eBISS European Business Intelligence Summer School

Data mining for local patterns

Toon Calders

ECOLE
POLYTECHNIQUE
DE BRUXELLES

Outline

PART I: Frequent itemset mining

- Definition & Applications
- Algorithms for Frequent Itemset Mining
- Extensions to other pattern types

PART II:

- Pattern explosion & Redundancy problem
- Methods to remove redundancy
  - Condensed representations
  - Statistical methods
  - Minimal Description Length

Motivation

- Originally stated in the context of *Market – Basket Analysis*
  - Data consists of transactions

  - Find unexpected associations between sets of products.
    - Store layout
    - Promotions
    - …

Definitions

- Transaction database:

TID	a	b	c
1	1	1	0
2	1	1	1
3	0	0	1

or,
more compact

TID	Items
1	a, b
2	a, b, c
3	c

Itemset I = set of items

supp(I) = # transactions containing I

EXAMPLE:

supp(abc) = 1

supp(ab) = 2

Association Rules and Confidence

- Association rule: X=>Y
  - X, Y non-empty itemsets
  - Meaning: *"Occurrence of X implies Y"*

E.g. A,B => C

- *"People who buy A and B, tend to buy C as well"*

- Confidence: "*Strength*" of the implication X=>Y

  conf(X=>Y) = support(XY) / support(X)

- Support of a rule X=>Y = support(XY)

Association Rule Mining Problem

- Given:
  - transaction database D
  - $0 \leq minsup \leq |D|$
  - $0 \leq minconf \leq 1$

- Find all rules X=>Y such that
  - support(X=>Y) $\geq$ minsup
  - conf(X=>Y) $\geq$ minconf

Association Rule Mining Problem

- **Minsup = 3**
- **Minconf = 65%**

Rule	Support	Confidence
A => B	4	67%
B => A	4	~~57%~~
A => C	3	~~50%~~
C => A	3	~~50%~~
B => C	5	71%
C => B	5	83%
A => BC	~~2~~	~~40%~~
AB => C	~~2~~	~~50%~~
AC => B	~~2~~	66%
B => AC	~~2~~	~~29%~~
BC => A	~~2~~	~~40%~~
C => AB	~~2~~	~~33%~~

TID	Items
1	A, B
2	B, C
3	B, C
4	A, B
5	A
6	B, C
7	A, C
8	A, B, C
9	A, B, C

Association Rule Mining Problem

- Typical approach:
  - First find all itemsets I s.t. support(I) ≥ minsup
  - Then: for all subsets X of I:
    - Test if confidence(X=>(I / X)) ≥ minconf

Frequent Itemset Mining Problem:

Given:

- Database D
- 0 ≤ minsup ≤ |D|

Find: all itemsets I such that

- support(I) ≥ minsup

Association Rule Mining

- **Minsup = 3**
  Minconf = 65%

TID	Items
1	A, B
2	B, C
3	B, C
4	A, B
5	A
6	B, C
7	A, C
8	A, B, C
9	A, B, C

→

Frequent Itemsets

set	supp
A	6
B	7
C	6
AB	4
AC	3
BC	5
ABC	~~2~~

Association Rule Mining

- **Minsup = 3**
  Minconf = 65%

TID	Items
1	A, B
2	B, C
3	B, C
4	A, B
5	A
6	B, C
7	A, C
8	A, B, C
9	A, B, C

→

Frequent Itemsets

set	supp
A	6
B	7
C	6
AB	4
AC	3
BC	5
ABC	~~2~~

Rules

Rule	Conf
A => B	67%
B => A	~~57%~~
A => C	~~50%~~
C => A	~~50%~~
B => C	71%
C => B	83%

Other Measures of Rule Quality

- **Confidence often criticized:**
  - **Beer => Snack (300) 75%**
  - **Beer => Diapers (200) 50%**
- **However:**
  - **Overall population:**
    - **86% buys snack**
    - **42% buys diapers**

TID	Items
1-100	Beer, Snack
101-200	Beer, Diapers, Snack
201-300	Beer, Diapers
301-400	Beer, Snack
401-500	Diapers, Snack
501-600	Snack
601-700	Snack

Beer has a negative effect on snacks, and a positive effect on diapers !

Other Measures of Rule Quality

- **Alternative measure:**
  - **Lift(X=>Y) = conf(X=>Y) / (support(Y) / |D|)**
    I.e., by which factor does P(Y) change if X is present?

  - **Beer => Snack 0.87**
  - **Beer => Diapers 1.72**

- **There exist many other measures as well:**
  - **Statistically based**
  - **Information theory based**

TID	Items
1-100	Beer, Snack
101-200	Beer, Diapers, Snack
201-300	Beer, Diapers
301-400	Beer, Snack
401-500	Diapers, Snack
501-600	Snack
601-700	Snack

Statistical-Based Measure: X^2-test

- **X^2-test for dependency between X and Y:**

Example: Beer => Snack

TID	Items
1-100	Beer, Snack
101-200	Beer, Diapers, Snack
201-300	Beer, Diapers
301-400	Beer, Snack
401-500	Diapers, Snack
501-600	Snack
601-700	Snack

observed

	¬X	X	
¬Y	0	100	100
Y	300	300	600
	300	400	700

Expected (indep.)

	¬X	X	
¬Y	42.9	57.1	100
Y	257.1	342.9	600
	300	400	700

$$\chi^2 = \sum_{i=1}^{r} \sum_{j=1}^{c} \frac{(O_{i,j} - E_{i,j})^2}{E_{i,j}}.$$

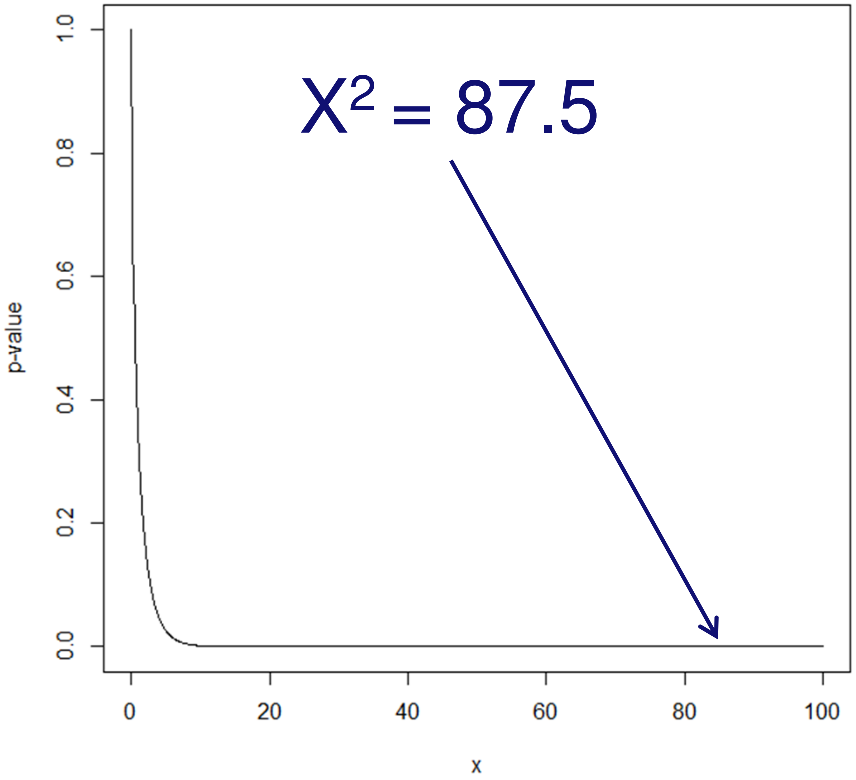
Statistical-Based Measure: X^2-test

observed

	¬X	X	
¬Y	0	100	100
Y	300	300	600
	300	400	700

Expected (indep.)

	¬X	X	
¬Y	42.9	57.1	100
Y	257.1	342.9	600
	300	400	700



$X^2 = 87.5$

- **P-value = probability of having a X^2 value at least as big as what we observe, *by chance***

Outline

PART I: Frequent itemset mining

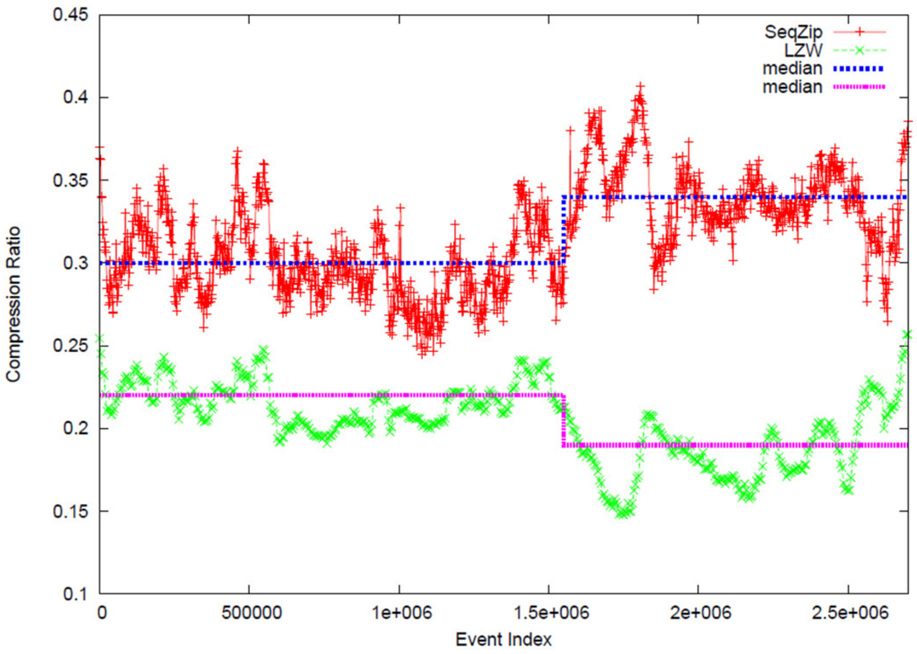
- Definition & Applications
- Algorithms for Frequent Itemset Mining
- Extensions to other pattern types

PART II:

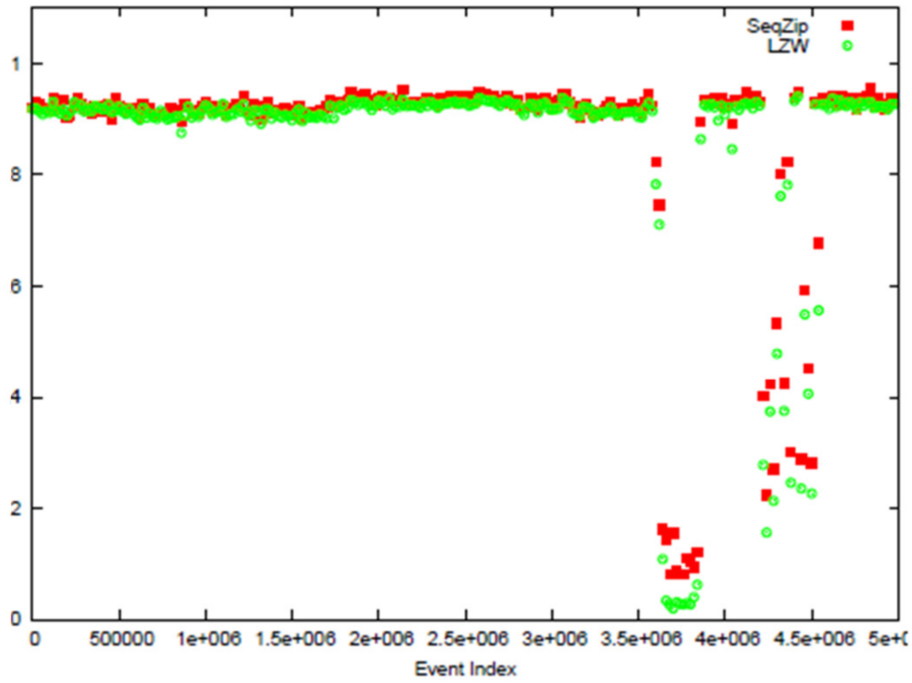
- Pattern explosion & Redundancy problem
- Methods to remove redundancy
    - Condensed representations
    - Statistical methods
    - Minimal Description Length

Why Itemset/Association Rule Mining?

- **Explorative data analysis**
  - **Find associations beyond simple correlation**
  - **Compute huge amounts of statistics at the same time**
  - **Changes in patterns can be significant**



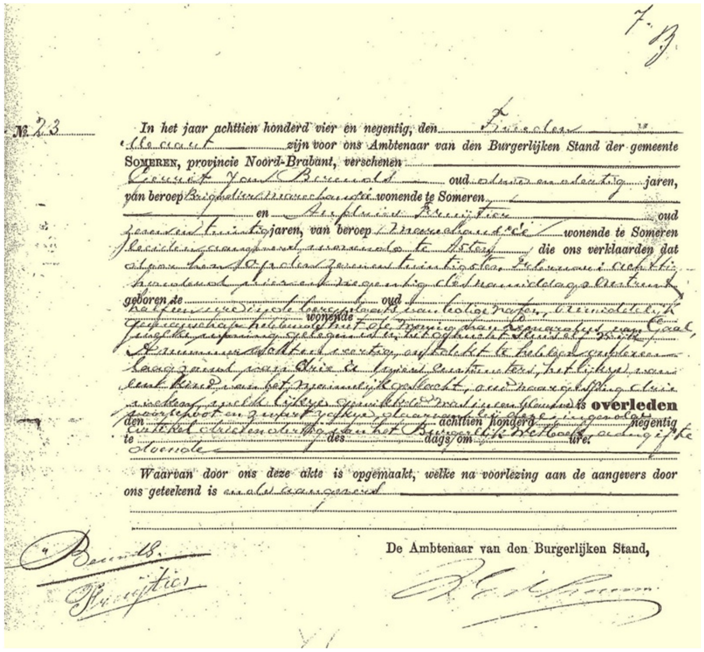
Machine log



World cup website visits

Why Itemset/Association Rule Mining?

- **Input to other data mining algorithms**



7. [illegible]

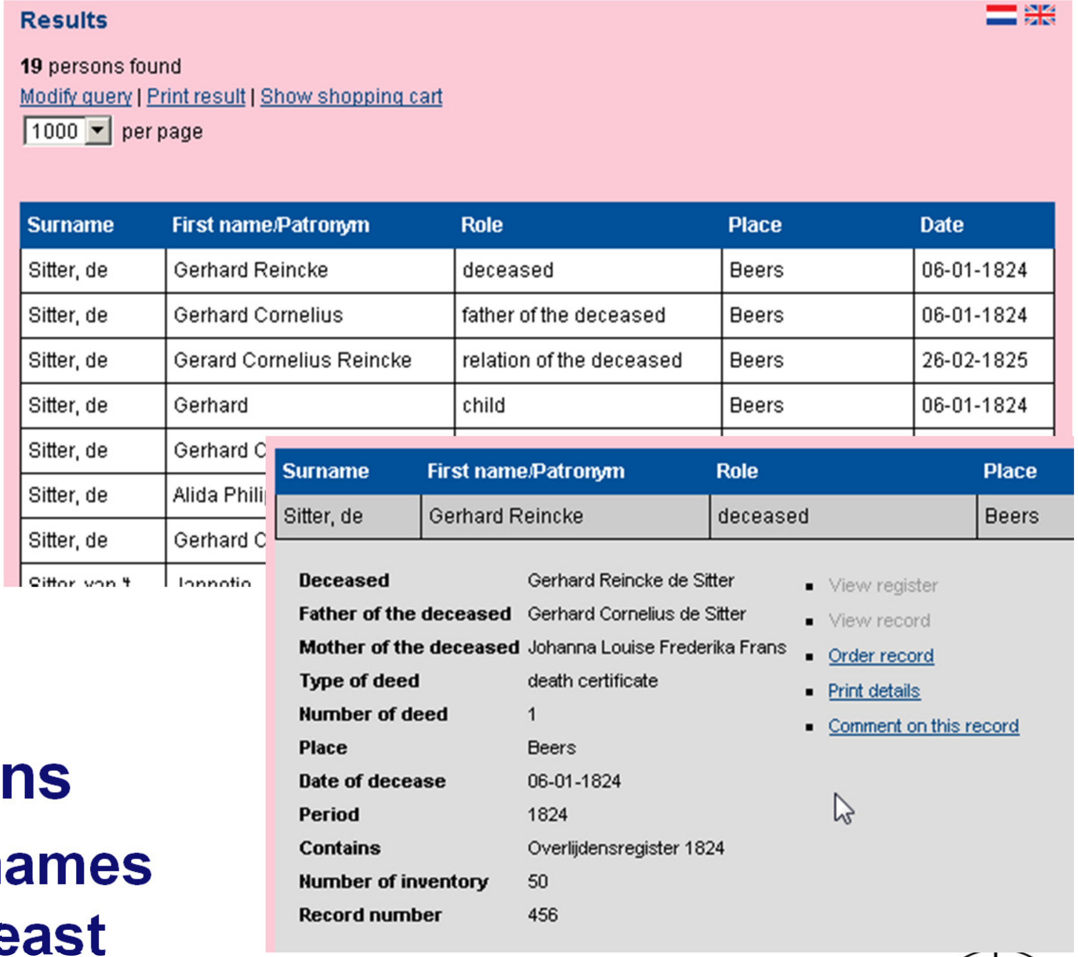
№ 23 In het jaar achttien honderd vier en negentig, den Zestien Maart zijn voor ons Ambtenaar van den Burgerlijken Stand der gemeente SOMEREN, provincie Noord-Brabant, verschenen Gerrit Jan Berends oud drie en dertig jaren, van beroep Brigadier Marechaussee wonende te Someren en Arplonis Fruijtier oud zeven en twintig jaren, van beroep Marechaussee wonende te Someren [illegible] te Aster die ons verklaarden dat op den [illegible] zeven en twintigsten Februari achttien honderd vier en negentig des namiddags omtrent geboren te oud half een [illegible] wonende [illegible] van Gaal [illegible] is **overleden** den achttien honderd negentig te des dags om ure. [illegible]

Waarvan door ons deze akte is opgemaakt, welke na voorlezing aan de aangevers door ons geteekend is [illegible]

Berends.

Fruijtier

De Ambtenaar van den Burgerlijken Stand,



Results

19 persons found

Modify query | Print result | Show shopping cart

1000 ▾ per page

Surname	First name/Patronym	Role	Place	Date
Sitter, de	Gerhard Reincke	deceased	Beers	06-01-1824
Sitter, de	Gerhard Cornelius	father of the deceased	Beers	06-01-1824
Sitter, de	Gerard Cornelius Reincke	relation of the deceased	Beers	26-02-1825
Sitter, de	Gerhard	child	Beers	06-01-1824
Sitter, de	Gerhard C			
Sitter, de	Alida Phili			
Sitter, de	Gerhard C			
Sitter, van 't	Jannetje			

Surname	First name/Patronym	Role	Place
Sitter, de	Gerhard Reincke	deceased	Beers

Deceased	Gerhard Reincke de Sitter
Father of the deceased	Gerhard Cornelius de Sitter
Mother of the deceased	Johanna Louise Frederika Frans
Type of deed	death certificate
Number of deed	1
Place	Beers
Date of decease	06-01-1824
Period	1824
Contains	Overlijdensregister 1824
Number of inventory	50
Record number	456

- **Finding name variations**
  - **Transaction = set of names co-occurring with at least 3 other names**

Why Itemset/Association Rule Mining?

- Data Summarization
  - What are the frequent patterns in my data?
  - Abstract away from infrequent patterns

social media
updat blog
thi weekend
social network
thi week
good luck
thi morn
michael jackson
give awai
iphon app
thi year
check thi video
complaint custom servic rep pleas respond
upload youtub video
join video chat tinychat
happi birthdai
thei ar
good morn
favorit youtub video
blog post

join chat peopl tinychat
blog post
thi time
iphon app
todai dai
thi morn
ha ad dailysourcecod
love thi
social media
justin bieber
love love
thi week
updat blog
good morn
complaint custom servic
youtub video
happi birthdai
subscrib channel youtub
thi year
good night

Illustration: eBISS registration

- Data: at eBISS registration
  - Highly interested in …
- 1 student → 1 transaction
  - Items = topics the student is highly interested in
- Result: toy dataset with 14 items & 36 transactions
  - Example: 0 0 0 1 1 0 1 1 0 0 1 1 1 1

    → { Ontologies, Semantic web, IR, DM, Graph mining, Cloud computing, Dist. Comp., Map Reduce }

Illustration: eBISS registration

```
0 0 0 1 1 0 1 1 0 0 1 1 1 1
1 1 0 1 0 0 0 1 1 0 0 1 1 1
1 1 1 1 1 0 1 1 1 1 1 0 0 0
1 1 0 0 0 1 1 1 0 0 0 0 1 1
1 1 0 0 0 0 0 1 0 0 0 1 0 0
0 0 0 0 0 0 1 1 0 1 1 0 0 1
1 1 0 0 1 0 0 1 0 0 0 0 0 0
1 1 0 0 0 0 1 1 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 1 1 1 1 0
0 0 0 0 0 0 0 1 0 0 0 1 1 0
1 1 0 1 1 0 0 1 0 0 0 0 0 0
0 0 0 1 1 1 1 1 0 0 0 0 0 0
1 1 0 0 0 0 0 1 1 0 0 0 0 0
0 0 0 0 0 1 0 1 1 0 0 0 0 0
1 1 0 0 0 0 0 1 1 1 1 1 1 1
1 1 0 1 1 0 0 0 0 0 0 0 0 0
1 1 0 0 0 0 0 0 0 1 0 0 0 1
1 1 0 0 0 0 0 0 0 1 0 0 0 1
1 1 0 0 0 0 0 0 0 1 0 0 0 1
```

```
1 1 0 0 0 0 0 0 0 1 0 0 0 1
1 1 0 0 0 0 1 0 0 1 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 1 0 0 1 0 1 0 0 0 1 1
1 1 1 0 0 0 1 1 1 0 1 1 1 1
0 1 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 0 1 0 1 0
1 1 1 1 1 0 1 1 1 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0 1 1
1 1 0 0 0 0 0 1 1 0 0 1 0 0
0 1 1 1 0 0 0 1 1 0 0 0 0 0
1 1 0 0 0 1 1 0 0 0 0 0 1 1
1 1 1 0 0 0 1 1 1 1 1 0 0 0
0 0 0 0 0 0 1 1 1 1 1 0 1 0
0 0 0 0 0 0 1 1 1 1 1 0 1 0
0 0 0 1 1 0 0 1 0 1 1 0 1 0
1 1 1 0 0 0 0 0 1 1 0 1 0 0
1 1 0 0 0 0 1 1 1 1 1 0 0 0
```

Frequent Sets (support 14 or more)

25 DW
24 DM
22 DB
16 IR
16 Visual analytics
15 Graph databases
14 Distributed computing
14 Map Reduce

22 DB DW
15 DW DM
14 DB DM
14 Visual analytics DM

14 DB DW DM

Illustration: eBISS registration

Lift	Conf	Supp	Rule
1.48	1.0	22	DB => DW
1.41	0.92	11	Graph mining => DM
2.31	1.0	7	GIS => Visual analytics
2.88	0.7	7	Ontologies => Semantic web
2.88	0.78	7	Semantic web => Ontologies
3.17	0.86	6	Semantic web, DM => Ontologies
3.08	0.75	6	Ontologies, DM => Semantic web
2.64	1.0	5	DB, Dist. Comp. => Map Reduce

Illustration: eBISS registration

Lift	Conf	Supp	Rule
1.54	0.5	8	IR => Graph mining
1.41	0.46	11	DM => Graph mining
1.85	0.6	9	Graph databases => Graph mining
2.06	0.67	8	IR, DM => Graph mining
3.08	1.0	8	DM, Graph databases => Graph mining

Outline

PART I: Frequent itemset mining

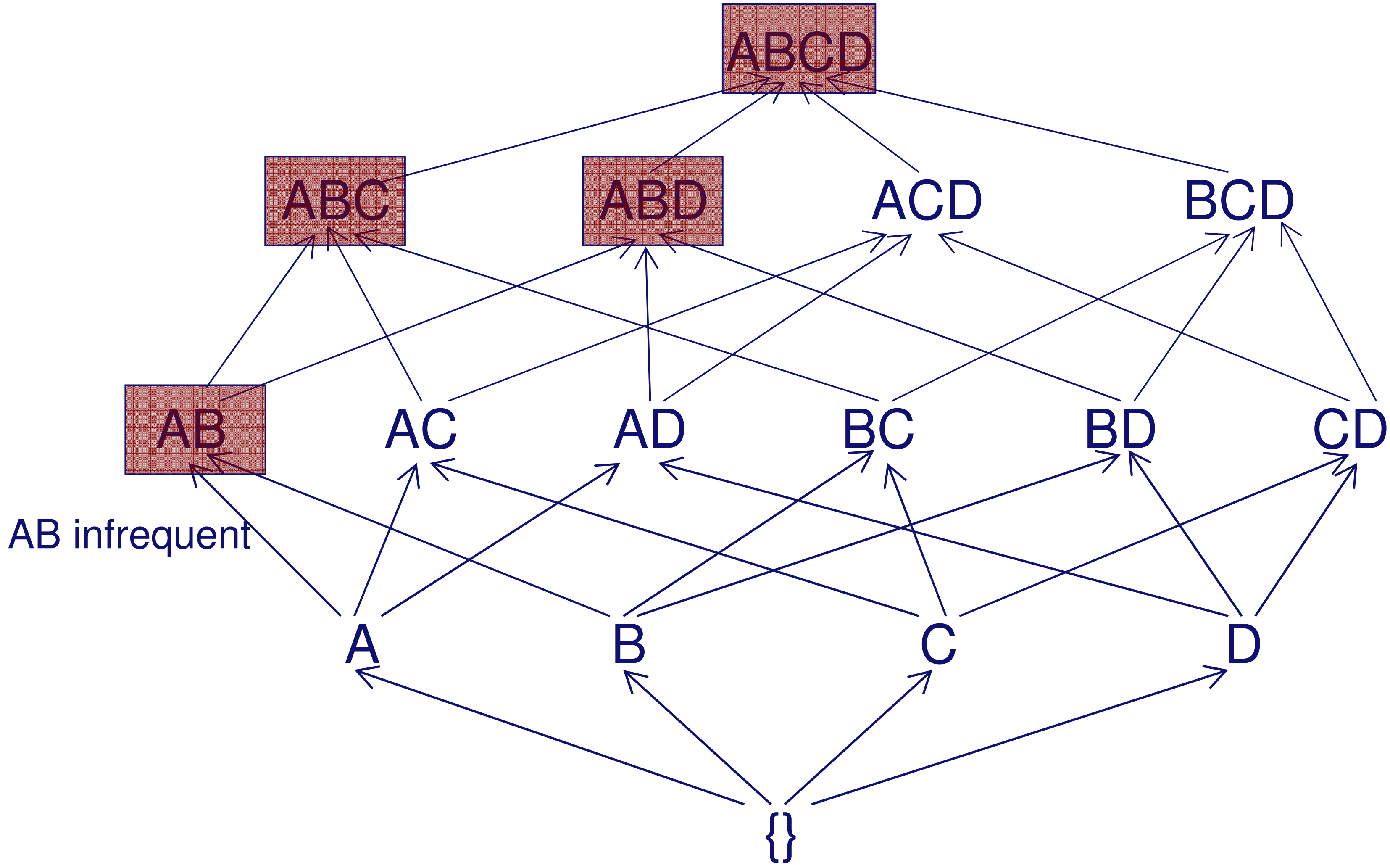
- Definition & Applications
- Algorithms for Frequent Itemset Mining
- Extensions to other pattern types

PART II:

- Pattern explosion & Redundancy problem
- Methods to remove redundancy
  - Condensed representations
  - Statistical methods
  - Minimal Description Length

Monotonicity Principle

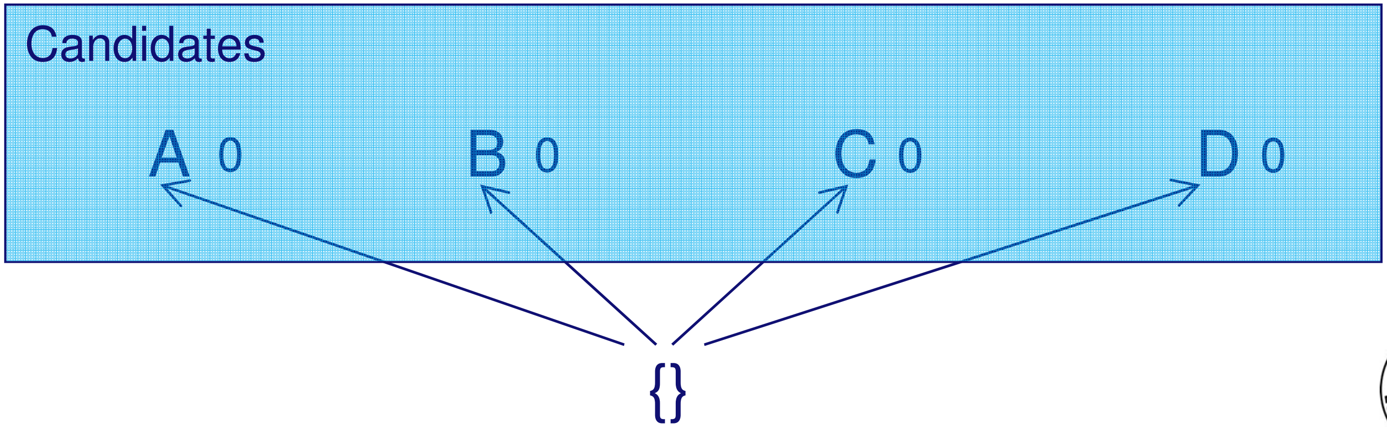
If X ⊆ Y, then support(X) ≥ support(Y)



Apriori

TID	A	B	C	D
1	0	1	1	0
2	0	1	1	0
3	1	0	1	1
4	1	1	1	1
5	0	1	0	1

minsup=2



Candidates

A 0

B 0

C 0

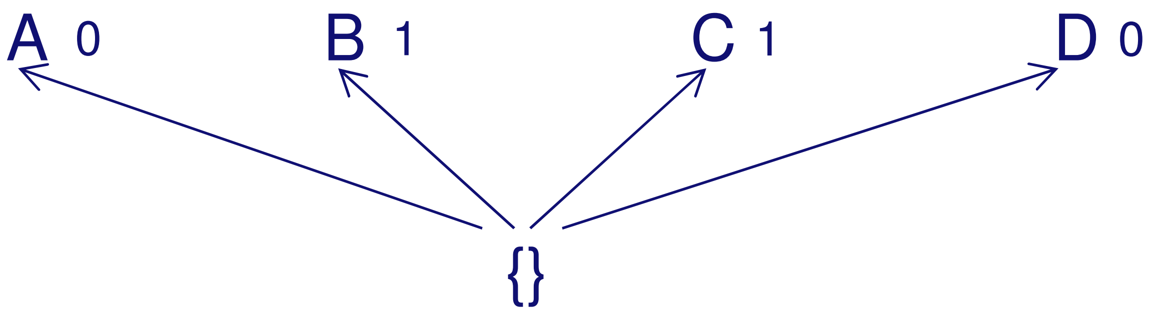
D 0

{}

Apriori

TID	A	B	C	D
1	0	1	1	0
2	0	1	1	0
3	1	0	1	1
4	1	1	1	1
5	0	1	0	1

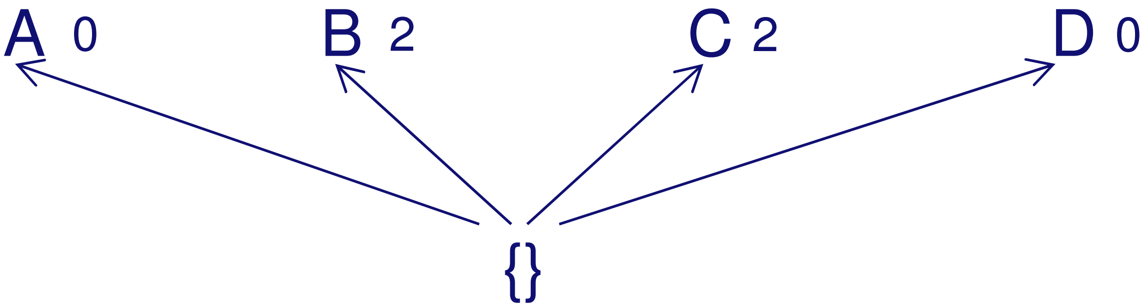
minsup=2



Apriori

TID	A	B	C	D
1	0	1	1	0
2	0	1	1	0
3	1	0	1	1
4	1	1	1	1
5	0	1	0	1

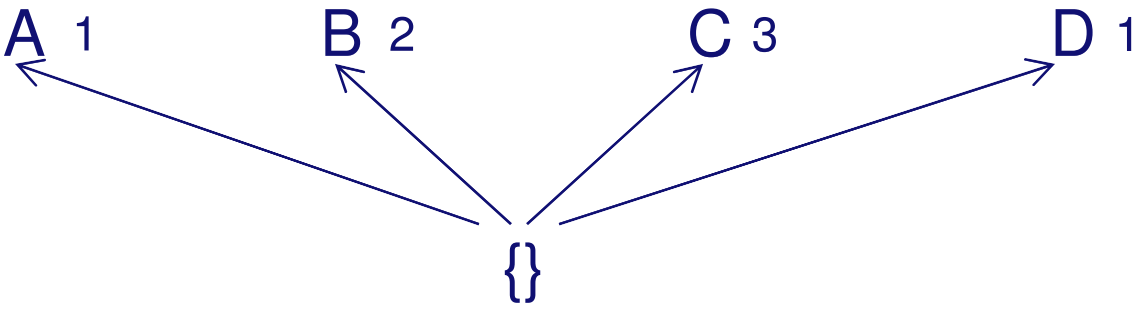
minsup=2



Apriori

TID	A	B	C	D
1	0	1	1	0
2	0	1	1	0
3	1	0	1	1
4	1	1	1	1
5	0	1	0	1

minsup=2



A 1

B 2

C 3

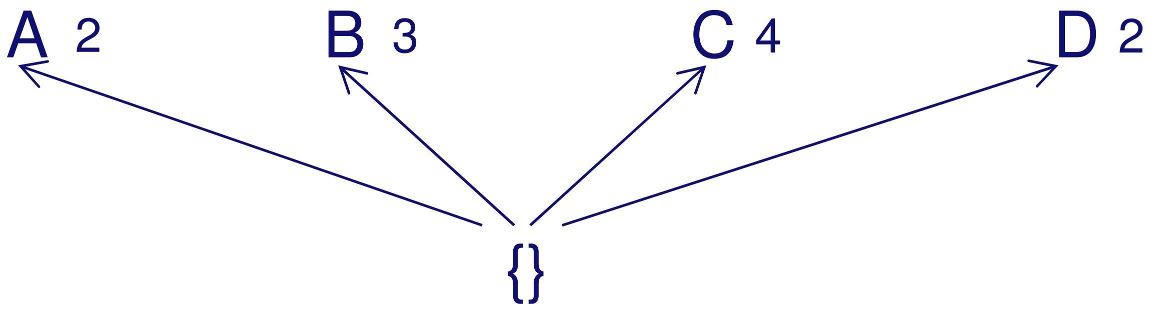
D 1

{}

Apriori

TID	A	B	C	D
1	0	1	1	0
2	0	1	1	0
3	1	0	1	1
4	1	1	1	1
5	0	1	0	1

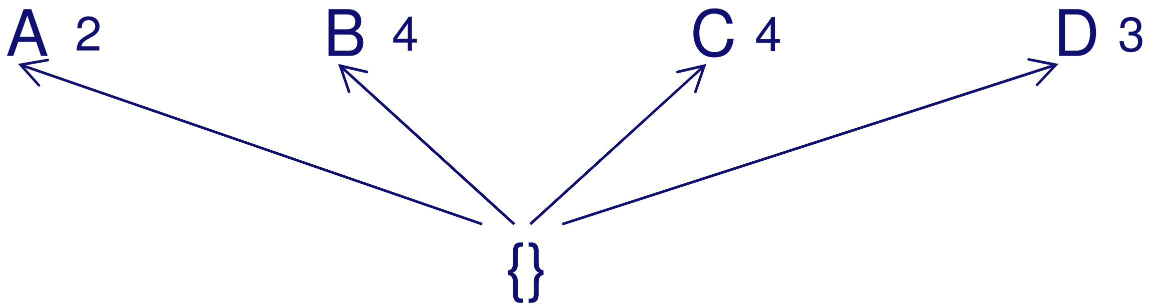
minsup=2



Apriori

TID	A	B	C	D
1	0	1	1	0
2	0	1	1	0
3	1	0	1	1
4	1	1	1	1
5	0	1	0	1

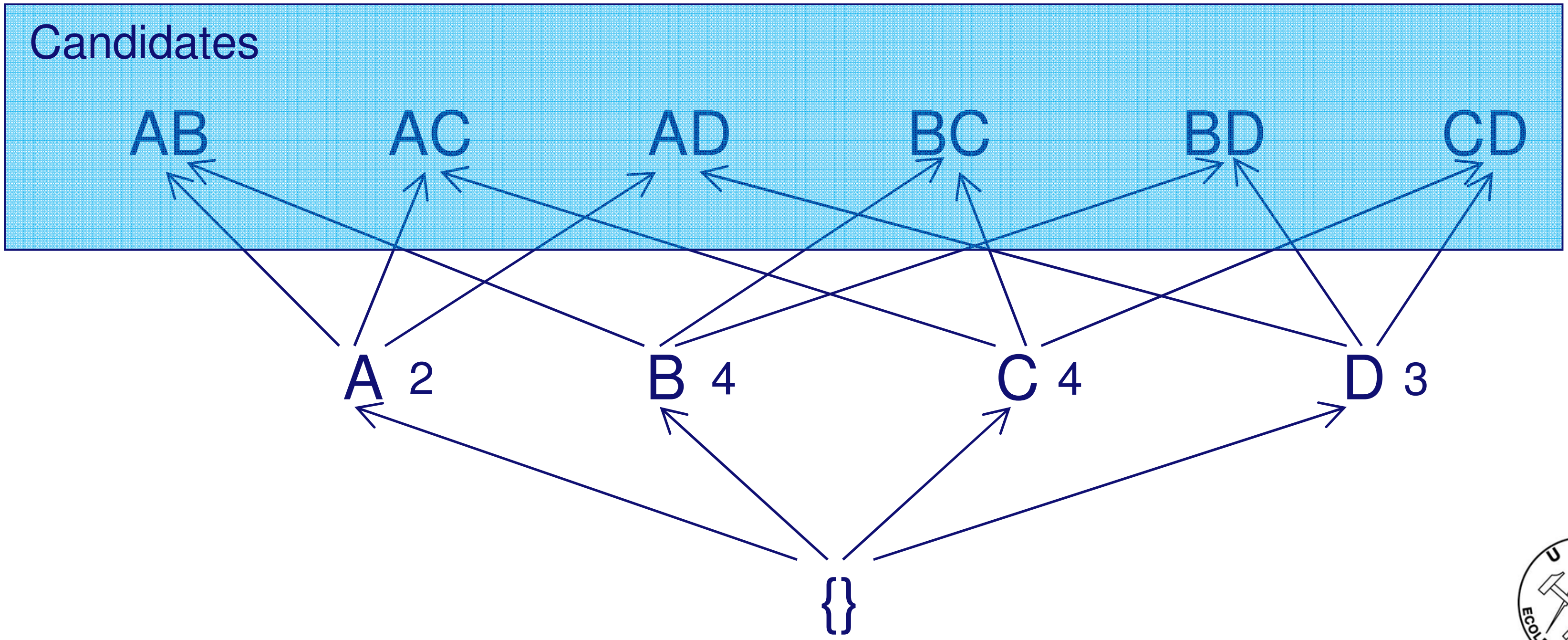
minsup=2



Apriori

TID	A	B	C	D
1	0	1	1	0
2	0	1	1	0
3	1	0	1	1
4	1	1	1	1
5	0	1	0	1

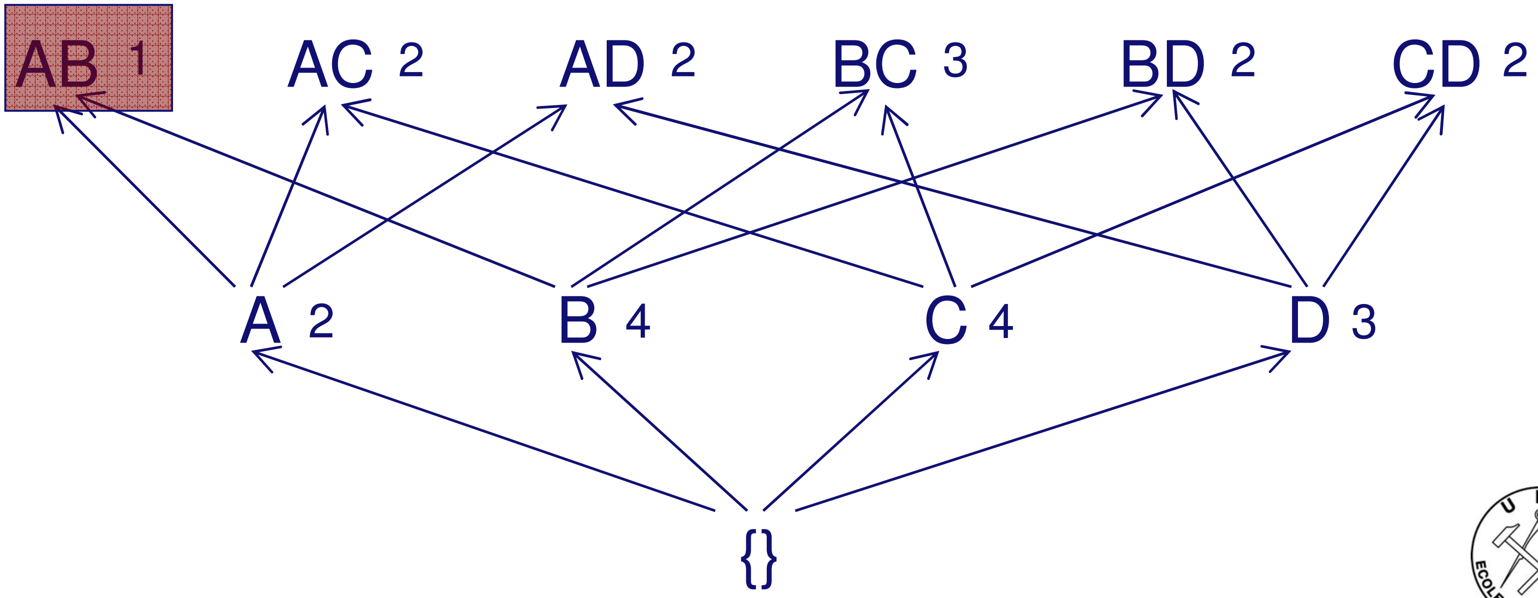
minsup=2



Apriori

TID	A	B	C	D
1	0	1	1	0
2	0	1	1	0
3	1	0	1	1
4	1	1	1	1
5	0	1	0	1

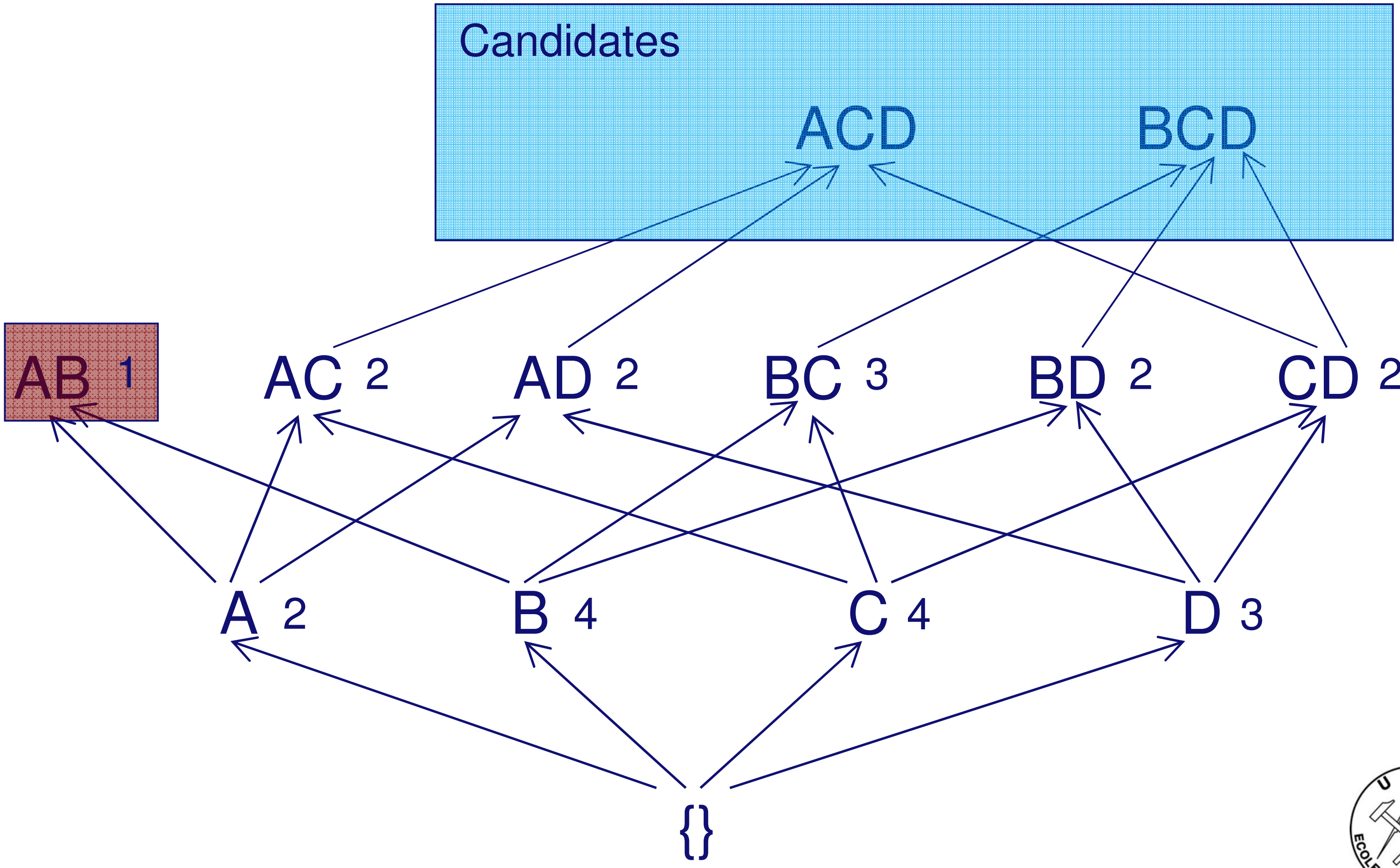
minsup=2



Apriori

TID	A	B	C	D
1	0	1	1	0
2	0	1	1	0
3	1	0	1	1
4	1	1	1	1
5	0	1	0	1

minsup=2



Candidates

ACD

BCD

AB 1

AC 2

AD 2

BC 3

BD 2

CD 2

A 2

B 4

C 4

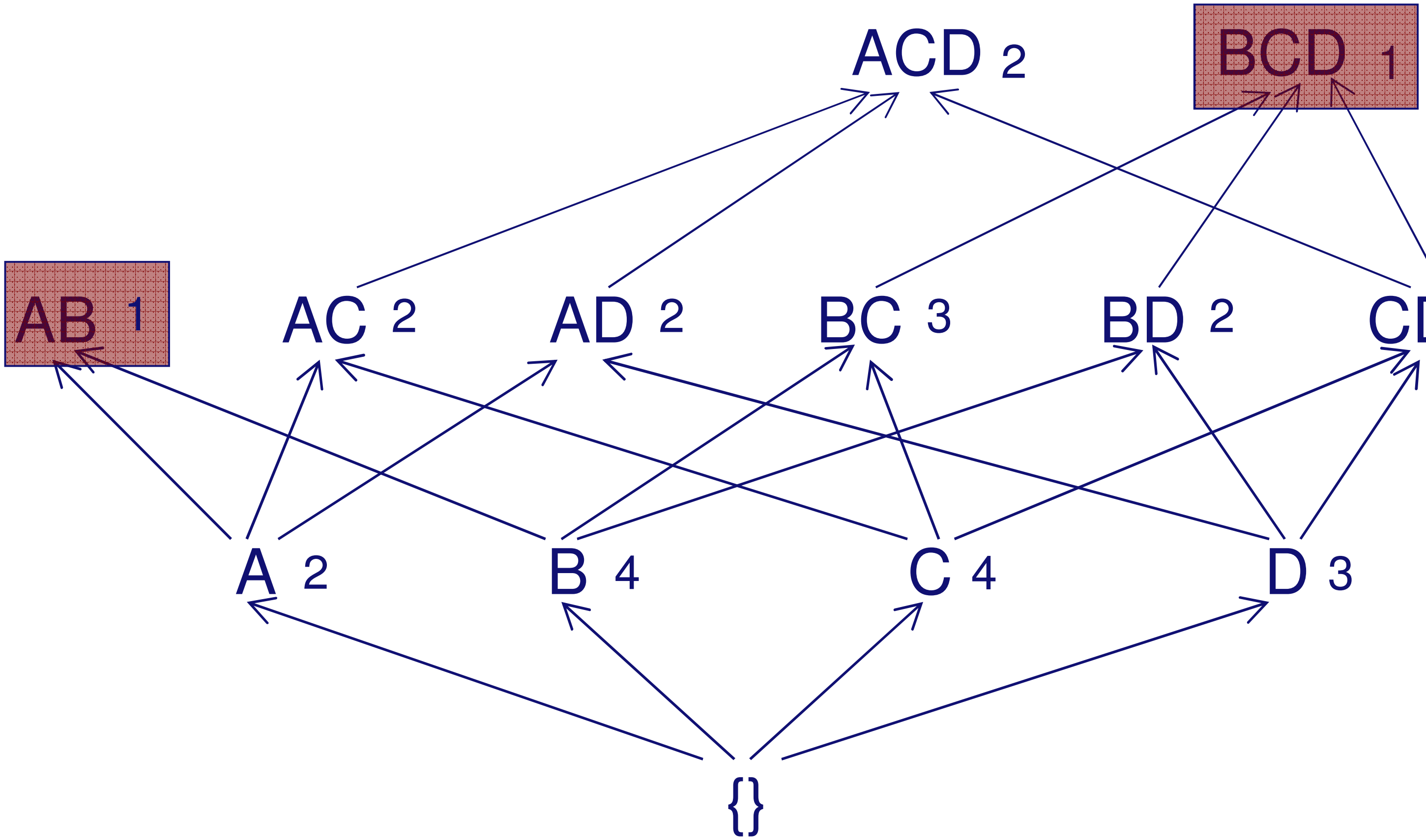
D 3

{}

Apriori

TID	A	B	C	D
1	0	1	1	0
2	0	1	1	0
3	1	0	1	1
4	1	1	1	1
5	0	1	0	1

minsup=2



ACD 2
BCD 1

AB 1
AC 2
AD 2
BC 3
BD 2
CD 2

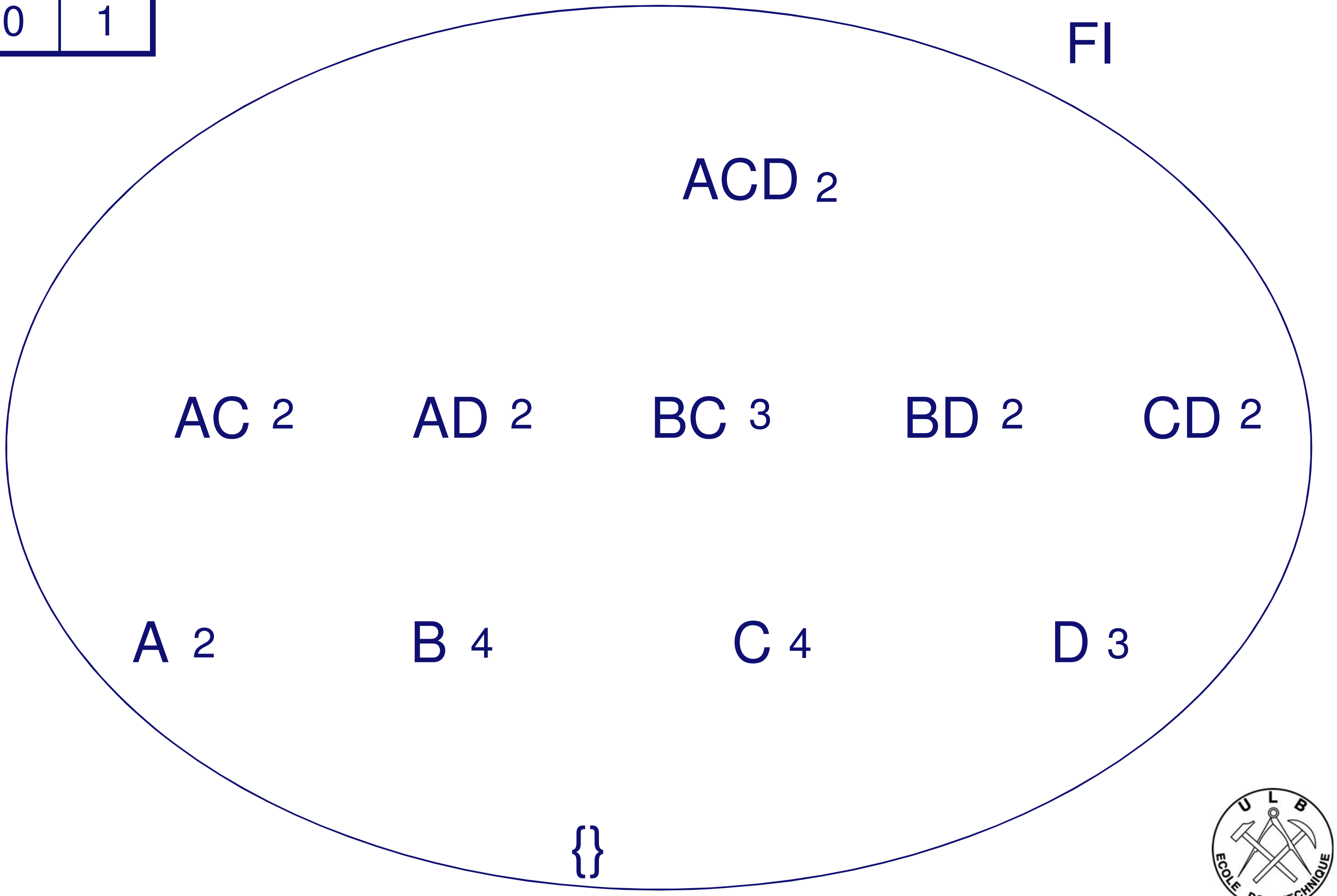
A 2
B 4
C 4
D 3

{}

Apriori

TID	A	B	C	D
1	0	1	1	0
2	0	1	1	0
3	1	0	1	1
4	1	1	1	1
5	0	1	0	1

minsup=2



FI

ACD 2

AC 2 AD 2 BC 3 BD 2 CD 2

A 2 B 4 C 4 D 3

{}

Apriori Algoritme

```
Apriori
Input: minsup, D
Output: Set of all frequent itemsets F

k := 1
$C_1$ := { {A} | A is an item }
Repeat until ($C_k$ = {}) {
   Count support of all itemsets in $C_k$ in 1 scan over D
      $F_k$ := { I $\in$ $C_k$ : I is frequent};
   Generate new candidates
      $C_{k+1}$ := { I : |I| = k+1 and all J $\subset$ I with |J|=k are in $F_k$};
   k:=k+1
}
Return $\cup_{i=1...k-1}$ $F_i$
```

Apriori: Summary

- Candidate generation is optimal:
  - If only information we can get from the database is whether or not an itemset I is frequent
  - Number of database scans is minimal (parallel queries to the database)

- What if:
  - we can load database into memory and transform the database?
  - we know the frequencies?

Depth-First Algorithms

- **Depth-First algorithms:**
  + **allow for more efficient counting**
    - **are based on divide-and-conquer**

  - **do not fully exploit monotonicity principle**

- **Counting all itemsets with item a?**

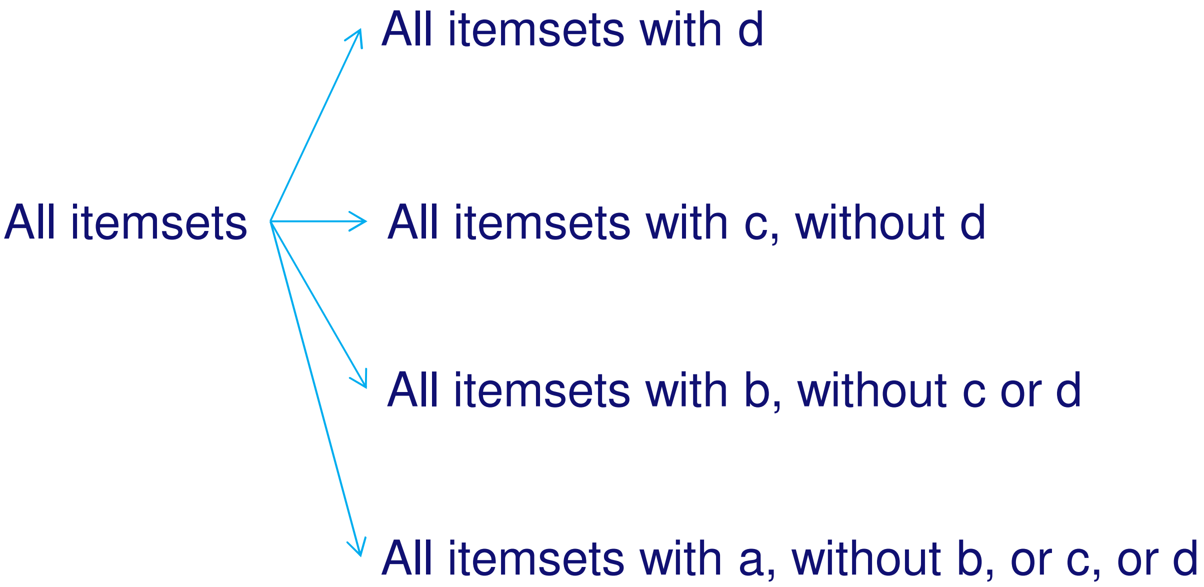
  → First reduce the database; remove all transactions without a

- **Counting all itemsets without a?**

  → Remove a from the database

Skip Depth-first

Divide-and-Conquer



All itemsets

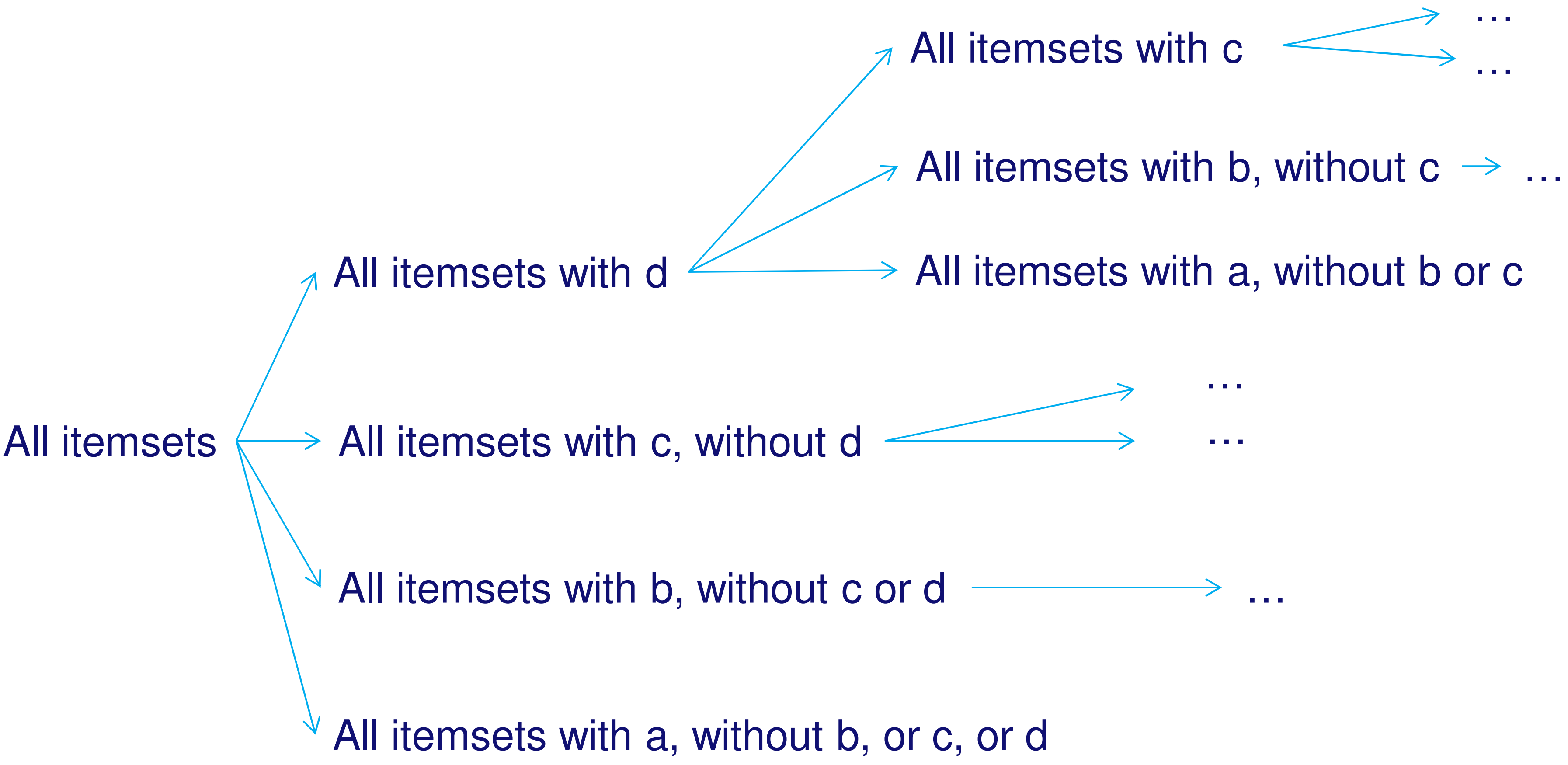
All itemsets with d

All itemsets with c, without d

All itemsets with b, without c or d

All itemsets with a, without b, or c, or d

Divide-and-Conquer



All itemsets

All itemsets with d

All itemsets with c

…

…

All itemsets with b, without c → …

All itemsets with a, without b or c

All itemsets with c, without d

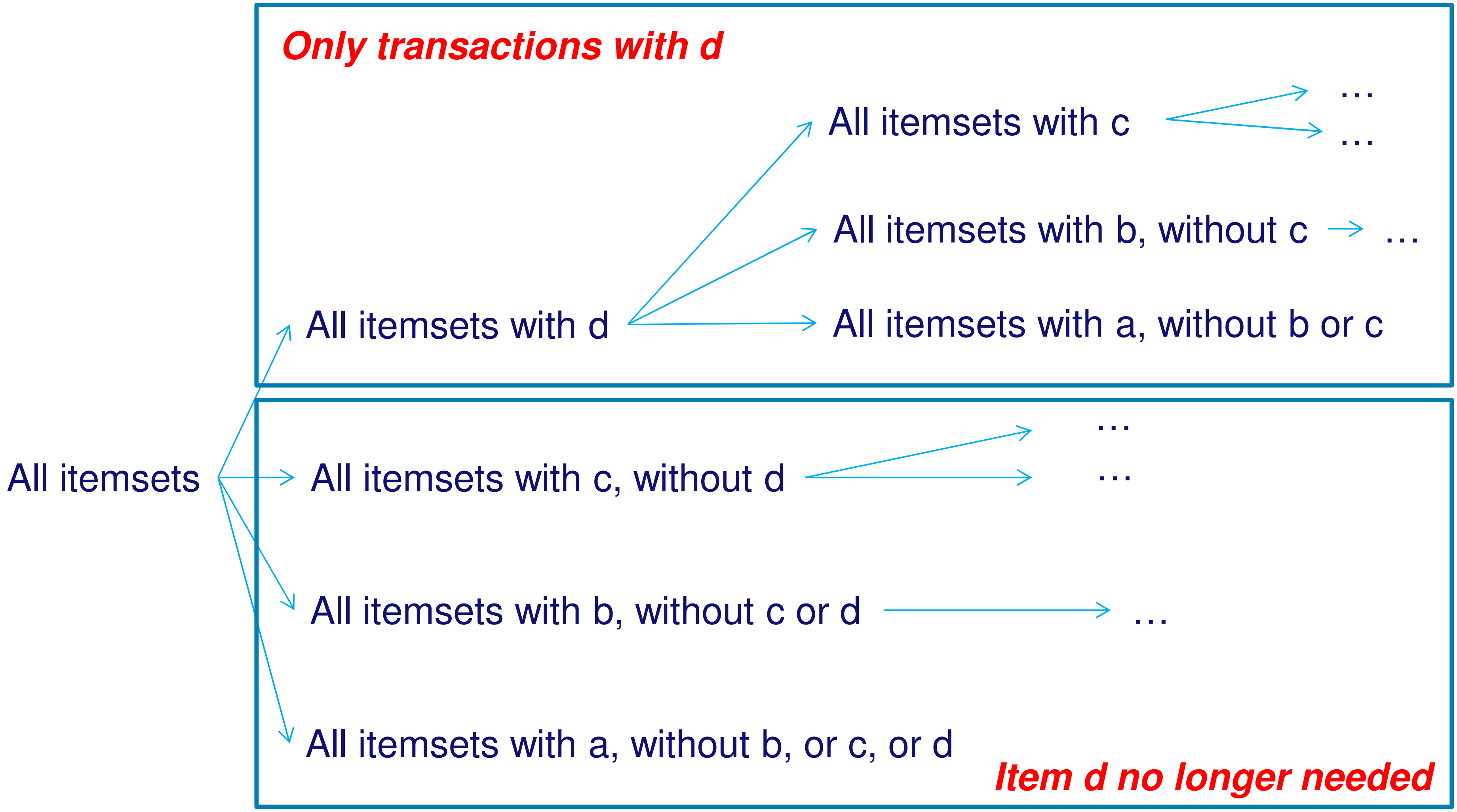
…

…

All itemsets with b, without c or d → …

All itemsets with a, without b, or c, or d

Divide-and-Conquer



Only transcriptions with d

All itemsets

All itemsets with d

All itemsets with c

…

…

All itemsets with b, without c

…

All itemsets with a, without b or c

All itemsets with c, without d

…

…

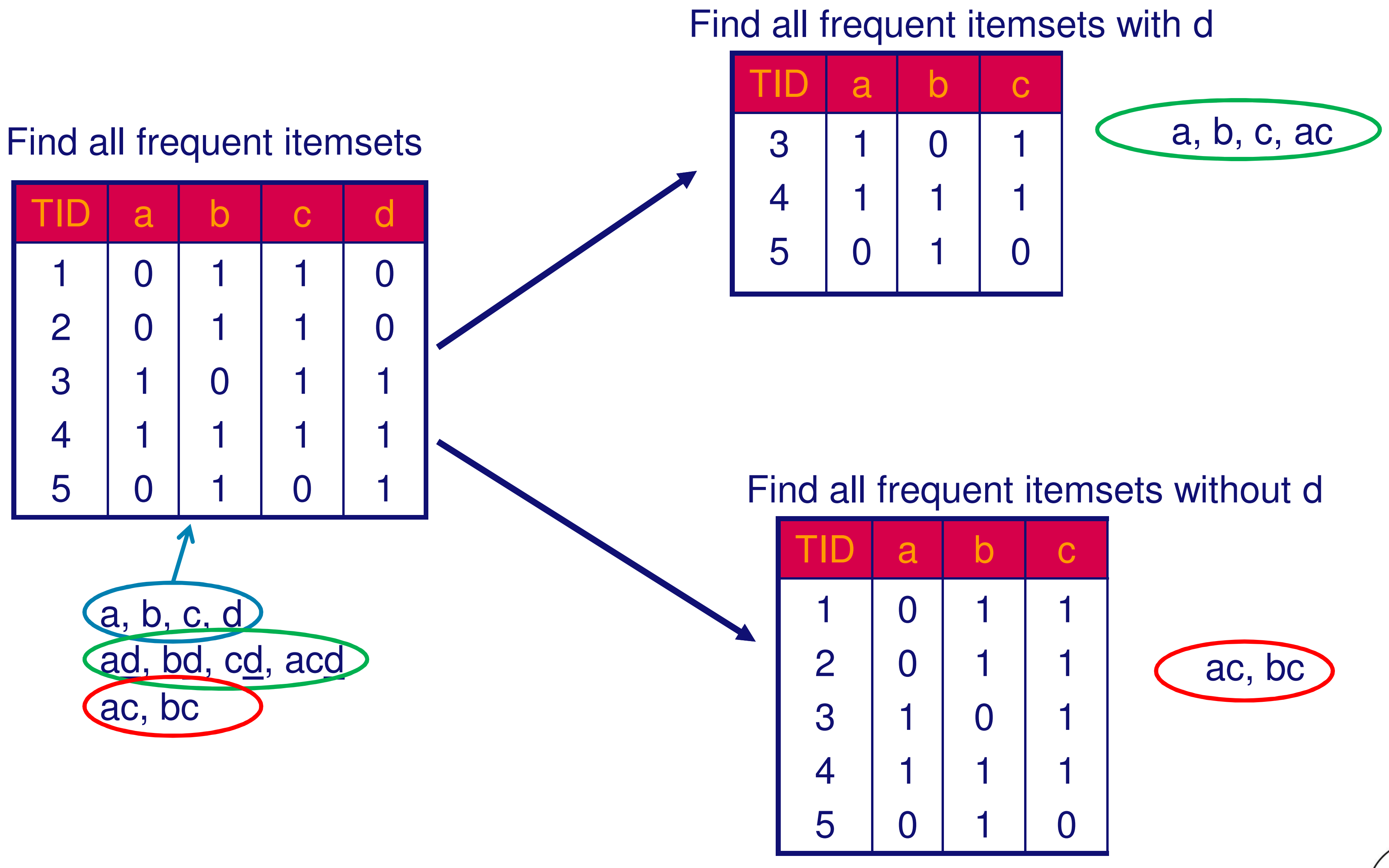
All itemsets with b, without c or d

…

All itemsets with a, without b, or c, or d

Item d no longer needed

Divide-and-Conquer



Find all frequent itemsets

TID	a	b	c	d
1	0	1	1	0
2	0	1	1	0
3	1	0	1	1
4	1	1	1	1
5	0	1	0	1

a, b, c, d

ad, bd, cd, acd

ac, bc

Find all frequent itemsets with d

TID	a	b	c
3	1	0	1
4	1	1	1
5	0	1	0

a, b, c, ac

Find all frequent itemsets without d

TID	a	b	c
1	0	1	1
2	0	1	1
3	1	0	1
4	1	1	1
5	0	1	0

ac, bc

Skip FPGrowth

Divide-and-Conquer

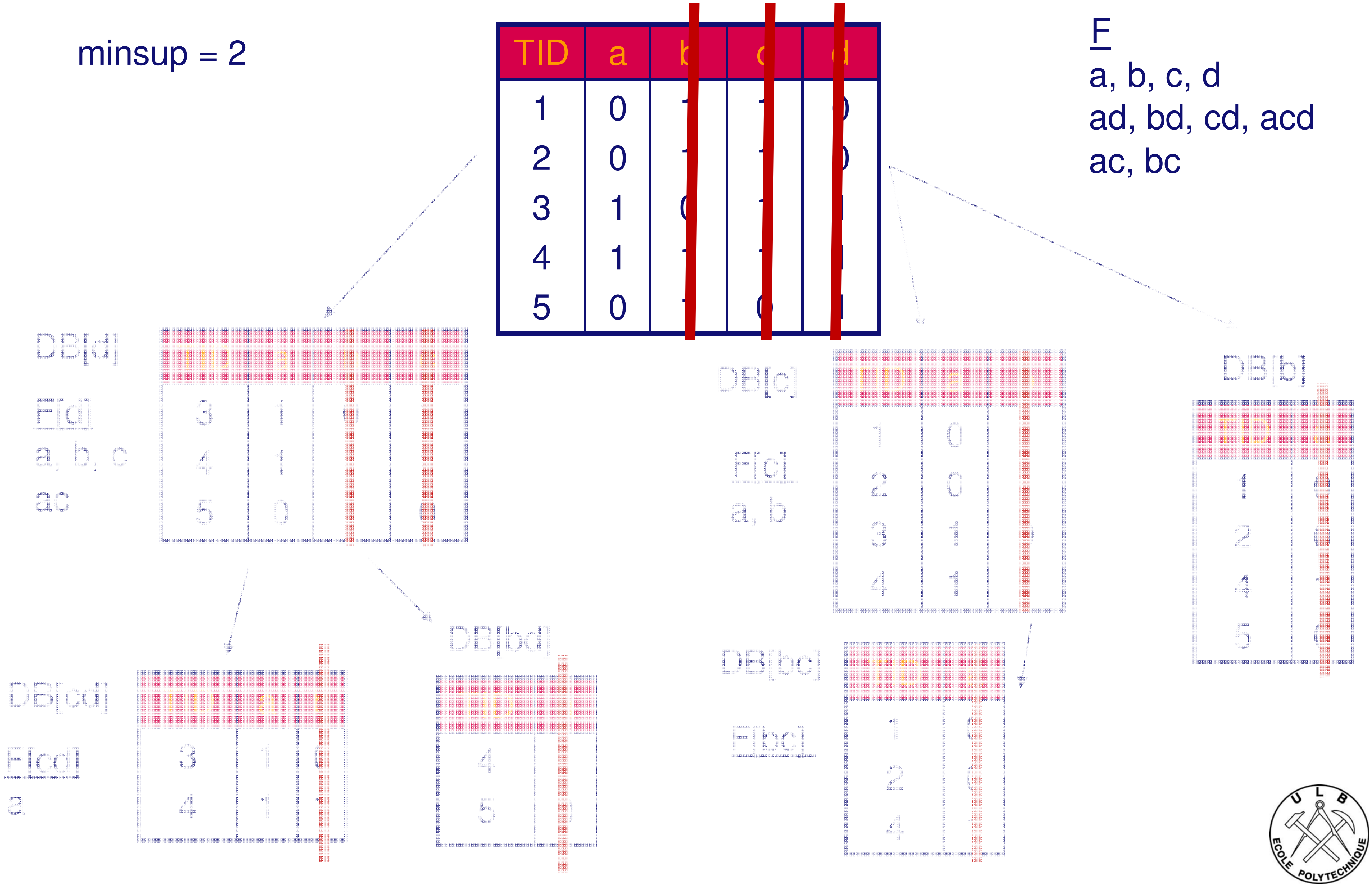
```
MineFrequent(DB)

1. $F := \{ a \mid a \text{ is frequent in } DB \}$
2. If $|F|<2$ return F                                ← base case
3. Remove infrequent items from DB
4. For every frequent item a, except the last:
   a. $DB[a] = \{ (tid, T \setminus \{a\}) \mid (tid, T) \in DB, a \text{ in } T \}$
   b. $F[a] := MineFrequent(DB[a])$                   ← recursion
   c. $F := F \cup \{ I \cup \{a\} \mid I \in F[a] \}$
   d. Remove a from DB
5. return F
```

Often skipped; a. then becomes:
$DB[a] = \{ (tid, T \cap O) \mid (tid, T) \in DB, a \text{ in } T \}$,
where O is the set of items not yet processed

Depth-First Algorithms

minsup = 2



TID	a	b	c	d
1	0	1	1	0
2	0	1	1	0
3	1	0	1	1
4	1	1	1	1
5	0	1	0	1

F
a, b, c, d
ad, bd, cd, acd
ac, bc

DB[d]

TID	a	b	c
3	1	0	1
4	1		
5	0		0

F[d]
a, b, c
ac

DB[cd]

TID	a	b
3	1	0
4	1	1

F[cd]
a

DB[bd]

TID	a
4	
5	0

DB[c]

TID	a	b
1	0	
2	0	
3	1	0
4	1	

F[c]
a, b

DB[bc]

TID	a
1	0
2	0
4	1

F[bc]

DB[b]

TID	a
1	0
2	0
4	1
5	0

Depth-First Algorithms

- Main difference between different algorithms:
  - Way to represent the database
    - Trie; tid-lists; …
- Database representation should allow for:
  - Selecting transaction containing a specific item
  - Building the conditional databases
- Most depth-first algorithms rely on an in-memory data structure
  - Random access important

Skip FPGrowth

FP-growth Algorithm

- Use a compressed representation of the database using an FP-tree

- Once an FP-tree has been constructed, it uses a recursive divide-and-conquer approach to mine the frequent itemsets

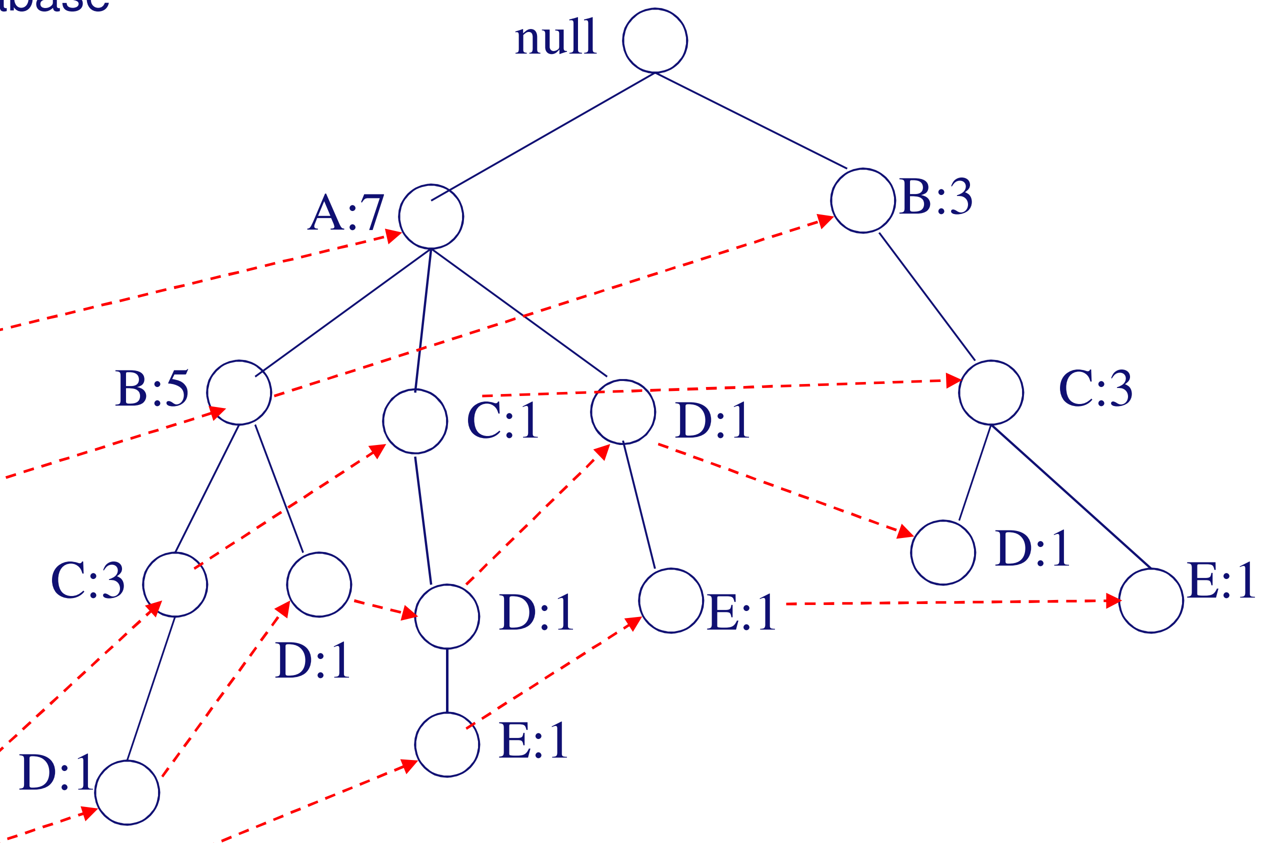
FP-Tree

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

Transaction Database

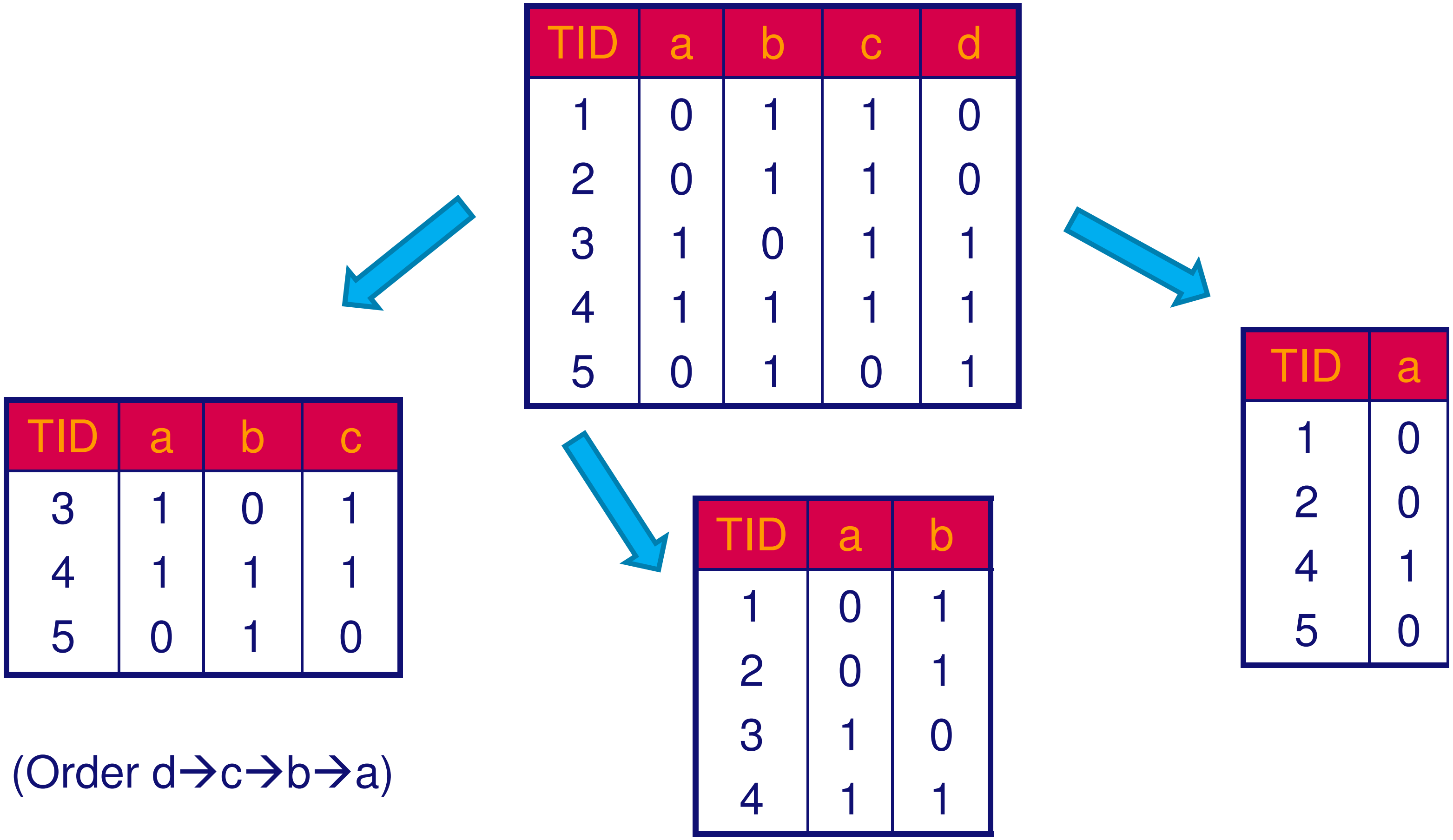
Header table

Item	Pointer
A	
B	
C	
D	
E	



Depth-First Algorithms

- **Building the conditional database:**
  - **DB[a] = { (tid,T∩O) | (tid, T) ∈ DB, a in T };**
    O is the set of items not yet processed



TID	a	b	c	d
1	0	1	1	0
2	0	1	1	0
3	1	0	1	1
4	1	1	1	1
5	0	1	0	1

TID	a	b	c
3	1	0	1
4	1	1	1
5	0	1	0

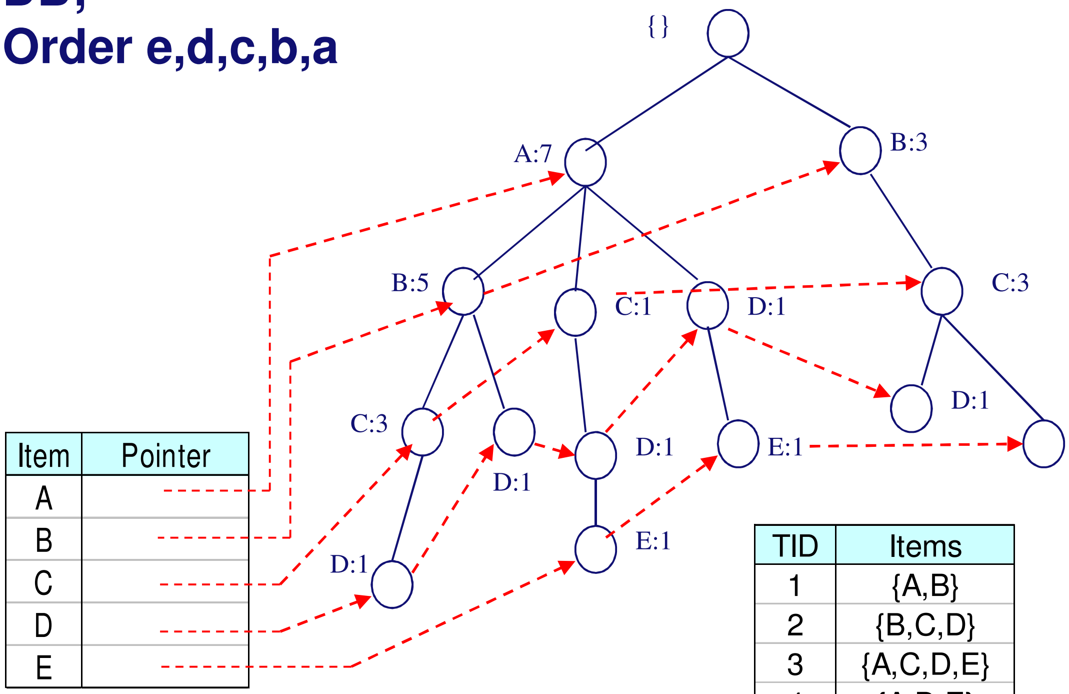
TID	a	b
1	0	1
2	0	1
3	1	0
4	1	1

TID	a
1	0
2	0
4	1
5	0

(Order d→c→b→a)

FP-Tree Operations: Example

DB;
Order e,d,c,b,a



Item	Pointer
A	
B	
C	
D	
E	

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

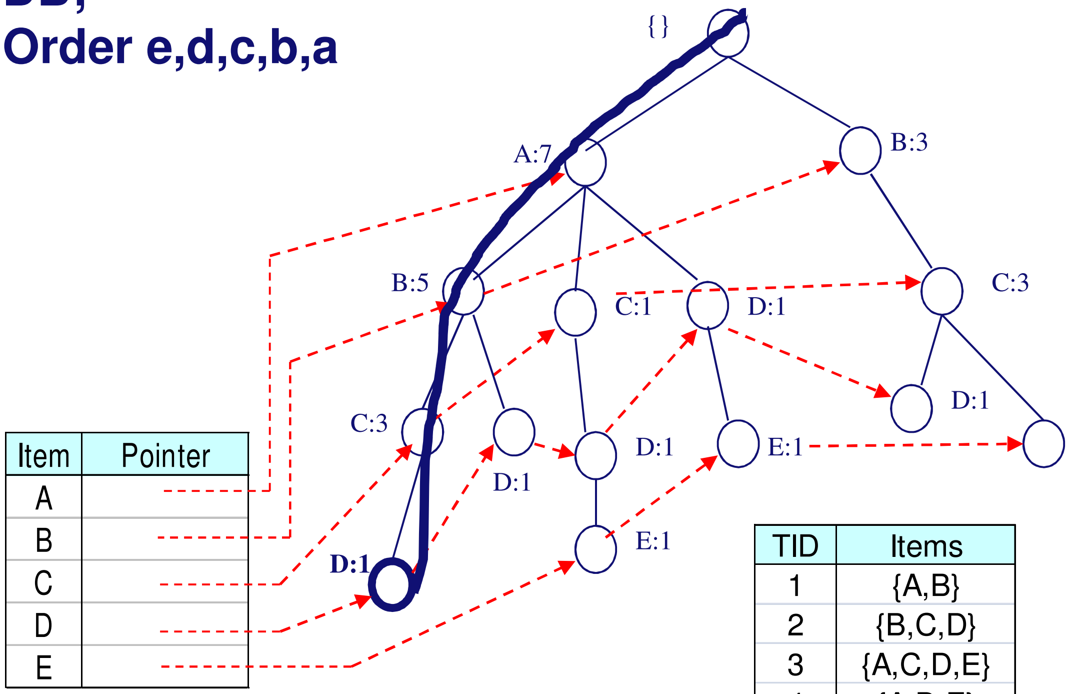
DB[d]

How to create the FPTree of DB[d] from the FPTree of DB?

TID	Items
2	{B,C}
3	{A,C}
4	{A}
6	{A,B,C}
9	{A,B}

FP-Tree Operations

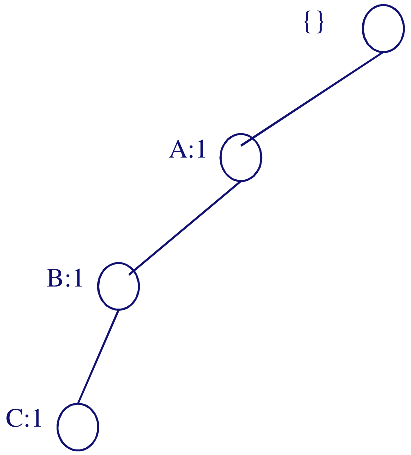
DB;
Order e,d,c,b,a



Item	Pointer
A	
B	
C	
D	
E	

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

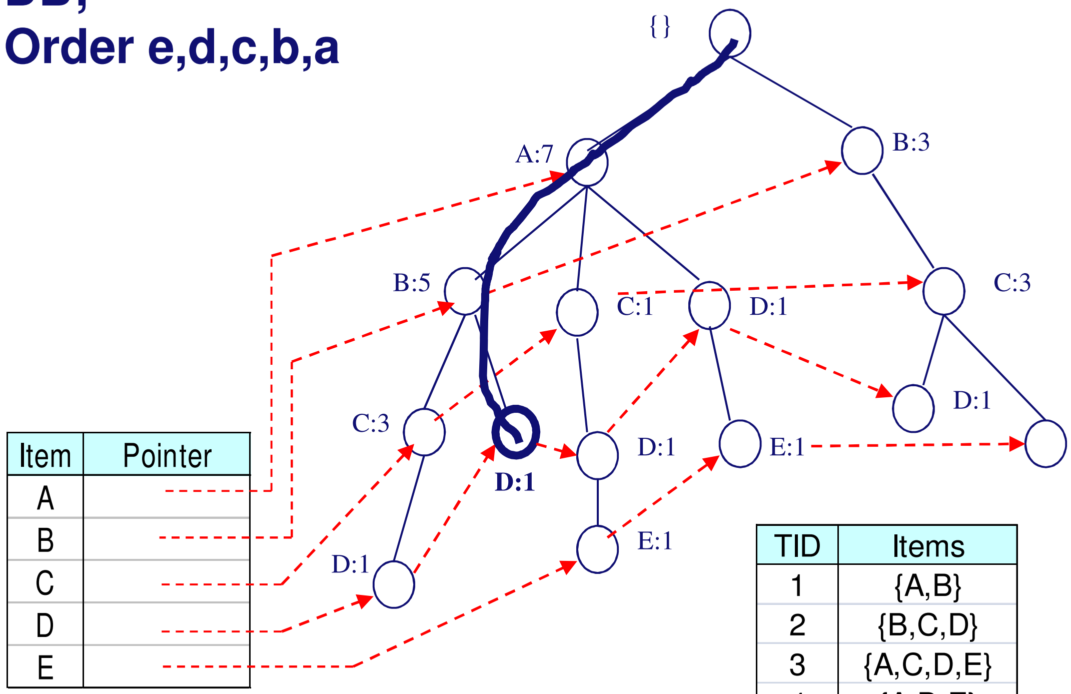
DB[d]



TID	Items
2	{B,C}
3	{A,C}
4	{A}
6	{A,B,C}
9	{A,B}

FP-Tree Operations

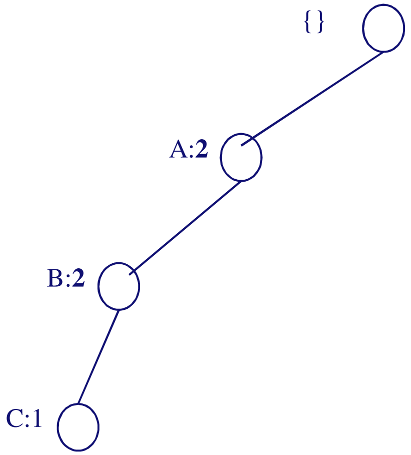
DB;
Order e,d,c,b,a



Item	Pointer
A	
B	
C	
D	
E	

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

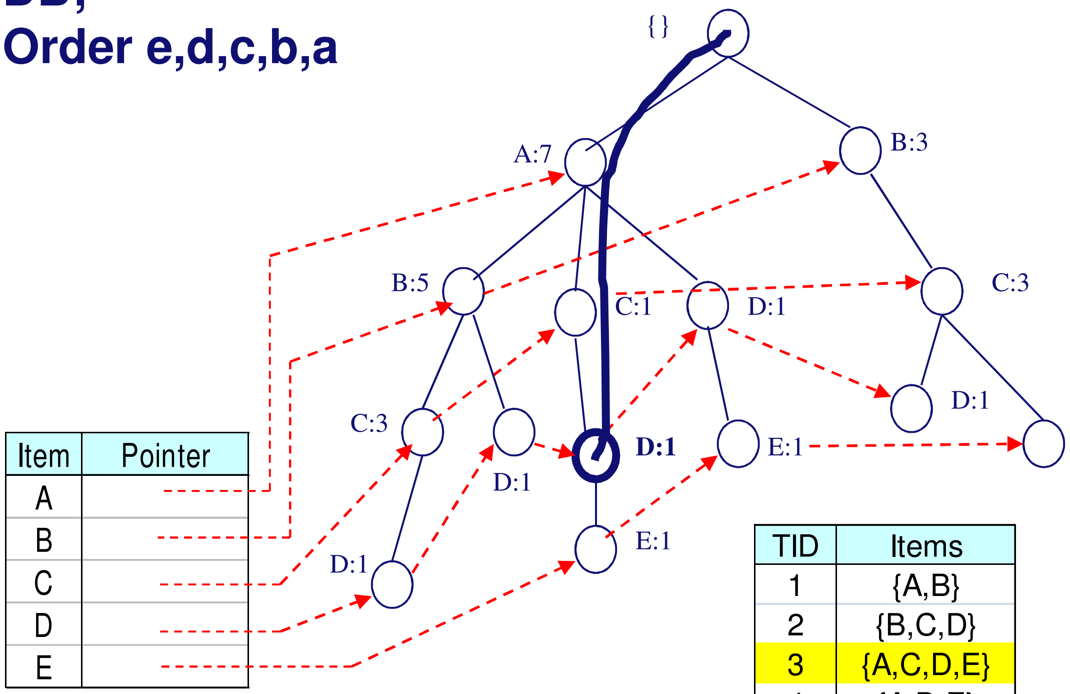
DB[d]



TID	Items
2	{B,C}
3	{A,C}
4	{A}
6	{A,B,C}
9	{A,B}

FP-Tree Operations

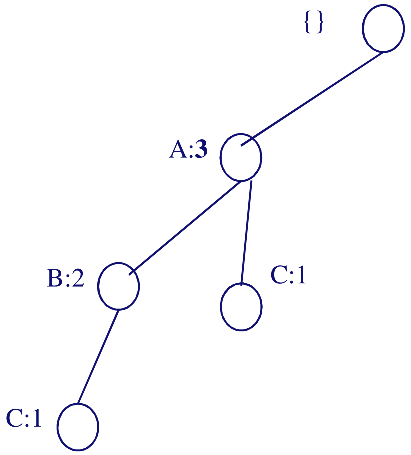
DB;
Order e,d,c,b,a



Item	Pointer
A	
B	
C	
D	
E	

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

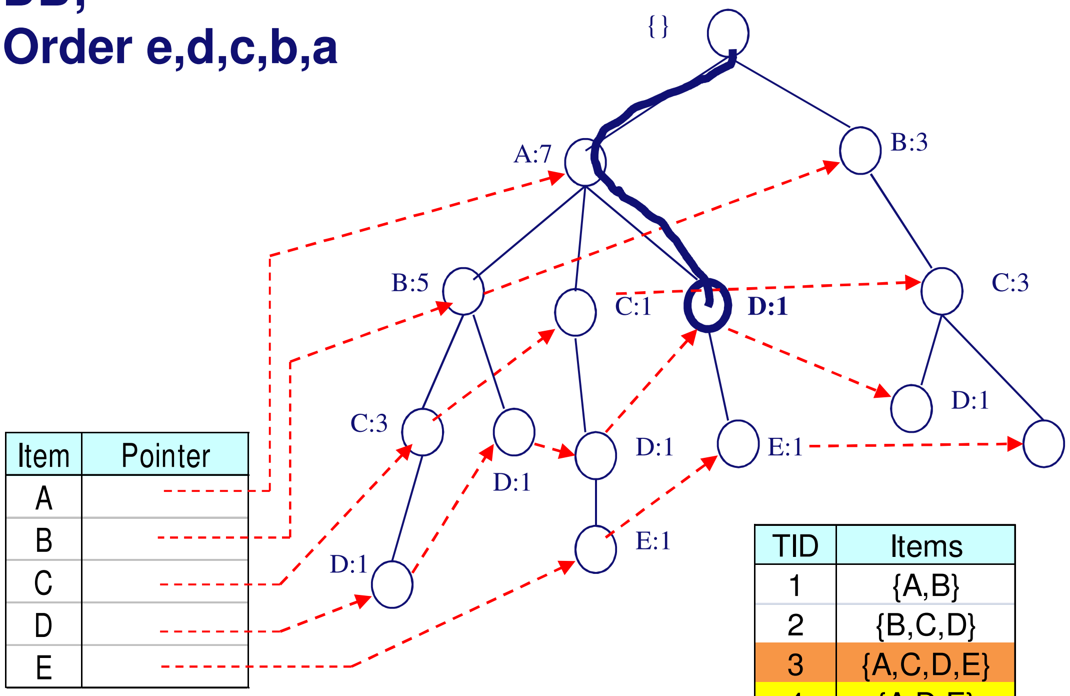
DB[d]



TID	Items
2	{B,C}
3	{A,C}
4	{A}
6	{A,B,C}
9	{A,B}

FP-Tree Operations

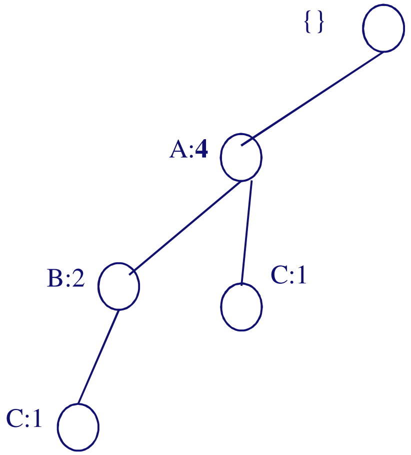
DB;
Order e,d,c,b,a



Item	Pointer
A	
B	
C	
D	
E	

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

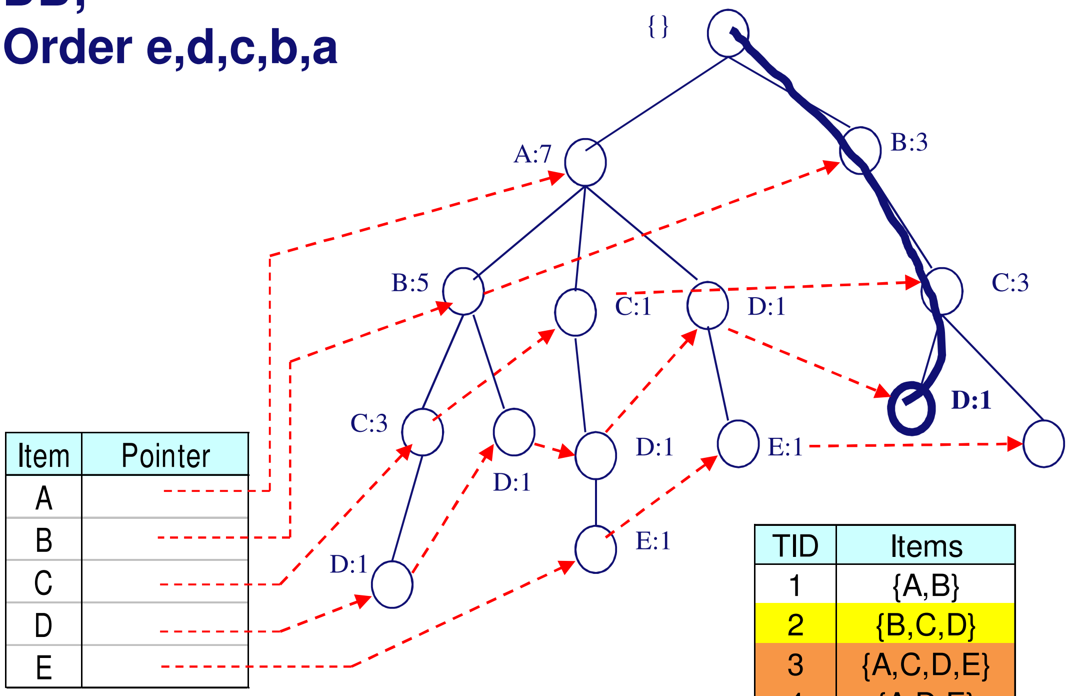
DB[d]



TID	Items
2	{B,C}
3	{A,C}
4	{A}
6	{A,B,C}
9	{A,B}

FP-Tree Operations

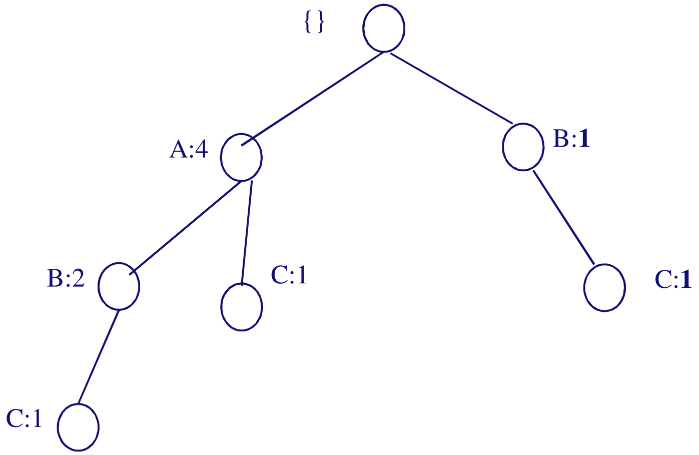
DB;
Order e,d,c,b,a



Item	Pointer
A	
B	
C	
D	
E	

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

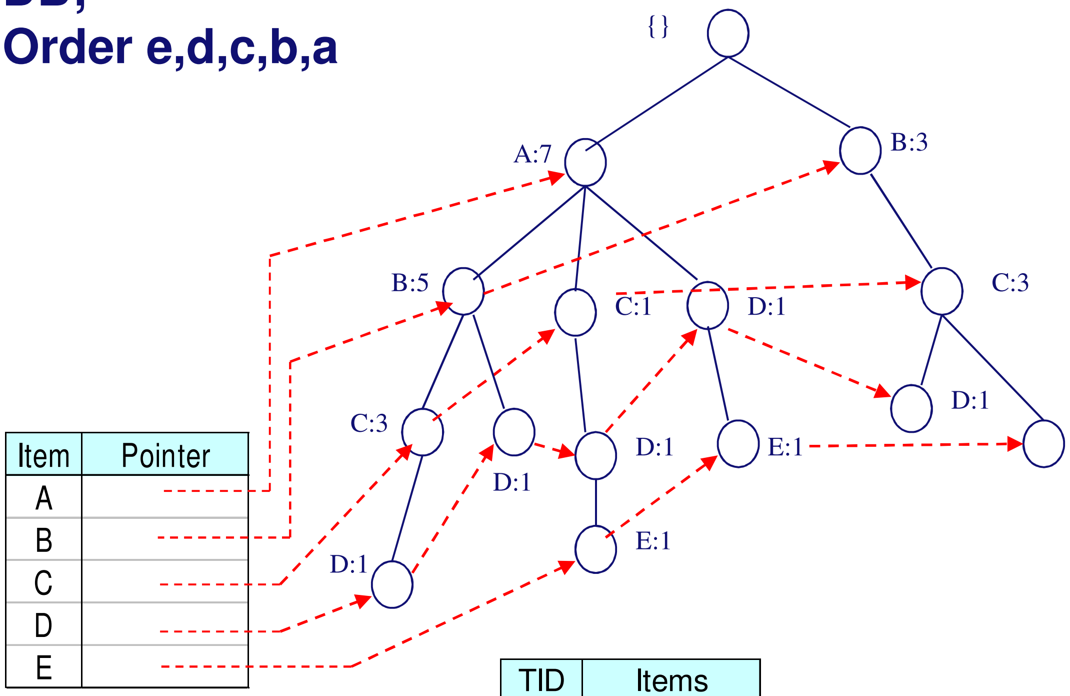
DB[d]



TID	Items
2	{B,C}
3	{A,C}
4	{A}
6	{A,B,C}
9	{A,B}

FP-Tree Operations

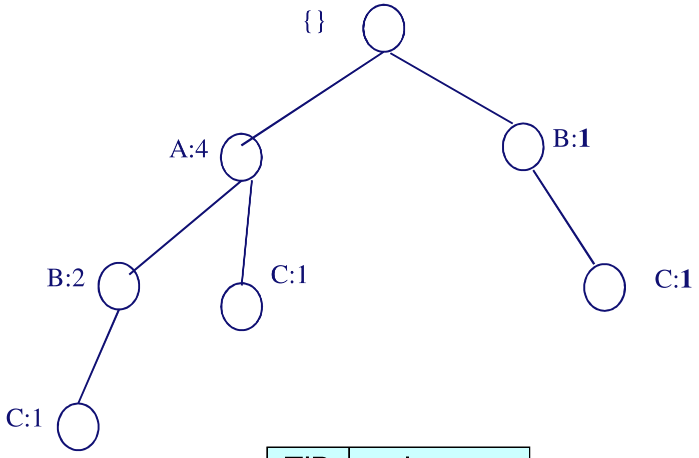
DB;
Order e,d,c,b,a



Item	Pointer
A	
B	
C	
D	
E	

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

DB[d]



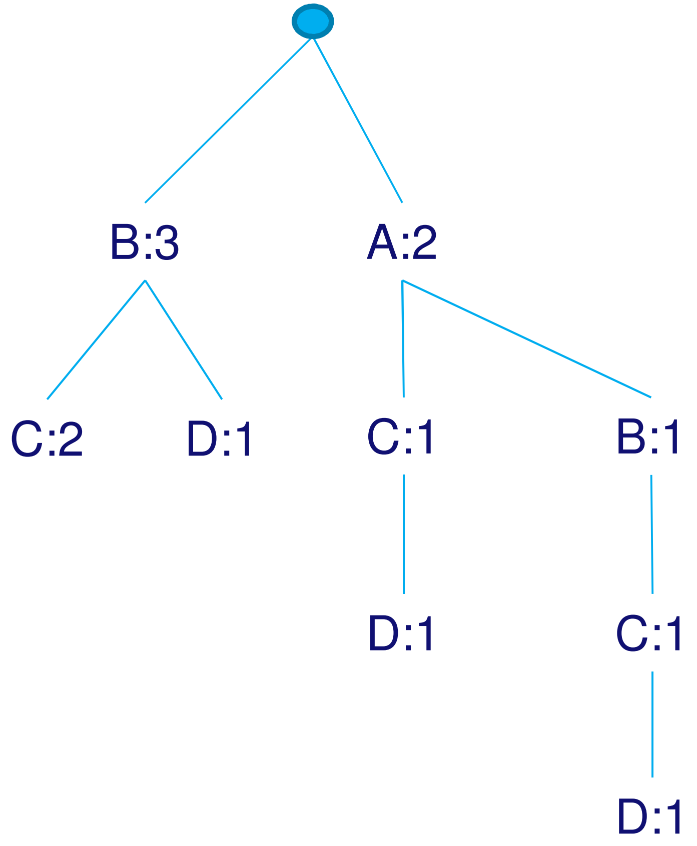
TID	Items
2	{B,C}
3	{A,C}
4	{A}
6	{A,B,C}
9	{A,B}

FPGrowth – Complete Example

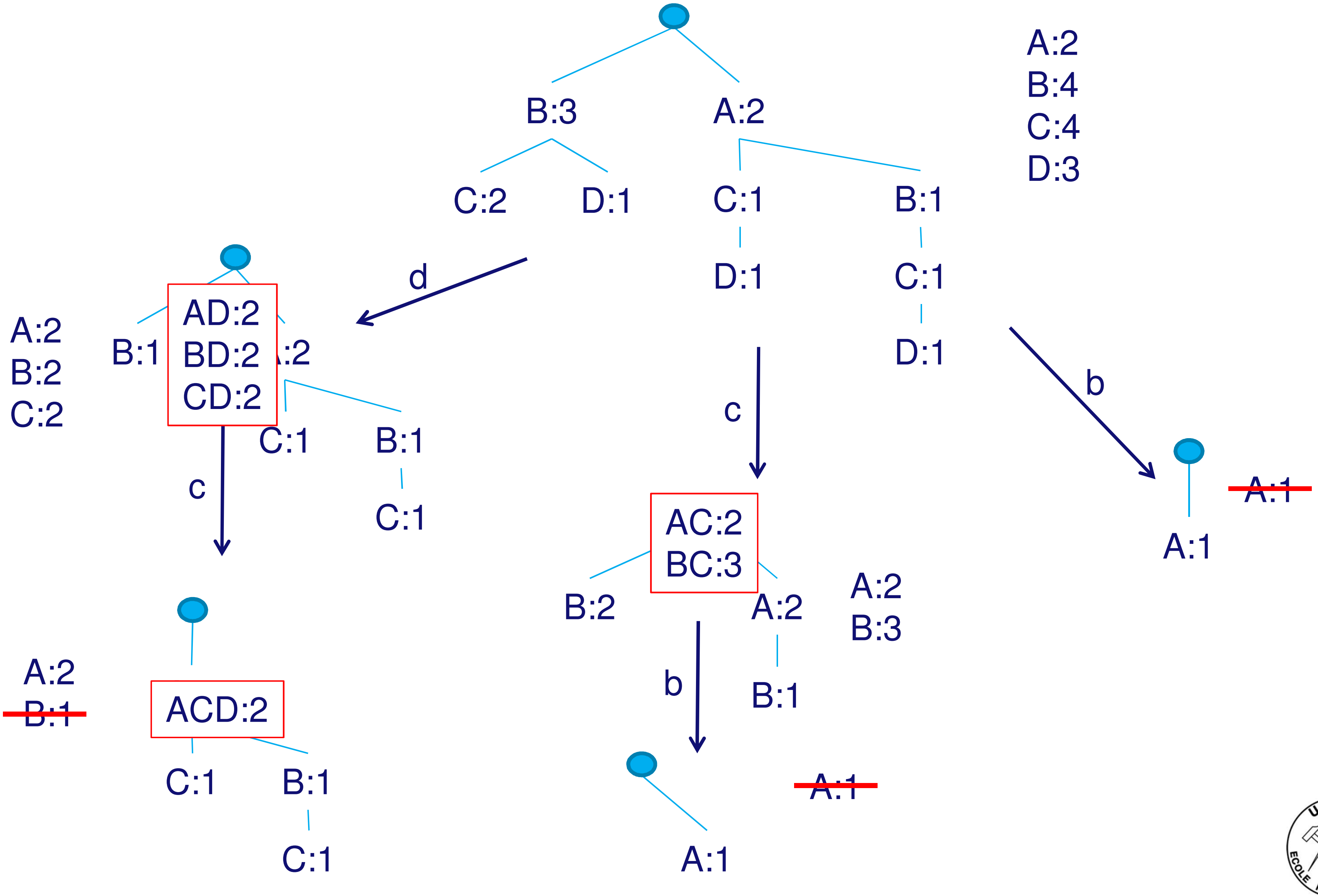
- **Step 1: create Initial FPTree**

minsup = 2

TID	a	b	c	d
1	0	1	1	0
2	0	1	1	0
3	1	0	1	1
4	1	1	1	1
5	0	1	0	1



FPGrowth – Complete Example



FPGrowth Summary

- Depth-first algorithm
  - Divide-and-conquer strategy
  - + More efficient counting
    - Reduce database in every step
  - - Not fully exploiting monotonicity
- FPTree data structure
  - + Allows for quickly projecting the database
  - - Kept in-memory
- Overall: if database fits in memory, depth-first algorithms rule

Frequent Itemset Mining: Summary

- Useful for exploration, feature selection, association discovery

- Many efficient algorithms exist
  - Monotonicity principle central property in all algorithms
  - General-to-specific exploration of the search space
  - Breadth-first algorithm: Apriori
  - Depth-first algorithm: FPGrowth

Outline

PART I: Frequent itemset mining

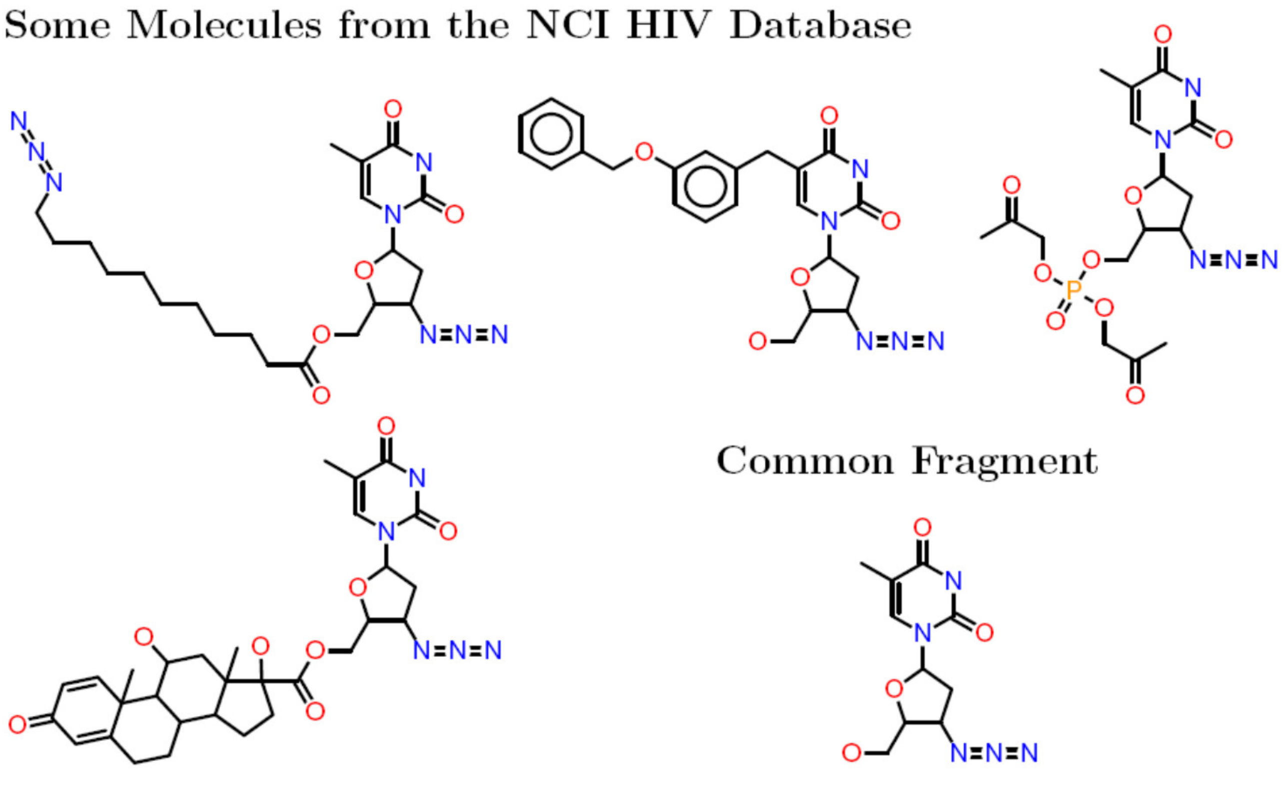
- Definition & Applications
- Algorithms for Frequent Itemset Mining
- Extensions to other pattern types

PART II:

- Pattern explosion & Redundancy problem
- Methods to remove redundancy
  - Condensed representations
  - Statistical methods
  - Minimal Description Length

Other Types of Patterns

- **Sequences**
- **Graphs**
- **Dynamic graphs**



Some Molecules from the NCI HIV Database

Common Fragment

Other Types of Patterns

- Sequences
  - Mining sequences of alarms
- Graphs
  - Finding common structures
    - Socially relevant
- Dynamic graphs
  - How do social graphs grow?
  - Patterns explaining growth over time.

Other Types of Patterns

- Sequences
- Graphs
- Dynamic graphs

- Breadth-first algorithms usually no longer work for more complex pattern types:
  - N^K sequences of size K with N items
  - $N^K 2^{K*(K-1)}$ directed graphs with N labels and K nodes
- Cannot hold this many patterns in memory
  - Monotonicity check requires random access
- Therefore: most algorithms are depth-first

Other Types of Patterns

- Sequences
- Graphs
- Dynamic graphs

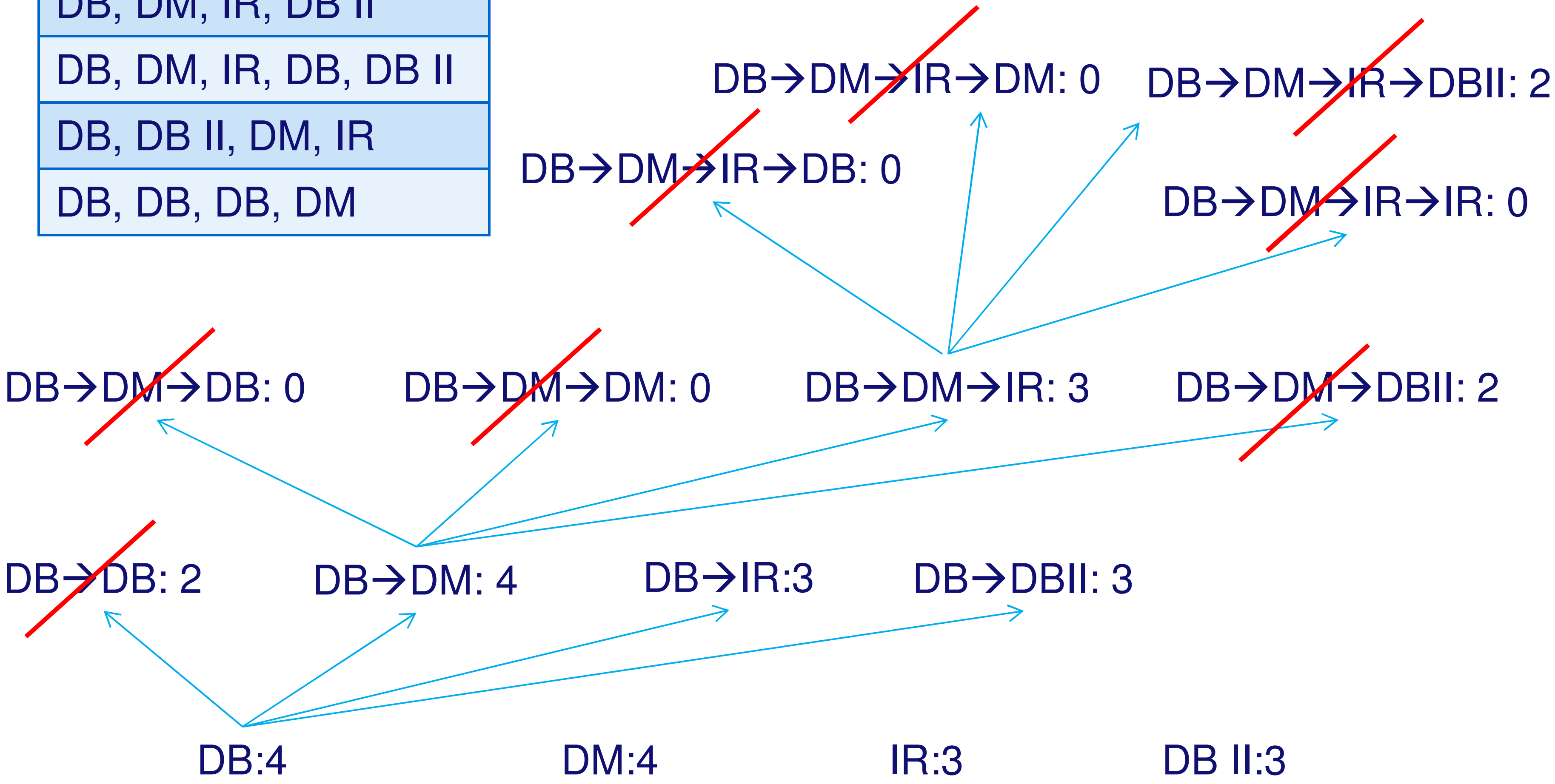
```
Generate(P)
    If supp(P) $\geq$ minsup :
        Write P to output
        Successors = extend(P)
    For c in Successors:
        Generate(C)
```

Sequence Mining

- Input data: database of sequences
  - Sequence of alarms in an event log
  - Order in which students followed courses
  - Text = sequence of words
- Two settings:
  - One large string
  - Database of strings
- Algorithms are very similar as for frequent itemset mining

Sequence Mining: Example

sequences
DB, DM, IR, DB II
DB, DM, IR, DB, DB II
DB, DB II, DM, IR
DB, DB, DB, DM



~~DB→DM→IR→DM: 0~~

~~DB→DM→IR→DBII: 2~~

~~DB→DM→IR→DB: 0~~

~~DB→DM→IR→IR: 0~~

~~DB→DM→DB: 0~~

~~DB→DM→DM: 0~~

DB→DM→IR: 3

~~DB→DM→DBII: 2~~

~~DB→DB: 2~~

DB→DM: 4

DB→IR:3

DB→DBII: 3

DB:4

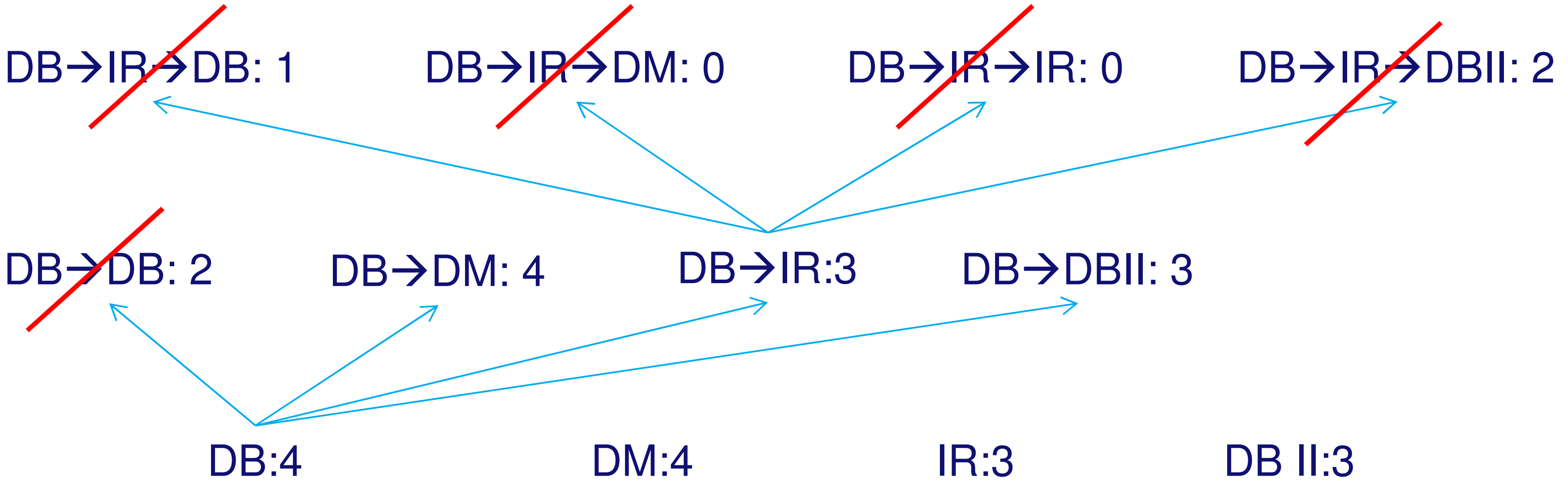
DM:4

IR:3

DB II:3

Sequence Mining: Example

sequences
DB, DM, IR, DB II
DB, DM, IR, DB, DB II
DB, DB II, DM, IR
DB, DB, DB, DM



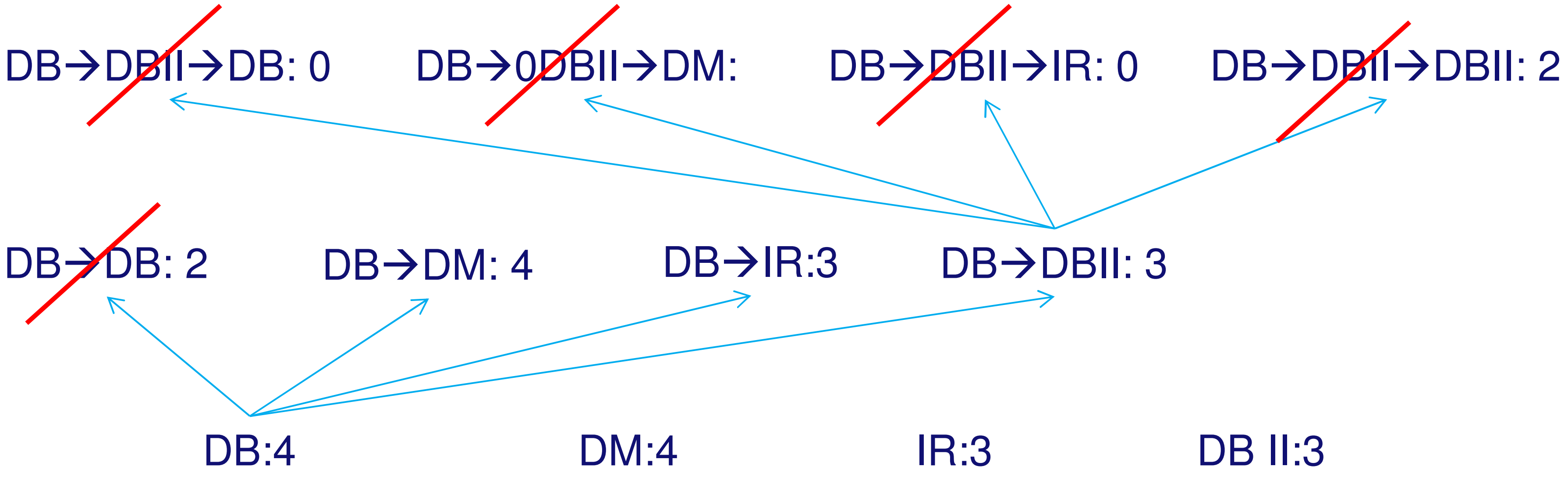
~~DB→IR→DB: 1~~ ~~DB→IR→DM: 0~~ ~~DB→IR→IR: 0~~ ~~DB→IR→DBII: 2~~

~~DB→DB: 2~~ DB→DM: 4 DB→IR:3 DB→DBII: 3

DB:4 DM:4 IR:3 DB II:3

Sequence Mining: Example

sequences
DB, DM, IR, DB II
DB, DM, IR, DB, DB II
DB, DB II, DM, IR
DB, DB, DB, DM



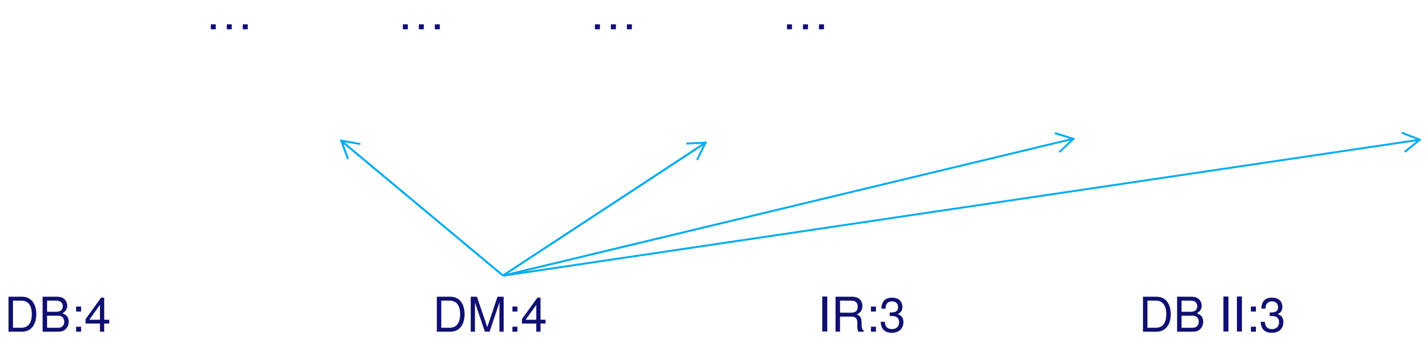
~~DB→DBII→DB: 0~~ ~~DB→0DBII→DM:~~ ~~DB→DBII→IR: 0~~ ~~DB→DBII→DBII: 2~~

~~DB→DB: 2~~ DB→DM: 4 DB→IR:3 DB→DBII: 3

DB:4 DM:4 IR:3 DB II:3

Sequence Mining: Example

sequences
DB, DM, IR, DB II
DB, DM, IR, DB, DB II
DB, DB II, DM, IR
DB, DB, DB, DM

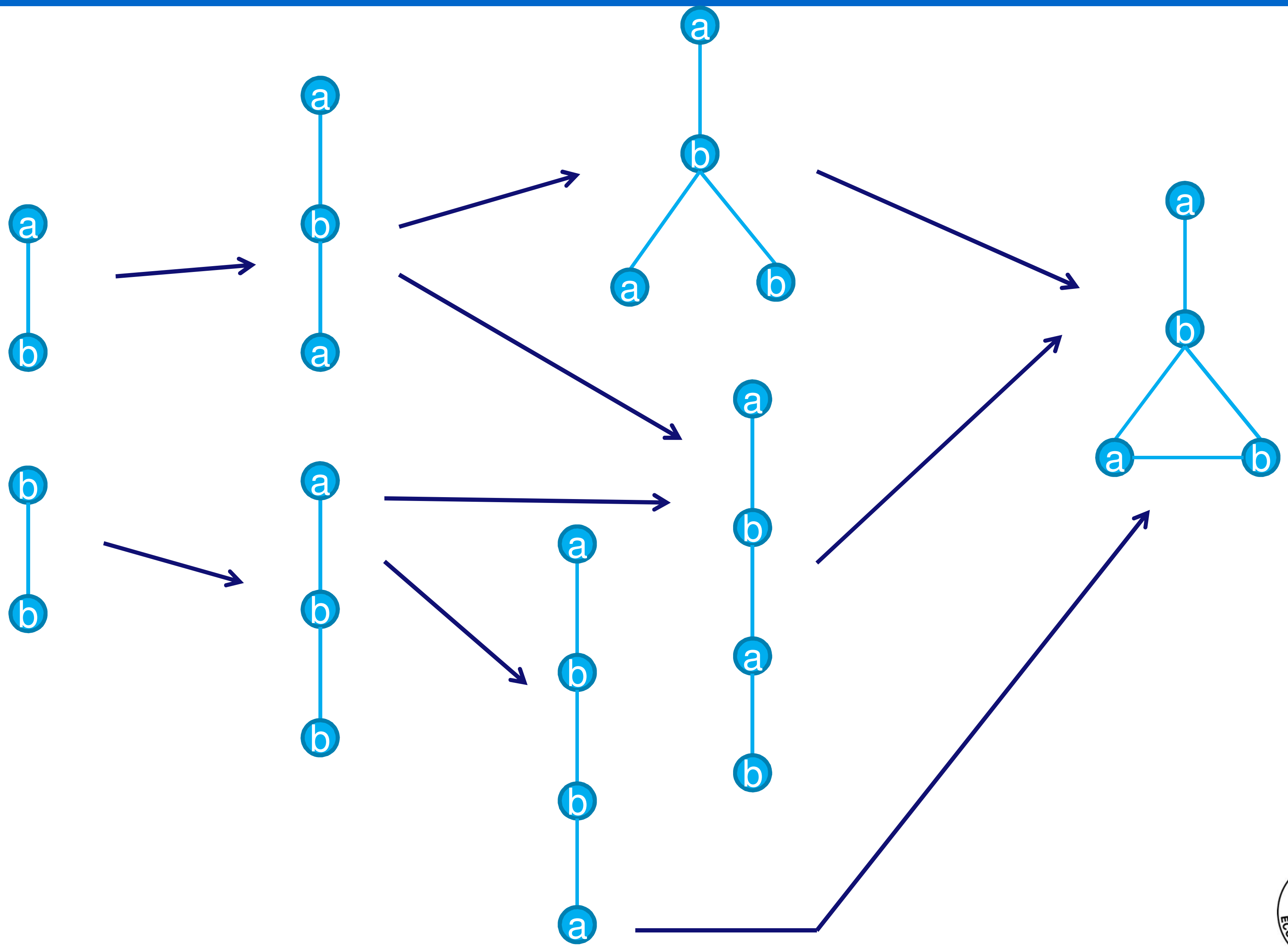


Other Types of Patterns

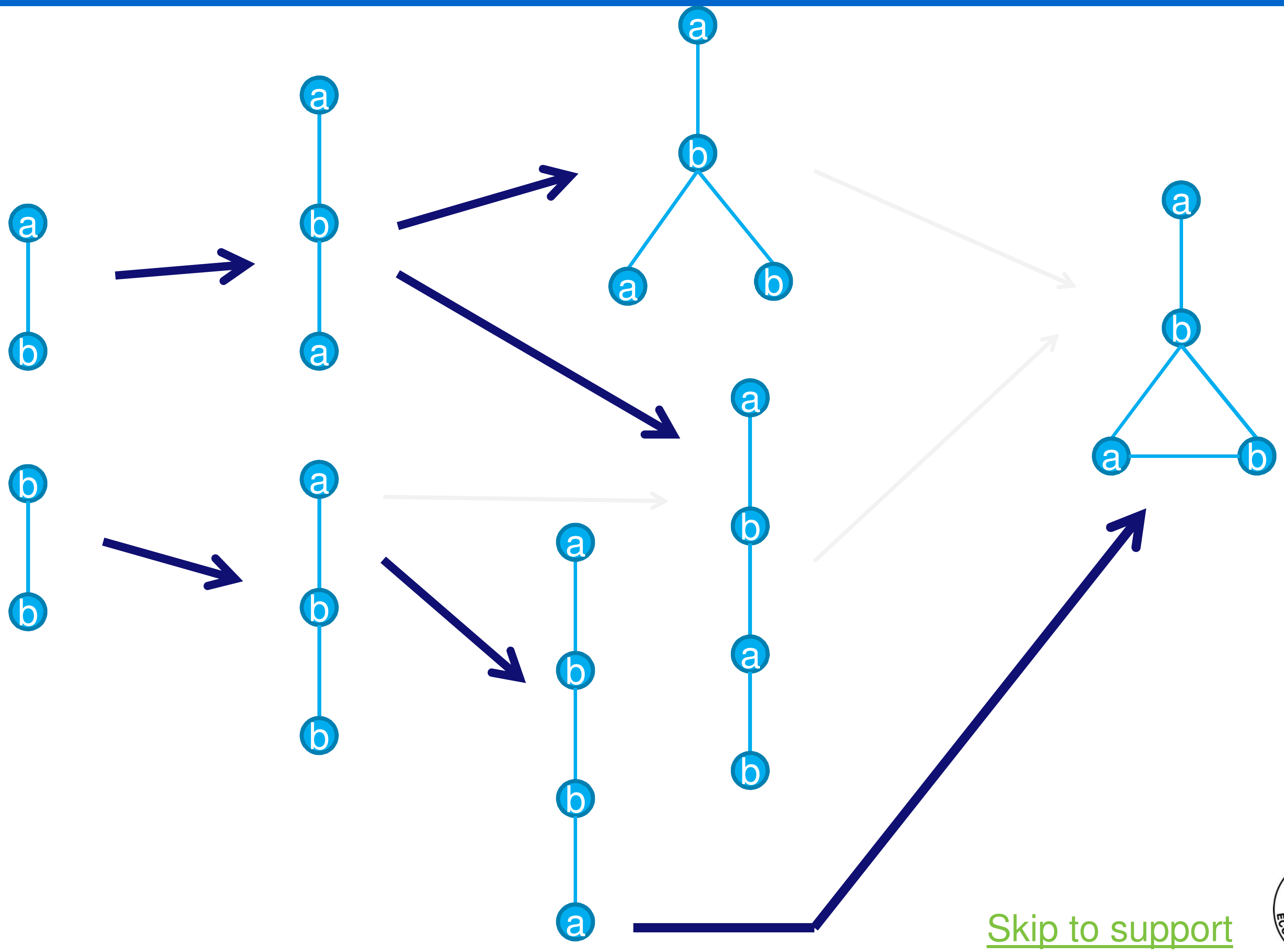
- Sequences
- Graphs
- Dynamic graphs

- Common problems:
  - How to generate all candidates without duplicates
  - How to count efficiently
  - Notion of "support" is not always straightforward
    - Must be anti-monotone and efficient to compute

Generate Graphs w.o. Duplicates



Generate Graphs w.o. Duplicates



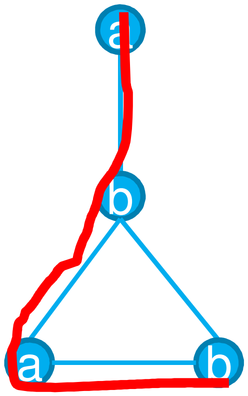
Skip to support

Generate Candidates w.o. Duplicates

- **Canonical representation**

  (1,2), (2,3), (3,4), (2,4) abab

  (1,2), (2,3), (2,4), (3,4) abab

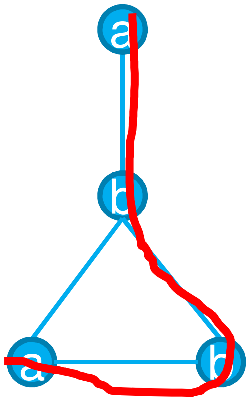


Generate Candidates w.o. Duplicates

- **Canonical representation**

  (1,2), (2,3), (3,4), (2,4) abab
  (1,2), (2,3), (2,4), (3,4) abab

  (1,2), (2,3), (3,4), (2,4) abba
  (1,2), (2,3), (2,4), (3,4) abba
  …



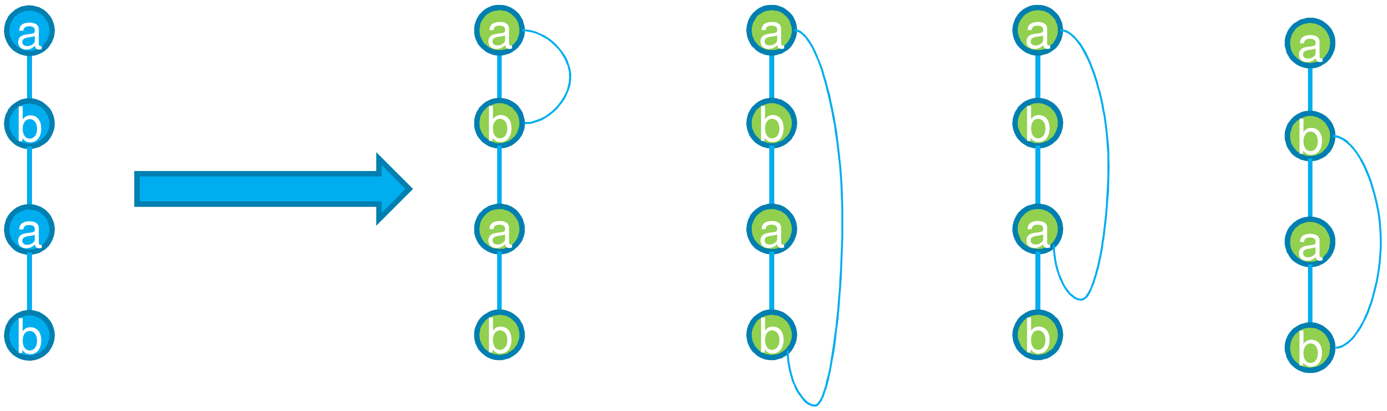
- **Canonical representation = lexicographically <u>first</u>**

0 1 0 0
0 0 1 1
0 0 0 1
0 0 0 0

0100001100010000 = 34308

Generate Candidates w.o. Duplicates

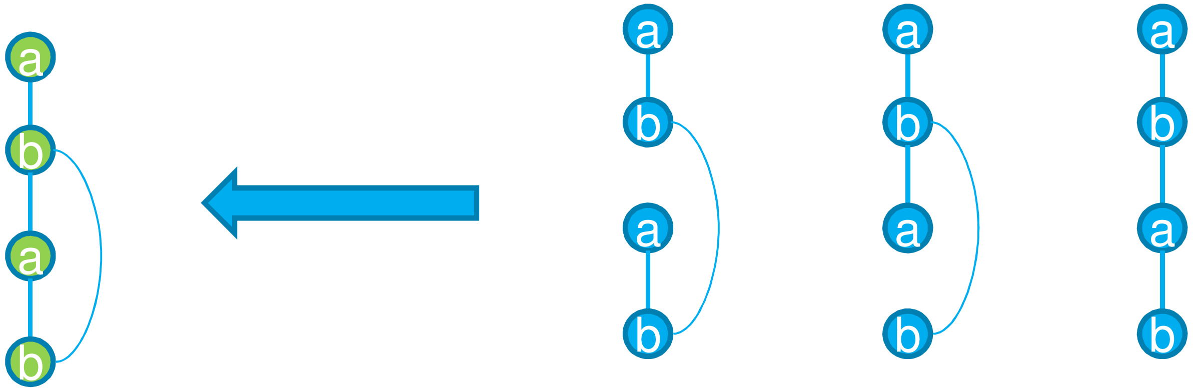
- **Generating successors:**
  - **Look at all direct successors of the pattern:**



Candidate successors

…

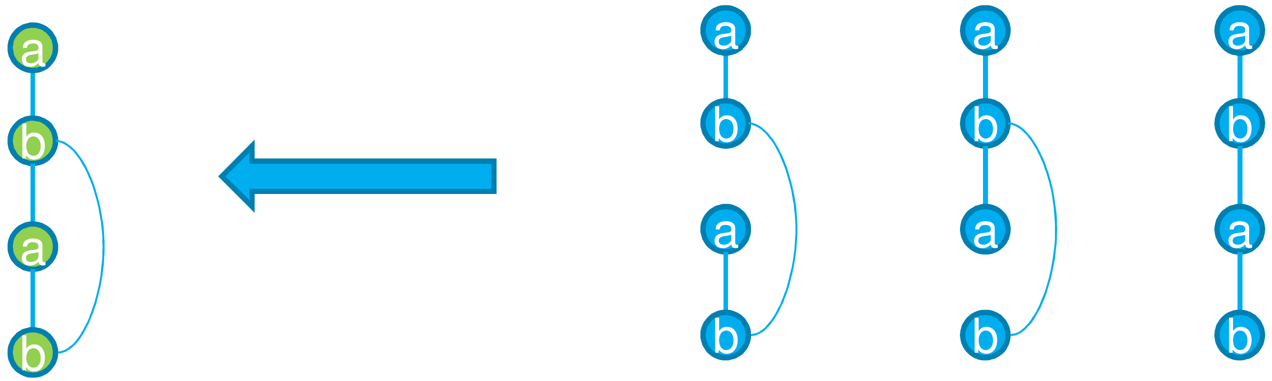
- **For all successors, look at all the predecessors that could have generated it**



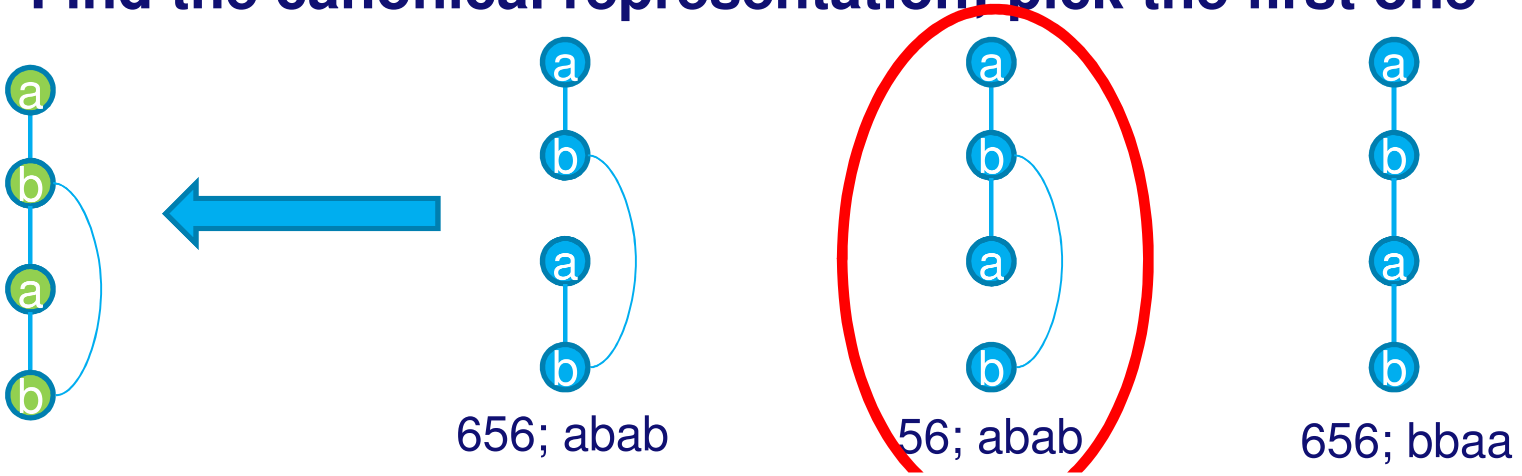
Candidate “real father”

Generate Candidates w.o. Duplicates

- **Generating successors:**
  - **For all successors, look at all the predecessors that could have generated it**

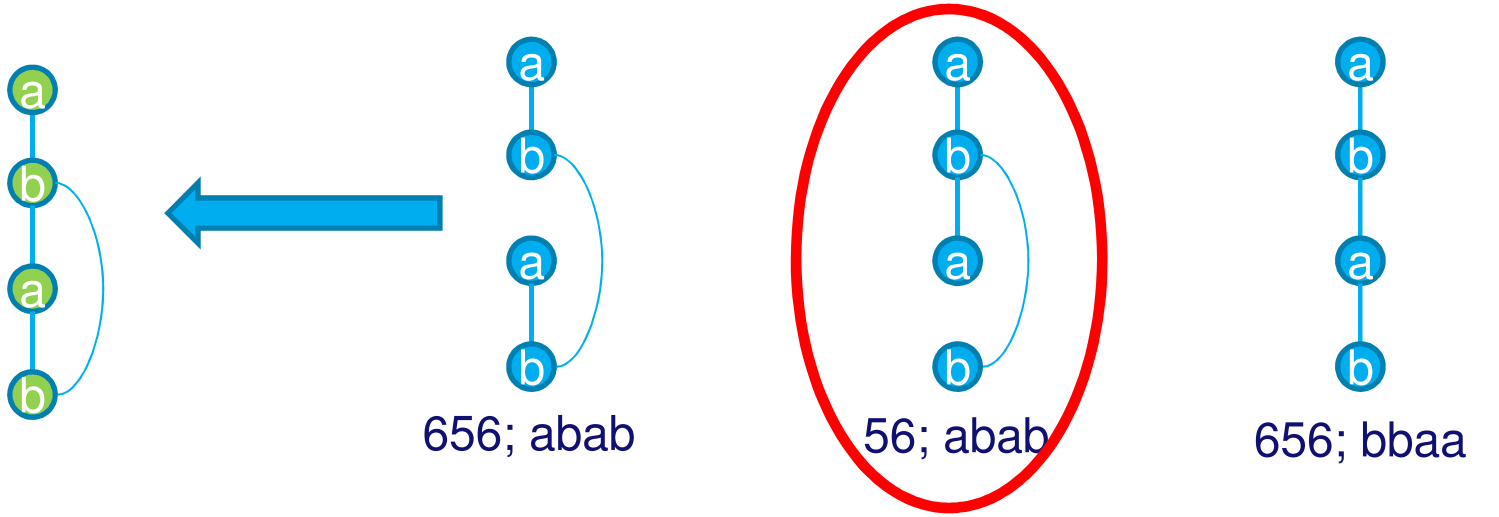


  - **Find the canonical representation; pick the first one**



Generate Candidates w.o. Duplicates

- **Generating successors:**
  - **Find the canonical representation; pick the first one**



  - **Only that pattern is allowed to generate the successor**
    - **Avoid the generation of duplicates while exploring of the search space depth-first**

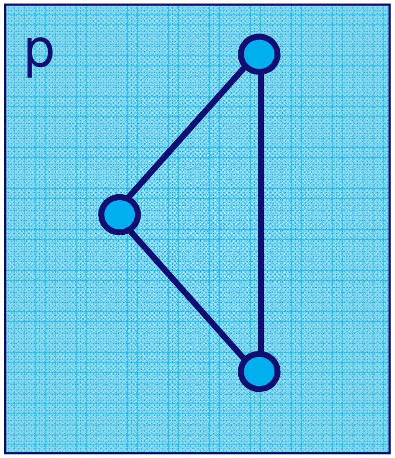
Other Types of Patterns

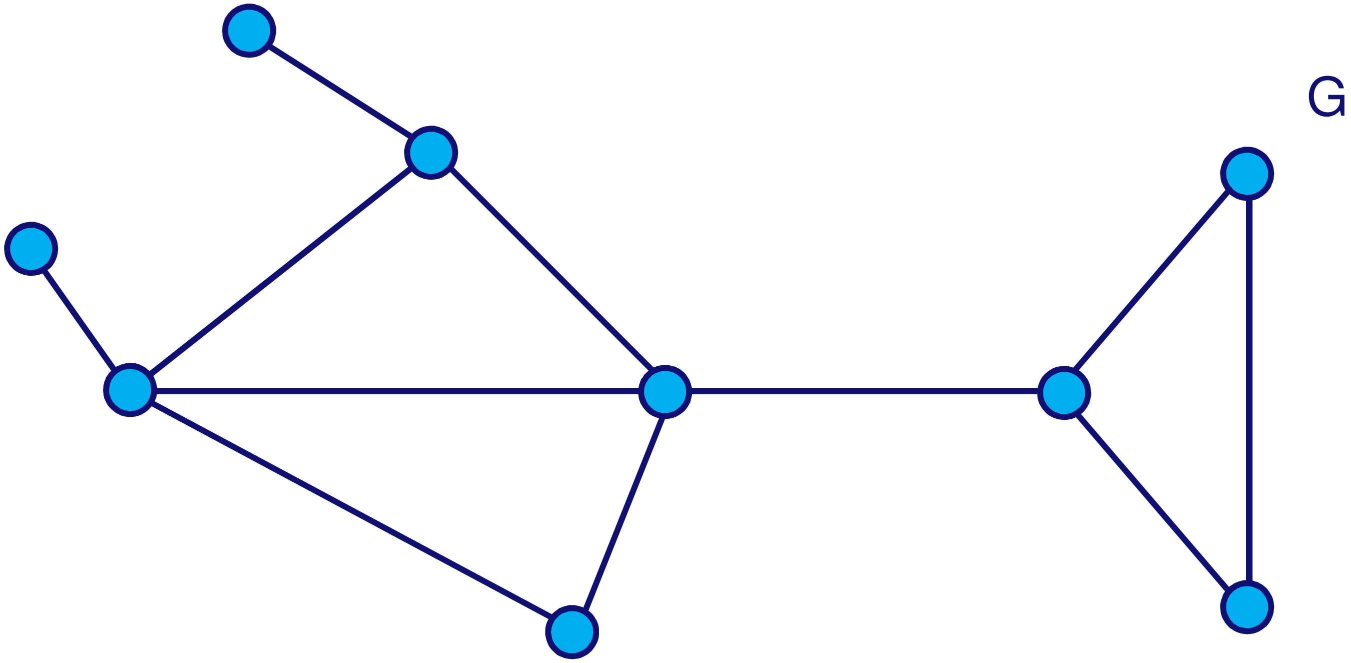
- Sequences
- Graphs
- Dynamic graphs

- Common problems:
  - How to generate all candidates without duplicates
  - How to count efficiently
  - Notion of “support” is not always straightforward
    - Must be anti-monotone and efficient to compute

Problems with Frequency

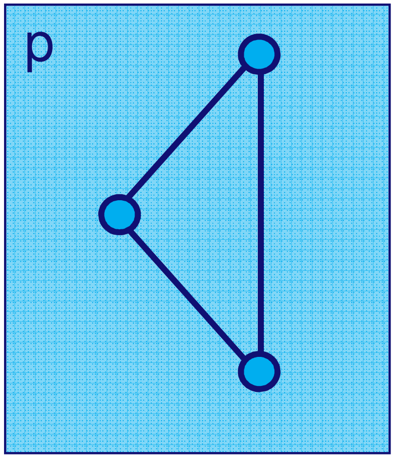
- **Counting instances does not work !**

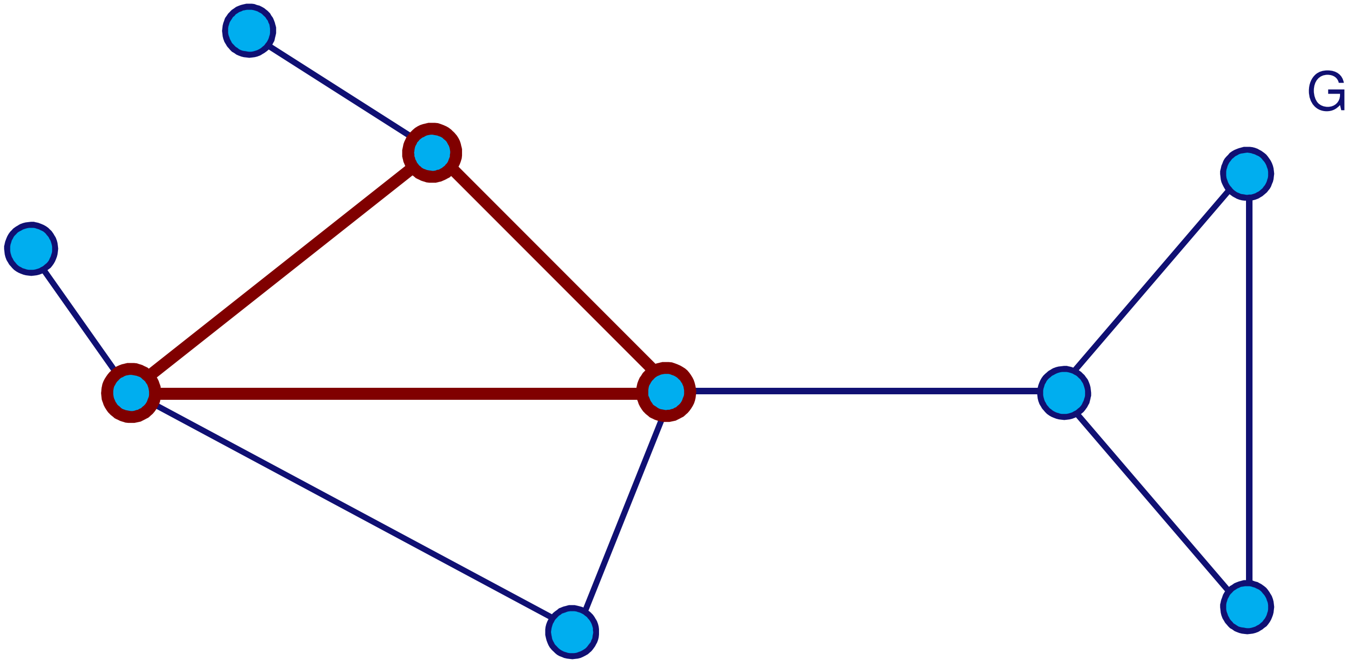




Problems with Frequency

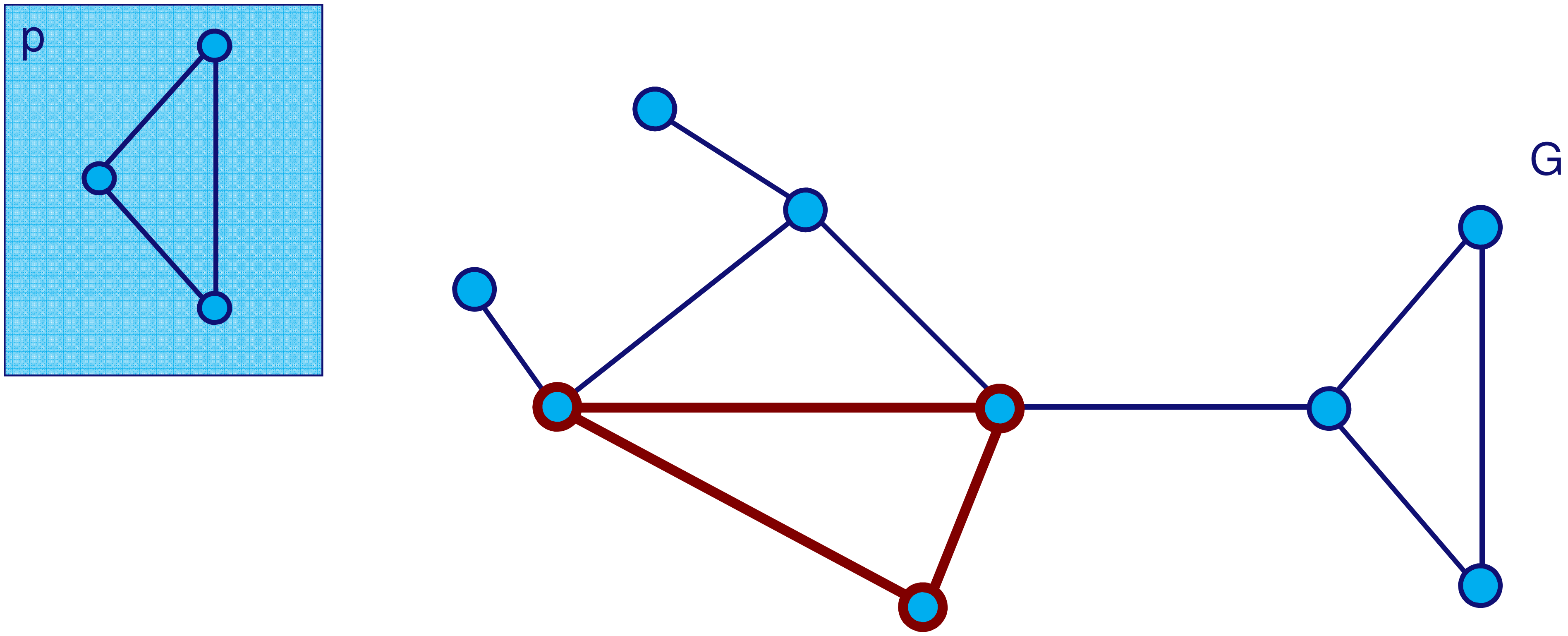
- **Counting instances does not work !**





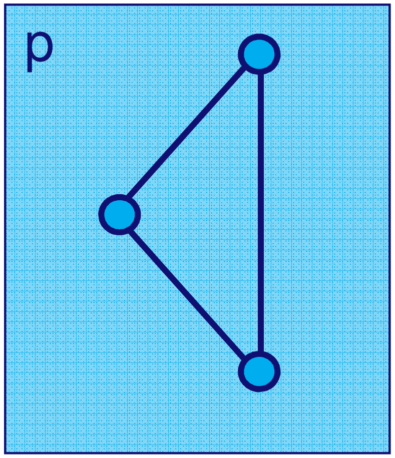
Problems with Frequency

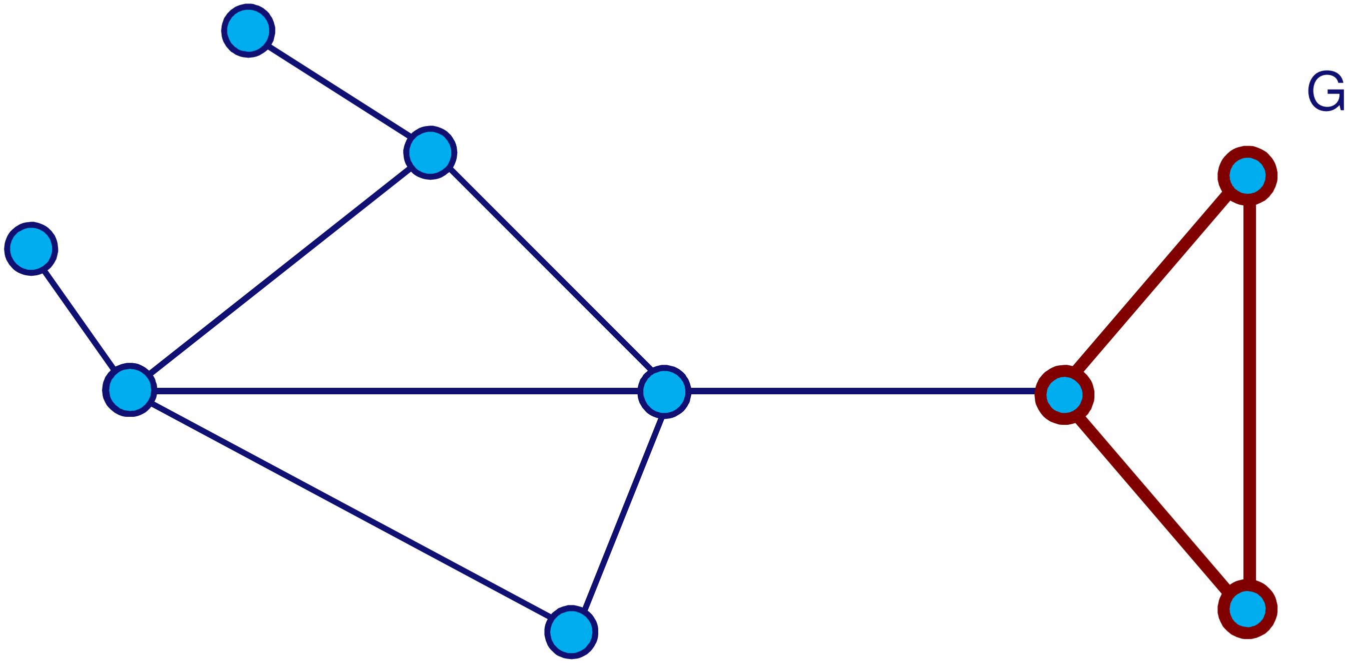
- **Counting instances does not work !**



Problems with Frequency

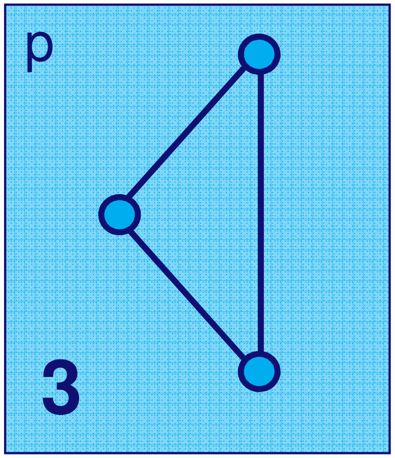
- **Counting instances does not work !**

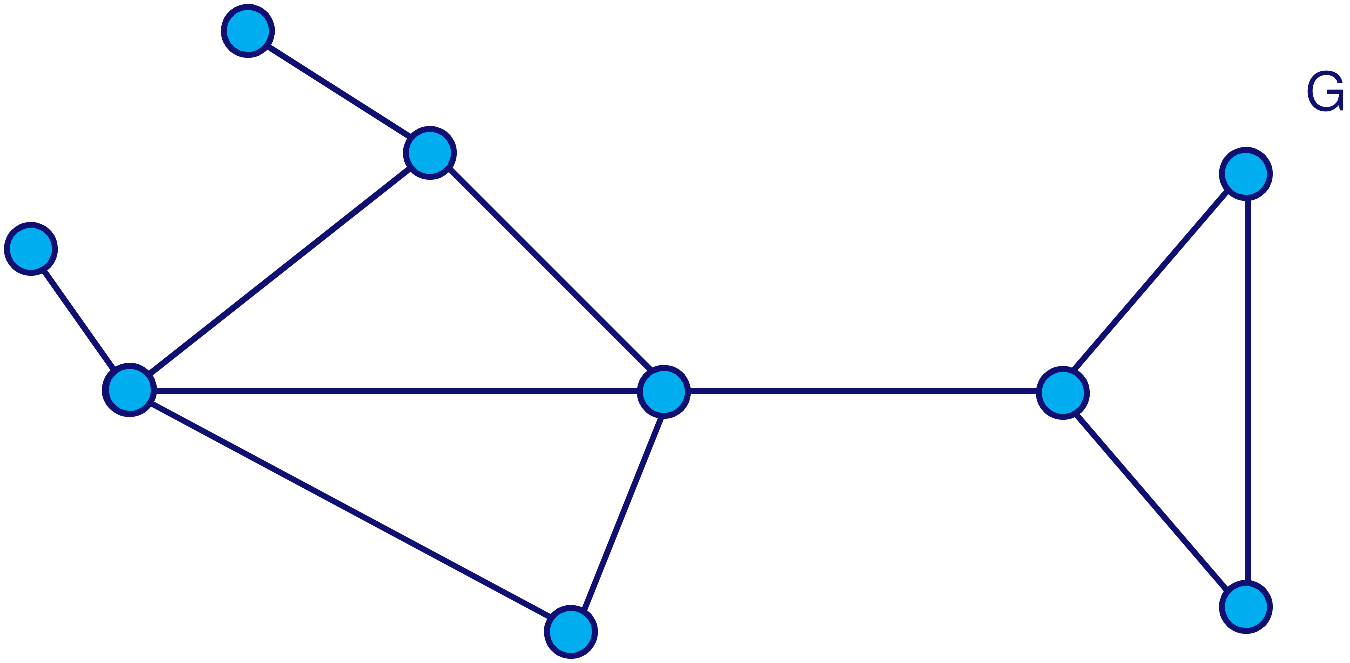




Problems with Frequency

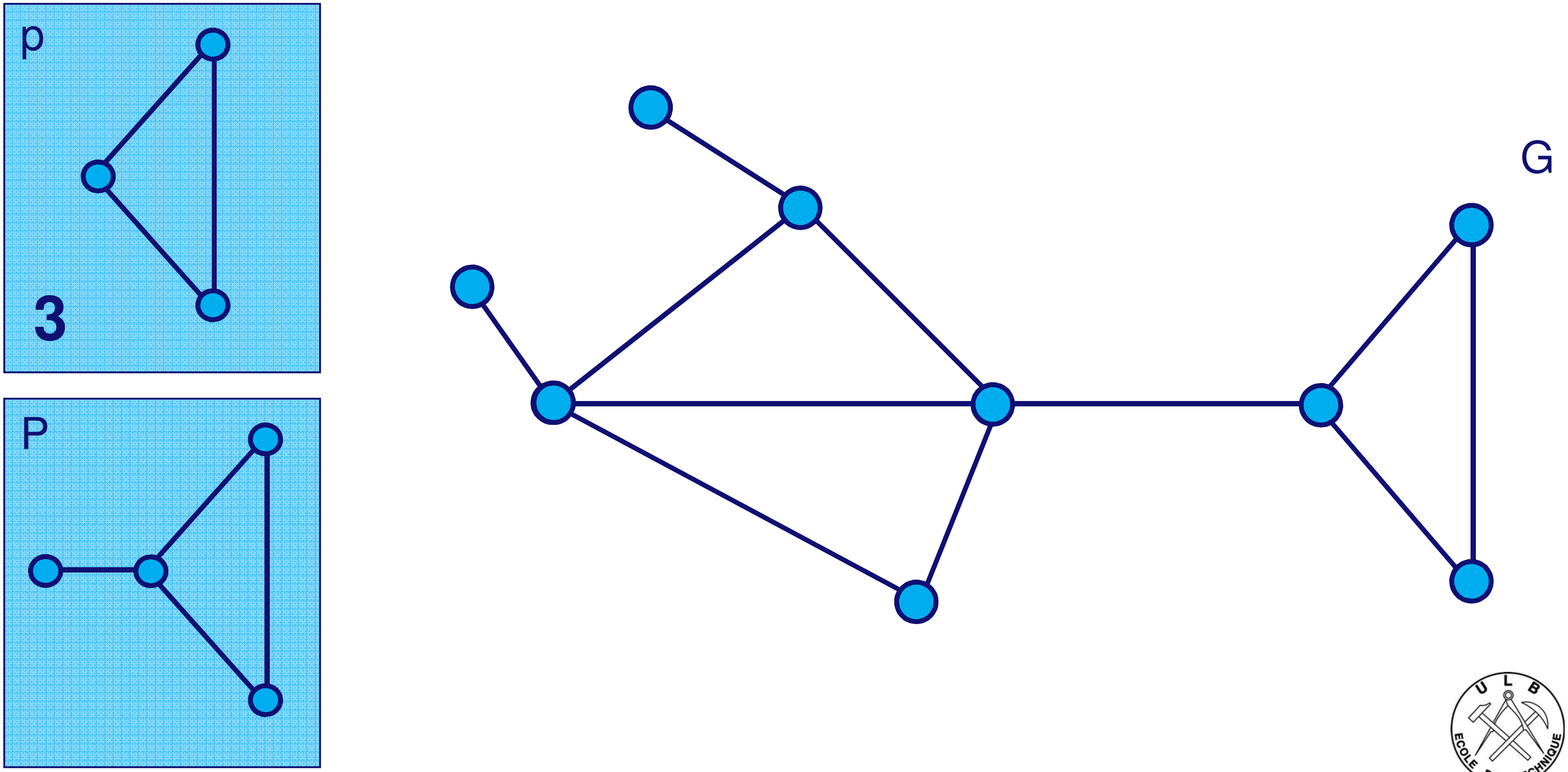
- **Counting instances does not work !**





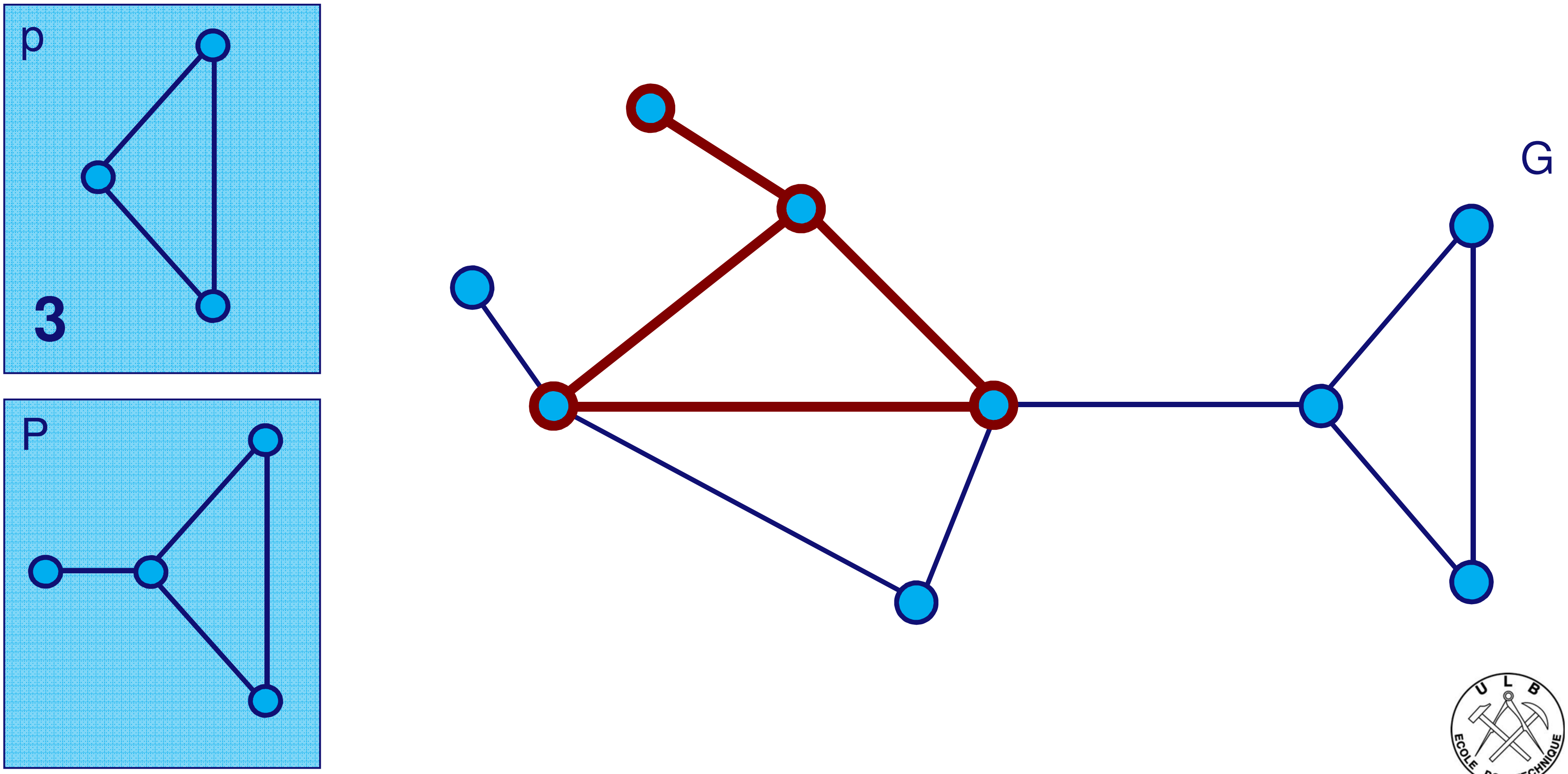
Problems with Frequency

- **Counting instances does not work !**



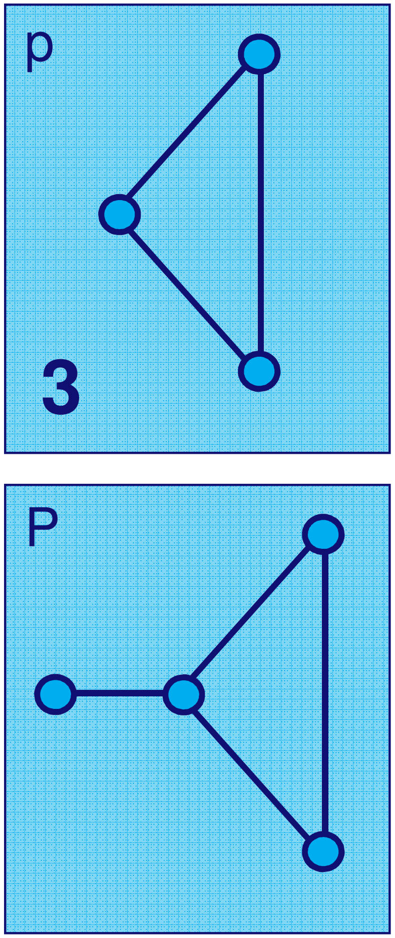
Problems with Frequency

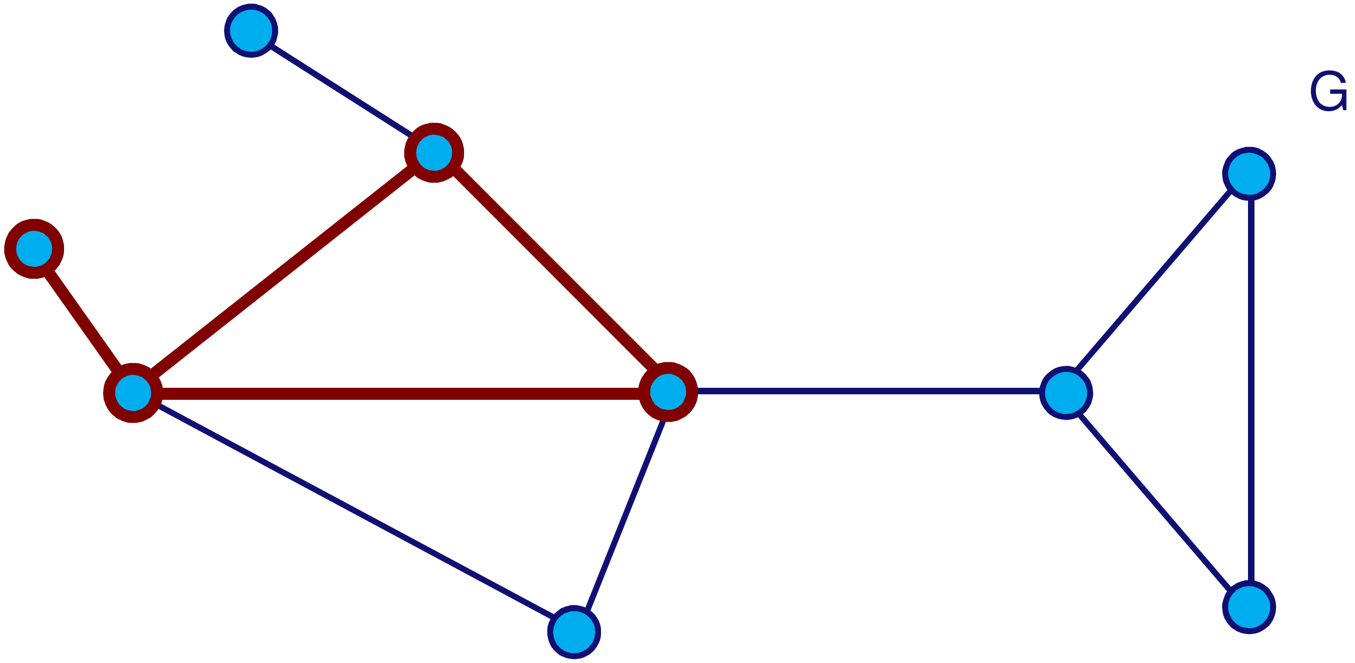
- **Counting instances does not work !**



Problems with Frequency

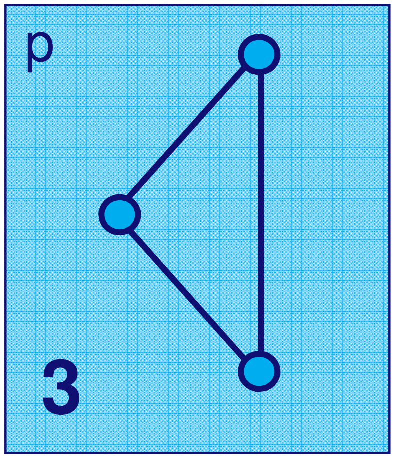
- **Counting instances does not work !**

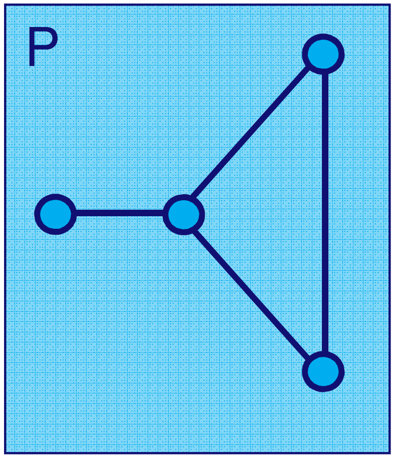


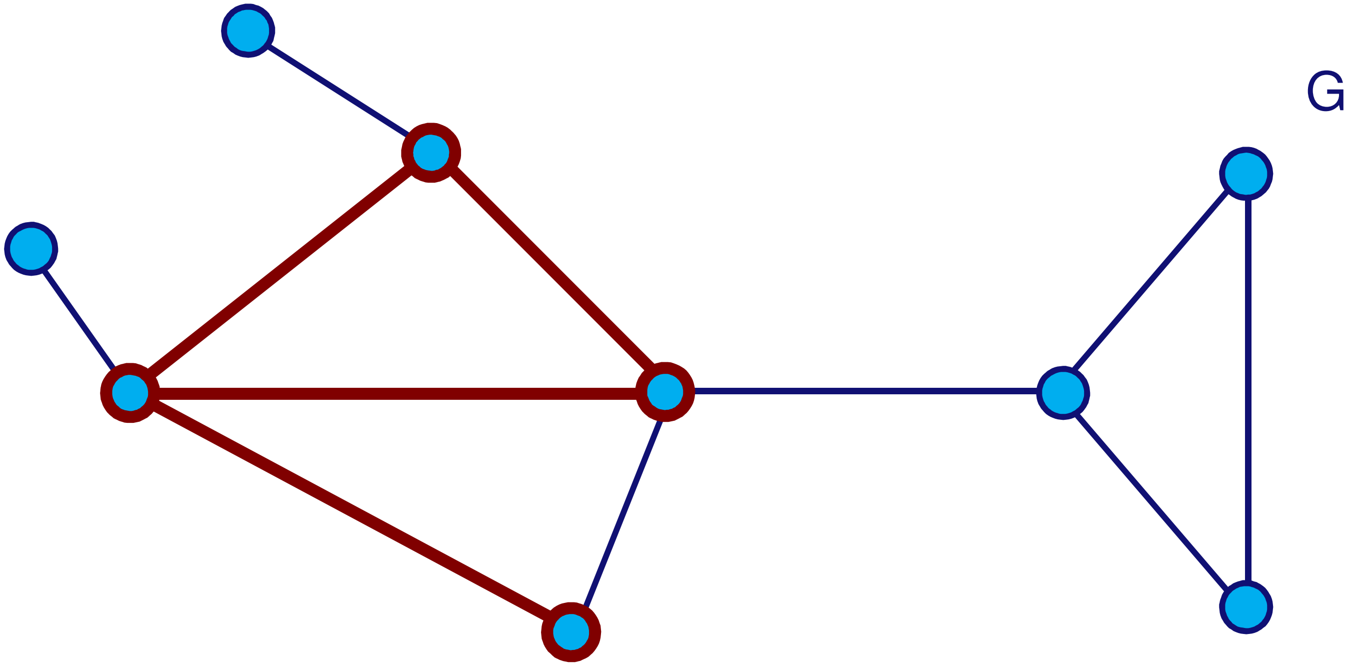


Problems with Frequency

- **Counting instances does not work !**

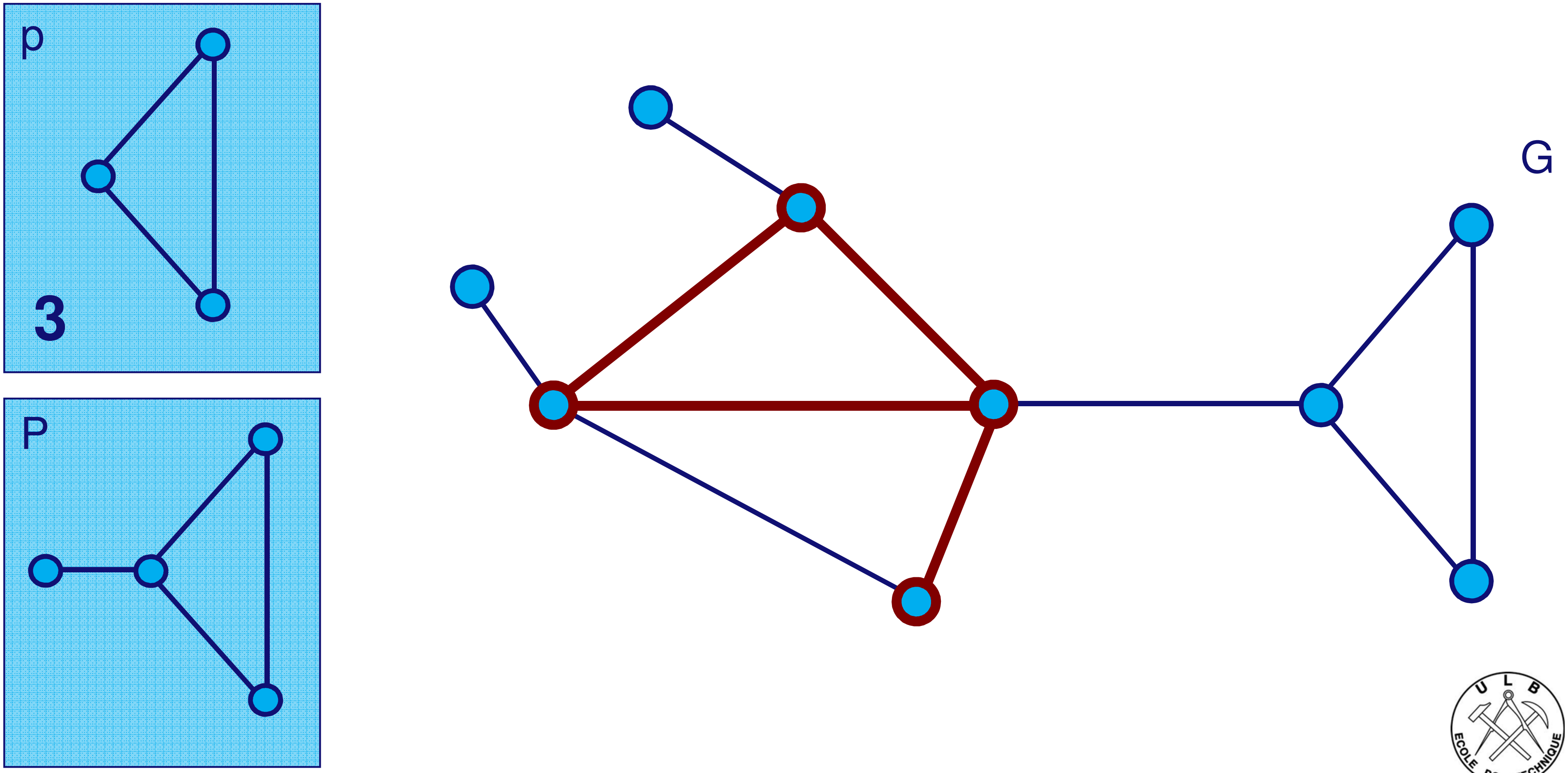






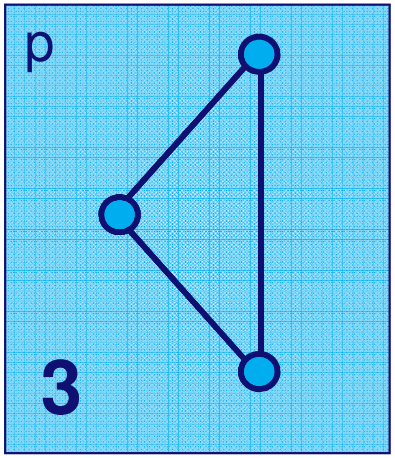
Problems with Frequency

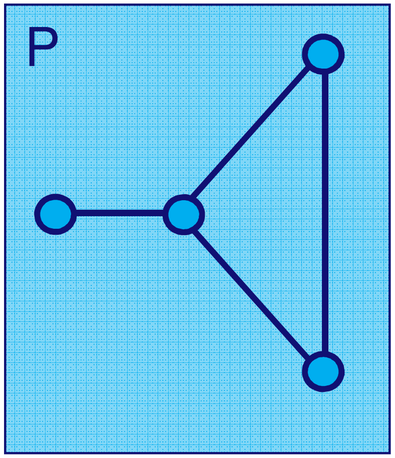
- **Counting instances does not work !**

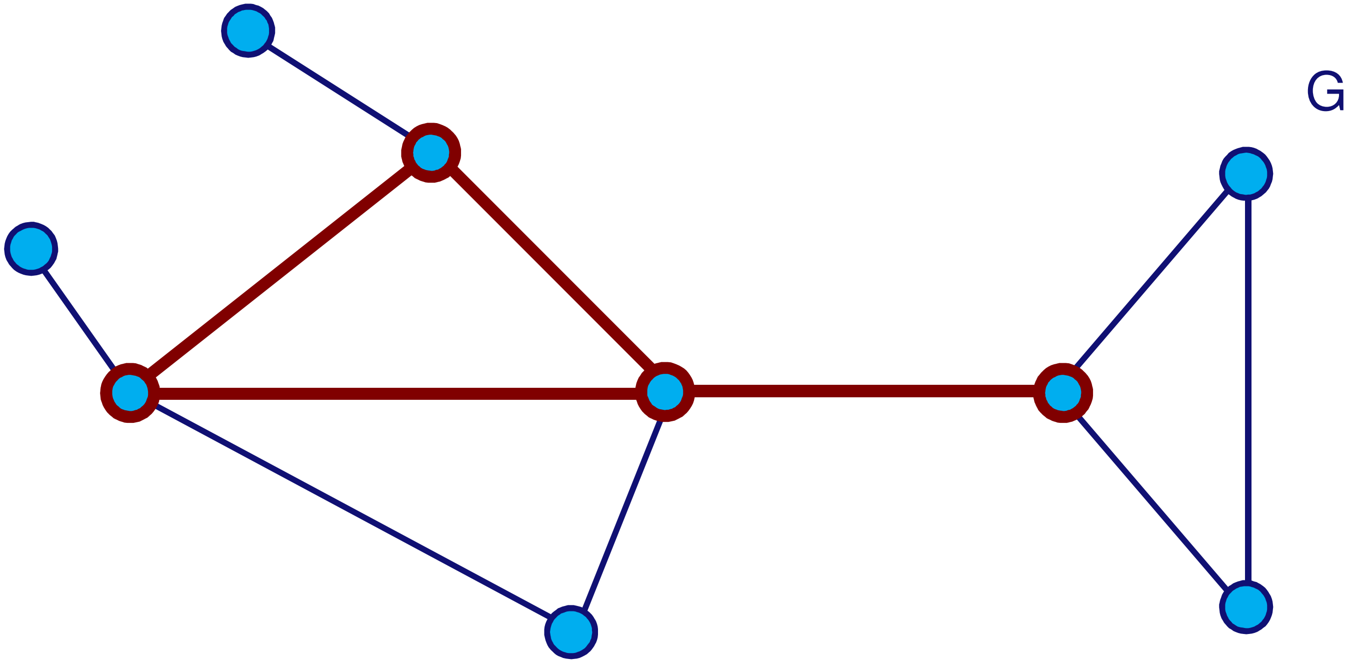


Problems with Frequency

- **Counting instances does not work !**

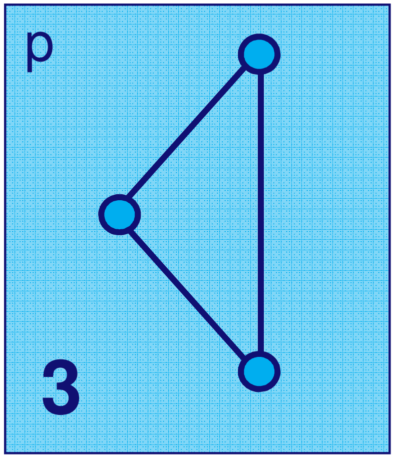


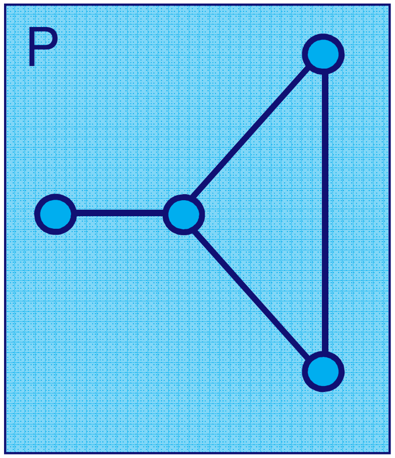


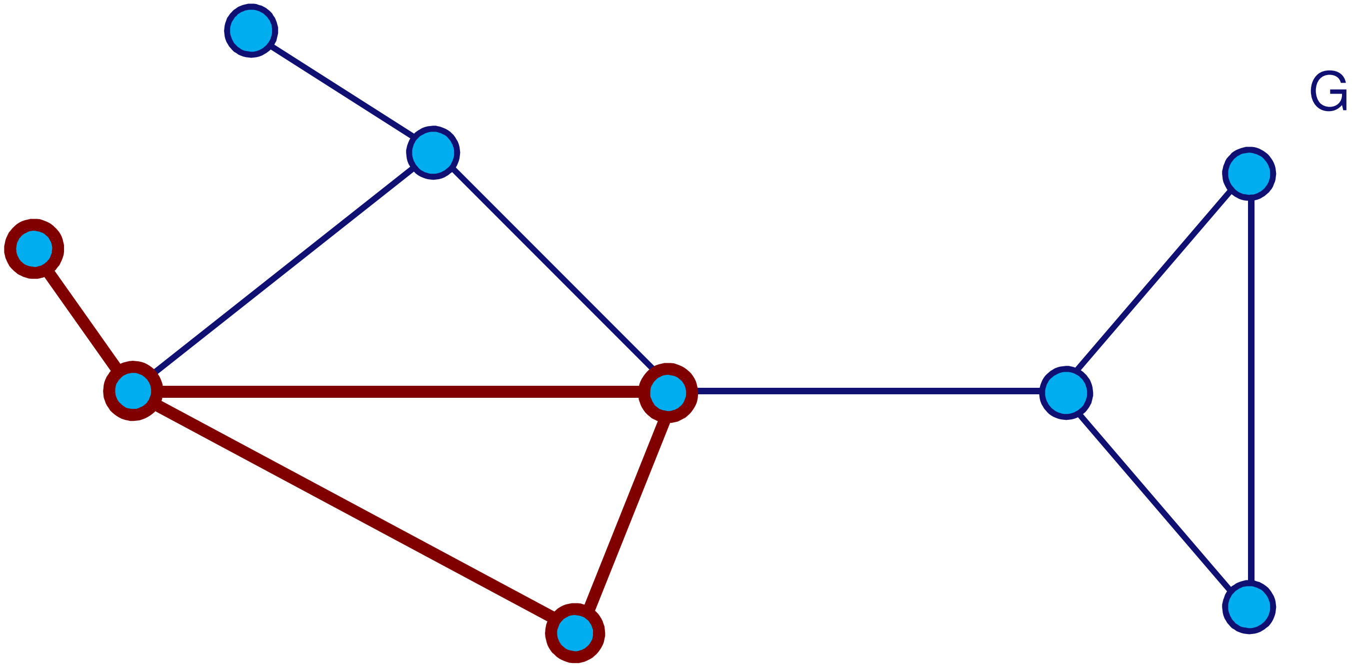


Problems with Frequency

- **Counting instances does not work !**

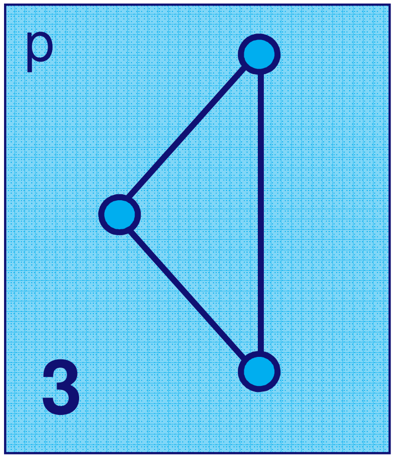


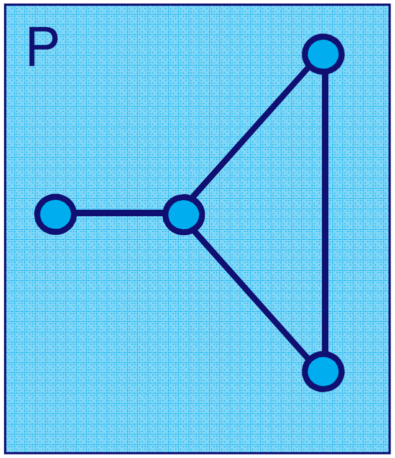


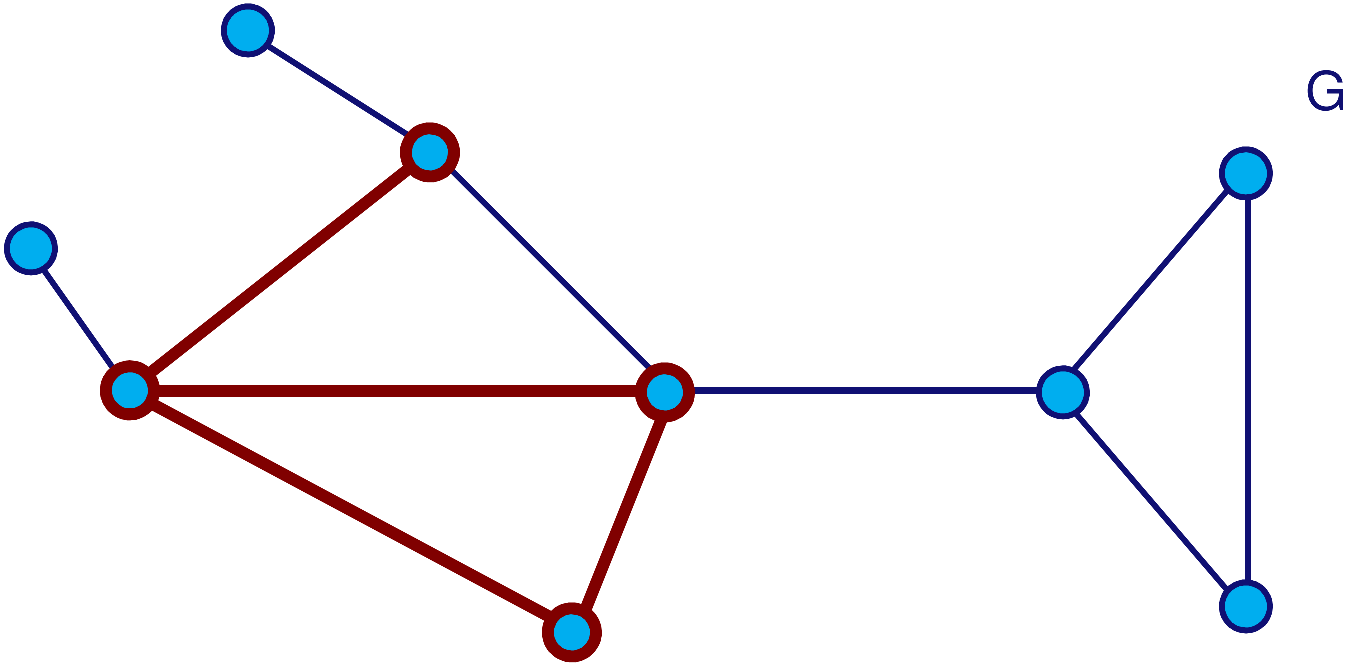


Problems with Frequency

- **Counting instances does not work !**

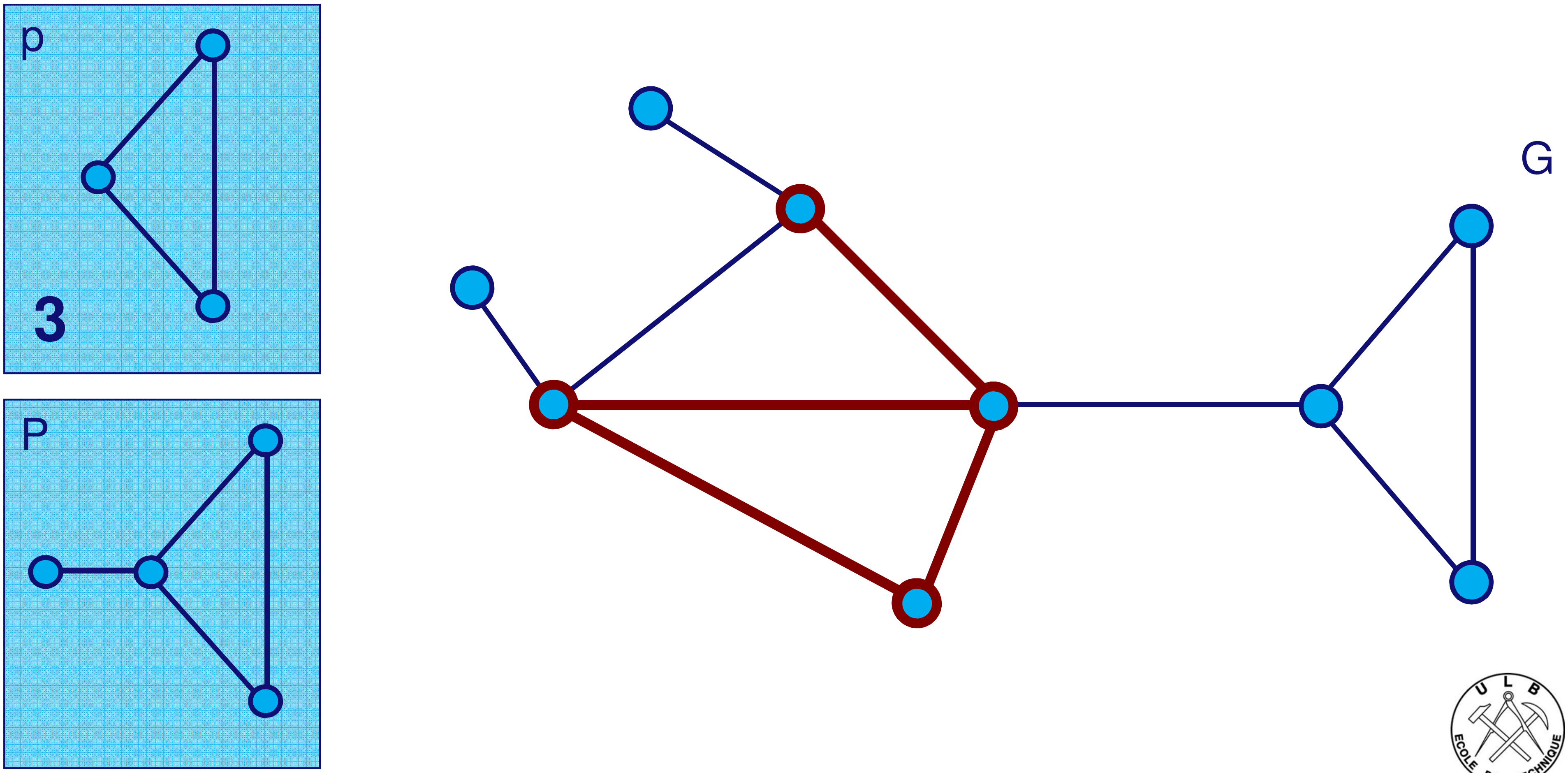






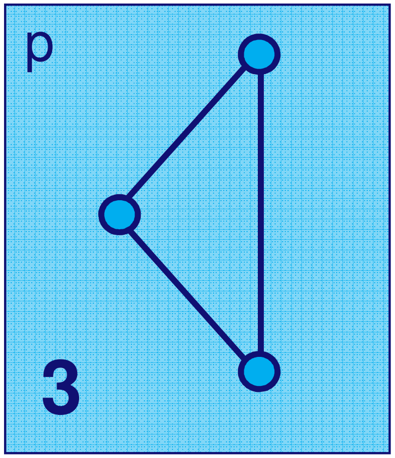
Problems with Frequency

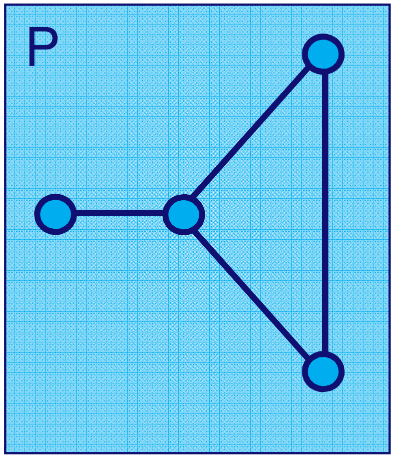
- **Counting instances does not work !**

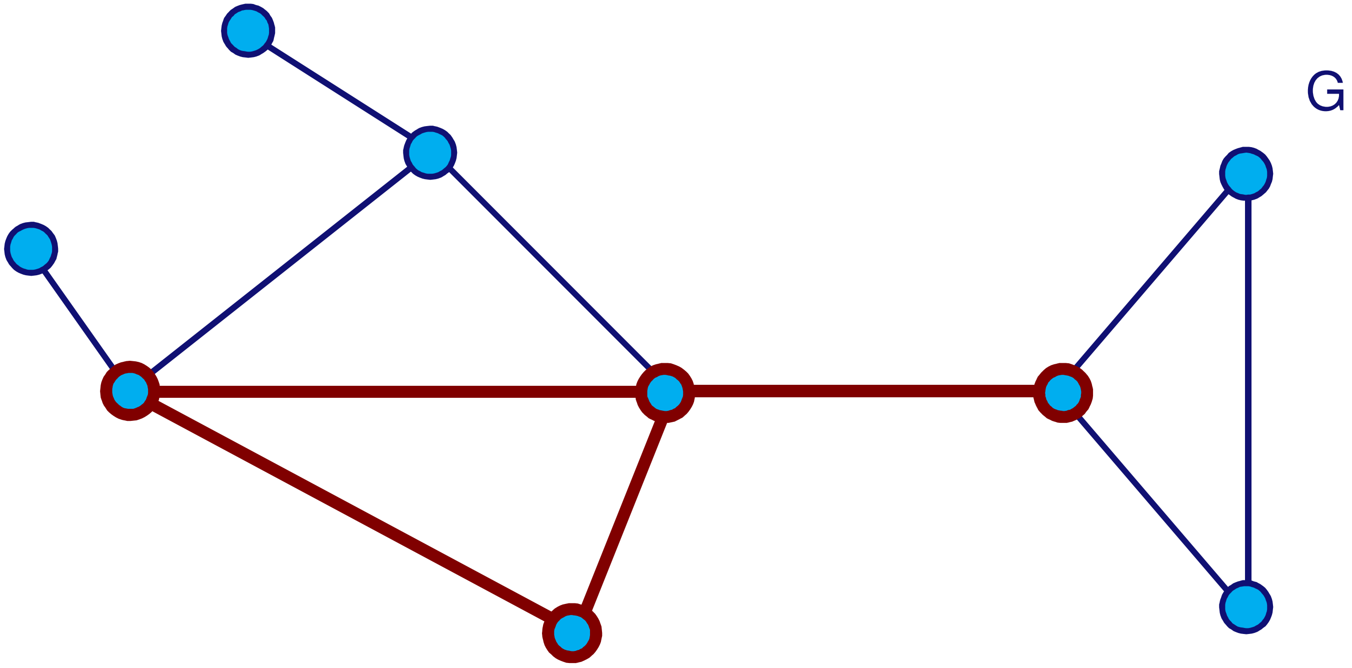


Problems with Frequency

- **Counting instances does not work !**

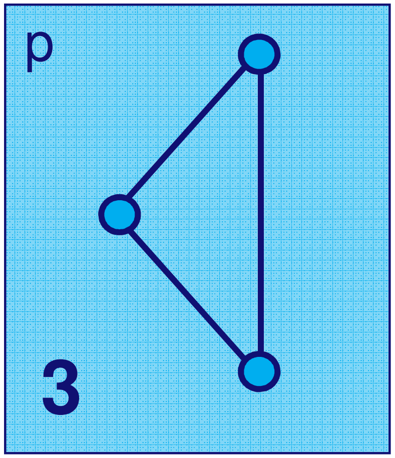


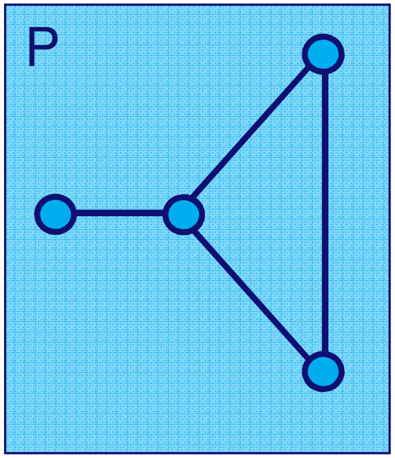


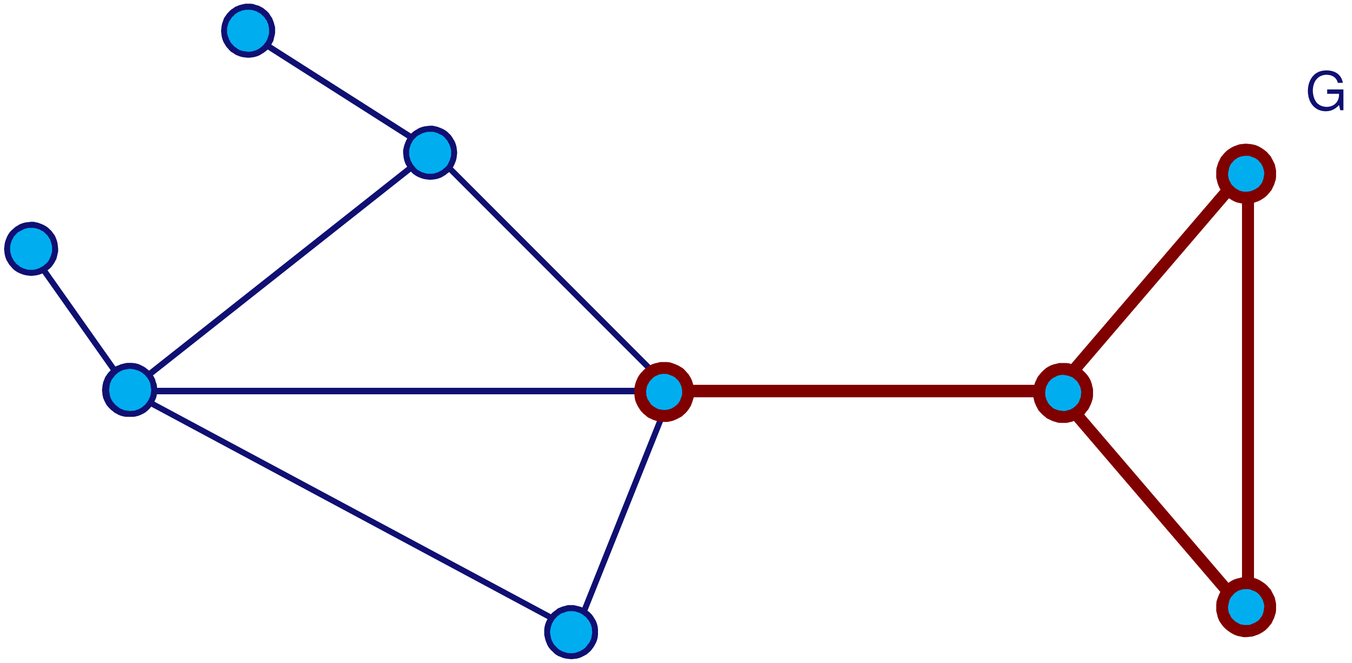


Problems with Frequency

- **Counting instances does not work !**

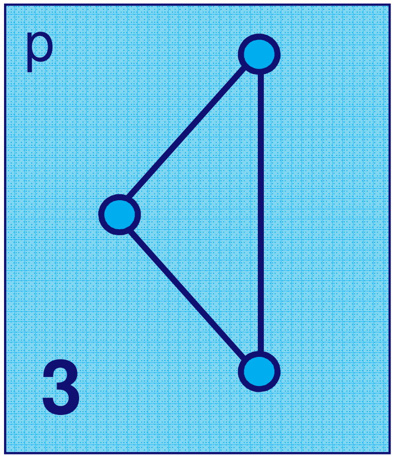


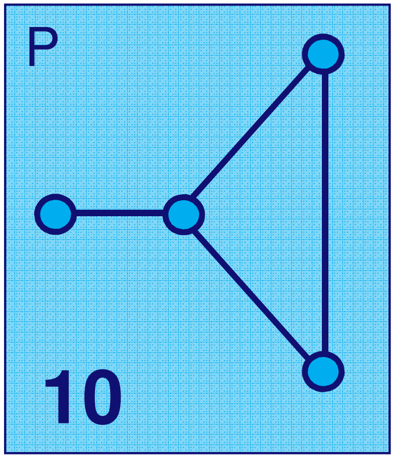


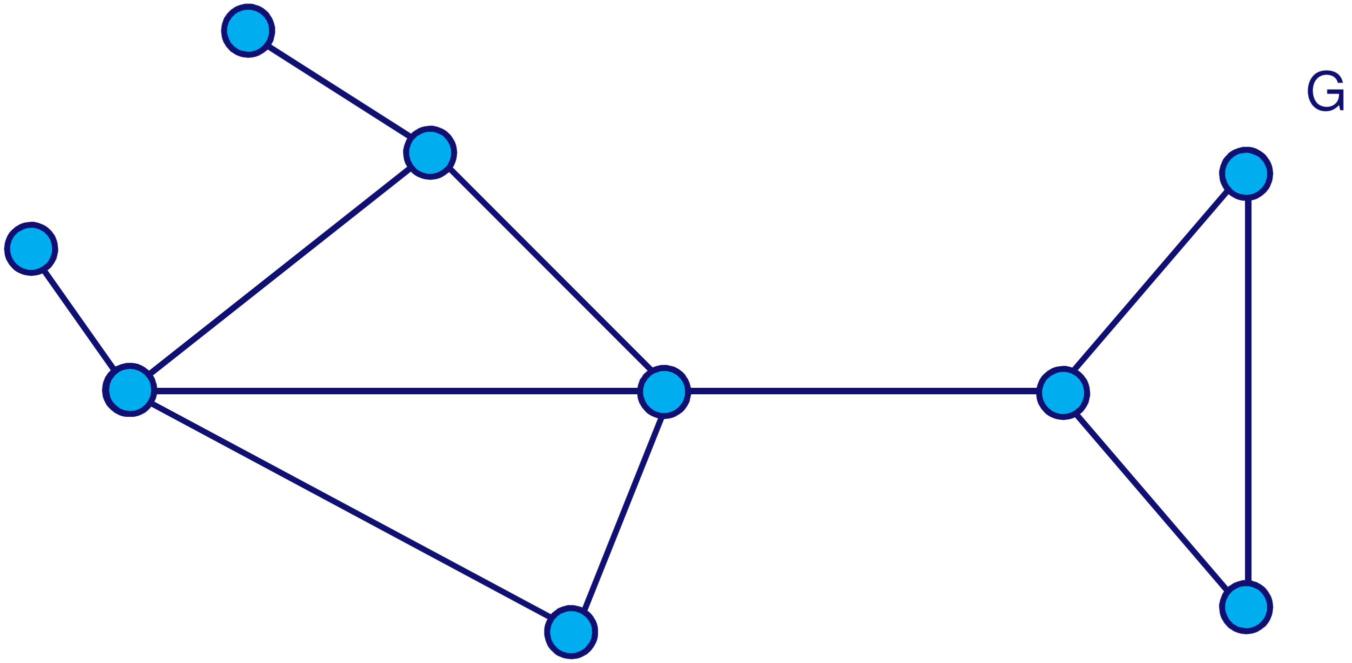


Problems with Frequency

- **Counting instances does not work !**

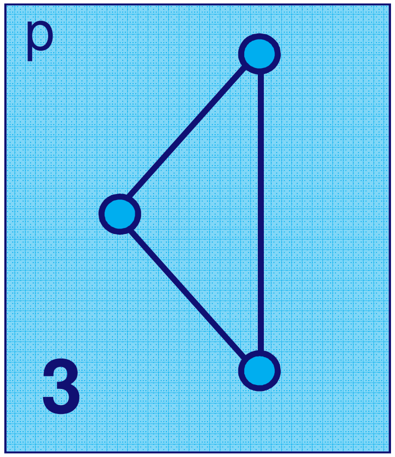




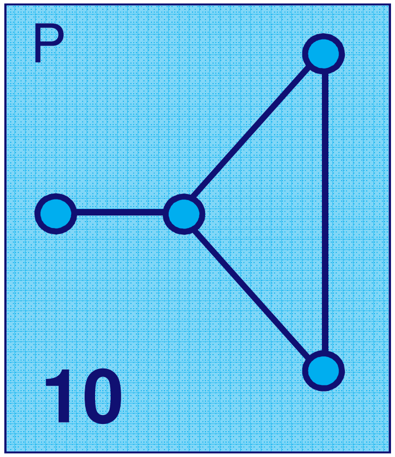


Problems with Frequency

- **Counting instances does not work !**
  - **Counter-intuitive**



Less frequent than



?!

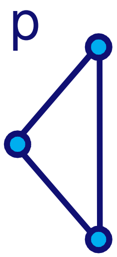
- **Algorithms rely critically on *anti-monotonicity* for pruning the search space**

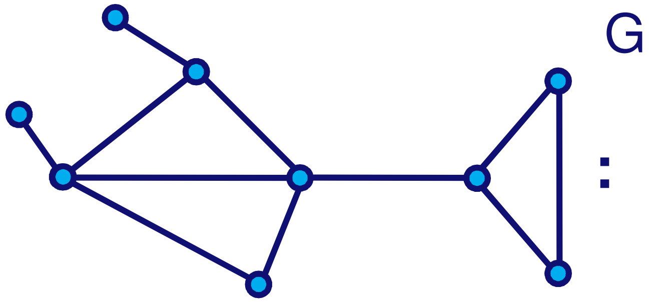
Skip to part II

Overlap Graph

- **Most algorithms for single graph mining base themselves on the *overlap-graph*:**

Example:

overlap graph of 

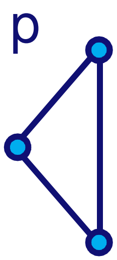
in 

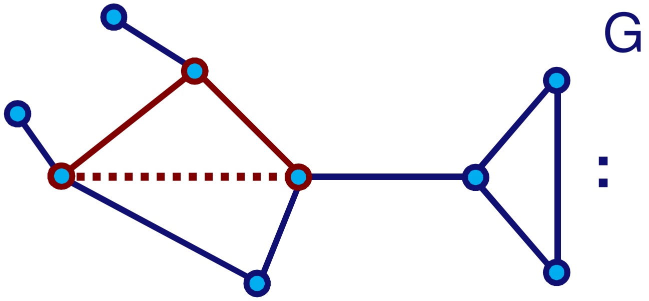
:

Overlap Graph

- Most algorithms for single graph mining base themselves on the *overlap-graph*:

Example:

overlap graph of 

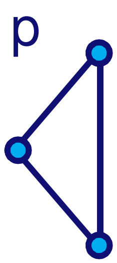
in 

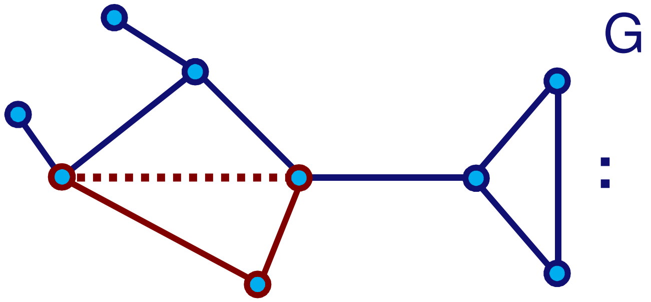
:

Overlap Graph

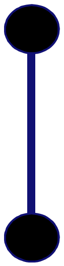
- Most algorithms for single graph mining base themselves on the *overlap-graph*:

Example:

overlap graph of 

in 

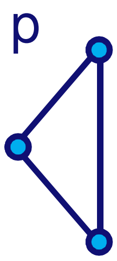
:

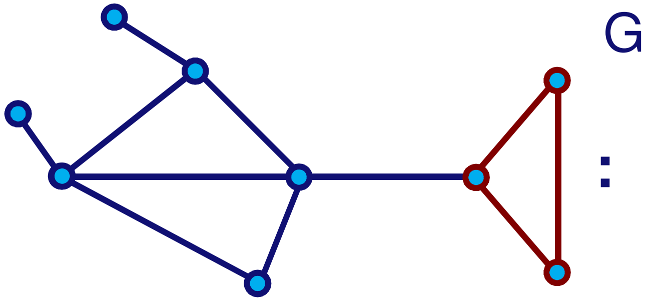


Overlap Graph

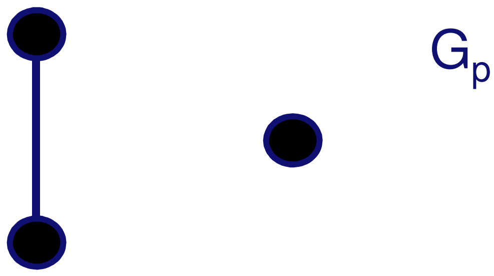
- Most algorithms for single graph mining base themselves on the *overlap-graph*:

Example:

overlap graph of 

in 

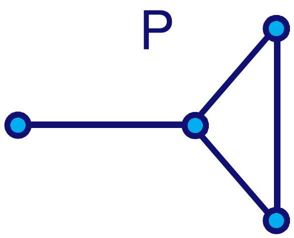
:

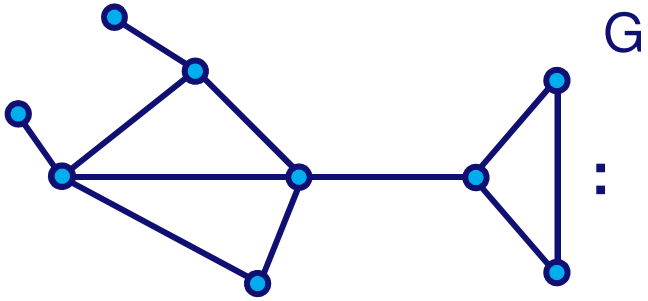


Overlap Graph

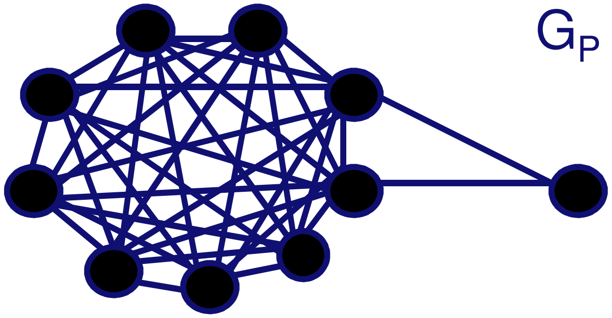
- **Most algorithms for single graph mining base themselves on the *overlap-graph*:**

Example:

overlap graph of 

in 

:



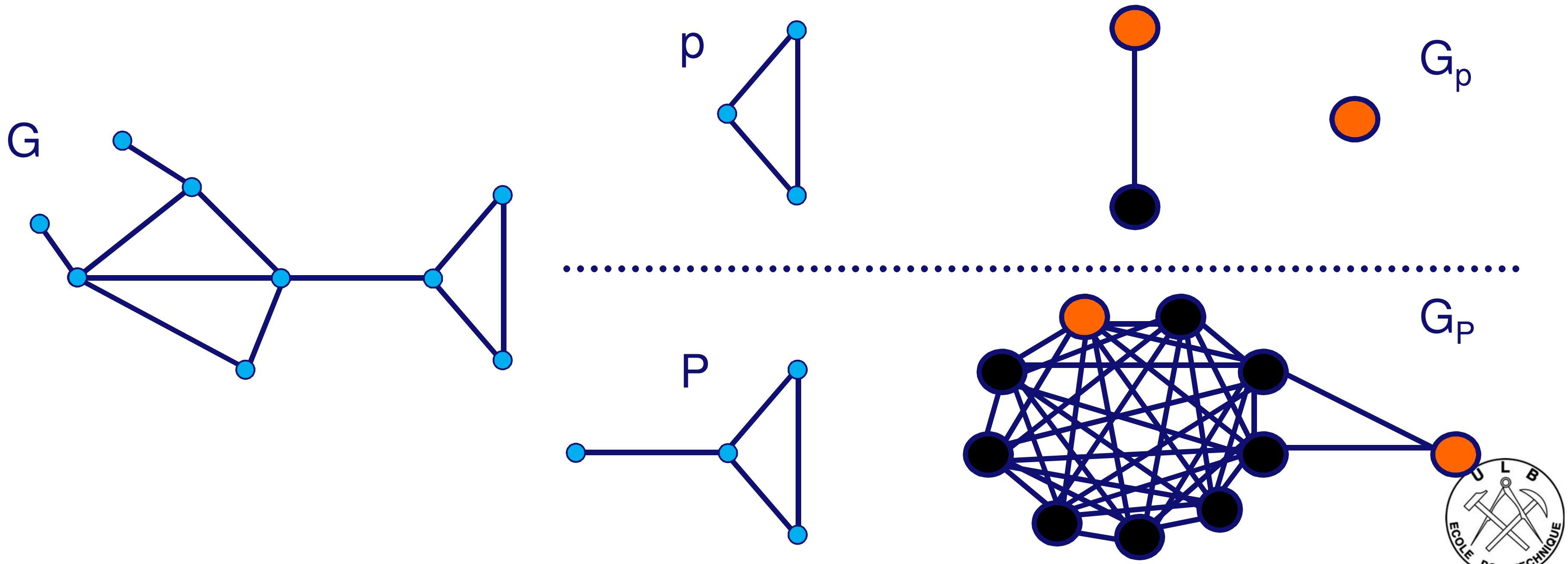
Overlap Graph

- Summarizes all instances, and *describes* how they *overlap*
  - vertex ←→ instance
  - edge ←→ overlap
- Notion extends straightforwardly to instances in labeled/directed graphs
- Yet, overlap graph is *always* an *unlabeled, undirected graph*

Maximum Independent Set

- **Anti-monotone measure on overlap graph:**
  - **size of the *Maximum Independent Set* of the overlap graph**

Data graph **pattern** **overlap graph**



Summary: Extension to Other Pattern Types

- Many extensions of frequent itemset mining exist
  - Sequences, partial orders, trees, graphs

- Most algorithms are depth-first
  - Too many patterns of same size for breadth-first

- Extensions become much more challenging
  - Pattern generation without duplicates
  - Define a good support measure
  - Counting support efficiently

Outline

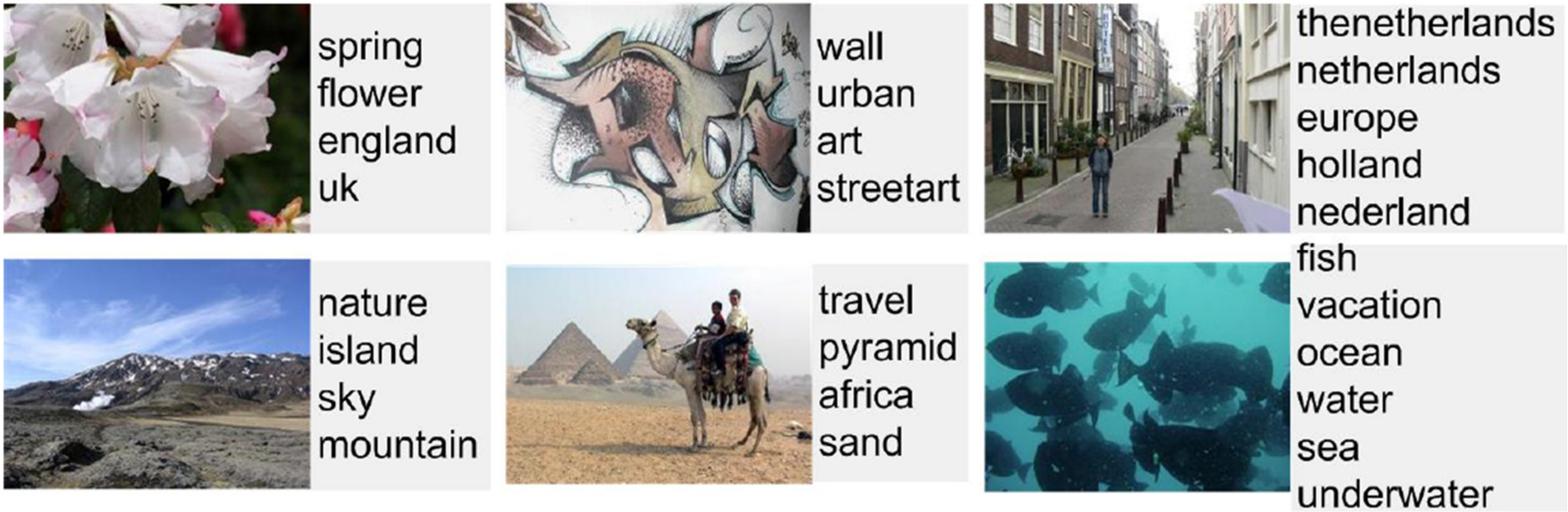
PART I: Frequent itemset mining

- Definition & Applications
- Algorithms for Frequent Itemset Mining
- Extensions to other pattern types

PART II:

- Pattern explosion & Redundancy problem
- Methods to remove redundancy
  - Condensed representations
  - Statistical methods
  - Minimal Description Length

Illustration: Tags dataset



Pictures from: Xirong Li, Cees G.M. Snoek, and Marcel Worring, Learning Social Tag Relevance by Neighbor Voting, IEEE Transactions on Multimedia, volume 11, issue 7, page 1310-1322, 2009

- **Flickr tags dataset**
  - **Question: What are popular tags?**

Illustration: Tags dataset (top-most frequent)

796	street
713	bridge
661	night
552	city
532	people
527	water
521	the
517	bus
495	dog
489	boat
487	sky
477	telephone
453	canon
433	kitchen
431	airplane
430	ship
423	new
409	blue
404	of
395	harbour
387	cityscape
381	flying

Illustration: Tags Dataset (Longest)

304 flight aeroplane travel aircraft plane

319 flight aeroplane aircraft plane

308 aeroplane travel aircraft plane

305 flight travel aircraft plane

304 flight aeroplane travel plane

304 flight aeroplane travel aircraft

639 airport aircraft plane

630 and black white

511 war protest demonstration

489 aeroplane aircraft plane

Redundancy Problem

- Frequent itemset / Association rule mining

  = find all itemsets / ARs satisfying thresholds

- Only some itemsets / association rules are interesting
  - Many are redundant

smoker → lung cancer

smoker, bald → lung cancer

pregnant → woman

pregnant, smoker → woman, lung cancer

Outline

PART I: Frequent itemset mining

- Definition & Applications
- Algorithms for Frequent Itemset Mining
- Extensions to other pattern types

PART II:

- Pattern explosion & Redundancy problem
- Methods to remove redundancy
  - Condensed representations
  - Statistical methods
  - Minimal Description Length

Compact Representation of Frequent Itemsets

- **Some itemsets are redundant because they have identical support as their supersets**

TID	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	1	1	1	1	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0	0	0
5	1	1	1	1	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0	0	0
6	0	0	0	0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0	0	0
7	0	0	0	0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0	0	0
8	0	0	0	0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0	0	0
9	0	0	0	0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0	0	0
10	0	0	0	0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0	0	0
11	0	0	0	0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1	1	1
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1

304 flight aeroplane travel aircraft plane
304 flight aeroplane travel plane
304 flight aeroplane travel aircraft

- **Number of frequent itemsets** $= 3 \times 2^{10} - 2 = 3070$

- **Need a compact representation**

Closed Itemsets

TID	A	B	C	D
1	0	1	1	0
2	0	1	1	0
3	1	0	1	1
4	1	1	1	1
5	0	1	0	1

- **The *support-set* of an itemset I is:**

  $$\mathrm{sset}(I) := \{\ \mathrm{TID} \mid (\mathrm{TID},J) \in D,\ I \subseteq J\ \}$$

- **Itemset I and J are said to be *equivalent* if:**

  $$\mathrm{sset}(I) = \mathrm{sset}(J)$$

Example:

$\mathrm{sset}(A) = \{\ 3, 4\ \}$

$\mathrm{sset}(AC) = \{\ 3, 4\ \}$

$\mathrm{sset}(BC) = \{\ 1, 2, 4\ \}$

Closed Itemsets

TID	A	B	C	D
1	0	1	1	0
2	0	1	1	0
3	1	0	1	1
4	1	1	1	1
5	0	1	0	1

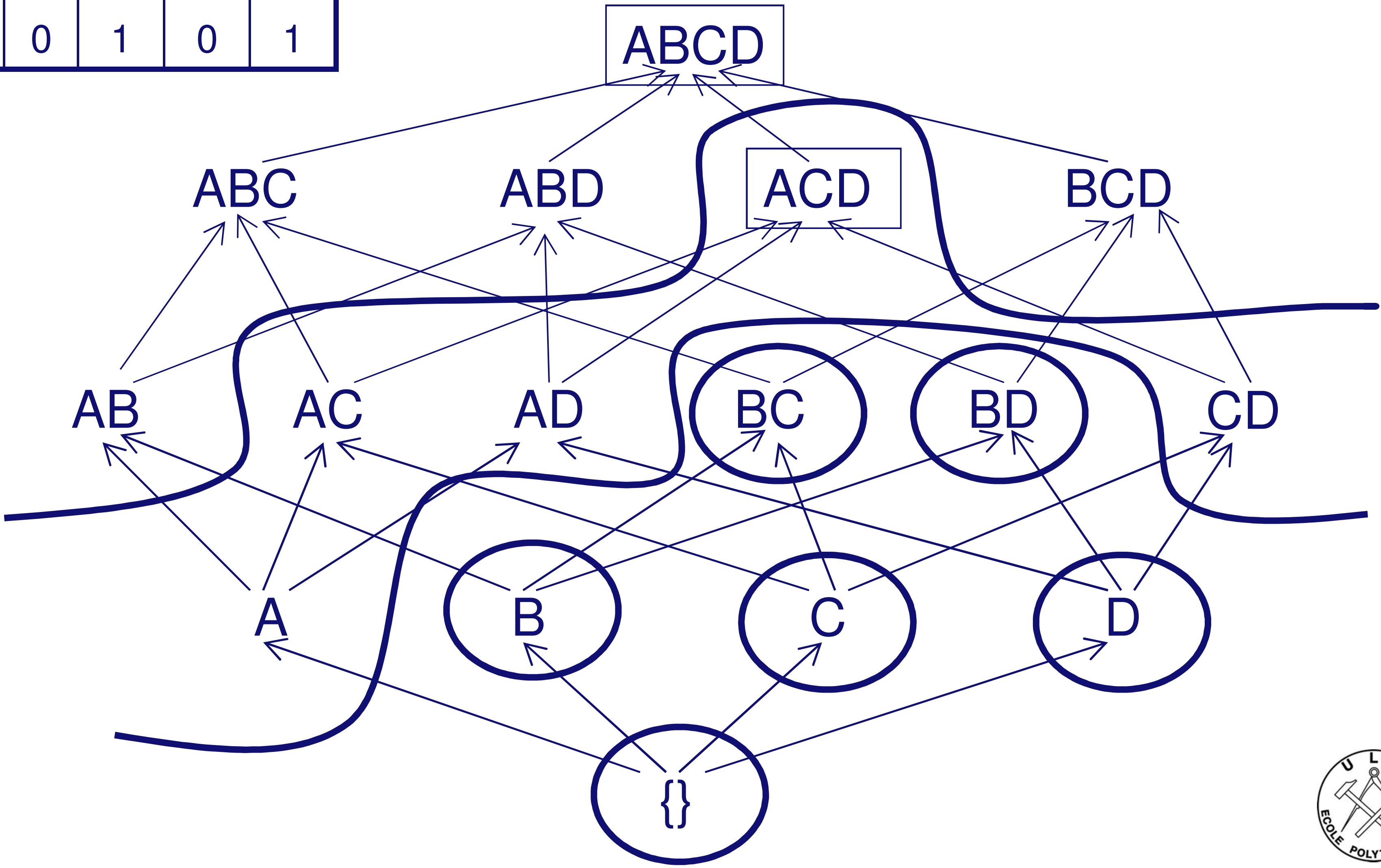
- **Let [I] denote the *equivalence class* of itemset I**
  - **For all $J \in [I]$: support(J) = support(I)**
    - **support(J) = | sset(J) | = | sset(I) |**
  - **[I] has a unique maximal element max([I])**
    - **If $X \in [I]$, $Y \in [I]$, then also $X \cup Y \in [I]$**
  - **The *closure cl(I)* of an itemset I is defined as max([I])**
  - **A set I is *closed* if I = cl(I)**

Example:

[ACD] = { A, AC, AD, ACD, CD }; hence ACD is closed

TID	A	B	C	D
1	0	1	1	0
2	0	1	1	0
3	1	0	1	1
4	1	1	1	1
5	0	1	0	1

Closed Itemsets



ABCD

ABC ABD ACD BCD

AB AC AD BC BD CD

A B C D

{}

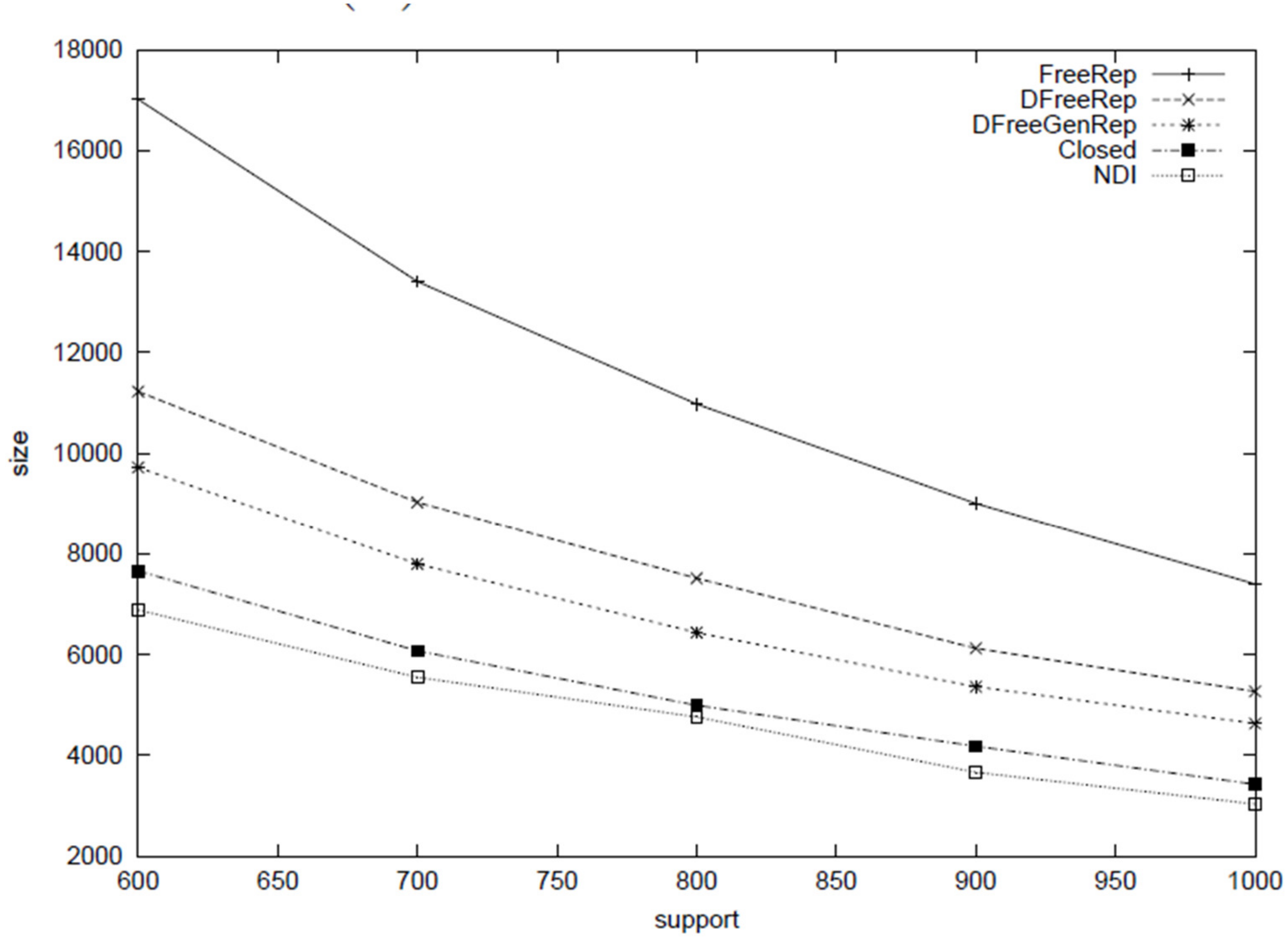
Closed Itemsets

- All sets in the same equivalence class have the same support
  - Occur in the same transactions
- Maximal element in an equivalence class is unique
  - If two itemsets occur in the same transactions, then so does their union

- Frequent Closed Itemset representation:

$$\{ I \mid I \in F \text{ and } I \text{ is closed}\}$$

Benefit of Condensed Representations



(d) Mushroom

Figure from: Toon Calders. Deducing Bounds on the Support of Itemsets. In: *Database Support for Data Mining Applications*: pp. 214-233 (2004)

Disadvantages of the “Combinatorial Method”

- Still too many rules/itemsets remain
  - Rules where head and tail are independent remain

conf(smoking => lung cancer) = 20%

conf(smoking & blue eyes => lung cancer) = 20%

  - Highly frequent items form together frequent itemsets

    → not very surprising

- Need a way to quantify what is “surprising”
  - Depends on what we expect

Closed Sets – Tags Dataset

304 flight aeroplane travel aircraft plane

319 flight aeroplane aircraft plane

308 aeroplane travel aircraft plane

305 flight travel aircraft plane

~~304 flight aeroplane travel plane~~

~~304 flight aeroplane travel aircraft~~

639 airport aircraft plane

630 and black white

511 war protest demonstration

489 aeroplane aircraft plane

458 ussmidway sandiego aircraftcarrier

449 aviation aircraft plane

Outline

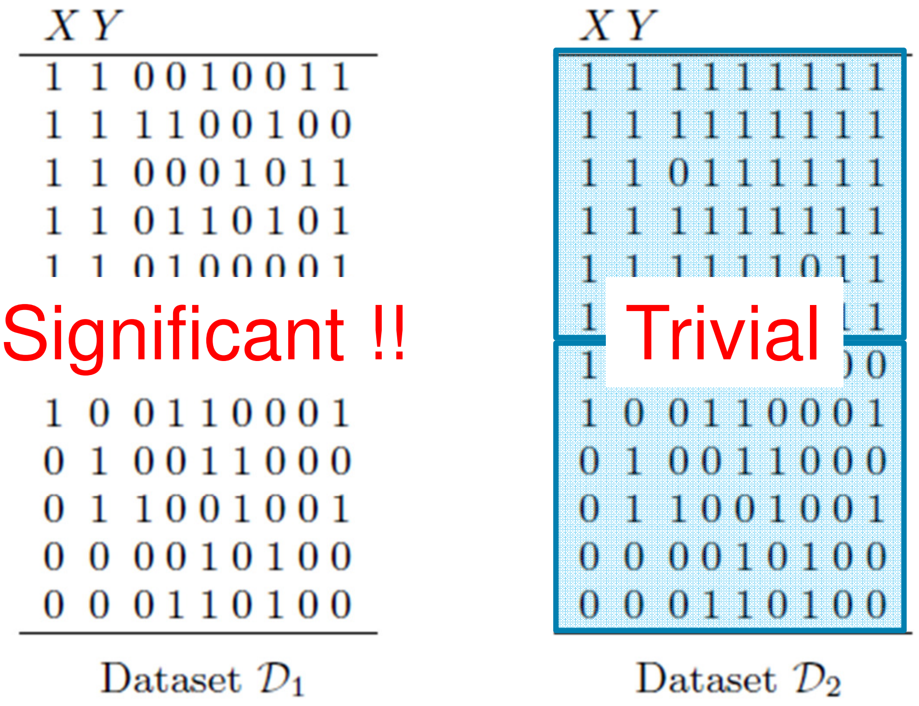
PART I: Frequent itemset mining

- Definition & Applications
- Algorithms for Frequent Itemset Mining
- Extensions to other pattern types

PART II:

- Pattern explosion & Redundancy problem
- Methods to remove redundancy
  - Condensed representations
  - Statistical methods
  - Minimal Description Length

Interestingness Depends on Expectation



X	Y							
1	1	0	0	1	0	0	1	1
1	1	1	1	0	0	1	0	0
1	1	0	0	0	1	0	1	1
1	1	0	1	1	0	1	0	1
1	1	0	1	0	0	0	0	1
1	0	0	1	1	0	0	0	1
0	1	0	0	1	1	0	0	0
0	1	1	0	0	1	0	0	1
0	0	0	0	1	0	1	0	0
0	0	0	1	1	0	1	0	0

Dataset $\mathcal{D}_1$

Significant !!

X	Y							
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	0	1	1
1							1	1
1								0
1	0	0	1	1	0	0	0	1
0	1	0	0	1	1	0	0	0
0	1	1	0	0	1	0	0	1
0	0	0	0	1	0	1	0	0
0	0	0	1	1	0	1	0	0

Dataset $\mathcal{D}_2$

Trivial

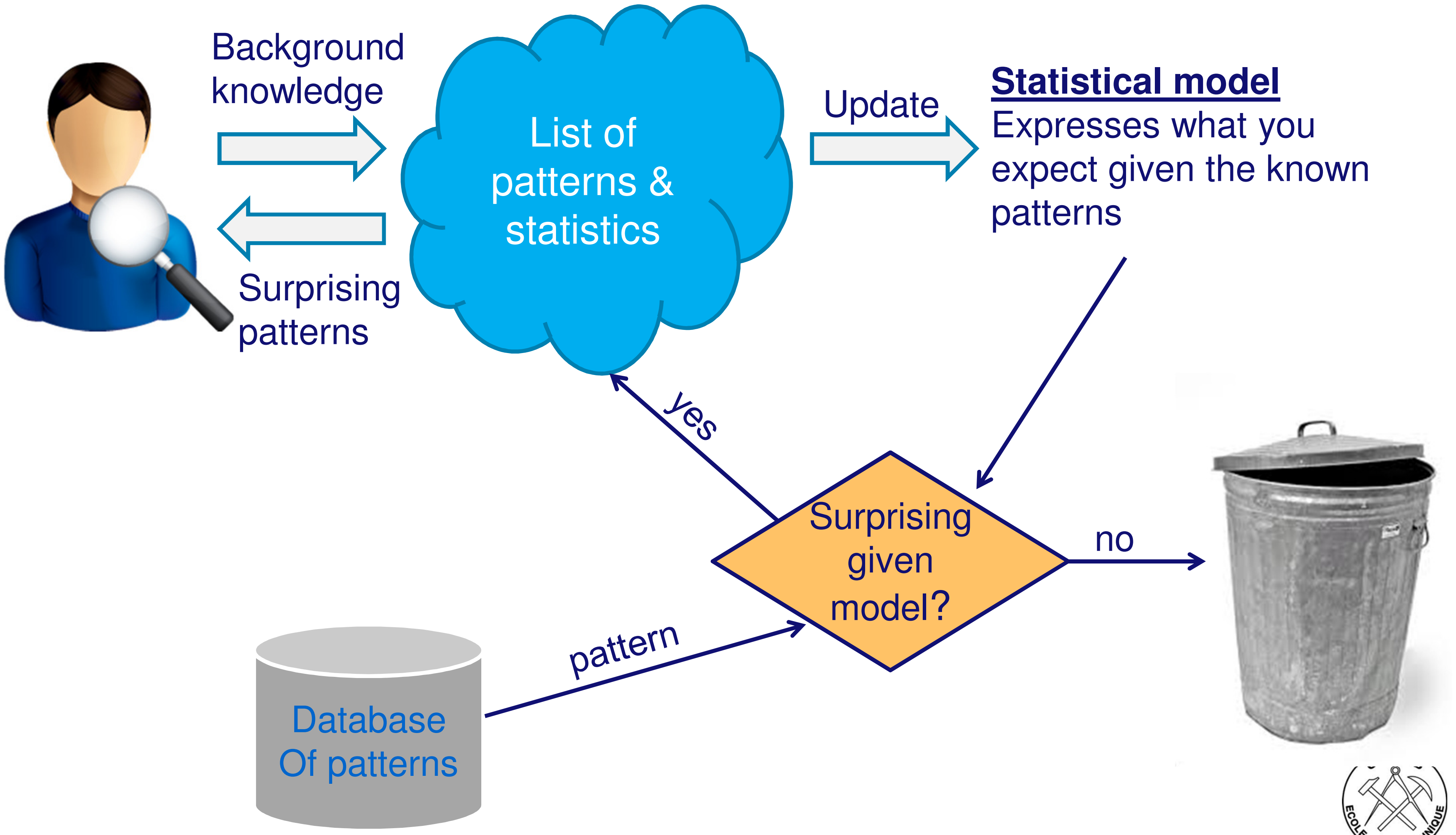
Buying a lot

Buying nothing

Figure 2: Examples of two 0–1 datasets, $\mathcal{D}_1$ and $\mathcal{D}_2$. In both cases we are interested in the correlation between columns (attributes) X and Y. The significance of the correlation result might depend on the overall context of the dataset

Picture from: A. Gionis, H. Mannila, T. Mielikäinen, P. Tsaparas: Assessing data mining results via swap randomization. TKDD 1(3): (2007)

The Modeling Method



Background knowledge

Surprising patterns

List of patterns & statistics

Update

Statistical model
Expresses what you expect given the known patterns

yes

Surprising given model?

no

pattern

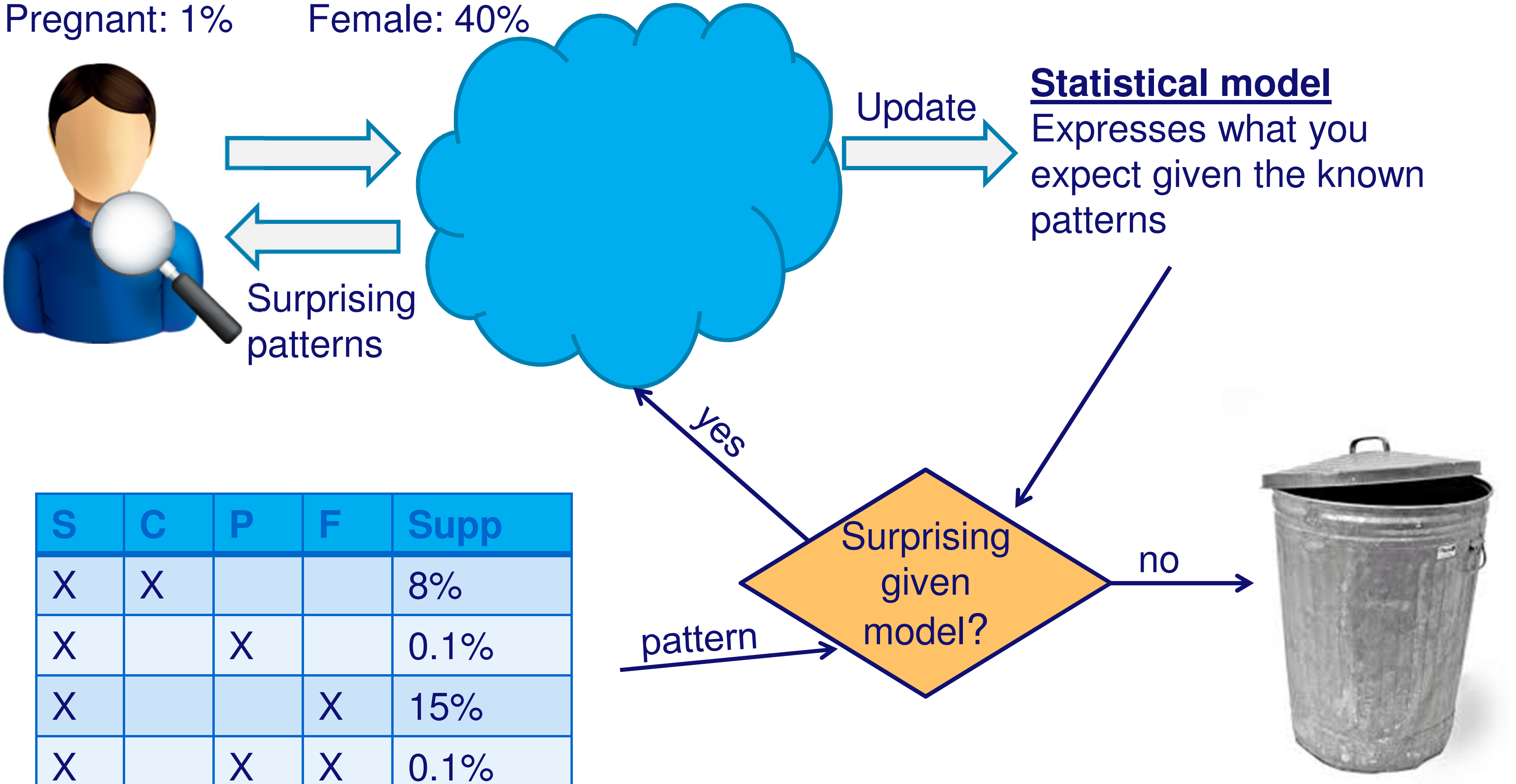
Database Of patterns

Example: Statistical Model

Pregnant => female

Smoking: 20%   Cancer: 10%

Pregnant: 1%   Female: 40%



Surprising patterns

Update

Statistical model
Expresses what you expect given the known patterns

yes

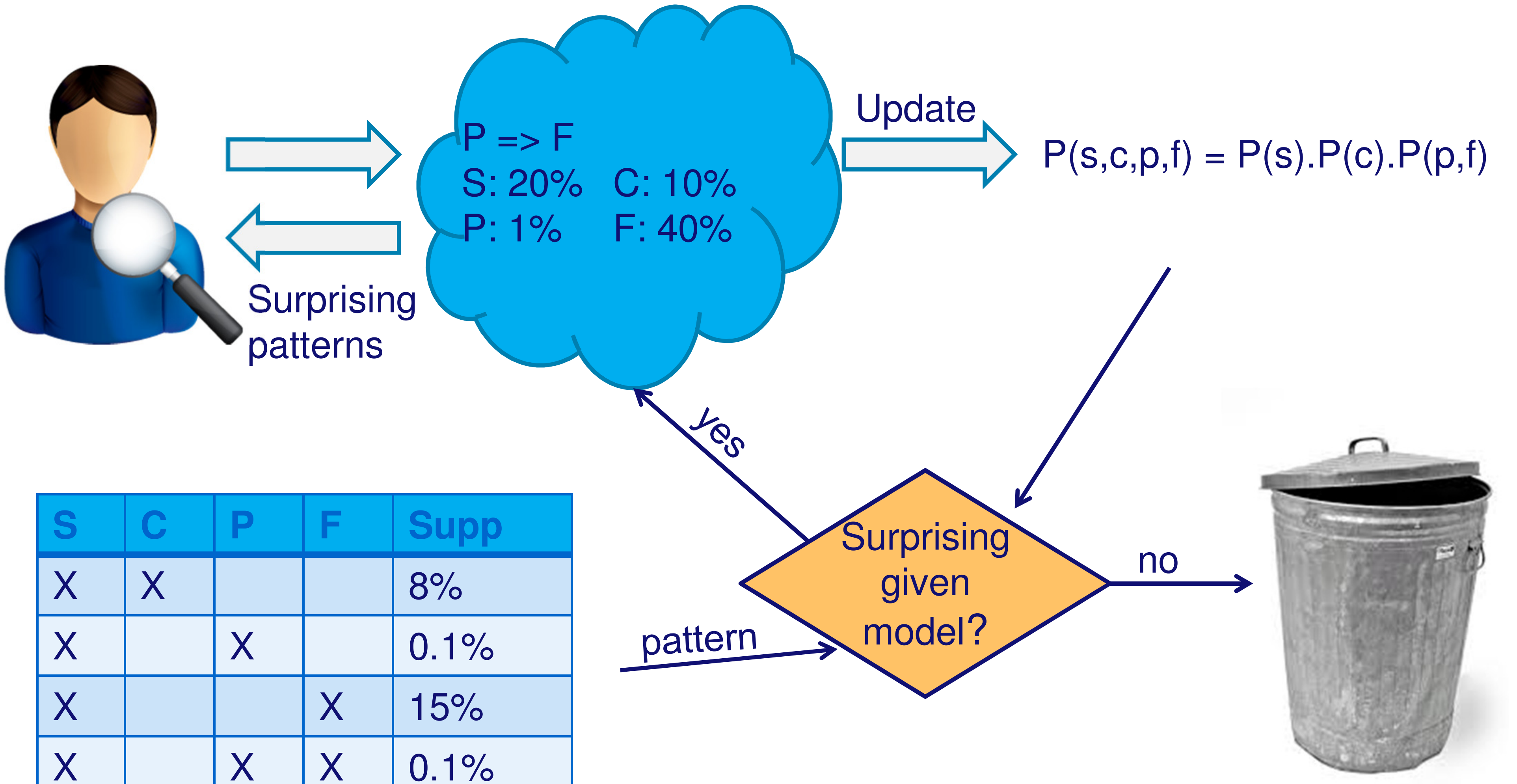
Surprising given model?

no

pattern

S	C	P	F	Supp
X	X			8%
X		X		0.1%
X			X	15%
X		X	X	0.1%
X	X		X	8%

Example: Statistical Model



P => F

S: 20% C: 10%

P: 1% F: 40%

Surprising patterns

Update

P(s,c,p,f) = P(s).P(c).P(p,f)

yes

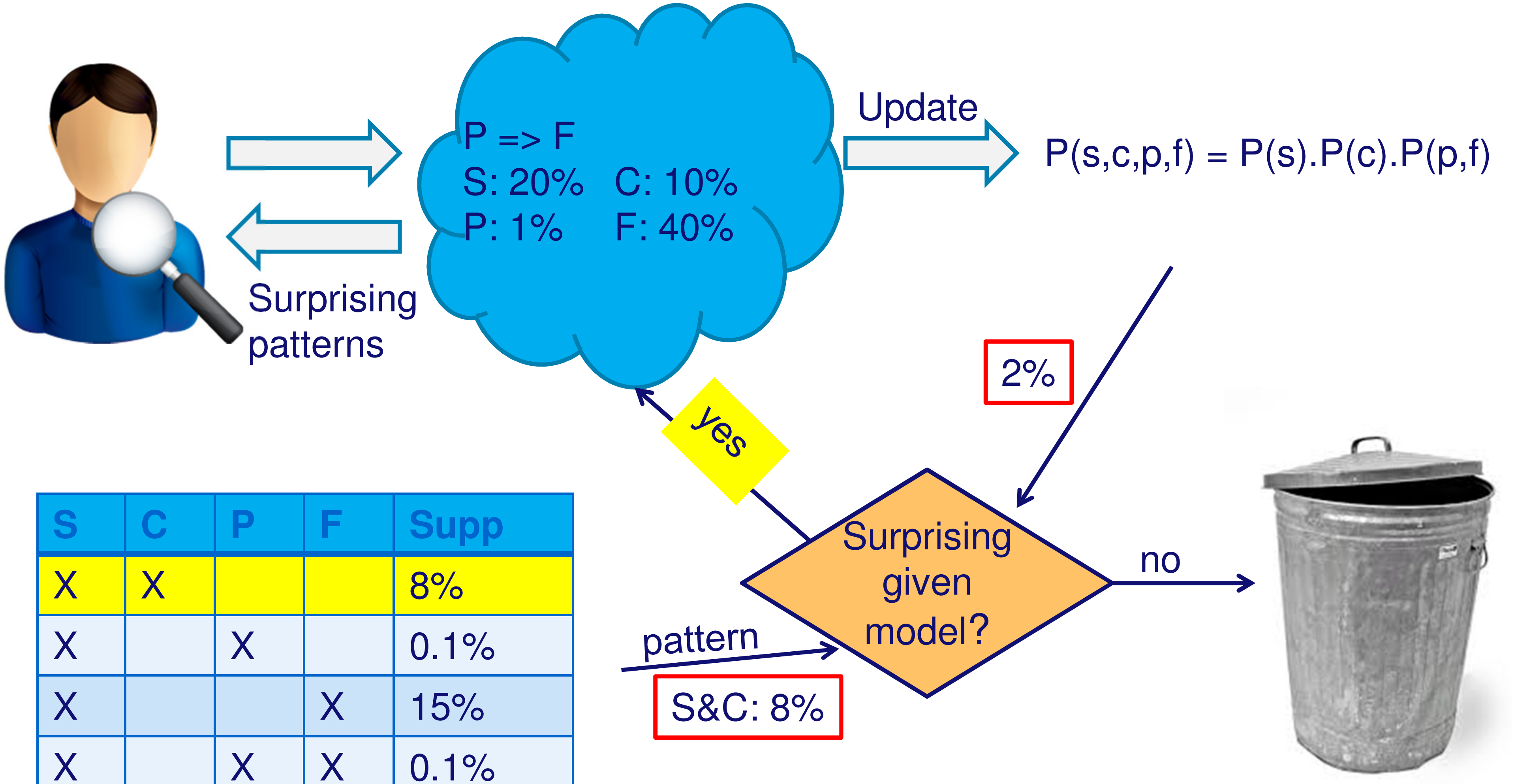
Surprising given model?

no

pattern

S	C	P	F	Supp
X	X			8%
X		X		0.1%
X			X	15%
X		X	X	0.1%
X	X		X	8%

Example: Statistical Model



Surprising patterns

P => F

S: 20% C: 10%

P: 1% F: 40%

Update

$P(s,c,p,f) = P(s).P(c).P(p,f)$

2%

yes

Surprising given model?

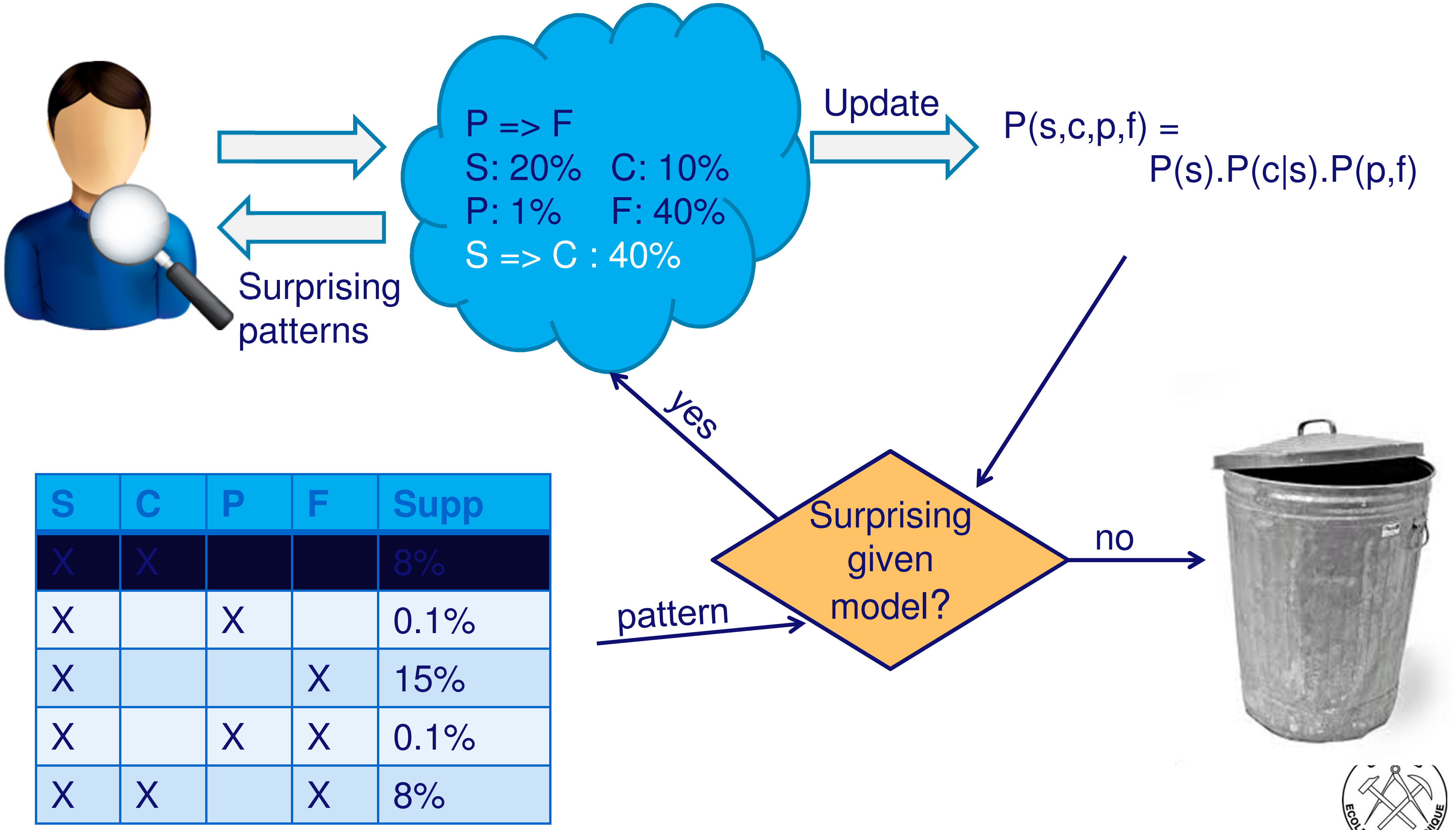
no

pattern

S&C: 8%

S	C	P	F	Supp
X	X			8%
X		X		0.1%
X			X	15%
X		X	X	0.1%
X	X		X	8%

Example: Statistical Model



P => F

S: 20% C: 10%

P: 1% F: 40%

S => C : 40%

Update

$P(s,c,p,f) = P(s).P(c|s).P(p,f)$

Surprising patterns

yes

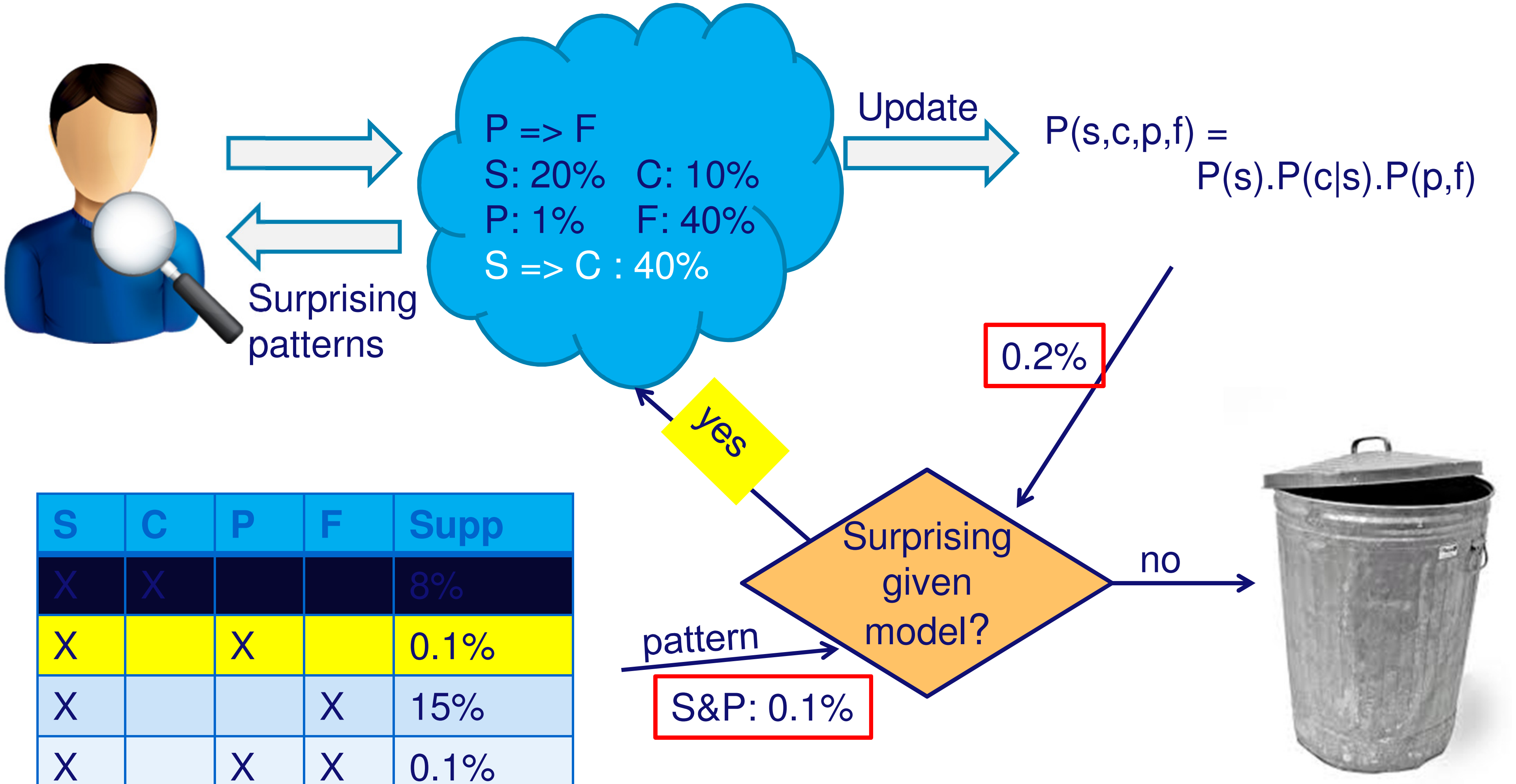
Surprising given model?

no

pattern

S	C	P	F	Supp
X	X			8%
X		X		0.1%
X			X	15%
X		X	X	0.1%
X	X		X	8%

Example: Statistical Model



Surprising patterns

P => F
S: 20% C: 10%
P: 1% F: 40%
S => C : 40%

Update

$P(s,c,p,f) = P(s).P(c|s).P(p,f)$

0.2%

yes

Surprising given model?

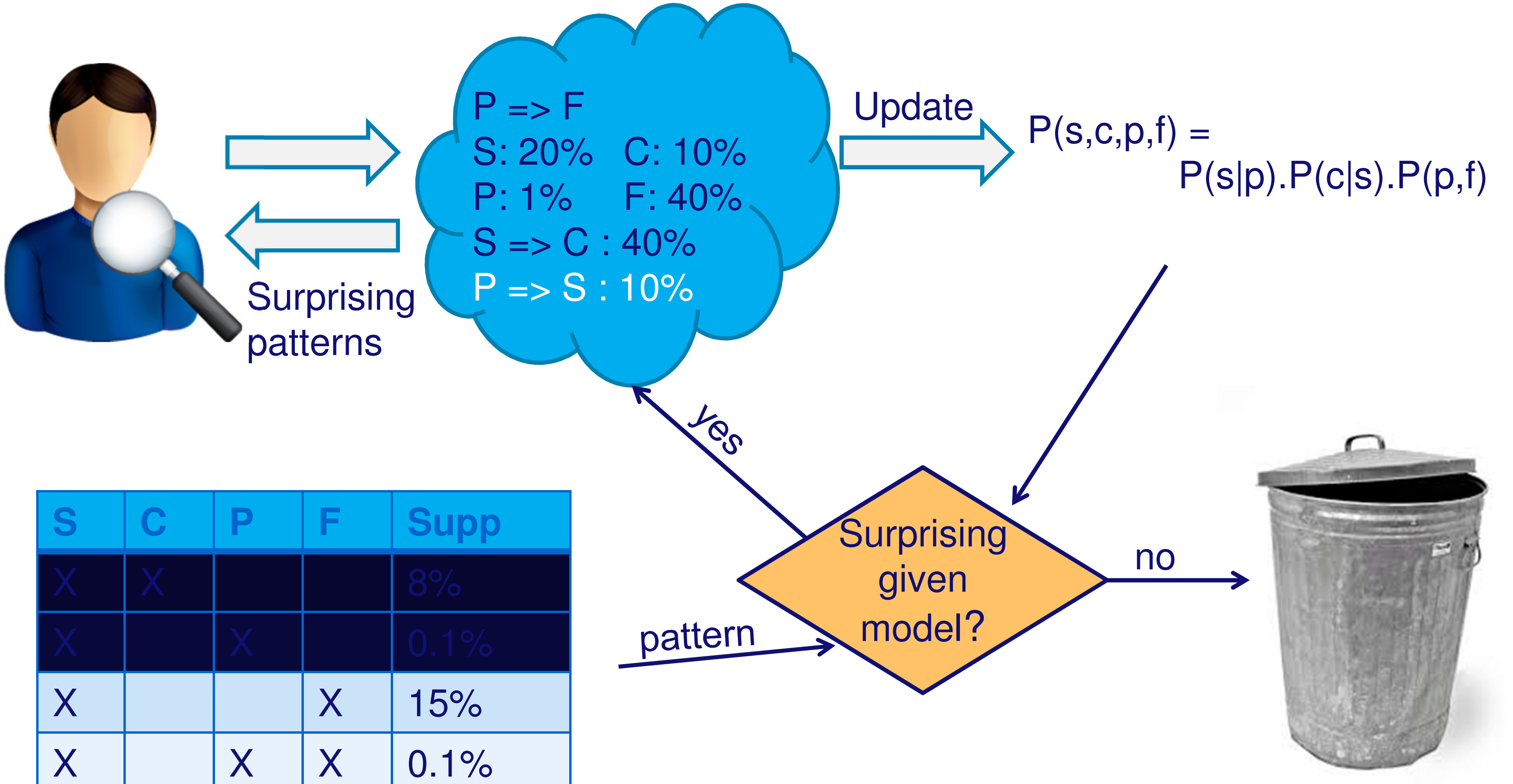
no

pattern

S&P: 0.1%

S	C	P	F	Supp
X	X			8%
X		X		0.1%
X			X	15%
X		X	X	0.1%
X	X		X	8%

Example: Statistical Model

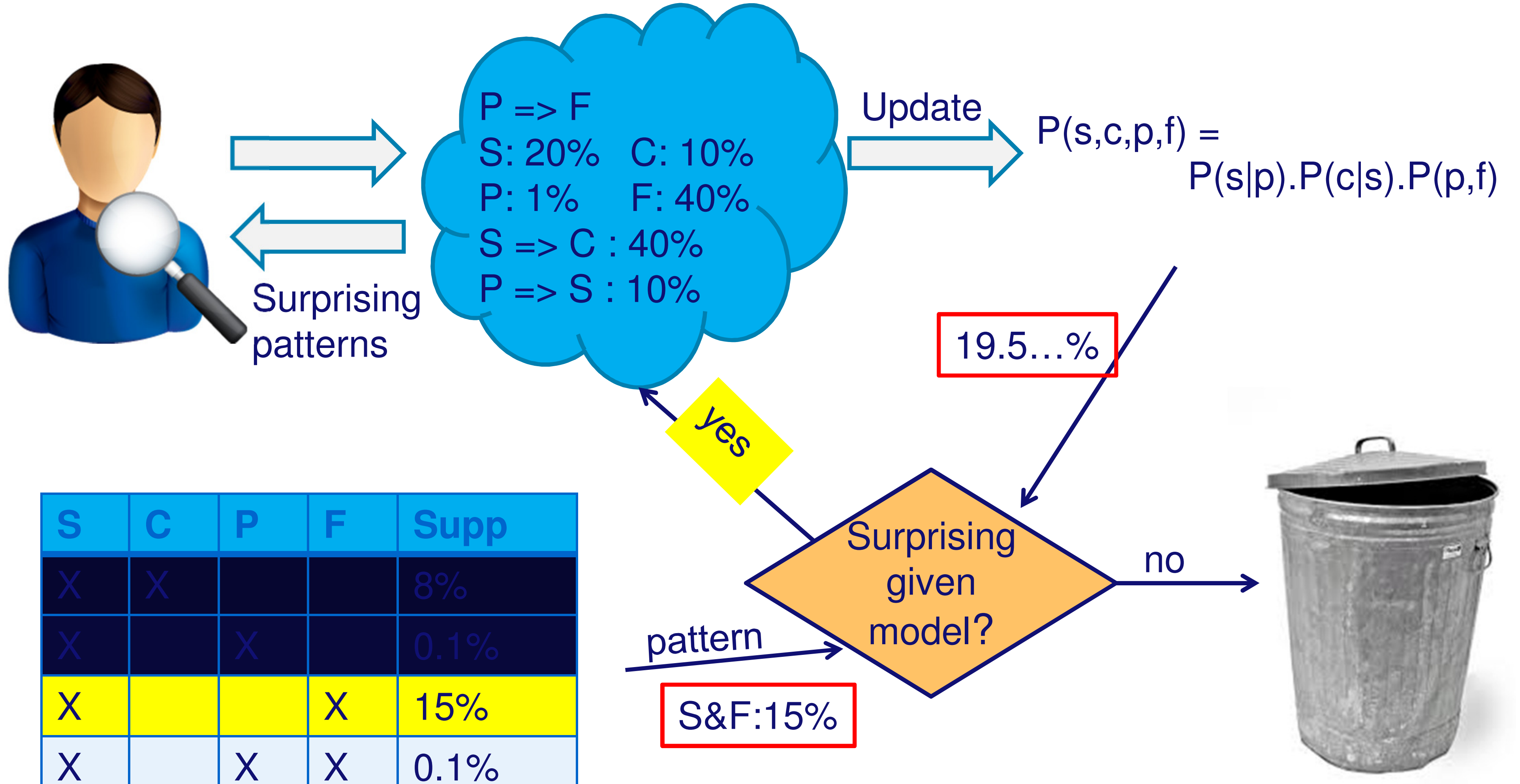


P => F
S: 20%   C: 10%
P: 1%   F: 40%
S => C : 40%
P => S : 10%

$$P(s,c,p,f) = P(s|p).P(c|s).P(p,f)$$

S	C	P	F	Supp
X	X			8%
X		X		0.1%
X			X	15%
X		X	X	0.1%
X	X		X	8%

Example: Statistical Model



Surprising patterns

P => F
S: 20% C: 10%
P: 1% F: 40%
S => C : 40%
P => S : 10%

Update

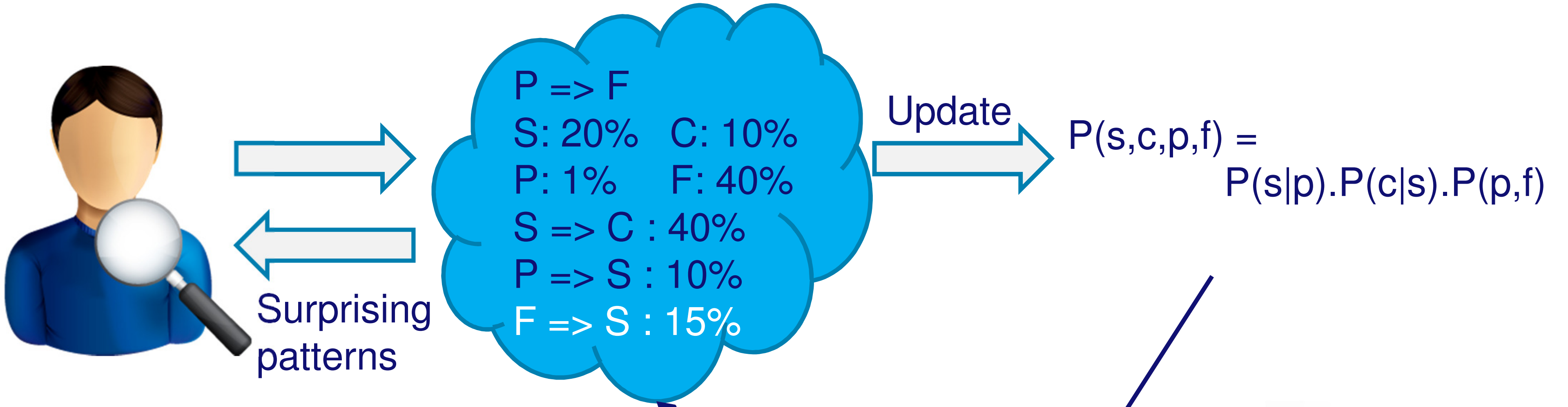
$P(s,c,p,f) = P(s|p).P(c|s).P(p,f)$

19.5…%

S&F:15%

S	C	P	F	Supp
X	X			8%
X		X		0.1%
X			X	15%
X		X	X	0.1%
X	X		X	8%

Example: Statistical Model

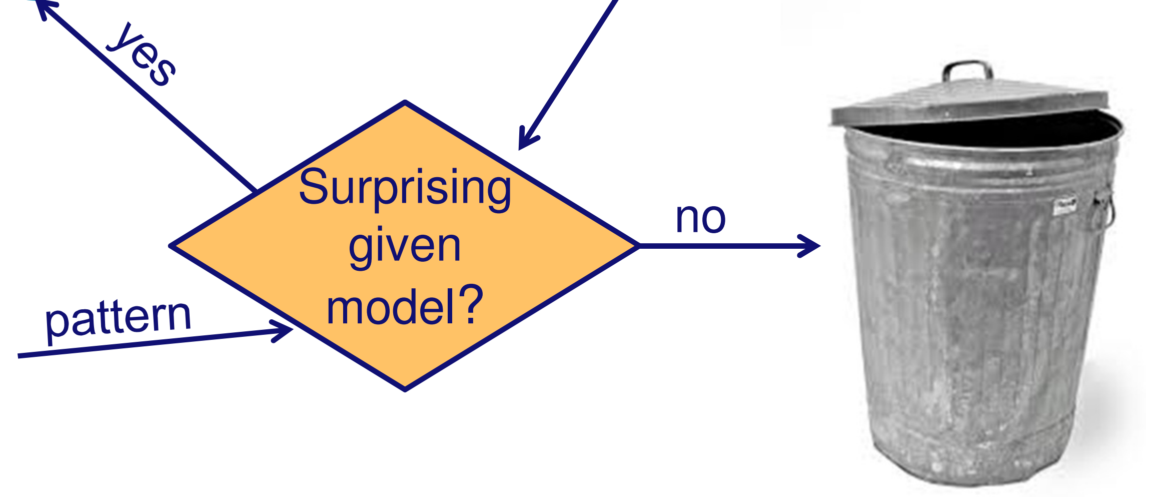


Surprising patterns

P => F
S: 20%   C: 10%
P: 1%   F: 40%
S => C : 40%
P => S : 10%
F => S : 15%

Update

$P(s,c,p,f) = P(s|p).P(c|s).P(p,f)$



yes

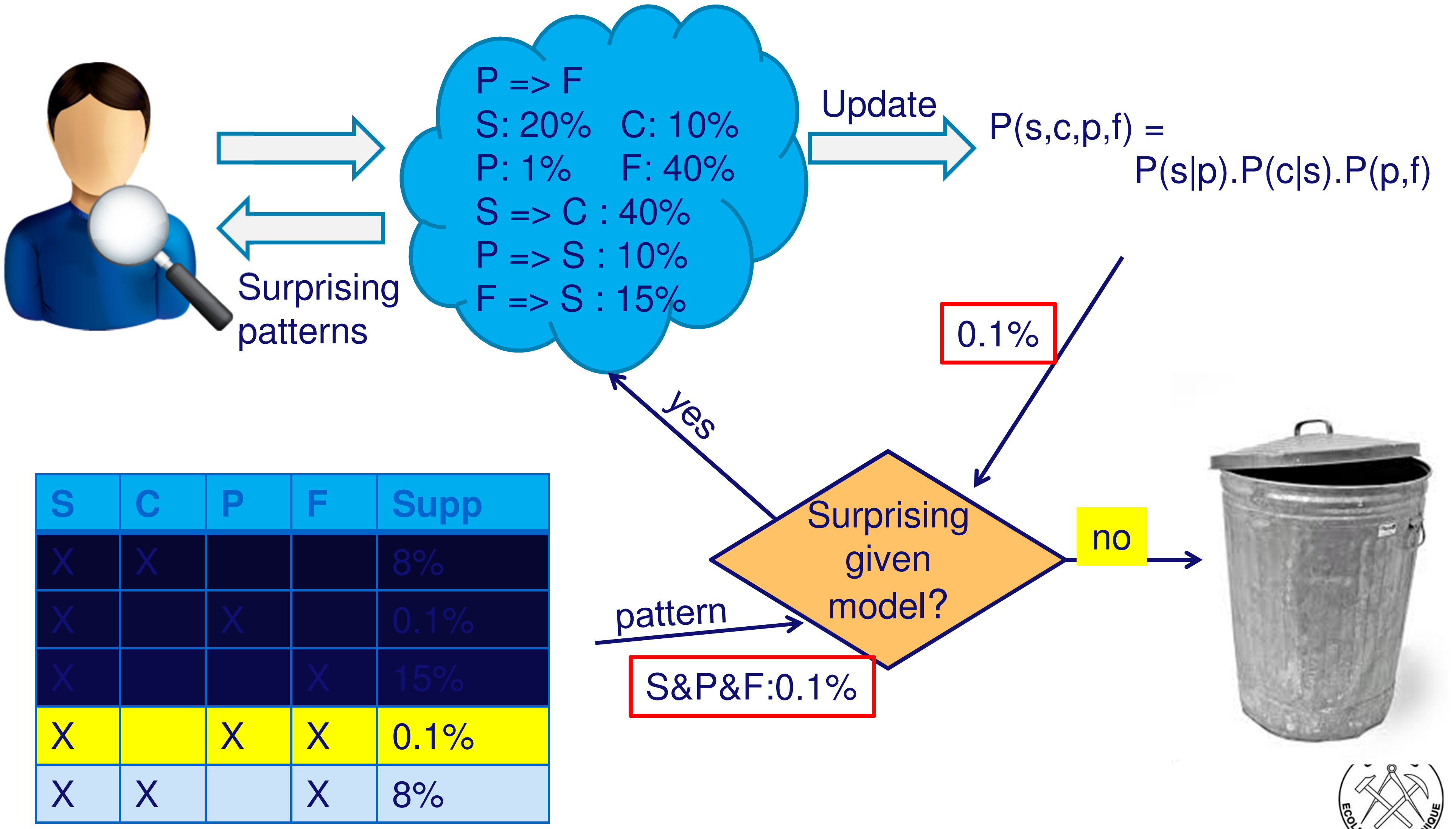
Surprising given model?

pattern

no

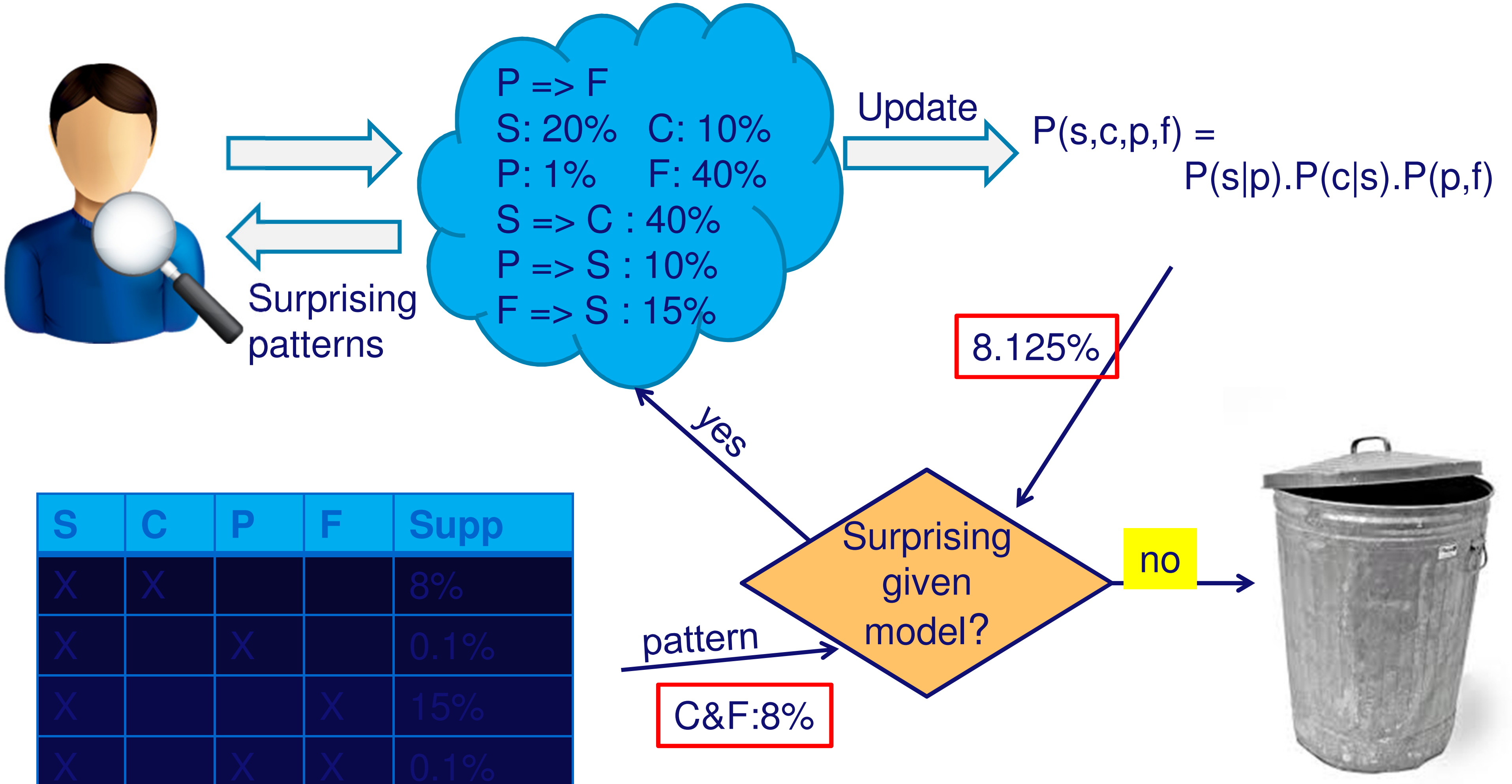
S	C	P	F	Supp
X	X			8%
X		X		0.1%
X			X	15%
X		X	X	0.1%
X	X		X	8%

Example: Statistical Model



S	C	P	F	Supp
X	X			8%
X		X		0.1%
X			X	15%
X		X	X	0.1%
X	X		X	8%

Example: Statistical Model



P => F
S: 20%   C: 10%
P: 1%   F: 40%
S => C : 40%
P => S : 10%
F => S : 15%

Update

$P(s,c,p,f) = P(s|p).P(c|s).P(p,f)$

8.125%

Surprising given model?

yes

no

pattern

C&F:8%

Surprising patterns

S	C	P	F	Supp
X	X			8%
X		X		0.1%
X			X	15%
X		X	X	0.1%
	X		X	8%

MTV – Statistics Based Filter

geo geotagged lat lon
airplane plane flying aircraft
boat ship
city nyc new york
two people
and white black
night exposure long
b w
protest demonstration
airplane flying aviation
san francisco
diamondclassphotographer flickrdiamond

Tell me what I need to know: Succinctly summarizing data with itemsets. *Michael Mampaey, Nikolaj Tatti, and Jilles Vreeken.* In Proceedings of the 17th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), 2011.

Outline

PART I: Frequent itemset mining

- Definition & Applications
- Algorithms for Frequent Itemset Mining
- Extensions to other pattern types

PART II:

- **Pattern explosion & Redundancy problem**
- **Methods to remove redundancy**
  - **Condensed representations**
  - **Statistical methods**
  - **Minimal Description Length**

Minimal Description Length

- A good model helps us to compress the data and is compact
  - Let L(M) be the description length of the model,
  - Let L(D|M) be the size of the data when compressed by the model

- Find set of patterns (model M) that minimizes:

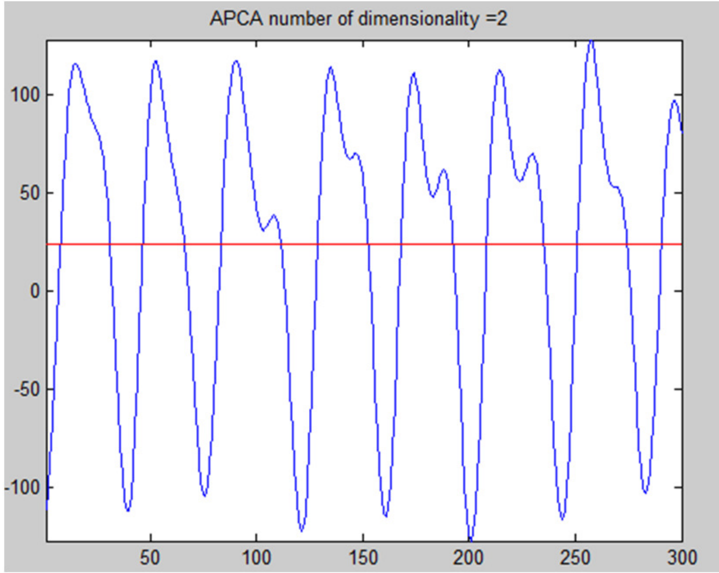
  L(M) + L(D|M)

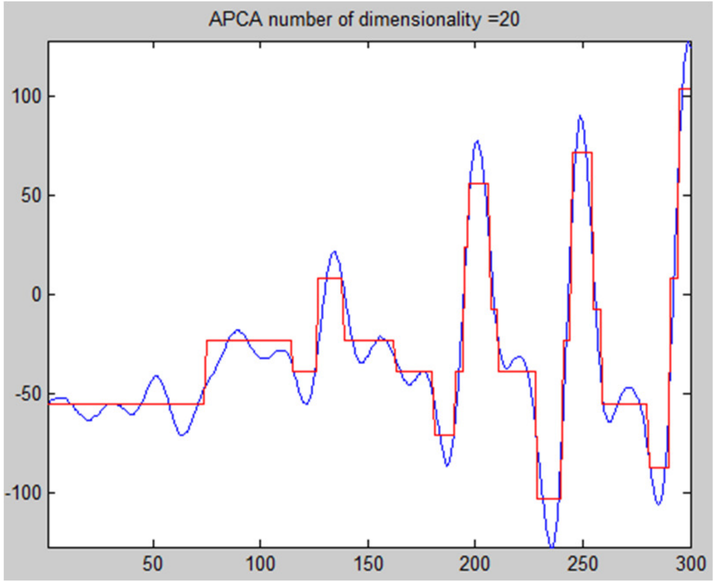
- Explicit trade-off; making a model more specific:
  - Increases L(M),
  - Decreases L(D|M)

Skip app

Minimal Description Length: Example

- Determining the intrinsic cardinality of a time series





- More segments will make the model more accurate
  - What is the optimal number of segments?

Minimal Description Length: Example

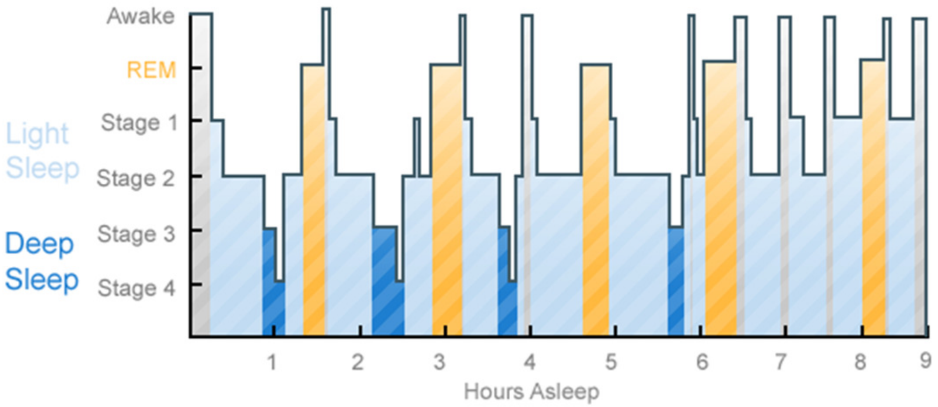
- What is the optimal number of segments?
- Increasing the number of segments
  - Increase model complexity
    = # bits to describe the model = L(M)
  - Decrease residuals
    = less bits for encoding the error = L(D|M)

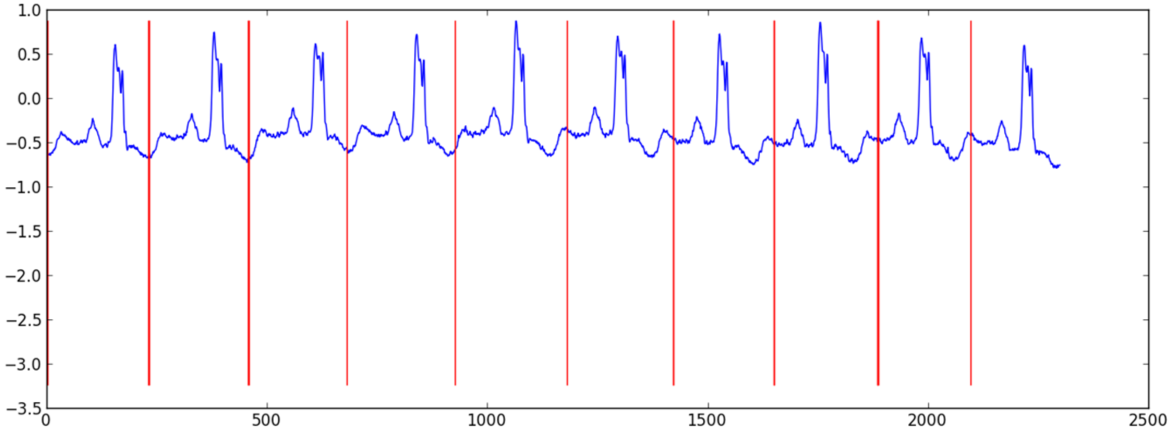
- Optimal point is determined by minimizing
  L(M) + L(D|M)

- L(M) + L(D|M) = amount of structure that can be exploited *usefully*

Application: Deep Sleep Prediction

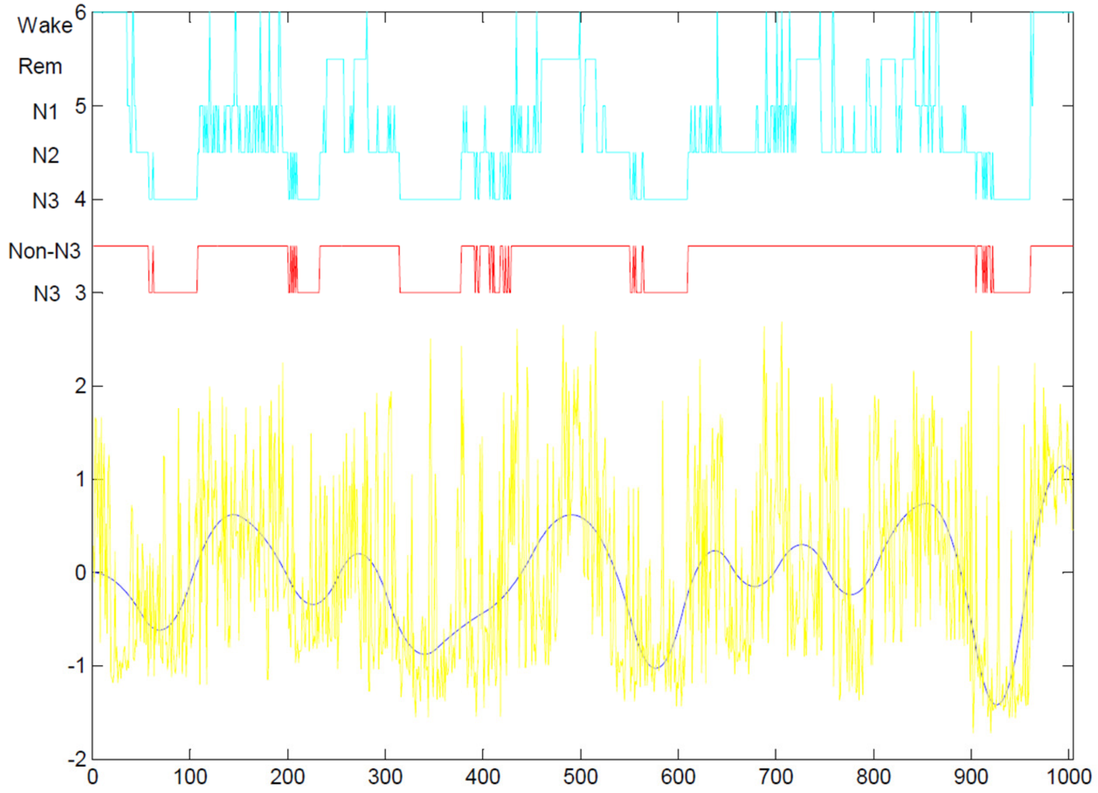
- **Based on ECG data predict if patient in deep sleep**
  - **Less intrusive than EEG**





A First Result

- **Use L(M) + L(D|M) to characterize regularity of the sequence**
  - **Window slides over the ECG; continuously compute L(M)+L(D|M) for the best model**



Stage	Mean
Wake	0.5119
REM	0.4596
N1	0.4700
N2	0.3256
N3	0.2053

Minimal Description Length

- **We can use patterns to code a database**

D

TID	Items
1	A
2	C
3	C
4	A,B
5	A,B,C
6	A,B,C

L(M)

pattern	code
A	00
B	01
C	10
AB	11

L(D\|M)

TID	Items
1	00
2	10
3	10
4	11
5	1110
6	1110

- **Find set of patterns that minimizes L(M)+L(D|M)**
  - **Heuristic approach**

Minimal Description Length

- **Rank itemsets according to how well they can be used to compress the dataset**
  - **Property of a set of patterns**

- **The "Krimp" algorithm was the first to use this paradigm in itemset mining**
  - **Assumes a seed set of patterns**
  - **A subset of these patterns is selected to form the "code book"**
  - **The best codebook is the one that gives the best compression**

Vreeken, Jilles, Matthijs Van Leeuwen, and Arno Siebes. "Krimp: mining itemsets that compress." *Data Mining and Knowledge Discovery* 23.1 (2011): 169-214

Tags Dataset - MDL

- jet landing gear airliner jetliner jetliners planes aeroplanes engines aircrafts airliners les avions tail motors cockpit fuselage flaps rudder aeroplano vliegtuig avi
- boat ship
- geo geotagged lat lon
- http library gov congress loc identifier hdl pnp purl elements
- airplane flying
- two people
- bridges bridgepix bridgepixing bridging
- photograph d set slr close nikonstunninggallery camera heigan martin mh
- jets aircraft airplanes aeroplane avion wings nose flugzeug
- nyc new york
- b w
- white black
- protest demonstration
- exposure long
- emergency fire truck vehicle

Summary: Redundancy problem

- Output of frequent set mining not useful in itself
  - Lots of redundant patterns
- Methods to remove redundancy
  - Element of “surprise”
  - Statistical: model expectation
  - MDL: how much structure can be exploited efficiently
- Mainly aimed towards summarization
  - Although also applications in change detection

Summary

- Frequent itemset mining
  - Simple definition, high complexity
  - Breadth-first and Depth-first algorithms
  - Many extensions to other pattern types

- Pattern explosion problem
  - Too many, redundant patterns are generated
  - Condensed representations → subset of all patterns
    - “Combinatorial” approach insufficient

- Recently new techniques emerged
  - statistically and MDL based
  - Model expectation / benefit of a set of patterns

Literature for Basics Frequent Pattern Mining

Dualize and Advance:

Dimitrios Gunopulos, Roni Khardon, Heikki Mannila, Hannu Toivonen: Data mining, Hypergraph Transversals, and Machine Learning. PODS 1997: 209-216

Frequent itemset mining definition:

Rakesh Agrawal, Tomasz Imielinski, Arun N. Swami: Mining Association Rules between Sets of Items in Large Databases. SIGMOD Conference 1993: 207-216

Apriori:

Rakesh Agrawal, Ramakrishnan Srikant: Fast Algorithms for Mining Association Rules in Large Databases. VLDB 1994: 487-499

FPGrowth:

Jiawei Han, Jian Pei, Yiwen Yin: Mining Frequent Patterns without Candidate Generation. SIGMOD Conference 2000: 1-12

Literature for Pattern Explosion

FIMI competition:

Roberto J. Bayardo Jr., Bart Goethals, Mohammed Javeed Zaki: FIMI '04, Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, Brighton, UK, November 1, 2004 CEUR-WS.org 2004

Closed Itemsets:

Nicolas Pasquier, Yves Bastide, Rafik Taouil, Lotfi Lakhal: Discovering Frequent Closed Itemsets for Association Rules. ICDT 1999: 398-416

Non-Derivable Itemsets:

Toon Calders, Bart Goethals: Non-derivable itemset mining. Data Min. Knowl. Discov. 14(1): 171-206 (2007)

Extending NDI:

Chedy Raïssi, Toon Calders, Pascal Poncelet: Mining conjunctive sequential patterns. Data Min. Knowl. Discov. 17(1): 77-93 (2008)

Reasoning about frequencies:

Toon Calders: The complexity of satisfying constraints on databases of transactions. Acta Inf. 44(7-8): 591-624 (2007)

Toon Calders: Itemset frequency satisfiability: Complexity and axiomatization. Theor. Comput. Sci. 394(1-2): 84-111 (2008)

Literature for Statistical Measures

Swap randomization:

Aristides Gionis, Heikki Mannila, Taneli Mielikäinen, Panayiotis Tsaparas: Assessing data mining results via swap randomization. TKDD 1(3): (2007)

Style “Nikolaj”:

Michael Mampaey, Nikolaj Tatti, Jilles Vreeken: Tell me what i need to know: succinctly summarizing data with itemsets. KDD 2011: 573-581

Style “De Bie”:

Tijl De Bie, Kleanthis-Nikolaos Kontonasios, Eirini Spyropoulou: A framework for mining interesting pattern sets. SIGKDD Explorations 12(2): 92-100 (2010)

Krimp - MDL:

Jilles Vreeken, Matthijs van Leeuwen, Arno Siebes: Krimp: mining itemsets that compress. Data Min. Knowl. Discov. 23(1): 169-214 (2011)

Thank You for Your Attention!