Constraint & Data Mining

Cours2

Master 2 - DS

Nadjib Lazaar
Ing - Phd - HDR - Professor - Paris-Saclay University - LISN - LaHDAK
lazaar@lisn.fr https://perso.lisn.upsaclay.fr/lazaar/

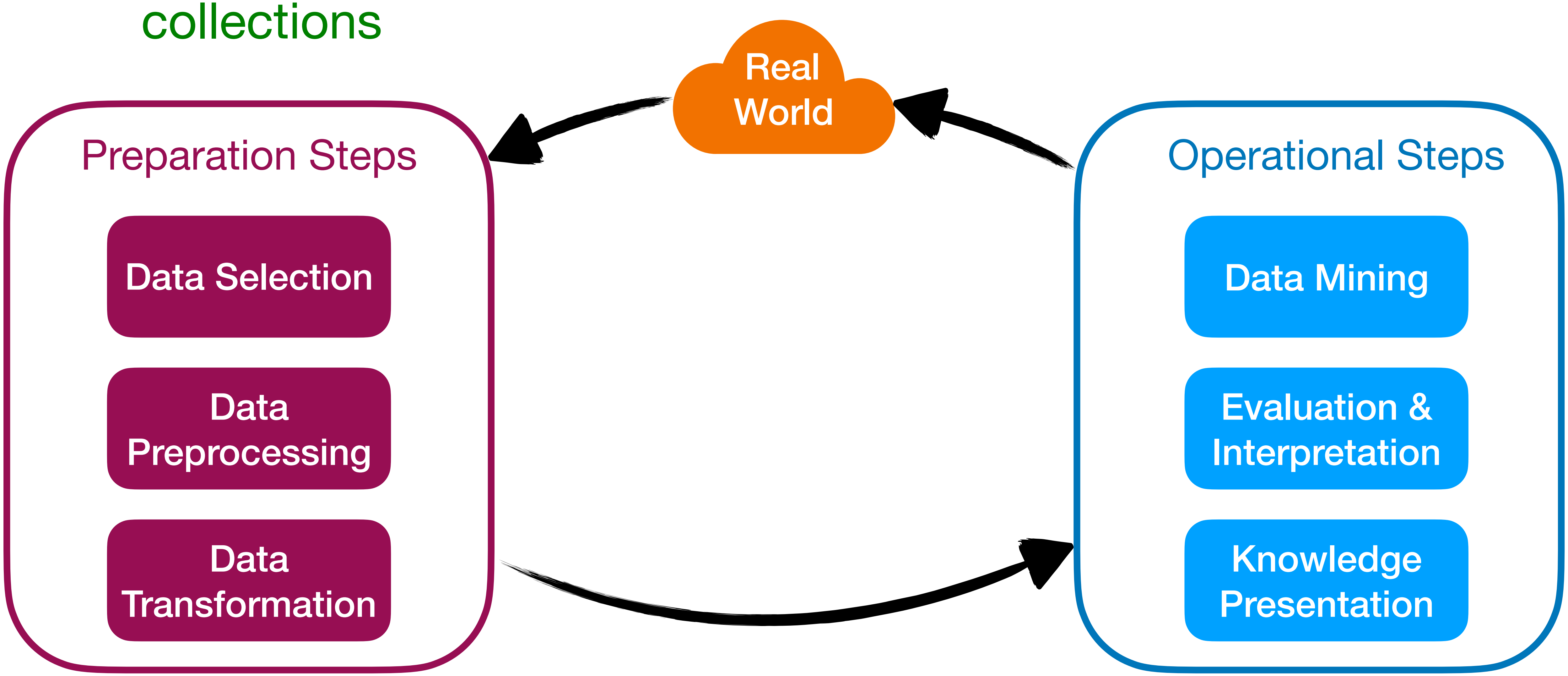
20/01/2025

Frequent Itemset Mining

Knowledge Discovery in Databases (KDD)

Definition and steps

- Knowledge Discovery in Databases (KDD) revolves around the investigation and creation of knowledge, processes, algorithms, and the mechanisms for retrieving potential knowledge from data collections



Frequent Itemset Mining

Motivations

- **Purpose:** A key method in market basket analysis, aimed at identifying patterns in customer purchasing behavior
- **Goal:** To discover frequently co-occurring products, which are items that customers often buy together
- **Applications of Frequent Itemsets:**
  - **Product Arrangement:** Optimize product placement on store shelves or within catalogs
  - **Cross-Selling:** Suggest related products to customers (e.g., recommending additional products during online shopping)
  - **Product Bundling:** Offer product bundles based on co-purchase patterns
  - **Other Applications:** Fraud detection, dependency analysis, fault localization, and more

Frequent Itemset Mining

Basic Notions

- Items: $I = \{p_1, \ldots, p_n\}$
- Itemset, transaction: $P, T \subseteq I$
- Transactional dataset: $D = \{T_1, \ldots, T_m\}$
- Language of itemsets: $\mathscr{L}_I = 2^I$
- Cover of an itemset: $cover(P) = \{T_i \in D : \; P \subseteq T_i\}$
- Absolute Frequency: $freq(P) = |cover(P)|$
- Relative Frequency: $freq(P) = \dfrac{|cover(P)|}{|D|}$

Frequent Itemset Mining

Problem Definition

- **Given:**
  - A set of items $I = \{p_1, \ldots, p_n\}$
  - A transactional dataset $D = \{T_1, \ldots, T_m\}$
  - A minimum support α
- **The need:**
  - The set of itemset P s.t.: $freq(P) \geq \alpha$

Frequent Itemset Mining

Example(1)

- $I = \{a, b, c, d, e\}$
- $D = \{T_1, \ldots, T_{10}\}$

$cover(bc) = ?$

$freq(bc) = ?$

H_D

1:	a			d	e
2:		b	c	d	
3:	a		c		e
4:	a		c	d	e
5:	a				e
6:	a		c	d	
7:		b	c		
8:	a		c	d	e
9:		b	c		e
10:	a			d	E

V_D

a	b	c	d	e
1	2	2	1	1
3	7	3	2	3
4	9	4	4	4
5		6	6	5
6		7	8	8
8		8	10	9
10		9		10

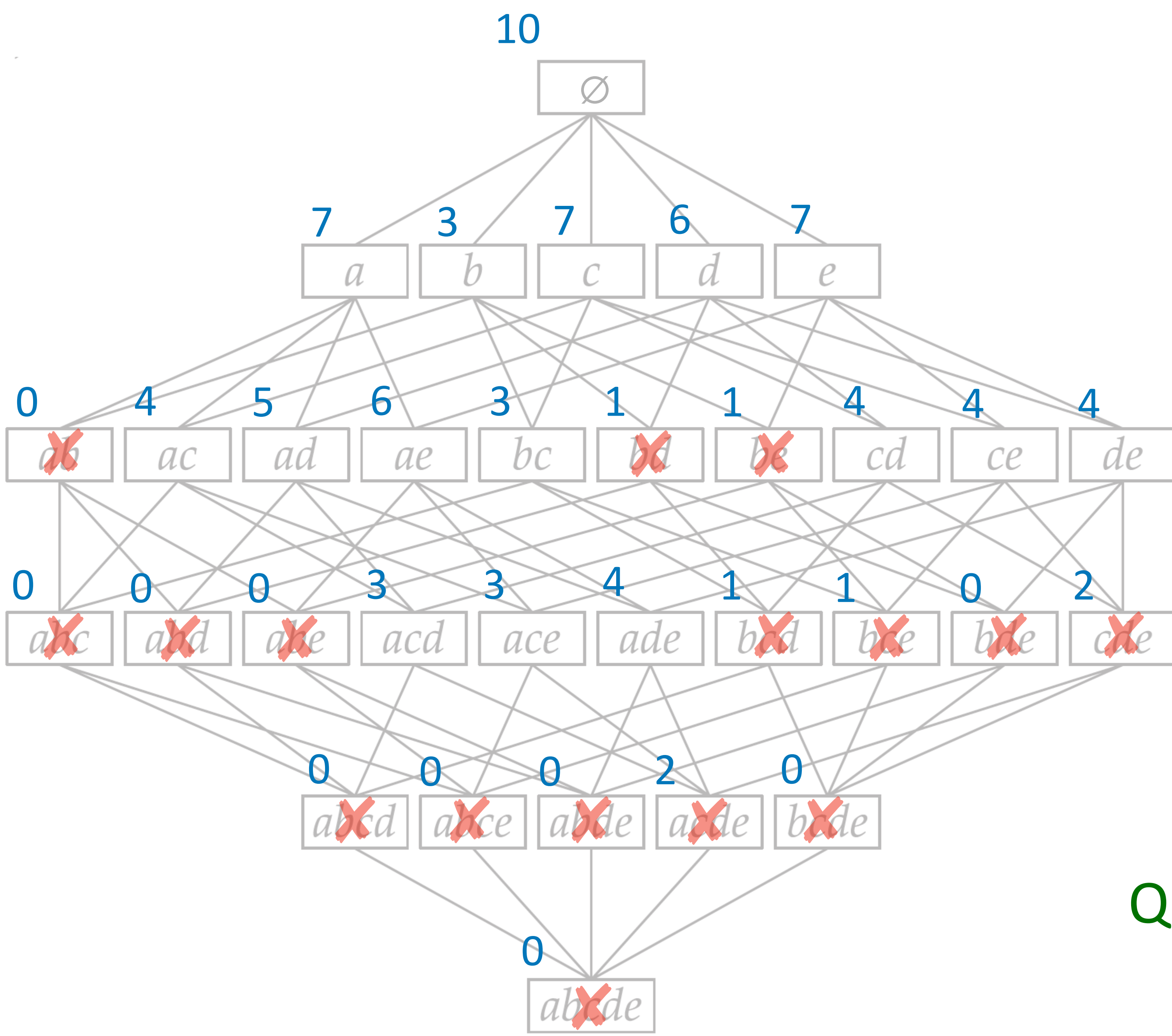
M_D

	a	b	c	d	e
1:	1	0	0	1	1
2:	0	1	1	1	0
3:	1	0	1	0	1
4:	1	0	1	1	1
5:	1	0	0	0	1
6:	1	0	1	1	0
7:	0	1	1	0	0
8:	1	0	1	1	1
9:	0	1	1	0	1
10:	1	0	0	1	1

Frequent Itemset Mining

Example(1)

Query-2: Frequent itemset with minimum support $\alpha = 3$?



M_D

	a	b	c	d	e
1:	1	0	0	1	1
2:	0	1	1	1	0
3:	1	0	1	0	1
4:	1	0	1	1	1
5:	1	0	0	0	1
6:	1	0	1	1	0
7:	0	1	1	0	0
8:	1	0	1	1	1
9:	0	1	1	0	1
10:	1	0	0	1	1

Query-1: Frequent itemset?

Searching for Frequent Itemsets

Naïve Search

- A naïve search that consists of enumerating and testing the frequency of itemset candidates in a given dataset is usually infeasible

Number of items (n)	Search space (2^n)
10	$\approx 10^3$
20	$\approx 10^6$
30	$\approx 10^9$
100	$\approx 10^{30}$
128	$\approx 10^{68}$ (atoms in the universe)
1000	$\approx 10^{301}$

Searching for Frequent Itemsets

Anti-Monotonicity Property

- Given a transaction database D over items I and two itemsets P and Q:

$$Q \subseteq P \Rightarrow cover(P) \subseteq cover(Q)$$

- That is,

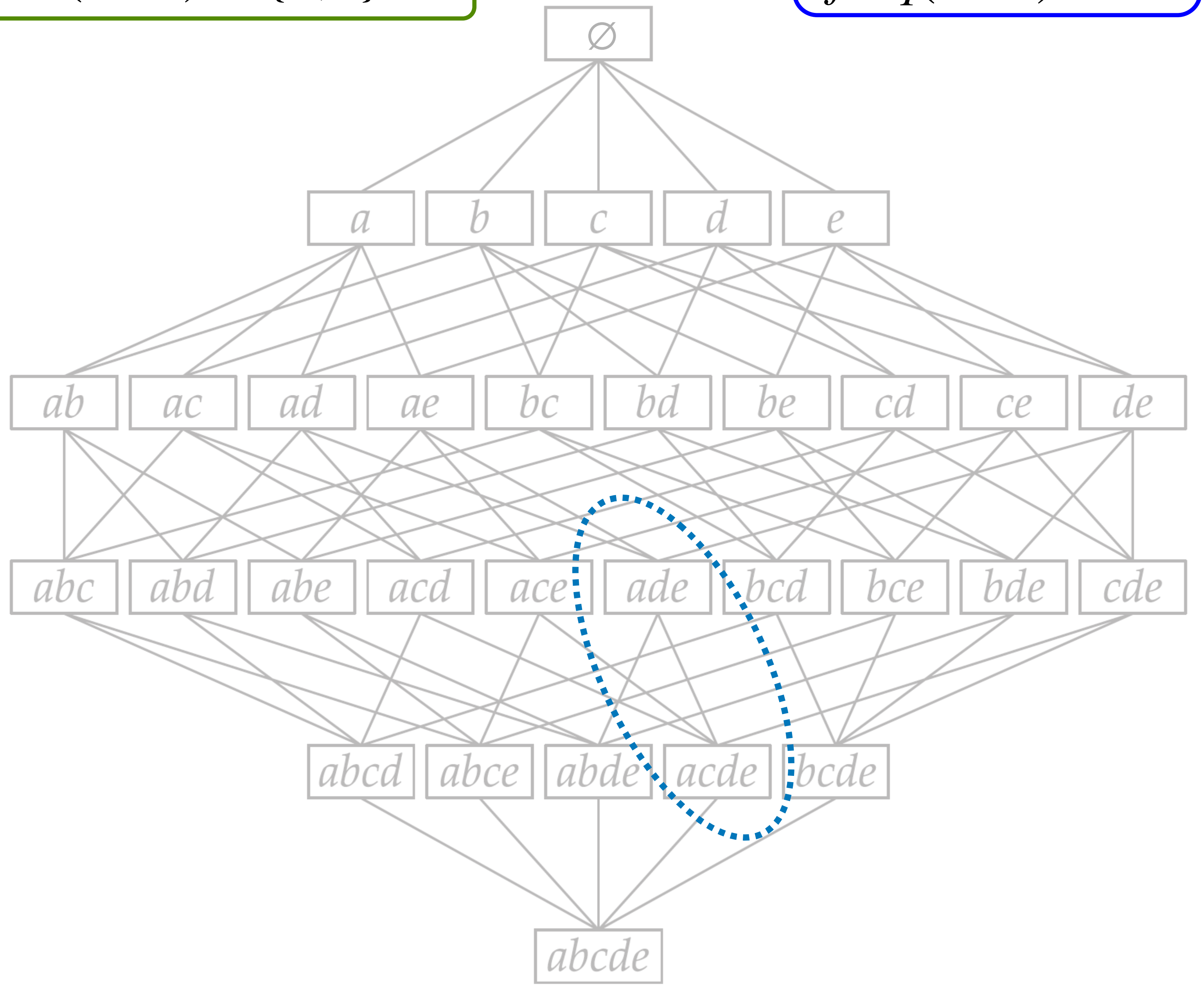
$$Q \subseteq P \Rightarrow freq(P) \leq freq(Q)$$

Anti-Monotonicity Property

Example(2)

$cover(ade) = \{1,4,8,10\}$
$cover(acde) = \{4,8\}$

$freq(ade) = 4$
$freq(acde) = 2$



M_D

	a	b	c	d	e
1:	1	0	0	1	1
2:	0	1	1	1	0
3:	1	0	1	0	1
4:	1	0	1	1	1
5:	1	0	0	0	1
6:	1	0	1	1	0
7:	0	1	1	0	0
8:	1	0	1	1	1
9:	0	1	1	0	1
10:	1	0	0	1	1

Searching for Frequent Itemsets

Apriori Property

- Given a transaction database D over items I, a minsup α and two itemsets P and Q:

$$Q \subseteq P \Rightarrow freq(P) \leq freq(Q)$$

- It follows: $Q \subseteq P \wedge freq(P) \geq \alpha \Rightarrow freq(Q) \geq \alpha$

All subsets of a frequent itemset are frequent!

- Contraposition: $Q \subseteq P \wedge freq(Q) < \alpha \Rightarrow freq(P) < \alpha$

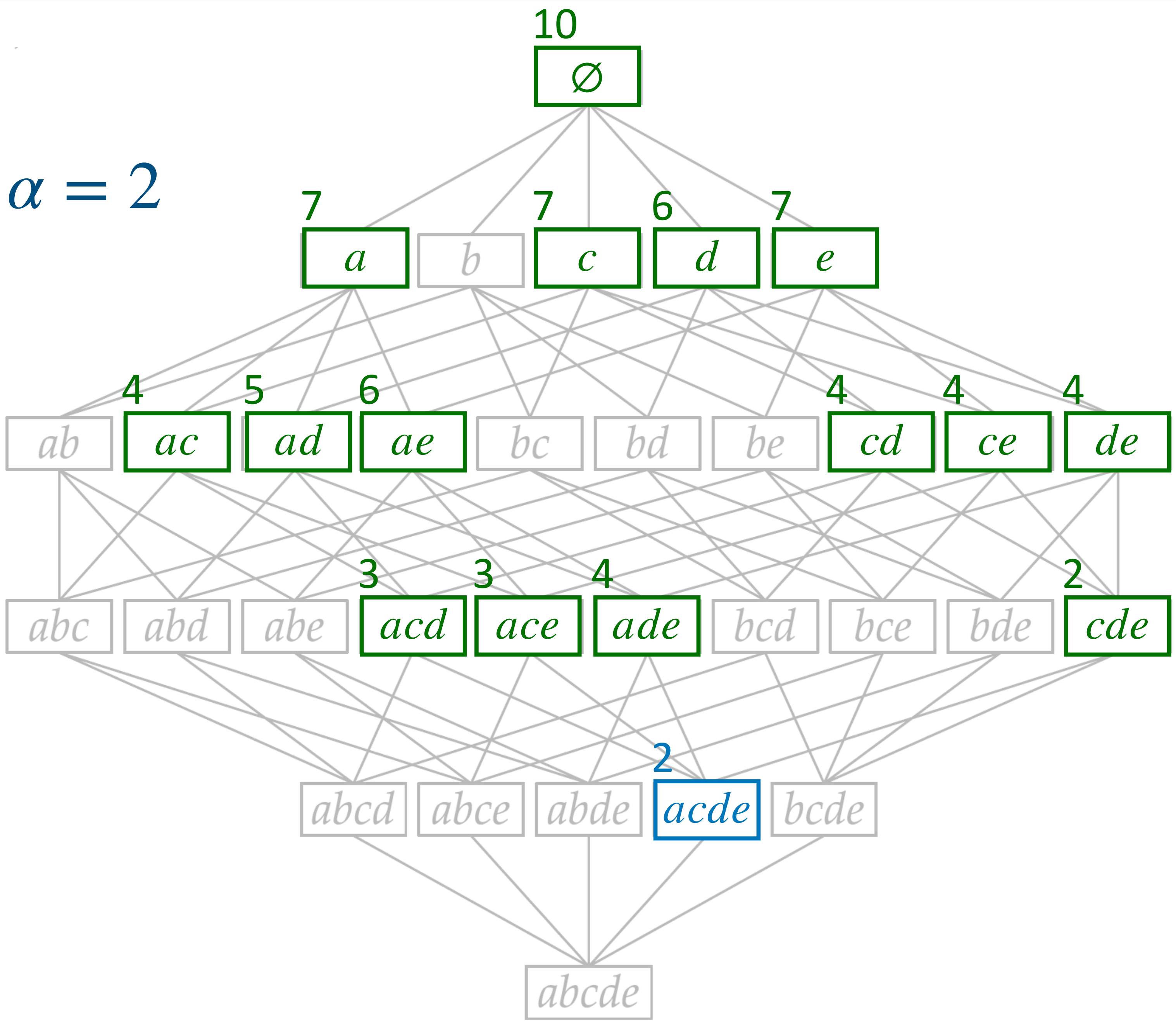
All supersets of an infrequent itemset are infrequent!

Apriori Property

Example(3)

All subsets of a frequent itemset are frequent!

$\alpha = 2$



M_D

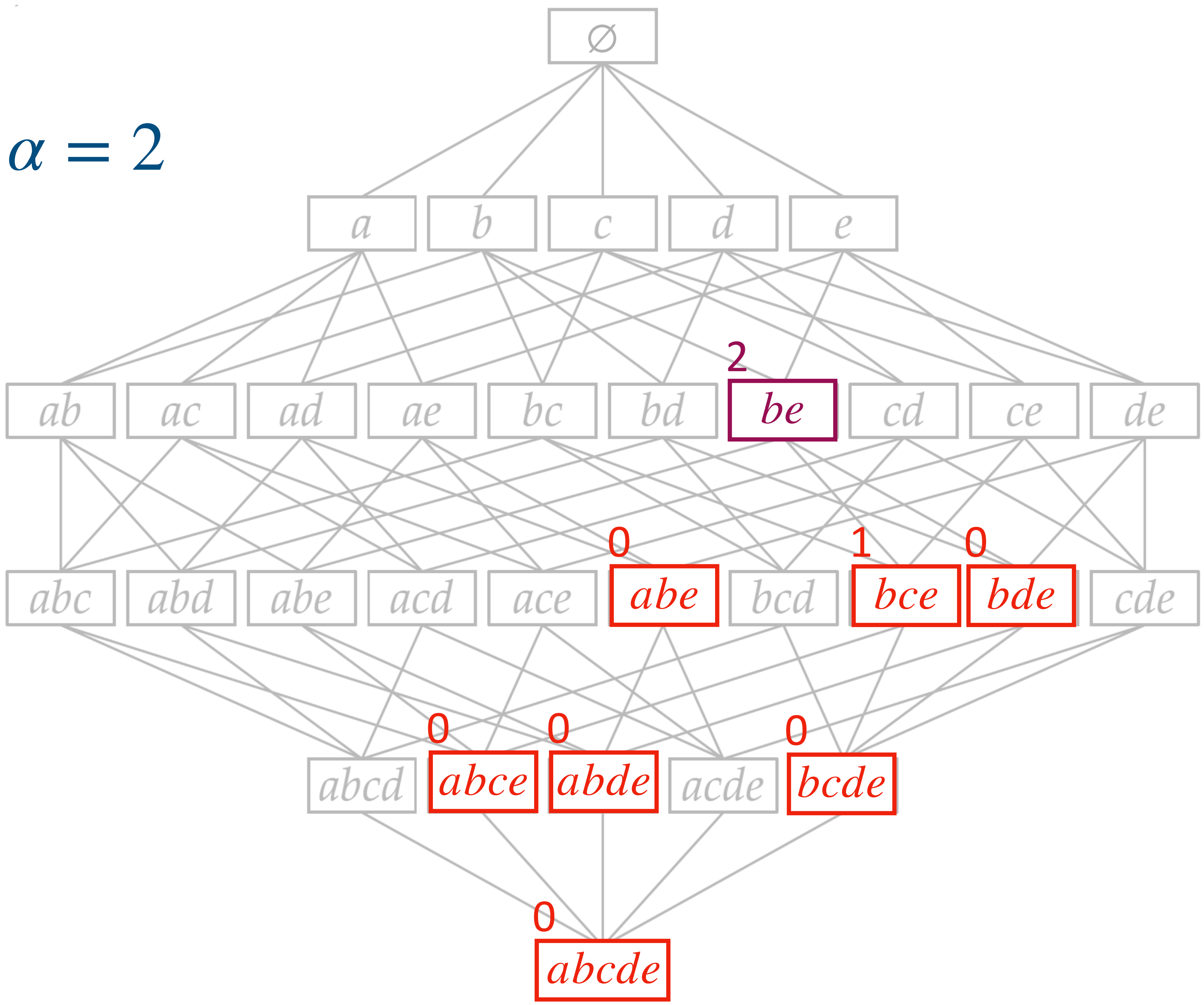
	a	b	c	d	e
1:	1	0	0	1	1
2:	0	1	1	1	0
3:	1	0	1	0	1
4:	1	0	1	1	1
5:	1	0	0	0	1
6:	1	0	1	1	0
7:	0	1	1	0	0
8:	1	0	1	1	1
9:	0	1	1	0	1
10:	1	0	0	1	1

Apriori Property

Example(3)

All supersets of an infrequent itemset are infrequent!

$\alpha = 2$



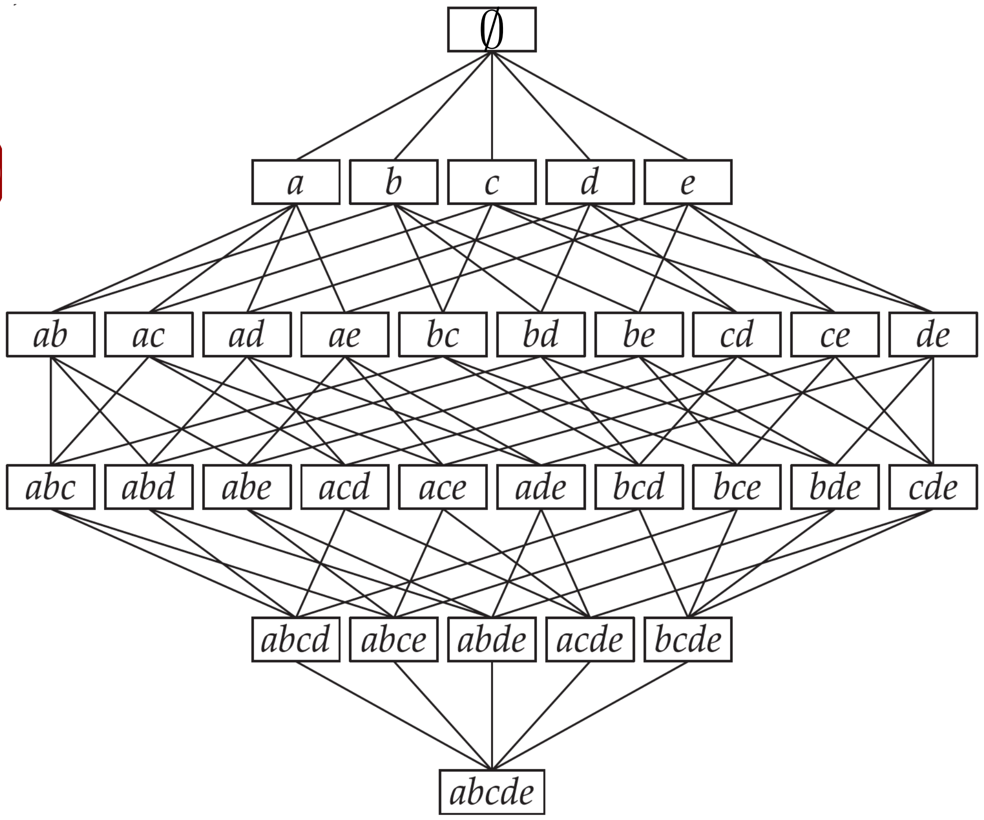
M_D

	a	b	c	d	e
1:	1	0	0	1	1
2:	0	1	1	1	0
3:	1	0	1	0	1
4:	1	0	1	1	1
5:	1	0	0	0	1
6:	1	0	1	1	0
7:	0	1	1	0	0
8:	1	0	1	1	1
9:	0	1	1	0	1
10:	1	0	0	1	1

Searching for Frequent Itemsets

Poset $(2^I, \subseteq\ )$

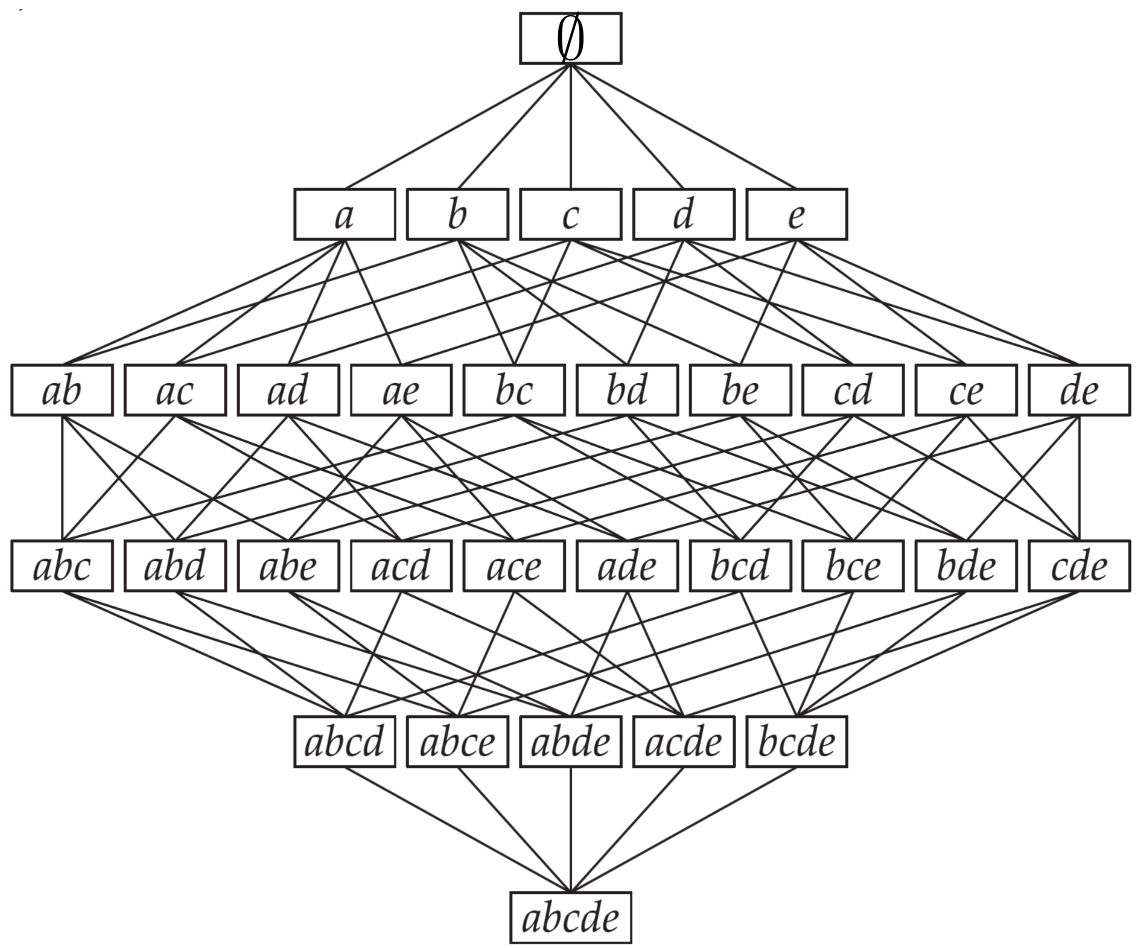
- Test A partial order is a binary relation R over a set S :
- $\forall x, y, z \in S$
- $x\ R\ x$ (reflexivity)
- $x\ R\ y \wedge y\ R\ x \Rightarrow x = y$ (anti-symmetry)
- $x\ R\ y \wedge y\ R\ z \Rightarrow x\ R\ z$ (transitivity)
- What are S and R in Itemset Mining?



Searching for Frequent Itemsets

Partially ordered sets (review)

- Comparable itemsets: $x \subseteq y \vee y \subseteq x$
- Incomparable itemsets: $x \nsubseteq y \wedge y \nsubseteq x$



Searching for Frequent Itemsets

Apriori Algorithm [Agrawal and Srikant 1994]

- Determine the support of the one-element item sets (i.e. singletons) and discard the infrequent items
- Form candidate itemsets with two items (both items must be frequent), determine their support, and discard the infrequent itemsets
- Form candidate item sets with three items (all contained pairs must be frequent), determine their support, and discard the infrequent itemsets
- And so on!

Based on candidate generation and pruning

Searching for Frequent Itemsets

Apriori Algorithm [Agrawal and Srikant 1994]

Algorithm 1: Apriori Algorithm

Input: Transaction database $\mathcal{D}$, minimum support threshold α
Output: Frequent itemsets

$k \leftarrow 1$;
$L_k \leftarrow \{p_i \mid p_i \in \mathcal{I} \wedge \mathtt{freq}(p_i) \geq \alpha\}$;
while $L_k \neq \emptyset$ **do**
  $C \leftarrow \mathtt{aprioriGen}(L_k)$;
  $k \leftarrow k + 1$;
  $L_k \leftarrow \{c \mid c \in C \wedge \mathtt{freq}(c) \geq \alpha\}$;
return $\bigcup_i L_i$;

Searching for Frequent Itemsets

Apriori Algorithm [Agrawal and Srikant 1994]

Function aprioriGen(L_k):

- $E \leftarrow \emptyset$;
- **for** *each pair of itemsets* $P', P'' \in L_k$ *such that* $P' = \{p_{i_1}, \ldots, p_{i_{k-1}}, p_{i_k}\}$ *and* $P'' = \{p_{i_1}, \ldots, p_{i_{k-1}}, p_{i'_k}\}$ **do**
  - **if** $p_{i_k} \neq p_{i'_k}$ **then**
    - $P \leftarrow P' \cup P''$;
    - **if** $\forall p_i \in P,\ P \setminus \{p_i\} \in L_k$ **then**
      - $E \leftarrow E \cup \{P\}$;
- **return** E ;

Apriori Algorithm [Agrawal and Srikant 1994]

Improving candidates generation

- Using `aprioriGen` function, an item of $k+1$ size can be generated in a δ possible ways:

$$\delta = \frac{k(k+1)}{2}$$

6 possibilities to generate (abcd)

	abc	abd	acd	bcd
abc	—	abcd	abcd	abcd
abd	abcd	—	abcd	abcd
acd	abcd	abcd	—	abcd
bcd	abcd	abcd	abcd	—

Apriori Algorithm [Agrawal and Srikant 1994]

Improving candidates generation

- Using `aprioriGen` function, an item of $k+1$ size can be generated in a δ possible ways:

$$\delta = \frac{k(k+1)}{2}$$

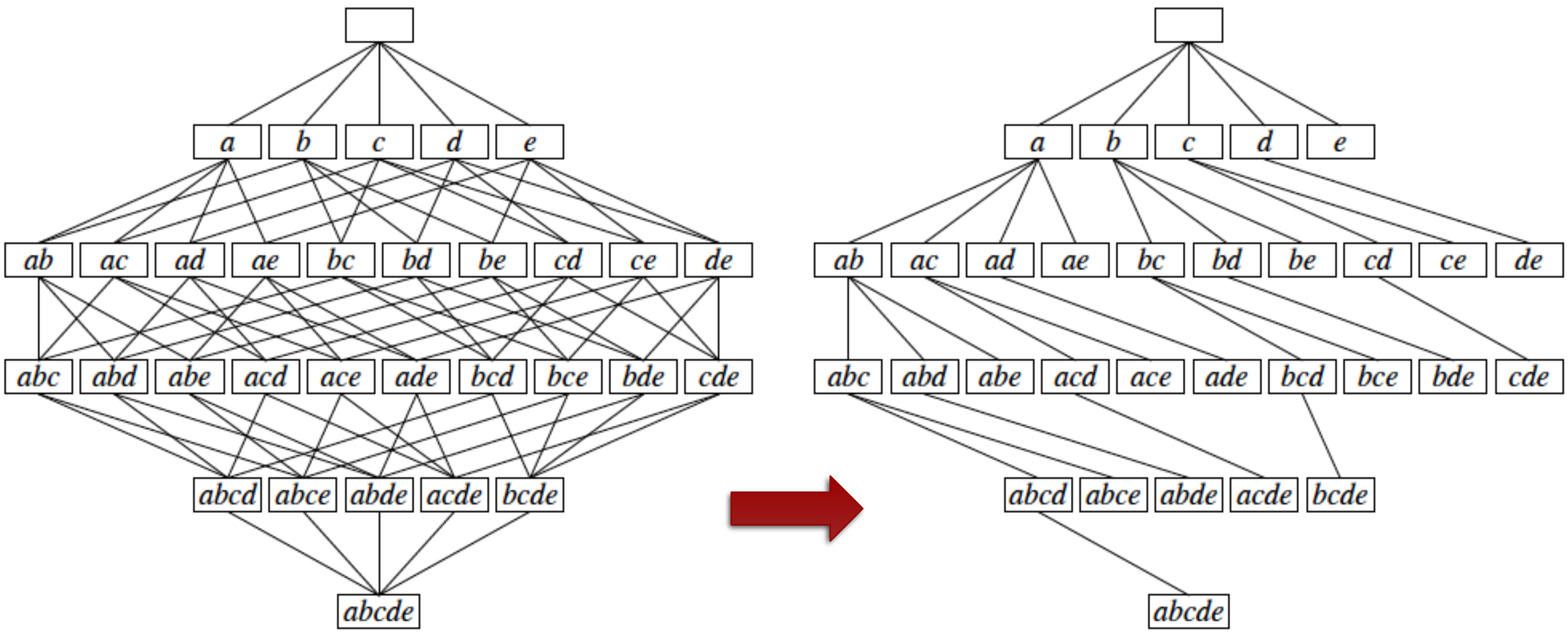
Need: Generate itemset candidate only once

How: Assign a unique parent itemset to each candidate, ensuring that it is generated only from its parent itemset

Apriori Algorithm [Agrawal and Srikant 1994]

Improving candidates generation

- Assigning unique parents turns the poset lattice into a tree:



Apriori Algorithm [Agrawal and Srikant 1994]

Canonical form for itemsets

- An itemset can be represented as a word over an alphabet I
  - Question: how many words of k items can we have?
  - Answer: k-permutations of k items: $k!$
- By imposing an arbitrary order (e.g., lexicography order) on the items, we can define a canonical form—a unique representation of itemsets that eliminates symmetries
  - Lex on items : $abc < acb < bac < bca \ldots$
  - $\kappa(abc) = \kappa(acb) = \kappa(bac) = \kappa(bca) = abc$
  - $\kappa(abc,1) = a;\ \kappa(abc,2) = b;\ \kappa(abc,3) = c$

Apriori Algorithm [Agrawal and Srikant 1994]

Recursive processing with Canonical forms

- For each itemset of a given size, generate all possible extensions of the itemset by adding one item, subject to the condition that:

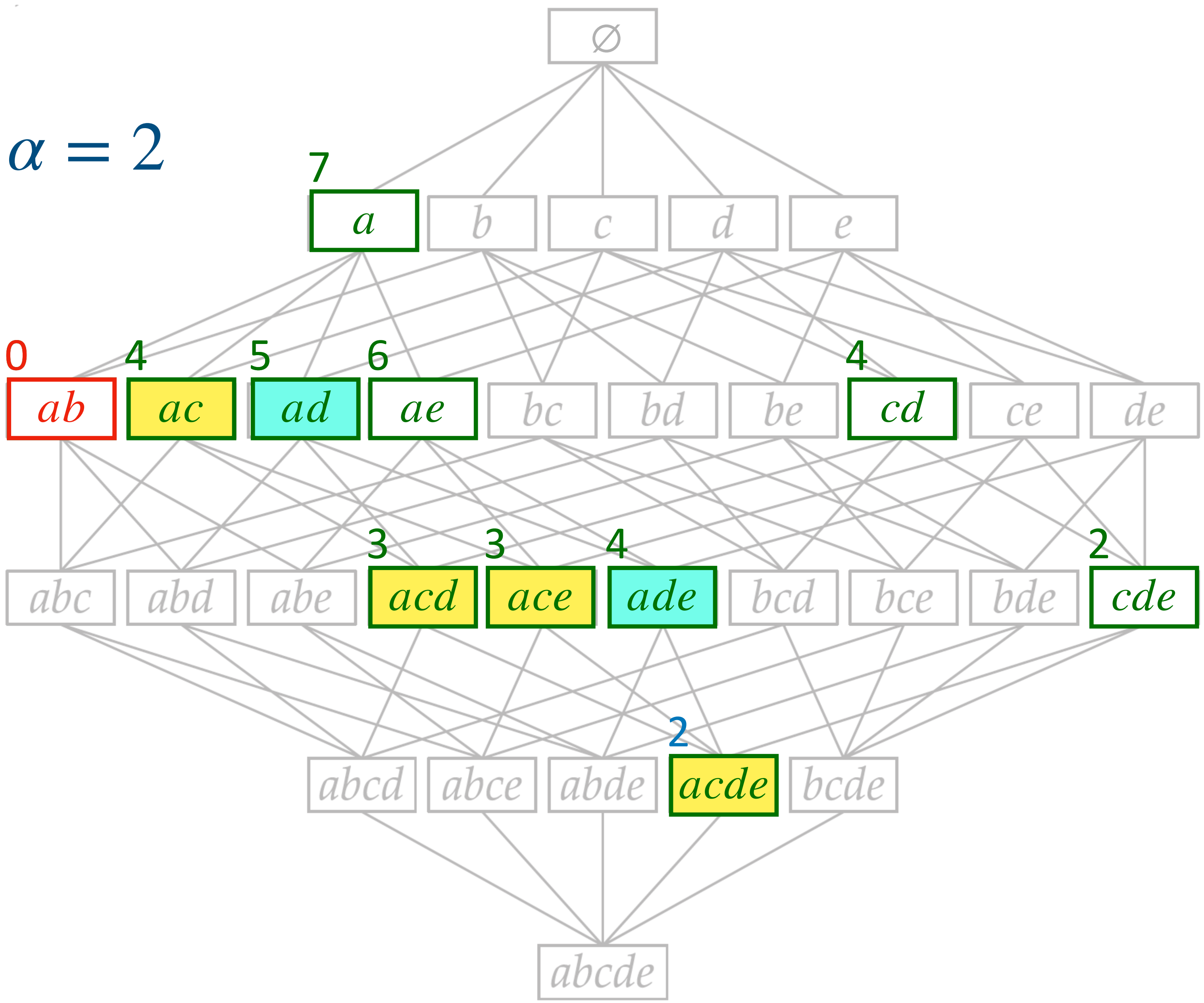
$$\boxed{child(P, \alpha) = \{P' : (P' = P \cup \{p_i\}) \wedge (\kappa(P, |P|) < p_i) \wedge (freq(P') \geq \alpha)\}}$$

Recursive processing with Canonical forms

Example(4)

$$child(P, \alpha) = \{P' : (P' = P \cup \{p_i\}) \wedge (\kappa(P, |P|) < p_i) \wedge (freq(P') \geq \alpha)\}$$

$\alpha = 2$



M_D

	a	b	c	d	e
1:	1	0	0	1	1
2:	0	1	1	1	0
3:	1	0	1	0	1
4:	1	0	1	1	1
5:	1	0	0	0	1
6:	1	0	1	1	0
7:	0	1	1	0	0
8:	1	0	1	1	1
9:	0	1	1	0	1
10:	1	0	0	1	1

Searching for Frequent Itemsets

Items Ordering

- **Any order can be used, but:**
  - The structure of the search space depends heavely on the chosen order
  - Algorithm efficiency varies significantly based on the item order
- **Advanced methods dynamically adjust the order during teh search:**
  - Use different but « compatible » orders in different branches

Searching for Frequent Itemsets

Items Ordering (heuristics)

- Sort the items w.r.t. their frequency (decreasing/increasing) : Frequent itemsets are composed of frequent items

- Sort items based on the size of the area they cover : considering both their frequency and the sizes of the transactions that include them

LISN LABORATOIRE INTERDISCIPLINAIRE DES SCIENCES DU NUMÉRIQUE

université PARIS-SACLAY

cnrs

Constraint & Data Mining

Cours2

Master 2 - DS

Nadjib Lazaar
Ing - Phd - HDR - Professor - Paris-Saclay University - LISN - LaHDAK
lazaar@lisn.fr https://perso.lisn.upsaclay.fr/lazaar/
20/01/2025