QM法
~Egisonにて~

Kentaro Honda

QM法?

- 真理値表

p	q	r	s	結果
#f	#f	#f	#f	#f
#f	#f	#f	#t	#f
#f	#f	#t	#f	#f
#f	#f	#t	#t	#f
#f	#t	#f	#f	#t
#f	#t	#f	#t	#f
#f	#t	#t	#f	#f
#f	#t	#t	#t	#f
#t	#f	#f	#f	#t
#t	#f	#f	#t	#t
#t	#f	#t	#f	#t
#t	#f	#t	#t	#t
#t	#t	#f	#f	#t
#t	#t	#f	#t	#f
#t	#t	#t	#f	#t
#t	#t	#t	#t	#t

QM法?

- 真理値表

p	q	r	s	結果
#f	#f	#f	#f	#f
#f	#f	#f	#t	#f
#f	#f	#t	#f	#f
#f	#f	#t	#t	#f
#f	#t	#f	#f	#t
#f	#t	#f	#t	#f
#f	#t	#t	#f	#f
#f	#t	#t	#t	#f
#t	#f	#f	#f	#t
#t	#f	#f	#t	#t
#t	#f	#t	#f	#t
#t	#f	#t	#t	#t
#t	#t	#f	#f	#t
#t	#t	#f	#t	#f
#t	#t	#t	#f	#t
#t	#t	#t	#t	#t

QM法?

- 真理値表

p	q	r	s	結果
#f	#t	#f	#f	#t
#t	#f	#f	#f	#t
#t	#f	#f	#t	#t
#t	#f	#t	#f	#t
#t	#f	#t	#t	#t
#t	#t	#f	#f	#t
#t	#t	#t	#f	#t
#t	#t	#t	#t	#t

QM法?

- 真理値表

p	q	r	s	結果
#f	#t	#f	#f	#t
#t	#f	#f	#f	#t
#t	#f	#f	#t	#t
#t	#f	#t	#f	#t
#t	#f	#t	#t	#t
#t	#t	#f	#f	#t
#t	#t	#t	#f	#t
#t	#t	#t	#t	#t

QM法?

- 真理値表

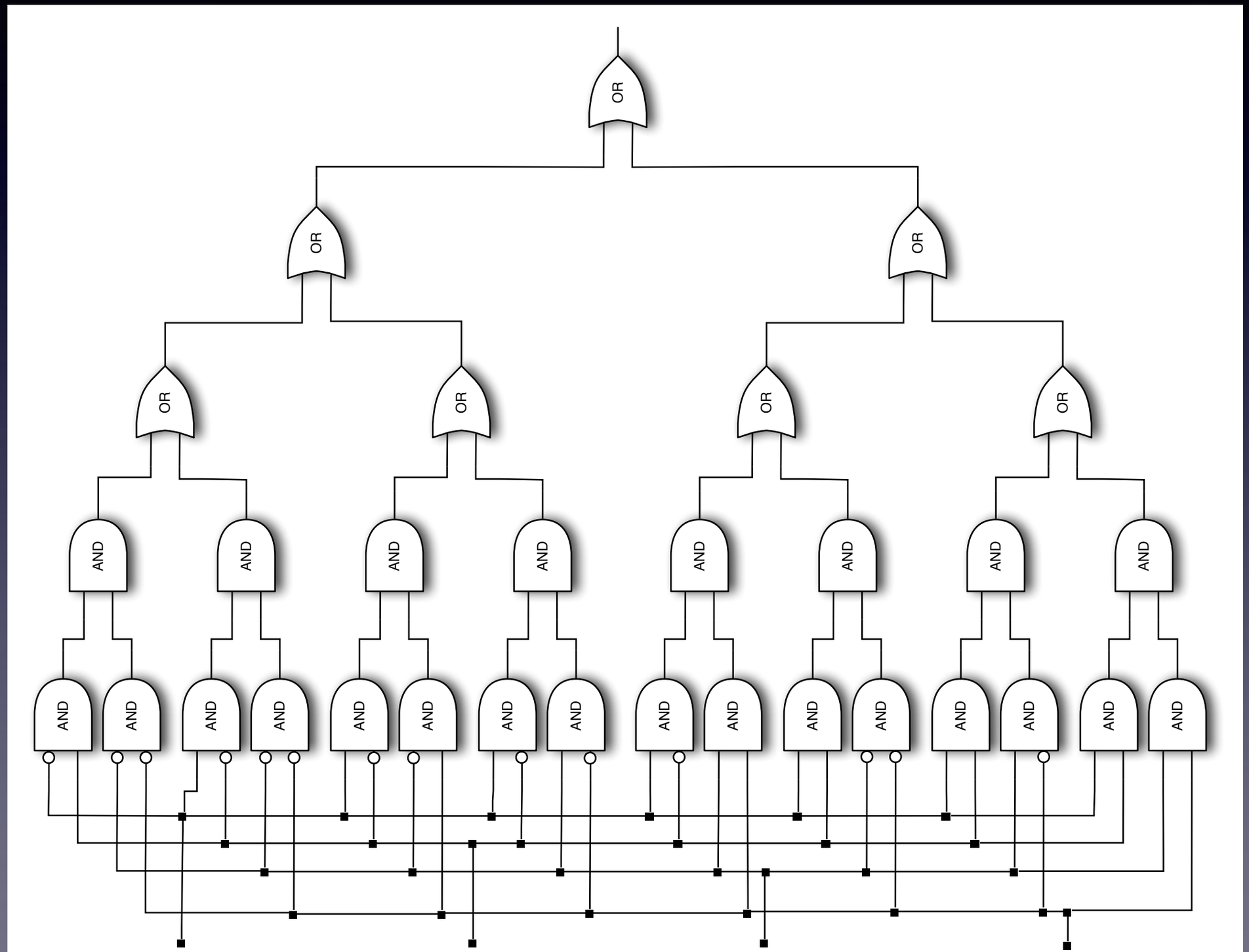
p	q	r	s
#f	#t	#f	#f
#t	#f	#f	#f
#t	#f	#f	#t
#t	#f	#t	#f
#t	#f	#t	#t
#t	#t	#f	#f
#t	#t	#t	#f
#t	#t	#t	#t

QM法?

- ブール関数

$$f(p,q,r,s) = \bar{p}q\bar{r}\bar{s} + p\bar{q}\bar{r}\bar{s} + p\bar{q}\bar{r}s + p\bar{q}r\bar{s} + p\bar{q}rs + pq\bar{r}\bar{s} + pqr\bar{s} + pqrs$$

QM法?


OR
OR
OR
OR
OR
OR
OR
AND
AND
AND
AND
AND
AND
AND
AND
AND
AND
AND
AND
AND
AND
AND
AND

QM法?

- 真理値表

p	q	r	s
#f	#t	#f	#f
#t	#f	#f	#f
#t	#f	#f	#t
#t	#f	#t	#f
#t	#f	#t	#t
#t	#t	#f	#f
#t	#t	#t	#f
#t	#t	#t	#t

QM法?

- 真理値表

p	q	r	s
#f	#t	#f	#f
#t	#f	#f	#f
#t	#f	#f	#t
#t	#f	#t	#f
#t	#f	#t	#t
#t	#t	#f	#f
#t	#t	#t	#f
#t	#t	#t	#t

QM法?

- 真理値表

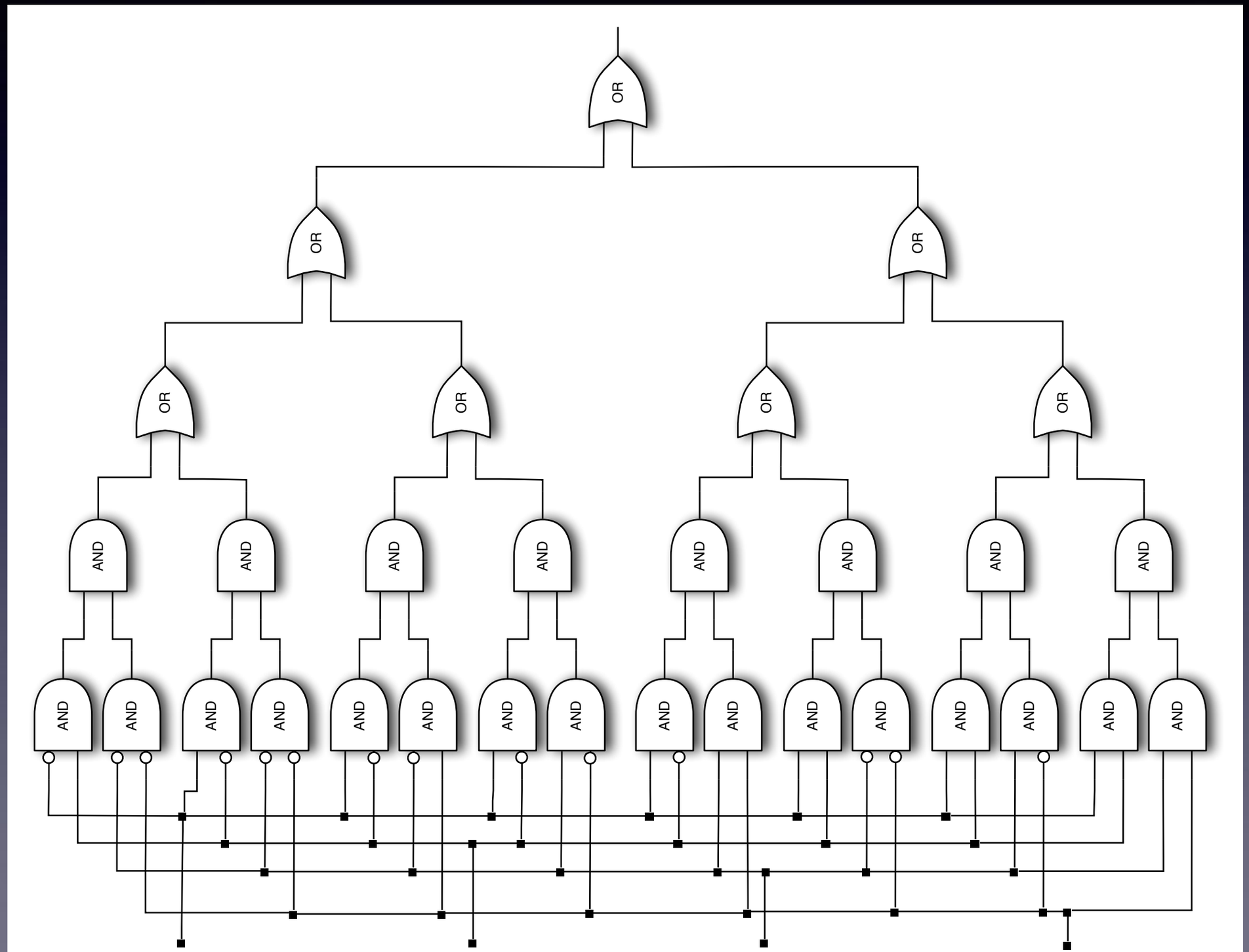
p	q	r	s
#f	#t	#f	#f
#t	#f	#f	#f
#t	#f	#f	#t
#t	#f	#t	#f
#t	#f	#t	#t
#t	#t	#f	#f
#t	#t	#t	#f
#t	#t	#t	#t

QM法?

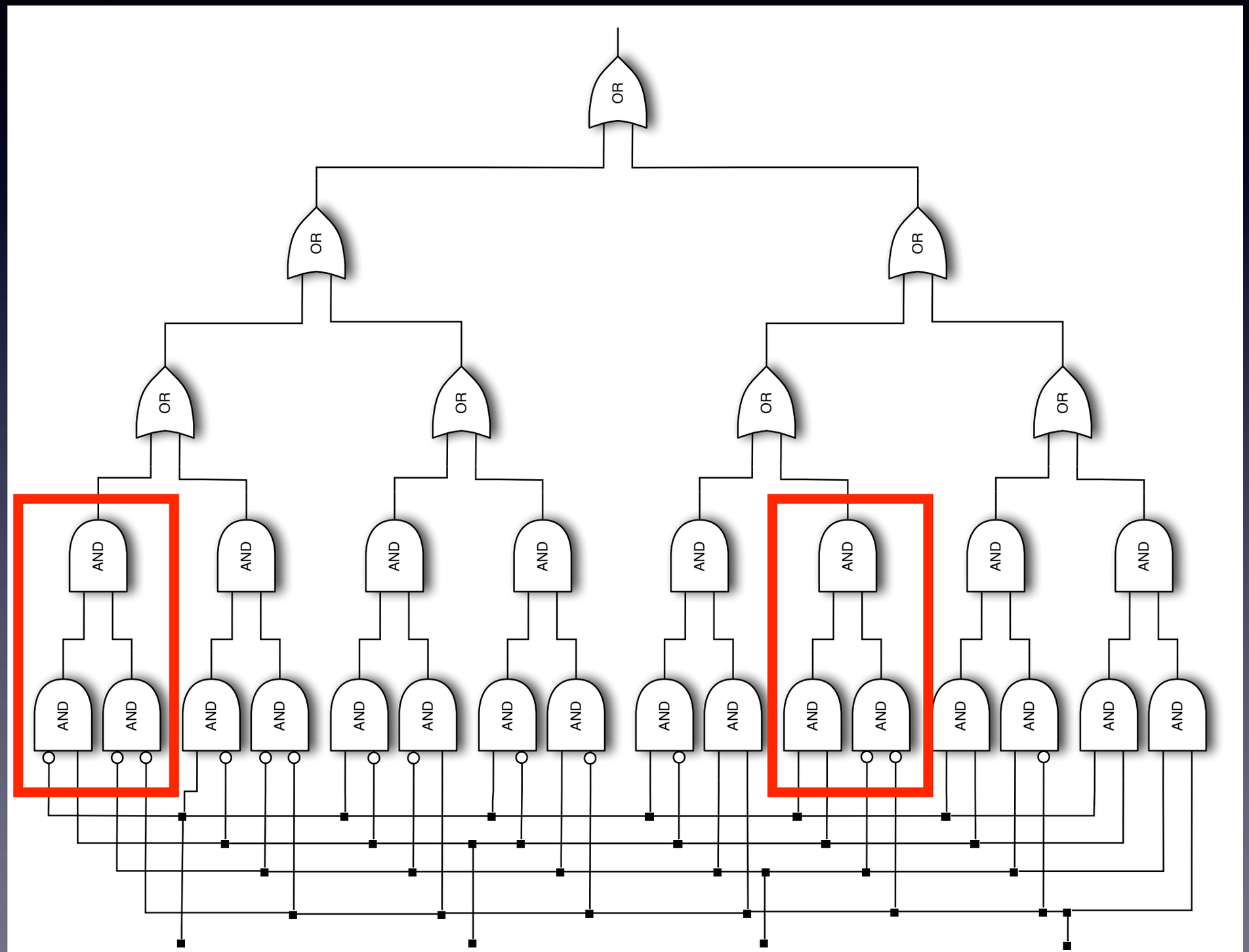
- 真理値表

p	q	r	s
-	#t	#f	#f
#t	#f	#f	#f
#t	#f	#f	#t
#t	#f	#t	#f
#t	#f	#t	#t
#t	#t	#f	#f
#t	#t	#t	#f
#t	#t	#t	#t

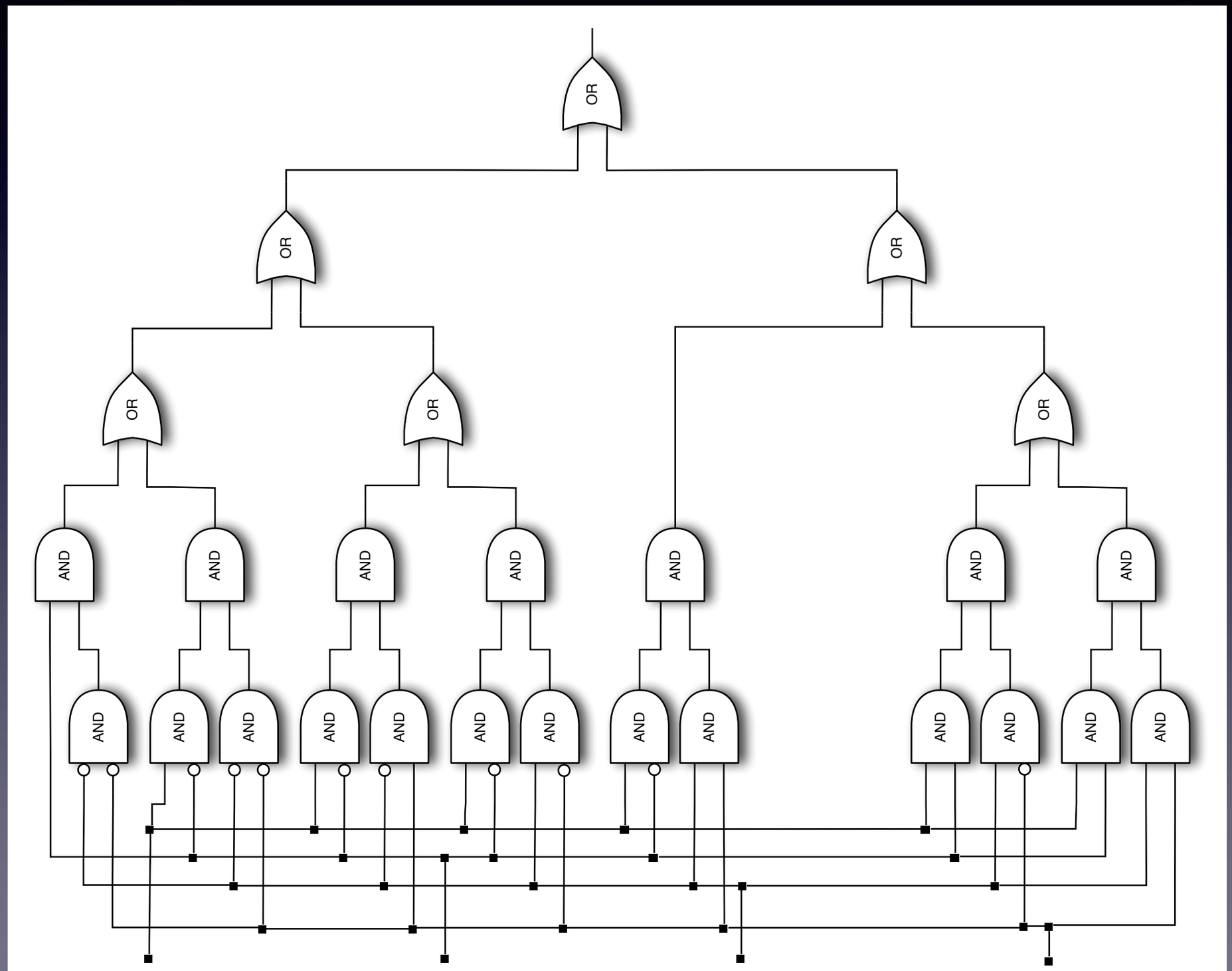
QM法?


OR
OR
OR
OR
OR
OR
OR
AND
AND
AND
AND
AND
AND
AND
AND
AND
AND
AND
AND
AND
AND
AND
AND

QM法?


OR
OR
OR
OR
OR
AND
AND
AND
AND
AND
AND
AND
AND
AND
AND
AND
AND
AND
AND
AND
AND

QM法?


OR
OR
OR
OR
OR
OR
AND
AND
AND
AND
AND
AND
AND
AND
AND
AND
AND
AND
AND
AND
AND
AND

QM法!

- ブール関数を簡単にしたい

  ➡ QM法

QM法：概要

1. 主項を求める
2. 必須項を求める
3. 最小被覆問題を解く

QM法：第1ステップ

- 主項を求める
- 各項をできるだけまとめる
- 主項：これ以上まとめられないもの

例：第1ステップ

p	q	r	s
#f	#t	#f	#f
#t	#f	#f	#f
#t	#f	#f	#t
#t	#f	#t	#f
#t	#f	#t	#t
#t	#t	#f	#f
#t	#t	#t	#f
#t	#t	#t	#t

例：第1ステップ

p	q	r	s
#f	#t	#f	#f
#t	#f	#f	#f
#t	#f	#f	#t
#t	#f	#t	#f
#t	#f	#t	#t
#t	#t	#f	#f
#t	#t	#t	#f
#t	#t	#t	#t

例：第1ステップ

p	q	r	s
#f	#t	#f	#f
#t	#f	#f	#f
#t	#f	#f	#t
#t	#f	#t	#f
#t	#f	#t	#t
#t	#t	#f	#f
#t	#t	#t	#f
#t	#t	#t	#t

p	q	r	s
-	#t	#f	#f
#t	-	#f	#f
#t	#f	-	#f
#t	#f	#f	-
#t	#f	-	#t
#t	-	#t	#f
#t	#f	#t	-
#t	-	#t	#t
#t	#t	-	#f
#t	#t	#t	-

例：第1ステップ

p	q	r	s
#f	#t	#f	#f
#t	#f	#f	#f
#t	#f	#f	#t
#t	#f	#t	#f
#t	#f	#t	#t
#t	#t	#f	#f
#t	#t	#t	#f
#t	#t	#t	#t

p	q	r	s
-	#t	#f	#f
#t	-	#f	#f
#t	#f	-	#f
#t	#f	#f	-
#t	#f	-	#t
#t	-	#t	#f
#t	#f	#t	-
#t	-	#t	#t
#t	#t	-	#f
#t	#t	#t	-

例：第1ステップ

p	q	r	s
#f	#t	#f	#f
#t	#f	#f	#f
#t	#f	#f	#t
#t	#f	#t	#f
#t	#f	#t	#t
#t	#t	#f	#f
#t	#t	#t	#f
#t	#t	#t	#t

p	q	r	s
-	#t	#f	#f
#t	-	#f	#f
#t	#f	-	#f
#t	#f	#f	-
#t	#f	-	#t
#t	-	#t	#f
#t	#f	#t	-
#t	-	#t	#t
#t	#t	-	#f
#t	#t	#t	-

例：第1ステップ

p	q	r	s
#f	#t	#f	#f
#t	#f	#f	#f
#t	#f	#f	#t
#t	#f	#t	#f
#t	#f	#t	#t
#t	#t	#f	#f
#t	#t	#t	#f
#t	#t	#t	#t

p	q	r	s
-	#t	#f	#f
#t	-	#f	#f
#t	#f	-	#f
#t	#f	#f	-
#t	#f	-	#t
#t	-	#t	#f
#t	#f	#t	-
#t	-	#t	#t
#t	#t	-	#f
#t	#t	#t	-

例：第1ステップ

p	q	r	s
#f	#t	#f	#f
#t	#f	#f	#f
#t	#f	#f	#t
#t	#f	#t	#f
#t	#f	#t	#t
#t	#t	#f	#f
#t	#t	#t	#f
#t	#t	#t	#t

p	q	r	s
-	#t	#f	#f
#t	-	#f	#f
#t	#f	-	#f
#t	#f	#f	-
#t	#f	-	#t
#t	-	#t	#f
#t	#f	#t	-
#t	-	#t	#t
#t	#t	-	#f
#t	#t	#t	-

例：第1ステップ

p	q	r	s
#f	#t	#f	#f
#t	#f	#f	#f
#t	#f	#f	#t
#t	#f	#t	#f
#t	#f	#t	#t
#t	#t	#f	#f
#t	#t	#t	#f
#t	#t	#t	#t

p	q	r	s
-	#t	#f	#f
#t	-	#f	#f
#t	#f	-	#f
#t	#f	#f	-
#t	#f	-	#t
#t	-	#t	#f
#t	#f	#t	-
#t	-	#t	#t
#t	#t	-	#f
#t	#t	#t	-

例：第1ステップ

p	q	r	s
#f	#t	#f	#f
#t	#f	#f	#f
#t	#f	#f	#t
#t	#f	#t	#f
#t	#f	#t	#t
#t	#t	#f	#f
#t	#t	#t	#f
#t	#t	#t	#t

p	q	r	s
-	#t	#f	#f
#t	-	#f	#f
#t	#f	-	#f
#t	#f	#f	-
#t	#f	-	#t
#t	-	#t	#f
#t	#f	#t	-
#t	-	#t	#t
#t	#t	-	#f
#t	#t	#t	-

例：第1ステップ

p	q	r	s
#f	#t	#f	#f
#t	#f	#f	#f
#t	#f	#f	#t
#t	#f	#t	#f
#t	#f	#t	#t
#t	#t	#f	#f
#t	#t	#t	#f
#t	#t	#t	#t

p	q	r	s
-	#t	#f	#f
#t	-	#f	#f
#t	#f	-	#f
#t	#f	#f	-
#t	#f	-	#t
#t	-	#t	#f
#t	#f	#t	-
#t	-	#t	#t
#t	#t	-	#f
#t	#t	#t	-

例：第1ステップ

p	q	r	s
#f	#t	#f	#f
#t	#f	#f	#f
#t	#f	#f	#t
#t	#f	#t	#f
#t	#f	#t	#t
#t	#t	#f	#f
#t	#t	#t	#f
#t	#t	#t	#t

p	q	r	s
-	#t	#f	#f
#t	-	#f	#f
#t	#f	-	#f
#t	#f	#f	-
#t	#f	-	#t
#t	-	#t	#f
#t	#f	#t	-
#t	-	#t	#t
#t	#t	-	#f
#t	#t	#t	-

例：第1ステップ

p	q	r	s
#f	#t	#f	#f
#t	#f	#f	#f
#t	#f	#f	#t
#t	#f	#t	#f
#t	#f	#t	#t
#t	#t	#f	#f
#t	#t	#t	#f
#t	#t	#t	#t

p	q	r	s
-	#t	#f	#f
#t	-	#f	#f
#t	#f	-	#f
#t	#f	#f	-
#t	#f	-	#t
#t	-	#t	#f
#t	#f	#t	-
#t	-	#t	#t
#t	#t	-	#f
#t	#t	#t	-

例：第1ステップ

p	q	r	s
#f	#t	#f	#f
#t	#f	#f	#f
#t	#f	#f	#t
#t	#f	#t	#f
#t	#f	#t	#t
#t	#t	#f	#f
#t	#t	#t	#f
#t	#t	#t	#t

p	q	r	s
-	#t	#f	#f
#t	-	#f	#f
#t	#f	-	#f
#t	#f	#f	-
#t	#f	-	#t
#t	-	#t	#f
#t	#f	#t	-
#t	-	#t	#t
#t	#t	-	#f
#t	#t	#t	-

例：第1ステップ

p	q	r	s
#f	#t	#f	#f
#t	#f	#f	#f
#t	#f	#f	#t
#t	#f	#t	#f
#t	#f	#t	#t
#t	#t	#f	#f
#t	#t	#t	#f
#t	#t	#t	#t

p	q	r	s
-	#t	#f	#f
#t	-	#f	#f
#t	#f	-	#f
#t	#f	#f	-
#t	#f	-	#t
#t	-	#t	#f
#t	#f	#t	-
#t	-	#t	#t
#t	#t	-	#f
#t	#t	#t	-

例：第1ステップ

p	q	r	s
-	#t	#f	#f
#t	-	#f	#f
#t	#f	-	#f
#t	#f	#f	-
#t	#f	-	#t
#t	-	#t	#f
#t	#f	#t	-
#t	-	#t	#t
#t	#t	-	#f
#t	#t	#t	-

例：第1ステップ

p	q	r	s
-	#t	#f	#f
#t	-	#f	#f
#t	#f	-	#f
#t	#f	#f	-
#t	#f	-	#t
#t	-	#t	#f
#t	#f	#t	-
#t	-	#t	#t
#t	#t	-	#f
#t	#t	#t	-

例：第1ステップ

p	q	r	s
-	#t	#f	#f
#t	-	#f	#f
#t	#f	-	#f
#t	#f	#f	-
#t	#f	-	#t
#t	-	#t	#f
#t	#f	#t	-
#t	-	#t	#t
#t	#t	-	#f
#t	#t	#t	-

p	q	r	s
#t	-	-	#f
#t	#f	-	-
#t	-	#t	-

例：第1ステップ

p	q	r	s
-	#t	#f	#f
#t	-	#f	#f
#t	#f	-	#f
#t	#f	#f	-
#t	#f	-	#t
#t	-	#t	#f
#t	#f	#t	-
#t	-	#t	#t
#t	#t	-	#f
#t	#t	#t	-

p	q	r	s
#t	-	-	#f
#t	#f	-	-
#t	-	#t	-

例：第1ステップ

p	q	r	s
-	#t	#f	#f
#t	-	#f	#f
#t	#f	-	#f
#t	#f	#f	-
#t	#f	-	#t
#t	-	#t	#f
#t	#f	#t	-
#t	-	#t	#t
#t	#t	-	#f
#t	#t	#t	-

p	q	r	s
#t	-	-	#f
#t	#f	-	-
#t	-	#t	-

例：第1ステップ

p	q	r	s
-	#t	#f	#f
#t	-	#f	#f
#t	#f	-	#f
#t	#f	#f	-
#t	#f	-	#t
#t	-	#t	#f
#t	#f	#t	-
#t	-	#t	#t
#t	#t	-	#f
#t	#t	#t	-

p	q	r	s
#t	-	-	#f
#t	#f	-	-
#t	-	#t	-

例：第1ステップ

p	q	r	s
-	#t	#f	#f
#t	-	#f	#f
#t	#f	-	#f
#t	#f	#f	-
#t	#f	-	#t
#t	-	#t	#f
#t	#f	#t	-
#t	-	#t	#t
#t	#t	-	#f
#t	#t	#t	-

p	q	r	s
#t	-	-	#f
#t	#f	-	-
#t	-	#t	-

例：第1ステップ

p	q	r	s
-	#t	#f	#f
#t	-	#f	#f
#t	#f	-	#f
#t	#f	#f	-
#t	#f	-	#t
#t	-	#t	#f
#t	#f	#t	-
#t	-	#t	#t
#t	#t	-	#f
#t	#t	#t	-

p	q	r	s
#t	-	-	#f
#t	#f	-	-
#t	-	#t	-

例：第1ステップ

p	q	r	s
-	#t	#f	#f
#t	-	#f	#f
#t	#f	-	#f
#t	#f	#f	-
#t	#f	-	#t
#t	-	#t	#f
#t	#f	#t	-
#t	-	#t	#t
#t	#t	-	#f
#t	#t	#t	-

p	q	r	s
#t	-	-	#f
#t	#f	-	-
#t	-	#t	-

例：第1ステップ

p	q	r	s
-	#t	#f	#f
#t	-	#f	#f
#t	#f	-	#f
#t	#f	#f	-
#t	#f	-	#t
#t	-	#t	#f
#t	#f	#t	-
#t	-	#t	#t
#t	#t	-	#f
#t	#t	#t	-

p	q	r	s
#t	-	-	#f
#t	#f	-	-
#t	-	#t	-

例：第1ステップ

p	q	r	s
#f	#t	#f	#f
#t	#f	#f	#f
#t	#f	#f	#t
#t	#f	#t	#f
#t	#f	#t	#t
#t	#t	#f	#f
#t	#t	#t	#f
#t	#t	#t	#t

p	q	r	s
-	#t	#f	#f
#t	-	#f	#f
#t	#f	-	#f
#t	#f	#f	-
#t	#f	-	#t
#t	-	#t	#f
#t	#f	#t	-
#t	-	#t	#t
#t	#t	-	#f
#t	#t	#t	-

p	q	r	s
#t	-	-	#f
#t	#f	-	-
#t	-	#t	-

例：第1ステップ

p	q	r	s
#f	#t	#f	#f
#t	#f	#f	#f
#t	#f	#f	#t
#t	#f	#t	#f
#t	#f	#t	#t
#t	#t	#f	#f
#t	#t	#t	#f
#t	#t	#t	#t

p	q	r	s
-	#t	#f	#f
#t	-	#f	#f
#t	#f	-	#f
#t	#f	#f	-
#t	#f	-	#t
#t	-	#t	#f
#t	#f	#t	-
#t	-	#t	#t
#t	#t	-	#f
#t	#t	#t	-

p	q	r	s
#t	-	-	#f
#t	#f	-	-
#t	-	#t	-

主項 ：これ以上まとめられないもの

例：第1ステップ

主項

p	q	r	s
-	#t	#f	#f
#t	-	-	#f
#t	#f	-	-
#t	-	#t	-

第1ステップ

- 主項を求める
- 「ハミング距離1のものをまとめる」をできなくなるまで繰り返す

第2ステップ

主項

p	q	r	s
-	#t	#f	#f
#t	-	-	#f
#t	#f	-	-
#t	-	#t	-

元の真理値表

p	q	r	s
#f	#t	#f	#f
#t	#f	#f	#f
#t	#f	#f	#t
#t	#f	#t	#f
#t	#f	#t	#t
#t	#t	#f	#f
#t	#t	#t	#f
#t	#t	#t	#t

第2ステップ

主項

p	q	r	s
-	#t	#f	#f
#t	-	-	#f
#t	#f	-	-
#t	-	#t	-

必須項

元の真理値表

p	q	r	s	
#f	#t	#f	#f	✔
#t	#f	#f	#f	
#t	#f	#f	#t	
#t	#f	#t	#f	
#t	#f	#t	#t	
#t	#t	#f	#f	
#t	#t	#t	#f	
#t	#t	#t	#t	

第2ステップ

主項

p	q	r	s
-	#t	#f	#f
#t	-	-	#f
#t	#f	-	-
#t	-	#t	-

元の真理値表

p	q	r	s
#f	#t	#f	#f
#t	#f	#f	#f
#t	#f	#f	#t
#t	#f	#t	#f
#t	#f	#t	#t
#t	#t	#f	#f
#t	#t	#t	#f
#t	#t	#t	#t

×

第2ステップ

主項

p	q	r	s
-	#t	#f	#f
#t	-	-	#f
#t	#f	-	-
#t	-	#t	-

必須項

元の真理値表

p	q	r	s
#f	#t	#f	#f
#t	#f	#f	#f
#t	#f	#f	#t
#t	#f	#t	#f
#t	#f	#t	#t
#t	#t	#f	#f
#t	#t	#t	#f
#t	#t	#t	#t

✔

第2ステップ

主項

p	q	r	s
-	#t	#f	#f
#t	-	-	#f
#t	#f	-	-
#t	-	#t	-

元の真理値表

p	q	r	s	
#f	#t	#f	#f	
#t	#f	#f	#f	
#t	#f	#f	#t	
#t	#f	#t	#f	×
#t	#f	#t	#t	
#t	#t	#f	#f	
#t	#t	#t	#f	
#t	#t	#t	#t	

第2ステップ

主項

p	q	r	s
-	#t	#f	#f
#t	-	-	#f
#t	#f	-	-
#t	-	#t	-

元の真理値表

p	q	r	s	
#f	#t	#f	#f	
#t	#f	#f	#f	
#t	#f	#f	#t	
#t	#f	#t	#f	
#t	#f	#t	#t	×
#t	#t	#f	#f	
#t	#t	#t	#f	
#t	#t	#t	#t	

第2ステップ

主項

p	q	r	s
-	#t	#f	#f
#t	-	-	#f
#t	#f	-	-
#t	-	#t	-

元の真理値表

p	q	r	s
#f	#t	#f	#f
#t	#f	#f	#f
#t	#f	#f	#t
#t	#f	#t	#f
#t	#f	#t	#t
#t	#t	#f	#f
#t	#t	#t	#f
#t	#t	#t	#t

×

第2ステップ

主項

p	q	r	s
-	#t	#f	#f
#t	-	-	#f
#t	#f	-	-
#t	-	#t	-

元の真理値表

p	q	r	s	
#f	#t	#f	#f	
#t	#f	#f	#f	
#t	#f	#f	#t	
#t	#f	#t	#f	
#t	#f	#t	#t	
#t	#t	#f	#f	
#t	#t	#t	#f	×
#t	#t	#t	#t	

第2ステップ

主項

p	q	r	s
-	#t	#f	#f
#t	-	-	#f
#t	#f	-	-
#t	-	#t	-

必須項

元の真理値表

p	q	r	s
#f	#t	#f	#f
#t	#f	#f	#f
#t	#f	#f	#t
#t	#f	#t	#f
#t	#f	#t	#t
#t	#t	#f	#f
#t	#t	#t	#f
#t	#t	#t	#t

✔

第2ステップ

主項

p	q	r	s
-	#t	#f	#f
#t	-	-	#f
#t	#f	-	-
#t	-	#t	-

元の真理値表

p	q	r	s
#f	#t	#f	#f
#t	#f	#f	#f
#t	#f	#f	#t
#t	#f	#t	#f
#t	#f	#t	#t
#t	#t	#f	#f
#t	#t	#t	#f
#t	#t	#t	#t

第2ステップ

主項

p	q	r	s	
-	#t	#f	#f	必須項
#t	-	-	#f	
#t	#f	-	-	必須項
#t	-	#t	-	必須項

元の真理値表

p	q	r	s	
#f	#t	#f	#f	✔
#t	#f	#f	#f	×
#t	#f	#f	#t	✔
#t	#f	#t	#f	×
#t	#f	#t	#t	×
#t	#t	#f	#f	×
#t	#t	#t	#f	×
#t	#t	#t	#t	✔

第2ステップ

- 必須項を求める
  - 「元の真理表の項のうち、ただ１つの主項にしか一致しないもの」を全て探す
    - 対応する主項が必須項

第3ステップ

- 「元の真理値表と一致する最小の主項の集合」を探す

- 必須項はこの集合に必ず入る

第3ステップ

主項

p	q	r	s
-	#t	#f	#f
#t	-	-	#f
#t	#f	-	-
#t	-	#t	-

必須項

必須項

必須項

元の真理値表

p	q	r	s
#f	#t	#f	#f
#t	#f	#f	#f
#t	#f	#f	#t
#t	#f	#t	#f
#t	#f	#t	#t
#t	#t	#f	#f
#t	#t	#t	#f
#t	#t	#t	#t

第3ステップ

主項

p	q	r	s	
-	#t	#f	#f	必須項
#t	-	-	#f	
#t	#f	-	-	必須項
#t	-	#t	-	必須項

元の真理値表

p	q	r	s
#f	#t	#f	#f
#t	#f	#f	#f
#t	#f	#f	#t
#t	#f	#t	#f
#t	#f	#t	#t
#t	#t	#f	#f
#t	#t	#t	#f
#t	#t	#t	#t

第3ステップ

主項

p	q	r	s	
-	#t	#f	#f	必須項
#t	-	-	#f	
#t	#f	-	-	必須項
#t	-	#t	-	必須項

元の真理値表

p	q	r	s
#t	#f	#f	#f
#t	#f	#f	#t
#t	#f	#t	#f
#t	#f	#t	#t
#t	#t	#t	#f
#t	#t	#t	#t

第3ステップ

主項

p	q	r	s
-	#t	#f	#f
#t	-	-	#f
#t	#f	-	-
#t	-	#t	-

必須項

必須項

必須項

元の真理値表

p	q	r	s
#t	#f	#f	#f
#t	#f	#f	#t
#t	#f	#t	#f
#t	#f	#t	#t
#t	#t	#t	#f
#t	#t	#t	#t

第3ステップ

主項

p	q	r	s	
-	#t	#f	#f	必須項
#t	-	-	#f	
#t	#f	-	-	必須項
#t	-	#t	-	必須項

元の真理値表

p	q	r	s
#t	#f	#f	#f
#t	#f	#f	#t
#t	#f	#t	#f
#t	#f	#t	#t
#t	#t	#t	#f
#t	#t	#t	#t

第3ステップ

主項

p	q	r	s	
-	#t	#f	#f	必須項
#t	-	-	#f	
#t	#f	-	-	必須項
#t	-	#t	-	必須項

元の真理値表

p	q	r	s
#t	#t	#t	#f
#t	#t	#t	#t

第3ステップ

主項

p	q	r	s	
-	#t	#f	#f	必須項
#t	-	-	#f	
#t	#f	-	-	必須項
#t	-	#t	-	必須項

元の真理値表

p	q	r	s
#t	#t	#t	#f
#t	#t	#t	#t

第3ステップ

主項

p	q	r	s	
-	#t	#f	#f	必須項
#t	-	-	#f	
#t	#f	-	-	必須項
#t	-	#t	-	必須項

元の真理値表

p	q	r	s

第3ステップ

主項

p	q	r	s	
-	#t	#f	#f	必須項
#t	-	-	#f	
#t	#f	-	-	必須項
#t	-	#t	-	必須項

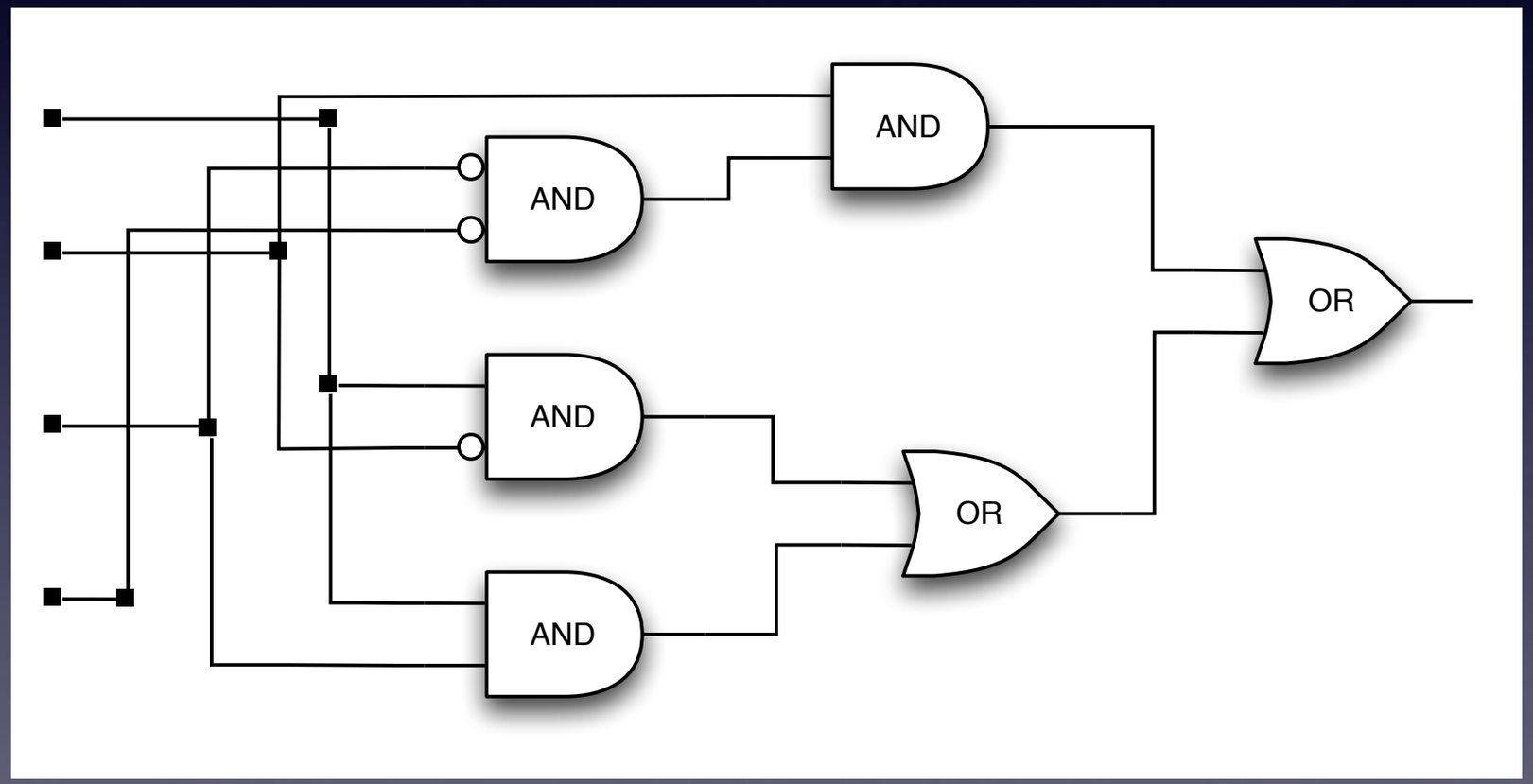
元の真理値表

p	q	r	s

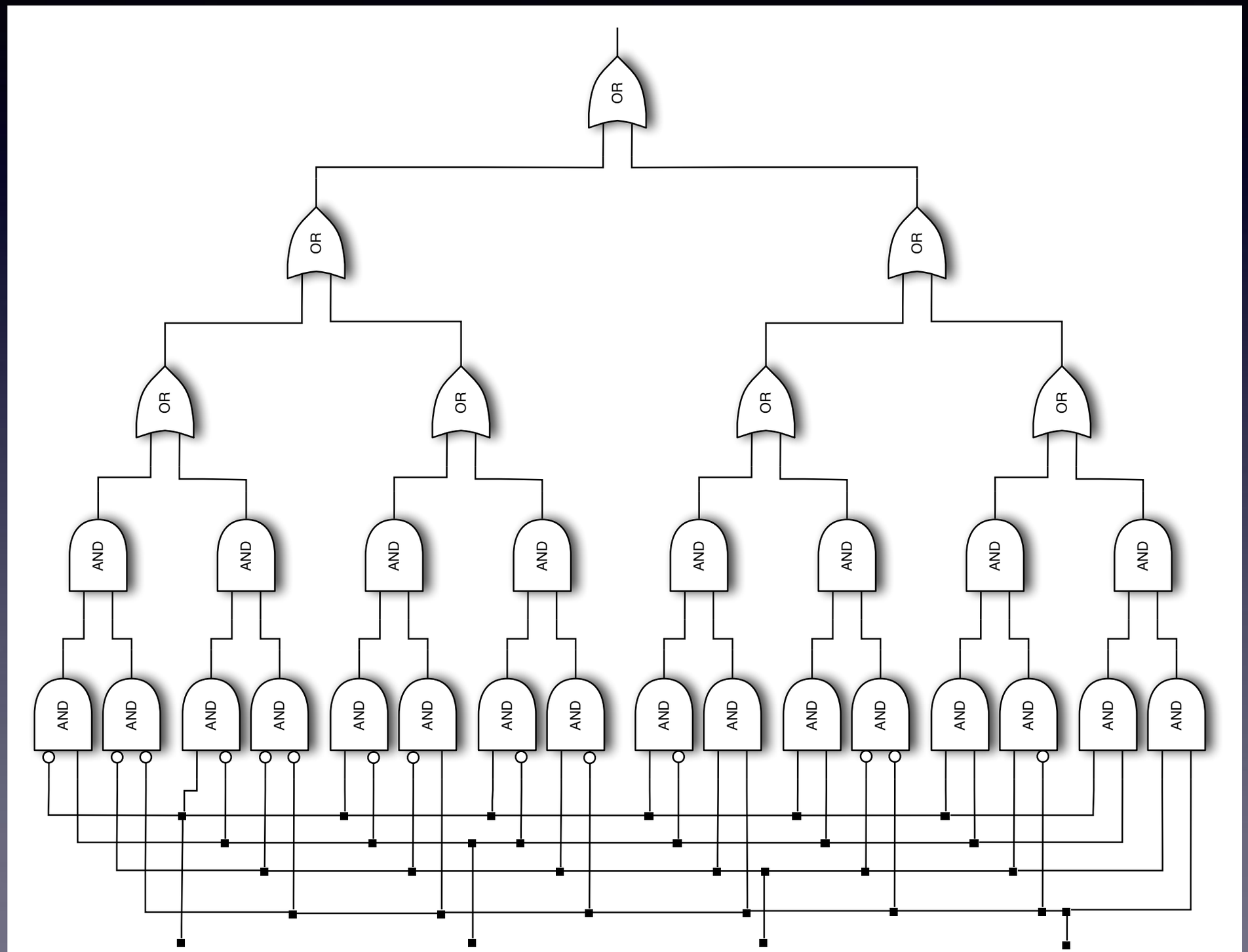
例：QM法

p	q	r	s
-	#t	#f	#f
#t	#f	-	-
#t	-	#t	-

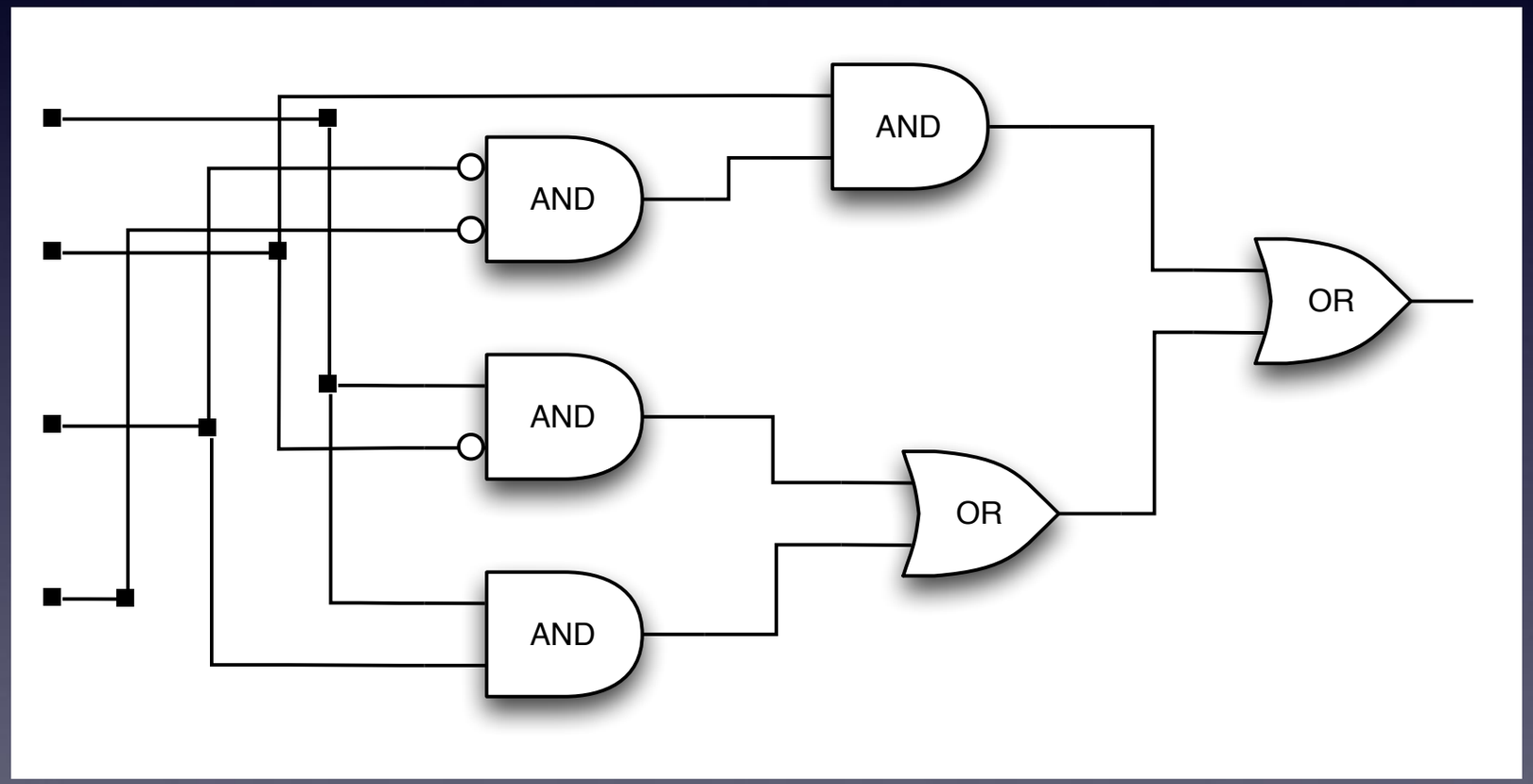
例：QM法



例：QM法


OR
OR
OR
OR
OR
OR
OR
AND
AND
AND
AND
AND
AND
AND
AND
AND
AND
AND
AND
AND
AND
AND
AND

例：QM法



データ型 in Egison

```
(define $Bool+
  (type
    {[$var-match (lambda [$tgt] {tgt})]
     [$inductive-match
      (destructor
        {[true []
          {[#t {[]}]
           [_ {}]}]
         [false []
          {[#f {[]}]
           [_ {}]}]
        [dontcare []
         {[<d> {[]}]
          [_ {}]}]})]
    [$=
     (lambda [$val $tgt]
       (match [val tgt] [Bool+ Bool+]
         {[[<true> <true>] #t]
          [[<false> <false>] #t]
          [[<dontcare> <dontcare>] #t]
          [[_ _] #f]}))]
    [$equal?
     (lambda [$val $tgt]
       (match [val tgt] [Bool+ Bool+]
         {[[$x ,x] #t]
          [[_ <dontcare>] #t]
          [[_ _] #f]}))]}))
```

- Bool+型
  - Bool値 + don’t care
  - 項：List Bool+
    - ex. {#t #f <d> <d>}
  - 真理値表：List List Bool+

第1ステップ

- 「ハミング距離1のものをまとめる」をできなくなるまで繰り返す

第1ステップ

```
(define $findPrimeImplicant
  (lambda [$as]
    (let {[$rs
           ((unique (List Bool+))
                    (match-all as (Multiset (List Bool+))
                      [<cons <join $xs <cons <false> $zs>> <cons ,{@xs #t @zs} _>> {@xs <d> @zs}]))]}
      (match rs (List Something)
        {[<nil> as]
         [_ {@(findPrimeImplicant rs) @(remove-collection-equal? as rs)}]}))))
```

第1ステップ

```
(define $findPrimeImplicant
  (lambda [$as]
    (let {[$rs
           ((unique (List Bool+))
                    (match-all as (Multiset (List Bool+))
                      [<cons <join $xs <cons <false> $zs>> <cons ,{@xs #t @zs} _>> {@xs <d> @zs}]))]}
      (match rs (List Something)
        {[<nil> as]
         [_ {@(findPrimeIm                    ion-equal? as rs)}]}))))
```

順番はどうでも良いのでMultisetでmatch

{@xs #f @zs}

第1ステップ

```
(define $findPrimeImplicant
  (lambda [$as]
    (let {[$rs
           ((unique (List Bool+))
                    (match-all as (Multiset (List Bool+))
                      [<cons <join $xs <cons <false> $zs>> <cons ,{@xs #t @zs} _>> {@xs <d> @zs}]))]}
      (match rs (List Something)
        {[<nil> as]
         [_ {@(findPrimeImplicant rs) @(remove-collection-equal? as rs)}]}))))
```

まとめられなかったもの

第1ステップ

```
(define $remove-all-equal?
  (lambda [$xs $x]
    ((remove-collection (List+ Bool+)) xs (filter (lambda [$y] ((type-ref (List+ Bool+) equal?) y x)) xs))))

(define $remove-collection-equal?
  (lambda [$xs $ys]
    (match ys (List Something)
      {[<nil> xs]
       [<cons $y $rs> (remove-collection-equal? (remove-all-equal? xs y) rs)]})))
```

第2ステップ

- 「元の真理表の項のうち、ただ１つの主項にしか一致しないもの」を全て探す

第2ステップ

```
(define $findMinimization
  (lambda [$aSet $bSet]
    (match [aSet bSet] [(Multiset (List+ Bool+)) (Multiset (List+ Bool+))]
      {[[<nil> _] {}]
       [[<cons $x $xs> <cons (& (? equal? x) $y) (& ^<cons (? equal? x) _> $zs)>]
        {y @(findMinimization (remove-all-equal? xs y) zs)}]
       [_ (findMinimumCover aSet bSet (size bSet) bSet)]})))
```

第2ステップ

```
(define $findMinimization
  (lambda [$aSet $bSet]
    (match [aSet bSet] [(Mu                                    )]
      {[[<nil> _] {}]
       [[<cons $x $xs> <cons (& (? equal? x) $y) (& ^<cons (? equal? x) _> $zs)>]
        {y @(findMinimization (re ove-all-equal? xs x) zs)}]
       [_ (findMinimumCover a                              b
```

&パターン：マッチした結果を束縛

Predパターン

Notパターン

Predパターン

- `(? fun val)`
  - `tgt`に対して
    if `(fun val tgt)` then matchする
    else matchしない
- `,x : (? = x)`

Notパターン

- ^pat : patにmatchしないときにmatch
  - e.g.

```
> (match-all {1 2 1 4 2 1} (Multiset Integer)
      [<cons $x ^<cons ,x _>> x])
{4}
```

第2ステップ

```
(define $findMinimization
  (lambda [$aSet $bSet]
    (match [aSet bSet] [(Multiset (List+ Bool+)) (Multiset (List+ Bool+))]
      {[[<nil> _] {}]
       [[<cons $x $xs> <cons (& (? equal? x) $y) (& ^<cons (? equal? x) _> $zs)>]
        {y @(findMinimization (remove-all-equal? xs y) zs)}]
       [_ (findMinimumCover aSet bSet (size bSet) bSet)]})))
```

第2ステップ

```
(define $findMinimization
  (lambda [$aSet $bSet]
    (match [aSet bSet] [(M                    l+))]
      {[[<nil> _] {}]
       [[<cons $x $xs> <cons (& (? equal? x) $y) (& ^<cons (? equal? x) _> $zs)>]
        {y @(findMinimization (remove-all-equal? xs y) zs)}]
       [_ (findMinimumCover aSet bSet (size bSet) bSet)]})))
```

xは１つの主項にしか一致しない

第3ステップ

- 「元の真理値表と一致する最小の主項の集合」を探す

第3ステップ

```
(define $findMinimumCover
  (lambda [$aSet $bSet $upper $uSet]
    (match [aSet bSet] [(Multiset (List+ Bool+)) (Multiset (List+ Bool+))]
      {[(& [_ <join (& ^(loop $l $i (between 2 upper) <cons _ l> <cons _ _>) $xs) $ys>]
           ^[<cons $z _> <join ^<cons (? equal? z) _> ,ys>])
        (findMinimumCover aSet bSet (size xs) xs)]
       [_ uSet]})))
```

第3ステップ

```
(define $findMinimumCover
  (lambda [$aSet $bSet $u
    (match [aSet bSet] [(Multiset (List+ Bool+)) (Multiset (List+ Bool+))]
      {[(& [_ <join (& ^(loop $l $i (between 2 upper) <cons _ l> <cons _ _>) $xs) $ys>]
          ^[<cons $z _> <join ^<cons (? equal? z) _> ,ys>]])
       (findMinimumCover aSet bSet (size xs) xs)]
       [_ uSet]})))
```

xsのサイズはupper以上ではない

第3ステップ

```
(define $findMinimumCover
  (lambda [$aSet $bSet $upper $uSet]
    (match [aSet bSet] [(Multiset (List+ Bool+)) (Multiset (List+ Bool+))]
      {[(& [_ <join (& ^(loop $l $i (between 2 upper) <cons _ l> <cons _ _>) $xs) $ys>]
           ^[<cons $z _> <join ^<cons (? equal? z) _> ,ys>]
        (findMinimumCover aSet bSet (size xs) xs)]
        [_ uSet]}))
```

「zと一致するxsの元が無い」ようなzはない

結果

```
> (test (QM input))
{{<d> #t #f #f} {#t #f <d> <d>} {#t <d> #t <d>}}
```

```
(define $findPrimeImplicant
  (lambda [$as]
    (let {[$rs
           ((unique (List Bool+))
                    (match-all as (Multiset (List Bool+))
                      [<cons <join $xs <cons <false> $zs>> <cons ,{@xs #t @zs} _>> {@xs <d> @zs}]))]}
      (match rs (List Something)
        {[<nil> as]
         [_ {@(findPrimeImplicant rs) @(remove-collection-equal? as rs)}]}))))

(define $remove-all-equal?
  (lambda [$xs $x]
    ((remove-collection (List+ Bool+)) xs (filter (lambda [$y] ((type-ref (List+ Bool+) equal?) y x)) xs))))

(define $remove-collection-equal?
  (lambda [$xs $ys]
    (match ys (List Something)
      {[<nil> xs]
       [<cons $y $rs> (remove-collection-equal? (remove-all-equal? xs y) rs)]})))
```

```
(define $findMinimumCover
  (lambda [$aSet $bSet $upper $uSet]
    (match [aSet bSet] [(Multiset (List+ Bool+)) (Multiset (List+ Bool+))]
      {[(& [_ <join (& ^(loop $l $i (between 2 upper) <cons _ l> <cons _ _>) $xs) $ys>]
           ^[<cons $z _> <join ^<cons (? equal? z) _> ,ys>])
        (findMinimumCover aSet bSet (size xs) xs)]
       [_ uSet]})))

(define $findMinimization
  (lambda [$aSet $bSet]
    (match [aSet bSet] [(Multiset (List+ Bool+)) (Multiset (List+ Bool+))]
      {[[<nil> _] {}]
       [[<cons $x $xs> <cons (& (? equal? x) $y) (& ^<cons (? equal? x) _> $zs)>]
        {y @(findMinimization (remove-all-equal? xs y) zs)}]
       [_ (findMinimumCover aSet bSet (size bSet) bSet)]})))

(define $QM
  (lambda [$aSet]
    (findMinimization aSet (findPrimeImplicant aSet))))
```

考察

- 集合のパターンマッチ
- NOTパターン
- Match-all