

# Inf-KDDM: Knowledge Discovery and Data Mining

Winter Term 2018/19

## Lecture 3: Frequent Itemsets and Association Rule Mining

Lectures: Prof. Dr. Matthias Renz

Exercises: Lukas Galke and Falk Bösch

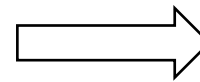
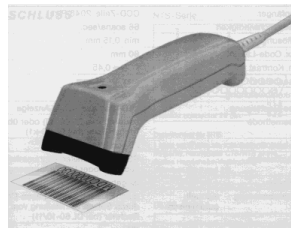
# Outline

---

- Introduction
- Basic concepts
- Frequent Itemsets Mining (FIM) – Apriori
- Association Rules Mining

# Introduction

- Frequent patterns are patterns that appear frequently in a dataset.
  - Patterns: items, substructures, subsequences ...
- Typical example: Market basket analysis



*transactions*

*items*

Customer transactions

| Tid | Transaction items                |
|-----|----------------------------------|
| 1   | Butter, Bread, Milk, Sugar       |
| 2   | Butter, Flour, Milk, Sugar       |
| 3   | Butter, Eggs, Milk, Salt         |
| 4   | Eggs                             |
| 5   | Butter, Flour, Milk, Salt, Sugar |

- We want to know: What products were often purchased together?

- e.g.: beer and diapers?



The parable of the beer and diapers:  
[http://www.theregister.co.uk/2006/08/15/beer\\_diapers/](http://www.theregister.co.uk/2006/08/15/beer_diapers/)

- Applications:

- Improving store layout, Sales campaigns, Cross-marketing, Advertising

## Applications beyond marked basket data

---

- Market basket analysis
  - Items are the products, transactions are the products bought by a customer during a supermarket visit
  - Example:  $\{\text{"Diapers"}\} \rightarrow \{\text{"Beer"}\} (0.5\%, 60\%)$
- Similarly in an online shop, e.g. Amazon
  - Example:  $\{\text{"Computer"}\} \rightarrow \{\text{"MS office"}\} (50\%, 80\%)$
- University library
  - Items are the books, transactions are the books borrowed by a student during the semester
  - Example:  $\{\text{"Kumar book"}\} \rightarrow \{\text{"Weka book"}\} (60\%, 70\%)$
- University
  - Items are the courses, transactions are the courses that are chosen by a student
  - Example:  $\{\text{"CS"}\} \wedge \{\text{"DB"}\} \rightarrow \{\text{"Grade A"}\} (1\%, 75\%)$
- ... and many other applications.
- Also, frequent pattern mining is fundamental in other DM tasks.

# Outline

---

- Introduction
- Basic concepts
- Frequent Itemsets Mining (FIM) – Apriori
- Association Rules Mining
- Homework
- Things you should know from this lecture

## Basic concepts: Items, itemsets and transactions 1/2

- **Items  $I$ :** the set of items  $I = \{i_1, \dots, i_m\}$ 
  - e.g. products in a supermarket, books in a bookstore
- **Itemset  $X$ :** A set of items  $X \subseteq I$
- **Itemset size:** the number of items in the itemset
- **$k$ -Itemset:** an itemset of size  $k$ 
  - e.g. {Butter, Bread, Milk, Sugar} is a 4-Itemset, {Butter, Bread} is a 2-Itemset
- **Transaction  $T$ :**  $T = (tid, X_T)$ 
  - e.g. products bought during a customer visit to the supermarket
- **Database  $DB$ :** A set of transactions  $T$ 
  - e.g. customers purchases in a supermarket during the last week
- Items in transactions or itemsets are lexicographically ordered
  - Itemset  $X = (x_1, x_2, \dots, x_k)$ , such as  $x_1 \leq x_2 \leq \dots \leq x_k$

| Tid | Transaction items                |
|-----|----------------------------------|
| 1   | Butter, Bread, Milk, Sugar       |
| 2   | Butter, Flour, Milk, Sugar       |
| 3   | Butter, Eggs, Milk, Salt         |
| 4   | Eggs                             |
| 5   | Butter, Flour, Milk, Salt, Sugar |

## Basic concepts: Items, itemsets and transactions 2/2

Let  $X$  be an itemset.

- **Itemset cover**: the set of transactions containing  $X$ :

$$\text{cover}(X) = \{tid \mid (tid, X_T) \in DB, X \subseteq X_T\}$$

- **(absolute) Support**/ support count of  $X$ : # transactions containing  $X$

$$\text{supportCount}(X) = |\text{cover}(X)|$$

- **(relative) Support** of  $X$ : fraction of transactions containing  $X$  (or the probability that a transaction contains  $X$ )

$$\text{support}(X) = P(X) = \text{supportCount}(X) / |DB|$$

- **Frequent itemset**: An itemset  $X$  is frequent in DB if its support is no less than a *minSupport* threshold  $s$ :

$$\text{support}(X) \geq s$$

- $L_k$ : the set of frequent  $k$ -itemsets

- $L$  comes from “Large” (“large itemsets”), another term for “frequent itemsets”

| Tid | Transaction items                |
|-----|----------------------------------|
| 1   | Butter, Bread, Milk, Sugar       |
| 2   | Butter, Flour, Milk, Sugar       |
| 3   | Butter, Eggs, Milk, Salt         |
| 4   | Eggs                             |
| 5   | Butter, Flour, Milk, Salt, Sugar |

## Example: Itemsets

- $I = \{\text{Butter, Bread, Eggs, Flour, Milk, Salt, Sugar}\}$

| Tid | Transaction items                |
|-----|----------------------------------|
| 1   | Butter, Bread, Milk, Sugar       |
| 2   | Butter, Flour, Milk, Sugar       |
| 3   | Butter, Eggs, Milk, Salt         |
| 4   | Eggs                             |
| 5   | Butter, Flour, Milk, Salt, Sugar |

- $\text{support}(\text{Butter}) = 4/5 = 80\%$ 
  - $\text{cover}(\text{Butter}) = \{1,2,3,4\}$
- $\text{support}(\text{Butter, Bread}) = 1/5 = 20\%$ 
  - $\text{cover}(\text{Butter, Bread}) = \dots$
- $\text{support}(\text{Butter, Flour}) = 2/5 = 40\%$ 
  - $\text{cover}(\text{Butter, Flour}) = \dots$
- $\text{support}(\text{Butter, Milk, Sugar}) = 3/5 = 60\%$ 
  - $\text{Cover}(\text{Butter, Milk, Sugar}) = \dots$

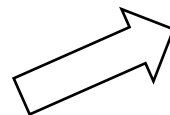


# The Frequent Itemsets Mining (FIM) problem

## Problem 1: Frequent Itemsets Mining (FIM)

- Given:
  - A set of items  $I$
  - A transactions database  $DB$  over  $I$
  - A *minSupport* threshold  $s$
- Goal: Find all frequent itemsets in  $DB$ , i.e.:  
 $\{X \subseteq I \mid \text{support}(X) \geq s\}$

| transactionID | items |
|---------------|-------|
| 2000          | A,B,C |
| 1000          | A,C   |
| 4000          | A,D   |
| 5000          | B,E,F |



Support of 1-Itemsets:

(A): 75%, (B), (C): 50%, (D), (E), (F): 25%,

Support of 2-Itemsets:

(A, C): 50%,

(A, B), (A, D), (B, C), (B, E), (B, F), (E, F): 25%

Support of 3-Itemsets:

(A, B, C), (B, E, F): 25%

Support of 4-Itemsets: -

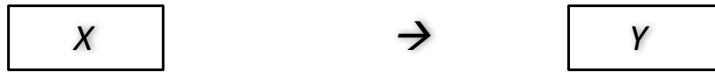
Support of 5-Itemsets: -

Support of 6-Itemsets: -

## Basic concepts: association rules, support, confidence

Let  $X, Y$  be two itemsets:  $X, Y \subseteq I$  and  $X \cap Y = \emptyset$ .

- **Association rules**: rules of the form



head or LHS (left-hand side) or antecedent of the rule

body or RHS (right-hand side) or consequent of the rule

- **Support**  $s$  of a rule: the percentage of transactions containing  $X \cup Y$  in the DB

$$\text{support}(X \rightarrow Y) = \text{support}(X \cup Y)$$

- **Confidence**  $c$  of a rule: the percentage of transactions containing  $X \cup Y$  in the set of transactions containing  $X$ . Or, in other words the conditional probability that a transaction containing  $X$  also contains  $Y$

$$\text{confidence}(X \rightarrow Y) = P(Y|X) = \frac{P(X \cap Y)}{P(X)} = \frac{\text{support}(X \cup Y)}{\text{support}(X)}$$

- Support and confidence are measures of rules' interestingness.
- Rules are usually written as follows:  **$X \rightarrow Y$  (support, confidence)**

Explain the rules:

- {Diapers}  $\rightarrow$  {Beer} (0.5%, 60%)
- {Toastbread}  $\rightarrow$  {Toast cheese} (50%, 90%)



## Example: association rules

- $I = \{\text{Butter, Bread, Eggs, Flour, Milk, Salt, Sugar}\}$

| Tid | Transaction items                |
|-----|----------------------------------|
| 1   | Butter, Bread, Milk, Sugar       |
| 2   | Butter, Flour, Milk, Sugar       |
| 3   | Butter, Eggs, Milk, Salt         |
| 4   | Eggs                             |
| 5   | Butter, Flour, Milk, Salt, Sugar |

### Sample rules:

- $\{\text{Butter}\} \rightarrow \{\text{Bread}\} (20\%, 25\%)$ 
  - $\text{support}(\text{Butter} \cup \text{Bread}) = 1/5 = 20\%$
  - $\text{support}(\text{Butter}) = 4/5 = 80\%$
  - $\text{Confidence} = 20\%/80\% = 1/4 = 25\%$
- $\{\text{Butter, Milk}\} \rightarrow \text{Sugar} (60\%, 75\%)$ 
  - $\text{support}(\text{Butter, Milk} \cup \text{Sugar}) = 3/5 = 60\%$
  - $\text{Support}(\text{Butter, Milk}) = 4/5 = 80\%$
  - $\text{Confidence} = 60\%/80\% = 3/4 = 75\%$

# The Association Rules Mining (ARM) problem

## Problem 2: Association Rules Mining (ARM)

### ■ Given:

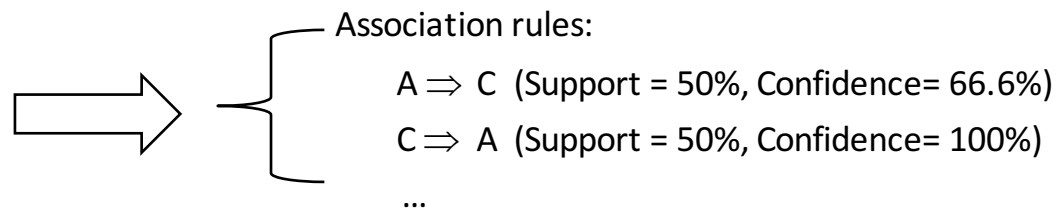
- A set of items  $I$
- A transactions database  $DB$  over  $I$
- A *minSupport* threshold  $s$  and a *minConfidence* threshold  $c$

### ■ Goal: Find all association rules $X \rightarrow Y$ in $DB$ w.r.t. minimum support $s$ and minimum confidence $c$ , i.e.:

$$\{X \rightarrow Y \mid \text{support}(X \cup Y) \geq s, \text{confidence}(X \rightarrow Y) \geq c\}$$

- These rules are called *strong*.

| transactionID | items |
|---------------|-------|
| 2000          | A,B,C |
| 1000          | A,C   |
| 4000          | A,D   |
| 5000          | B,E,F |



## Solving the problems

- Problem 1 (FIM): Find all frequent itemsets in  $DB$ , i.e.:  $\{X \subseteq I \mid \text{support}(X) \geq s\}$
- Problem 2 (ARM): Find all association rules  $X \rightarrow Y$  in  $DB$ , w.r.t. min support  $s$  and min confidence  $c$ , i.e.:  $\{X \rightarrow Y \mid \text{support}(X \cup Y) \geq s, \text{confidence}(X \rightarrow Y) \geq c, X, Y \subseteq I \text{ and } X \cap Y = \emptyset\}$
- Problem 1 is part of Problem 2:
  - Once we have  $\text{support}(X \cup Y)$  and  $\text{support}(X)$ , we can check if  $X \rightarrow Y$  is strong.
- 2-step method to extract the association rules:
  - Step 1: Determine the frequent itemsets w.r.t. min support  $s$ :
    - “Naïve” algorithm: count the frequencies for all  $k$ -itemsets
      - Inefficient!!! There are  $O\left(\binom{|I|}{k}\right)$  such subsets
      - Total cost:  $O(2^{|I|})$   
=> Apriori-algorithm and variants
  - Step 2: Generate the association rules w.r.t. min confidence  $c$ :  
from frequent itemsets  $X$ , generate  $Y \rightarrow (X - Y), Y \subset X, Y \neq \emptyset, Y \neq X$

FIM problem

*Step 1(FIM) is the most costly, so the overall performance of an association rules mining algorithm is determined by this step.*

## Itemset lattice complexity

- The number of itemsets can be really huge. Let us consider a small set of items:  $I = \{A, B, C, D\}$

- # 1-itemsets:  $\binom{4}{1} = \frac{4!}{(4-1)!*1!} = \frac{4!}{3!} = 4$

- # 2-itemsets:  $\binom{4}{2} = \frac{4!}{(4-2)!*2!} = \frac{4!}{2!*2!} = 6$

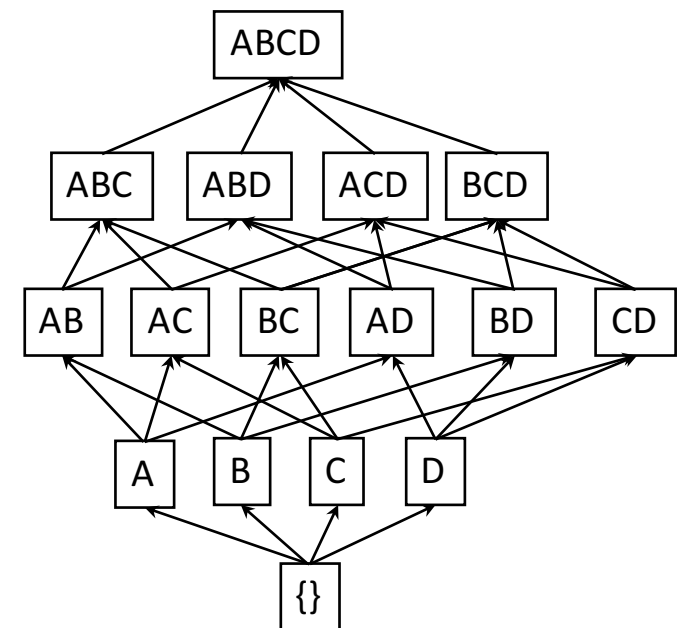
- # 3-itemsets:  $\binom{4}{3} = \frac{4!}{(4-3)!*3!} = \frac{4!}{3!} = 4$

- # 4-itemsets:  $\binom{4}{4} = \frac{4!}{(4-4)!*4!} = 1$

- In the general case, for  $|I|$  items, there exist:

$$\binom{|I|}{1} + \binom{|I|}{2} + \dots + \binom{|I|}{k} = 2^{|I|} - 1$$

- So, generating all possible combinations and computing their support is inefficient!



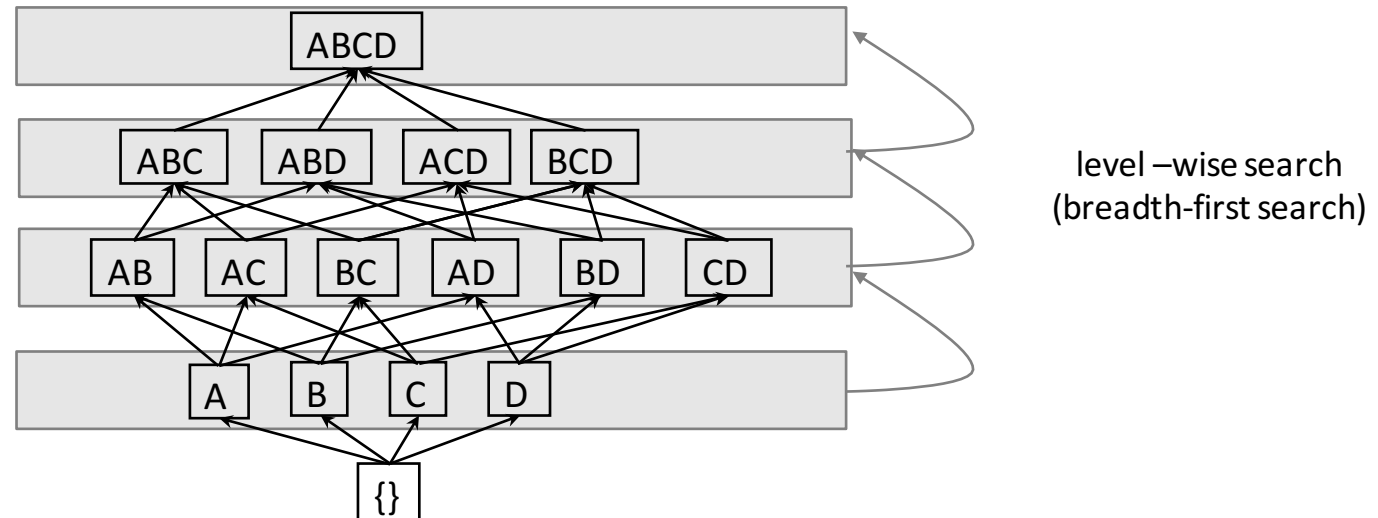
# Outline

---

- Introduction
- Basic concepts
- Frequent Itemsets Mining (FIM) – Apriori
- Association Rules Mining

# Apriori algorithm [Agrawal & Srikant @VLDB'94]

- Idea: First determine frequent 1-itemsets, then frequent 2-itemsets and so on



- Method overview:

- Initially, scan *DB* once to get frequent 1-itemset
- Generate length  $(k+1)$  candidate itemsets from length  $k$  frequent itemsets
- Test the candidates against *DB* (one scan)
- Terminate when no frequent or candidate set can be generated



## Apriori property

- **Naïve approach:** Count the frequency of all  $k$ -itemsets  $X$  from  $I$

- generate  $\sum_{k=1}^{|I|} \binom{|I|}{k} = 2^{|I|} - 1$  itemsets, i.e.,  $O(2^{|I|})$ .
- for each candidate itemset  $X$ , the algorithm evaluates whether  $X$  is frequent

→ To reduce complexity, the set of candidates should be as small as possible!!!

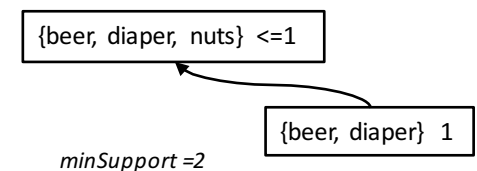
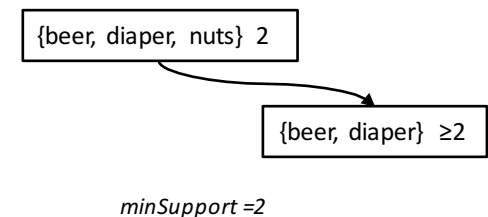
- **Downward closure property / Monotonic property/Apriori property** of frequent itemsets:

- If  $X$  is *frequent*, all its subsets  $Y \subseteq X$  are also *frequent*.

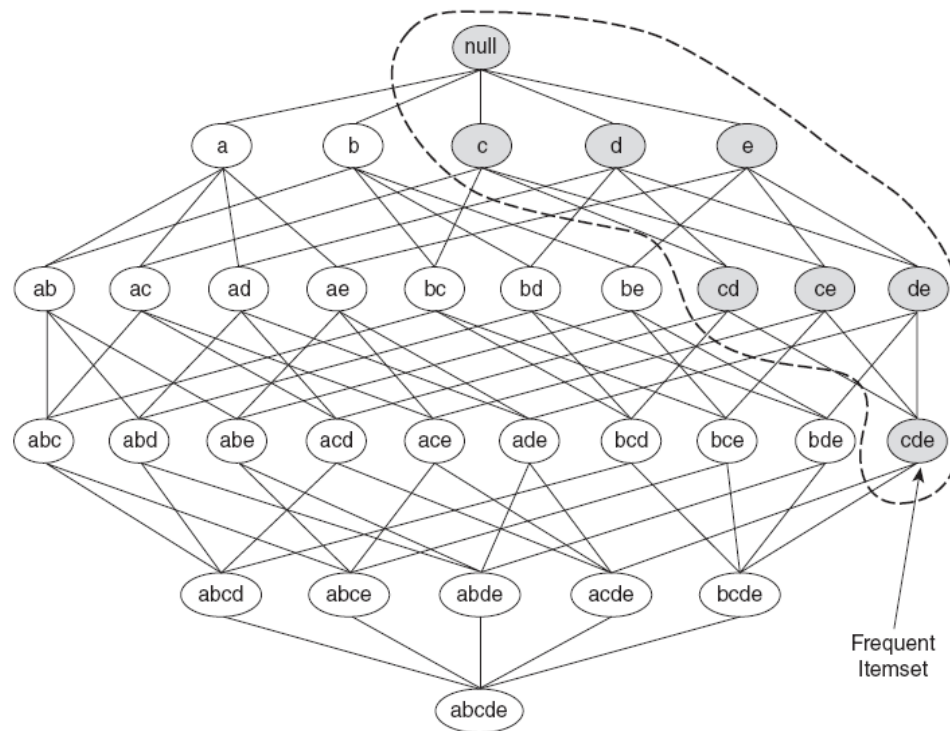
- e.g., if {beer, diaper, nuts} is frequent, so is {beer, diaper}
  - i.e., every transaction having {beer, diaper, nuts} also contains {beer, diaper}
- similarly for {diaper, nuts}, {beer, nuts}

- **On the contrary:** When  $X$  is *not frequent*, all its supersets are *not frequent* and thus they should not be generated/ tested!!! → reduce the candidate itemsets set

- e.g., if {beer, diaper} is not frequent, {beer, diaper, nuts} would not be frequent also



## Illustration of the Apriori property



**Figure 6.3.** An illustration of the *Apriori* principle. If  $\{c, d, e\}$  is frequent, then all subsets of this itemset are frequent.

## Search space and pruning

---

- Let us consider the following transaction database

Transaction Database

{Chips, Pizza}

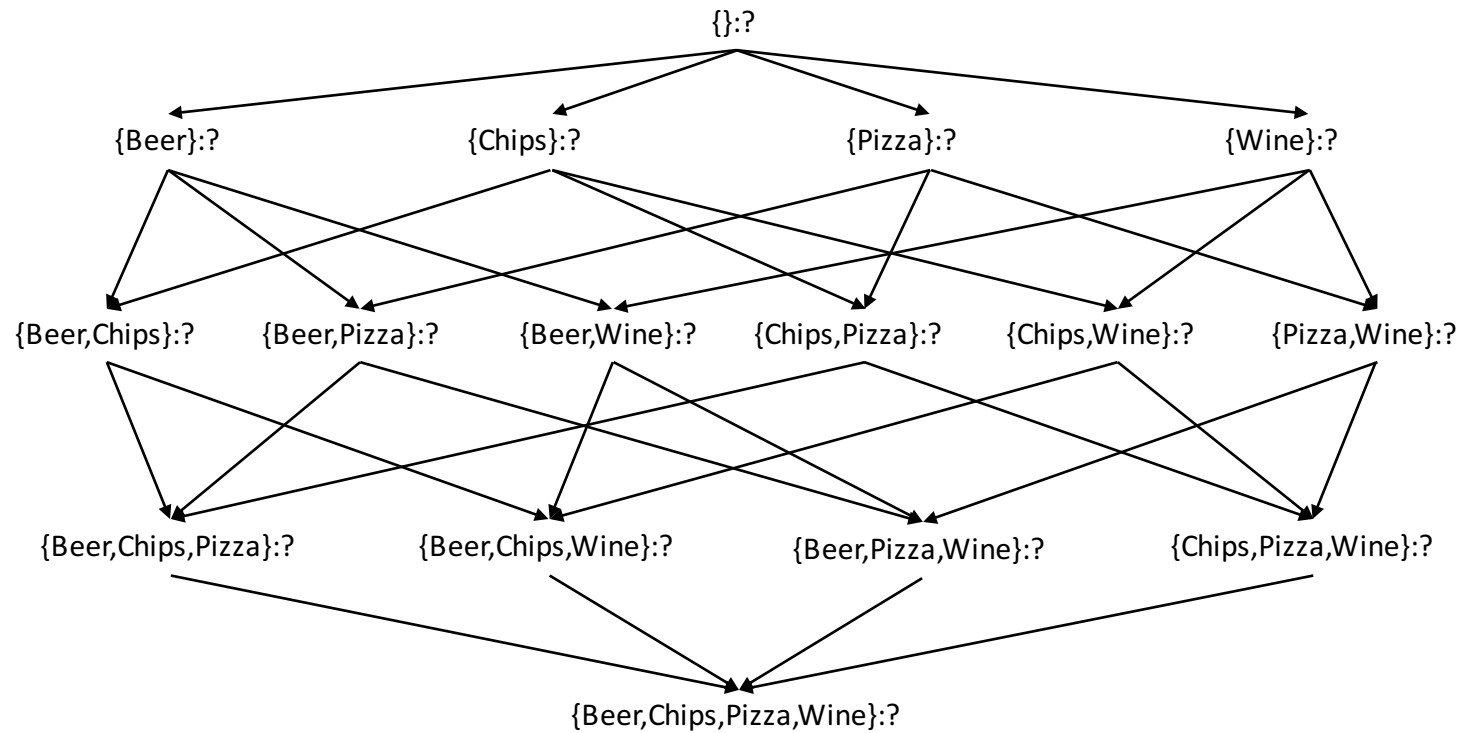
{Beer, Chips}

{Chips, Pizza, Wine}

{Wine}

- and a minSupport threshold  $minSupp = 2$

# Search space and pruning

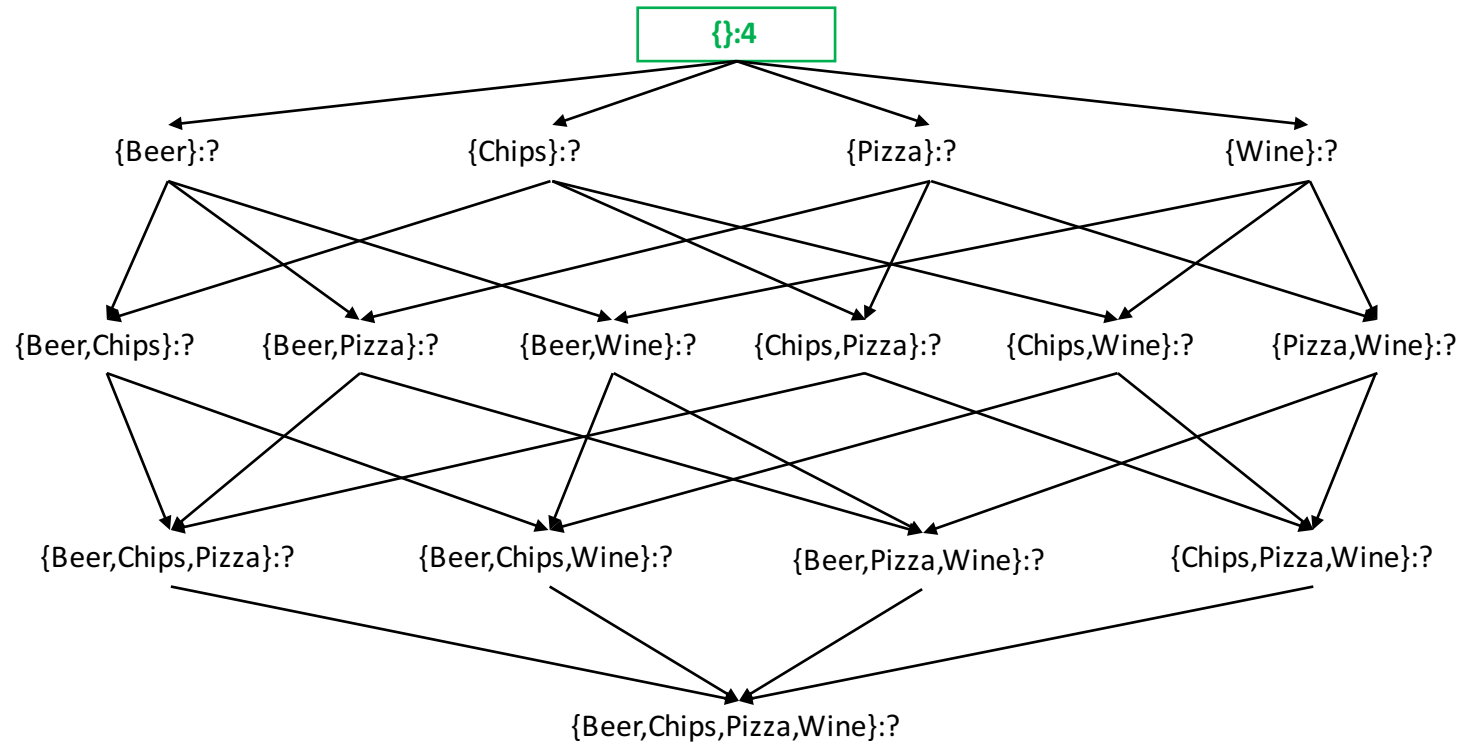


## Transaction Database

- {Chips, Pizza}
- {Beer, Chips}
- {Chips, Pizza, Wine}
- {Wine}

$minSupp = 2$

# Search space and pruning

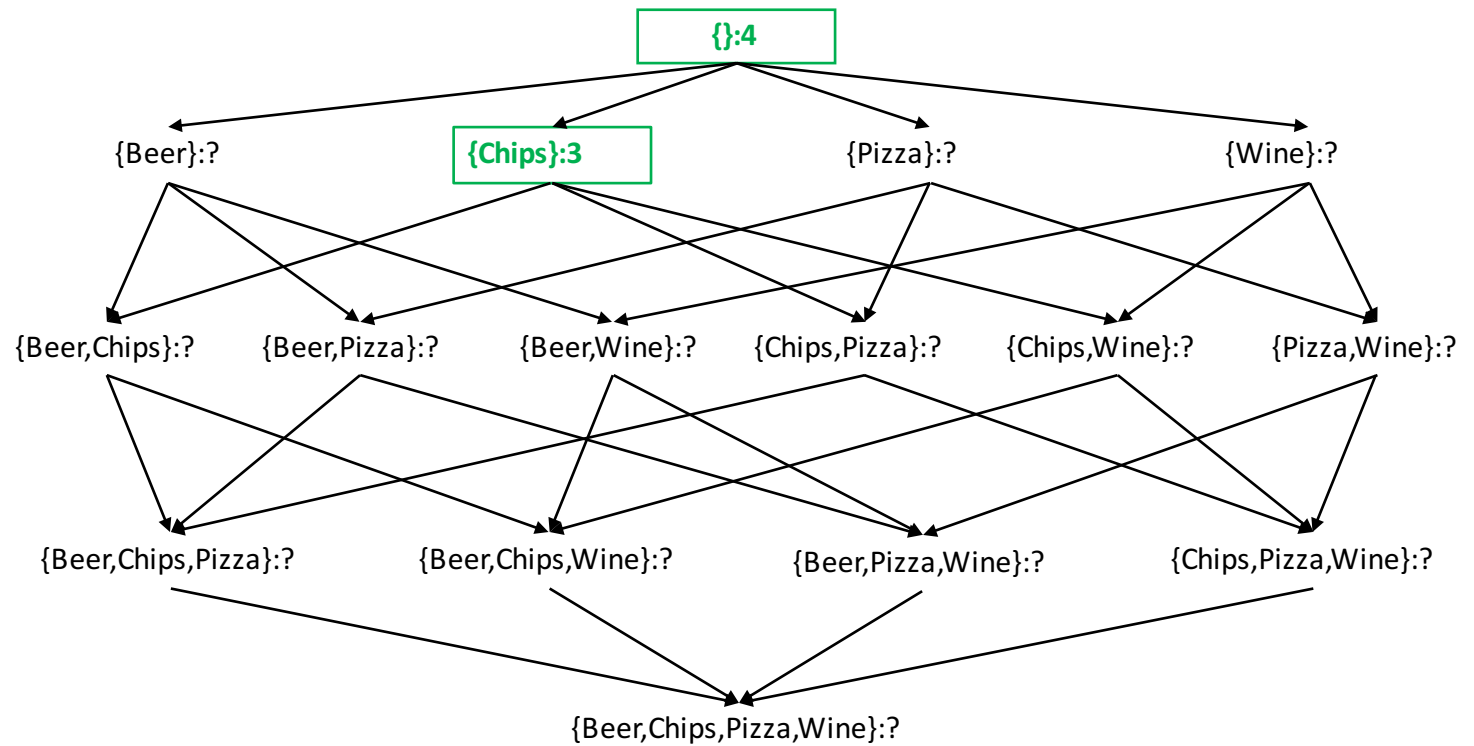


## Transaction Database

- {Chips, Pizza}
- {Beer, Chips}
- {Chips, Pizza, Wine}
- {Wine}

$minSupp = 2$

# Search space and pruning

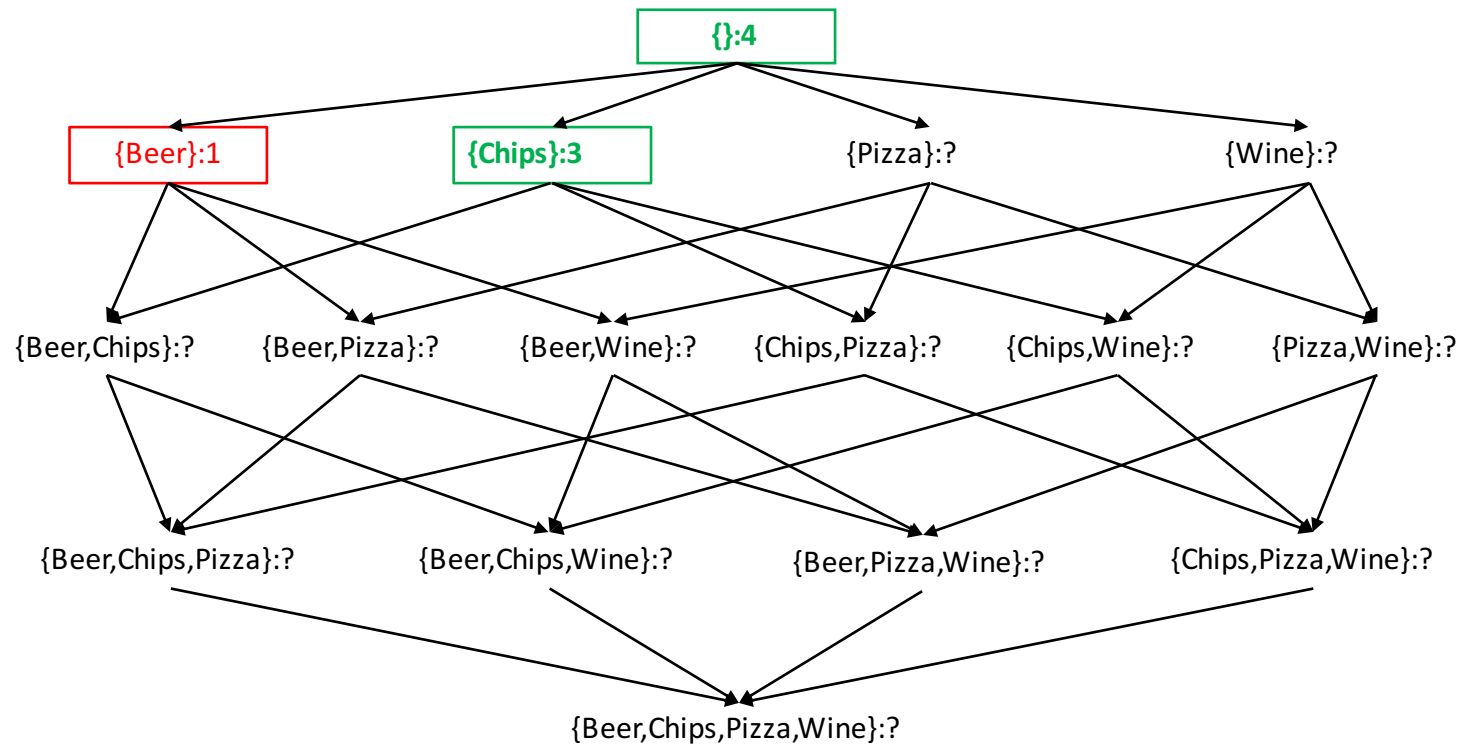


## Transaction Database

{Chips, Pizza}  
{Beer, Chips}  
{Chips, Pizza, Wine}  
{Wine}

$minSupp = 2$

# Search space and pruning

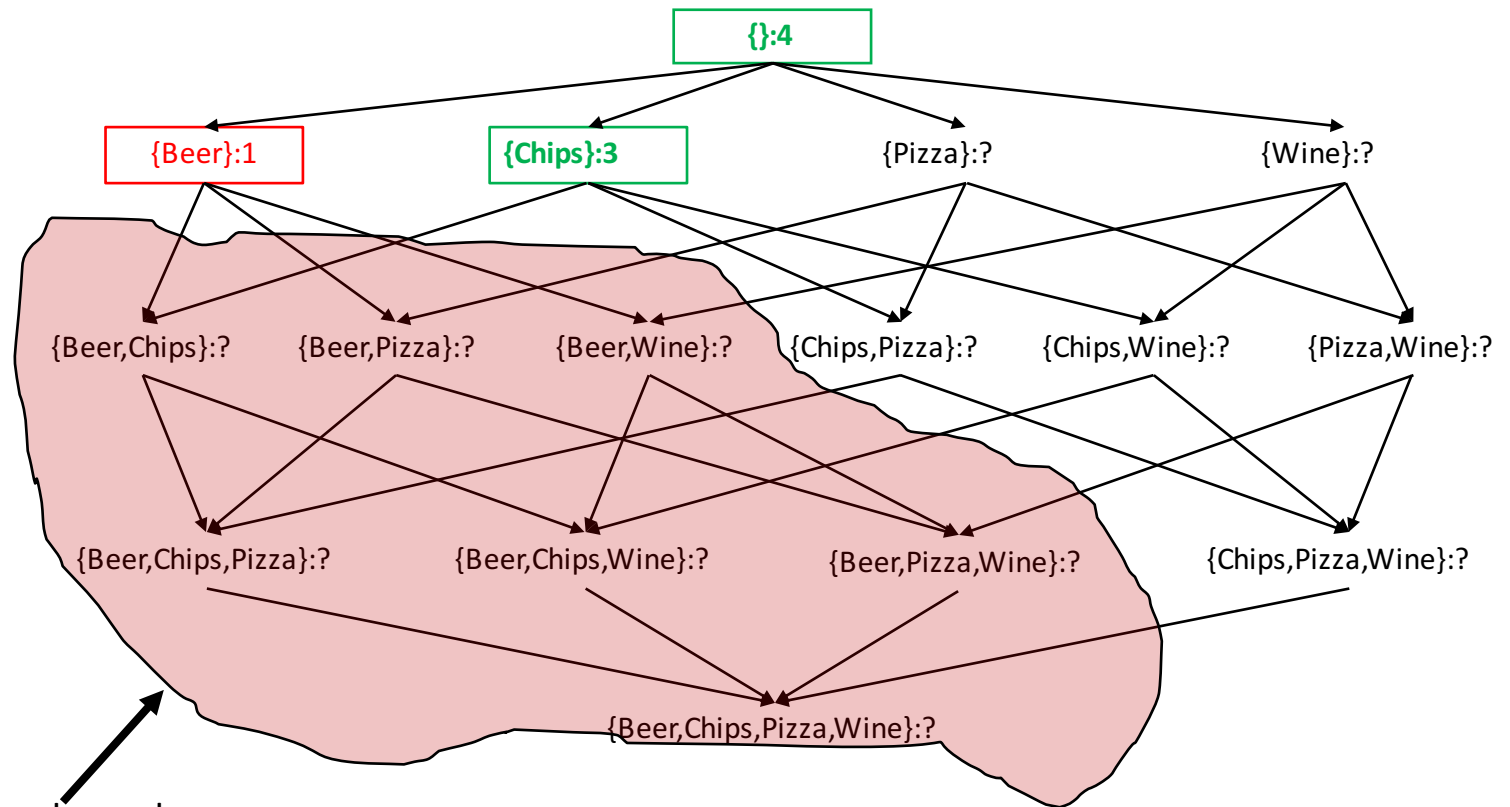


## Transaction Database

- {Chips, Pizza}
- {Beer, Chips}
- {Chips, Pizza, Wine}
- {Wine}

$minSupp = 2$

# Search space and pruning



Transaction Database

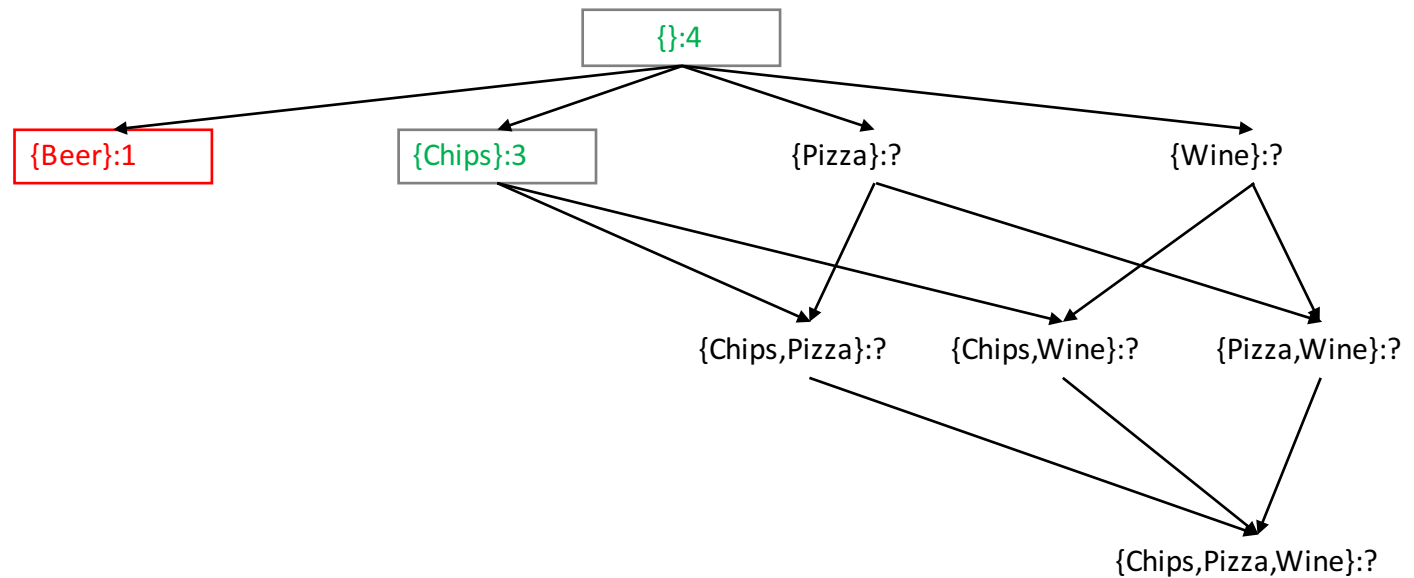
- {Chips, Pizza}
- {Beer, Chips}
- {Chips, Pizza, Wine}
- {Wine}

$minSupp = 2$

Pruned search space



# Search space and pruning

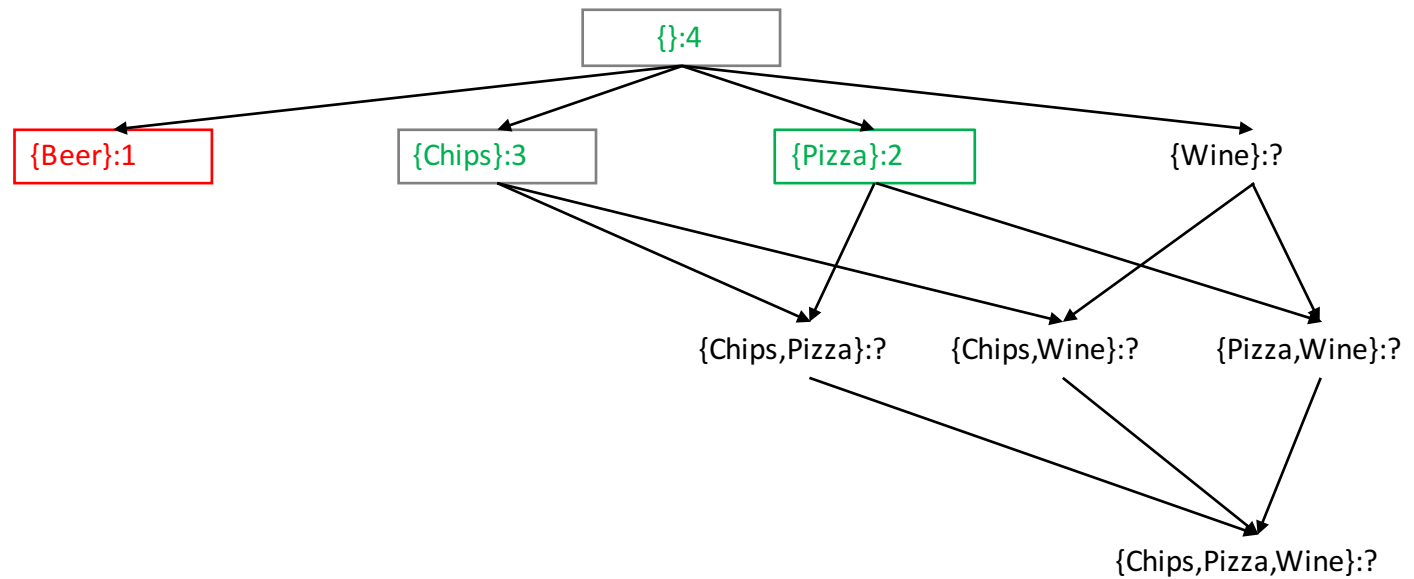


## Transaction Database

{Chips, Pizza}  
{Beer, Chips}  
{Chips, Pizza, Wine}  
{Wine}

$minSupp = 2$

# Search space and pruning

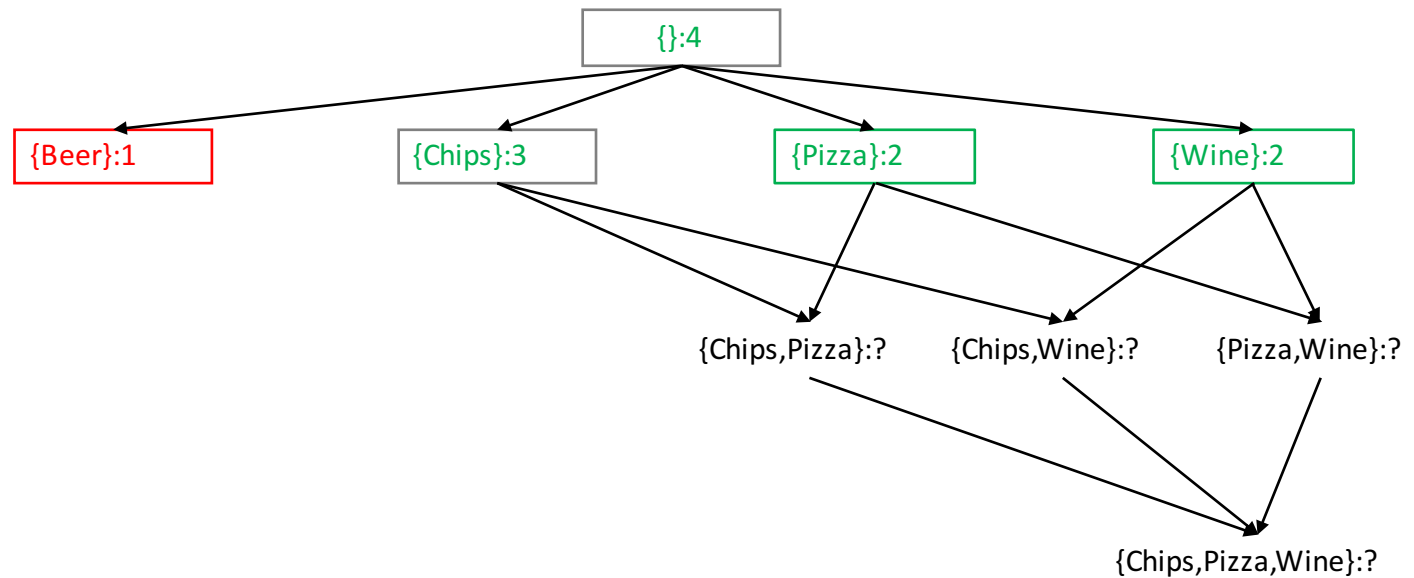


## Transaction Database

{Chips, Pizza}  
{Beer, Chips}  
{Chips, Pizza, Wine}  
{Wine}

$minSupp = 2$

# Search space and pruning

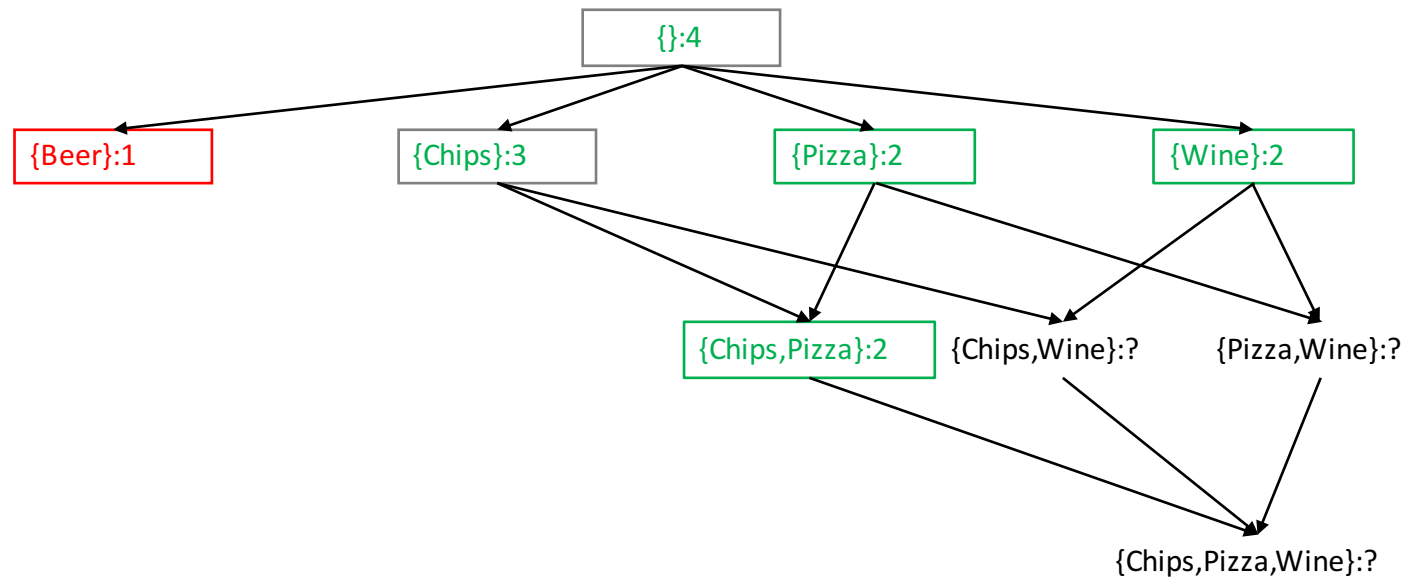


## Transaction Database

{Chips, Pizza}  
{Beer, Chips}  
{Chips, Pizza, Wine}  
{Wine}

$minSupp = 2$

# Search space and pruning

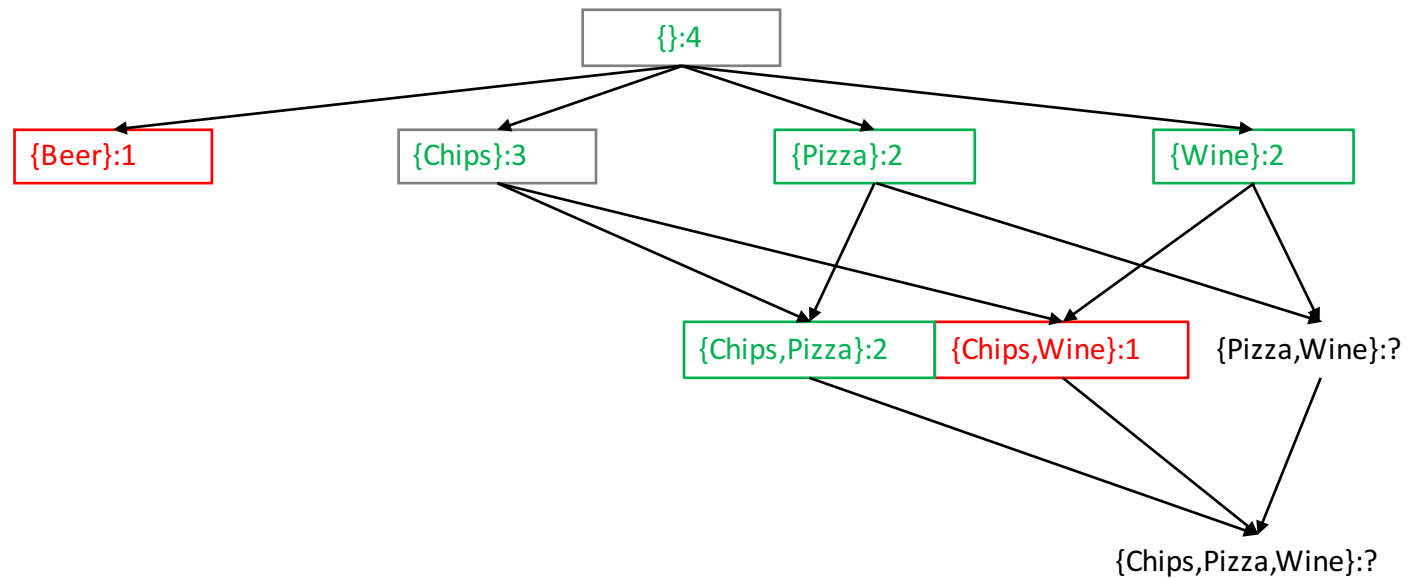


## Transaction Database

{Chips, Pizza}  
{Beer, Chips}  
{Chips, Pizza, Wine}  
{Wine}

$minSupp = 2$

# Search space and pruning

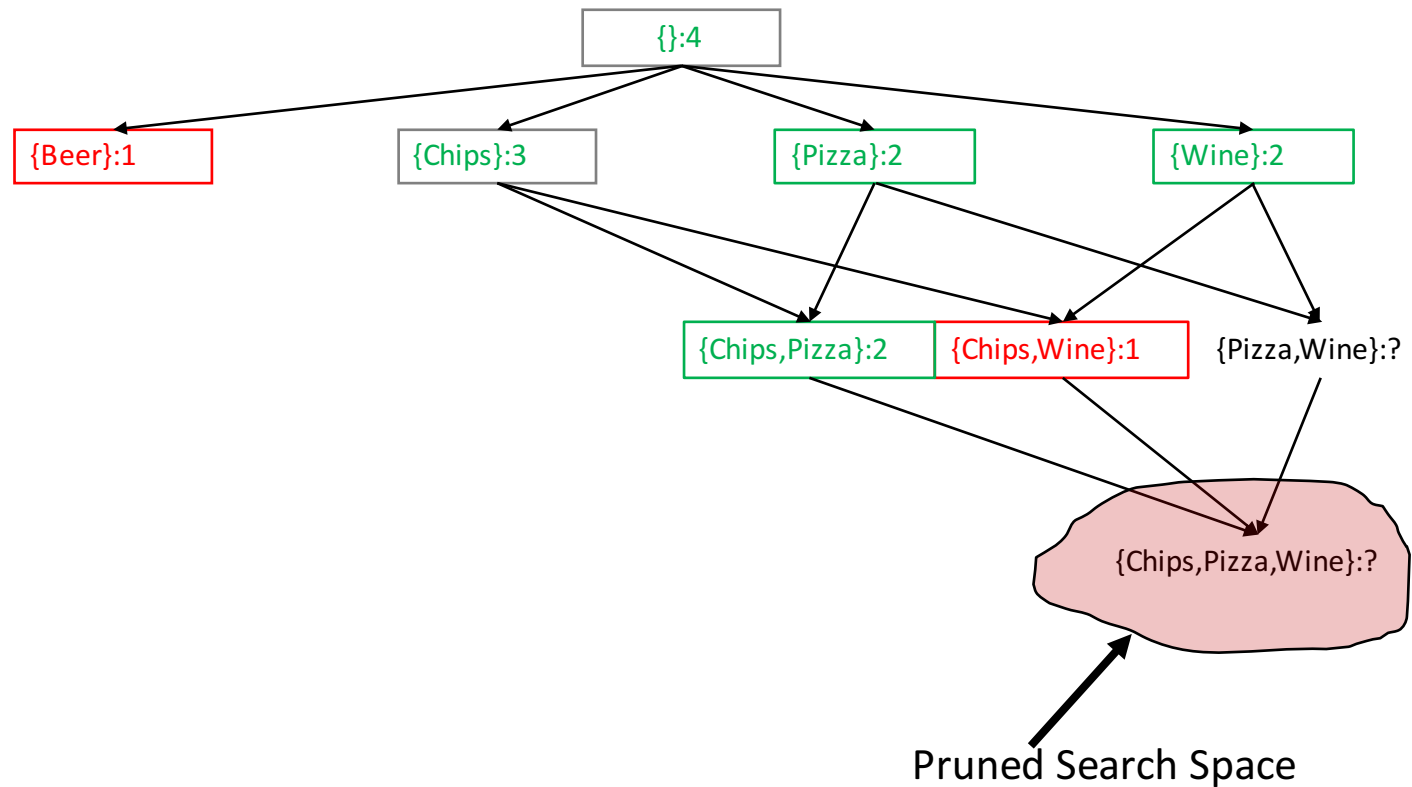


## Transaction Database

{Chips, Pizza}  
{Beer, Chips}  
{Chips, Pizza, Wine}  
{Wine}

$minSupp = 2$

# Search space and pruning



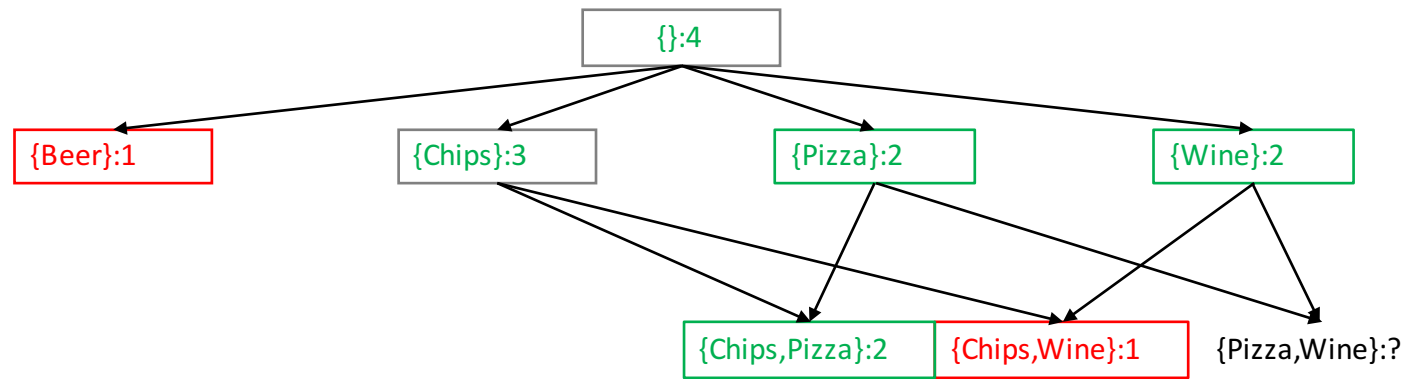
## Transaction Database

{Chips, Pizza}  
{Beer, Chips}  
{Chips, Pizza, Wine}  
{Wine}

$minSupp = 2$

Pruned Search Space

# Search space and pruning

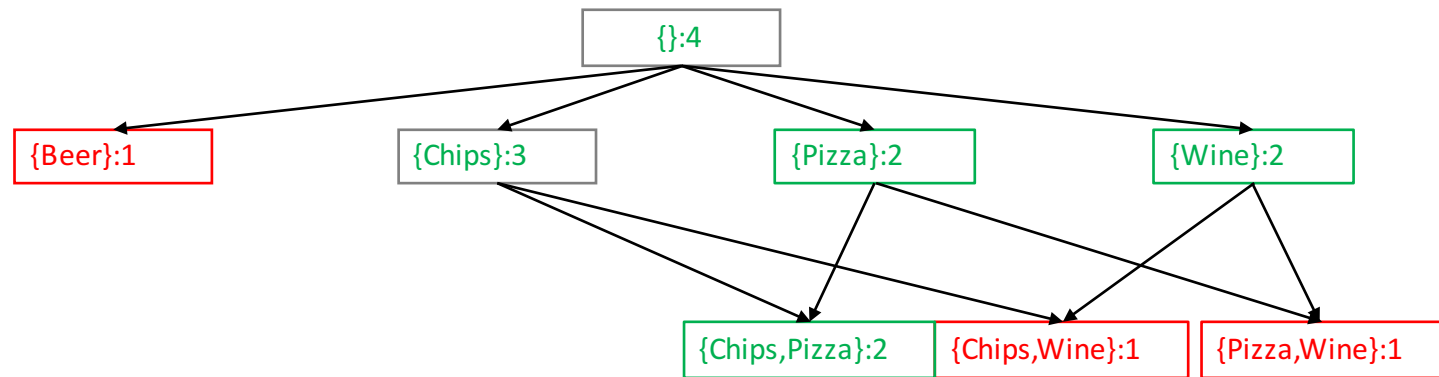


## Transaction Database

{Chips, Pizza}  
{Beer, Chips}  
{Chips, Pizza, Wine}  
{Wine}

$minSupp = 2$

## Search space and pruning



### Transaction Database

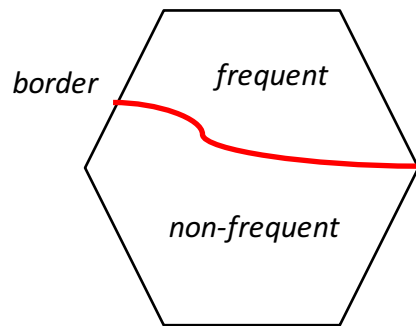
{Chips, Pizza}  
{Beer, Chips}  
{Chips, Pizza, Wine}  
{Wine}

$minSupp = 2$



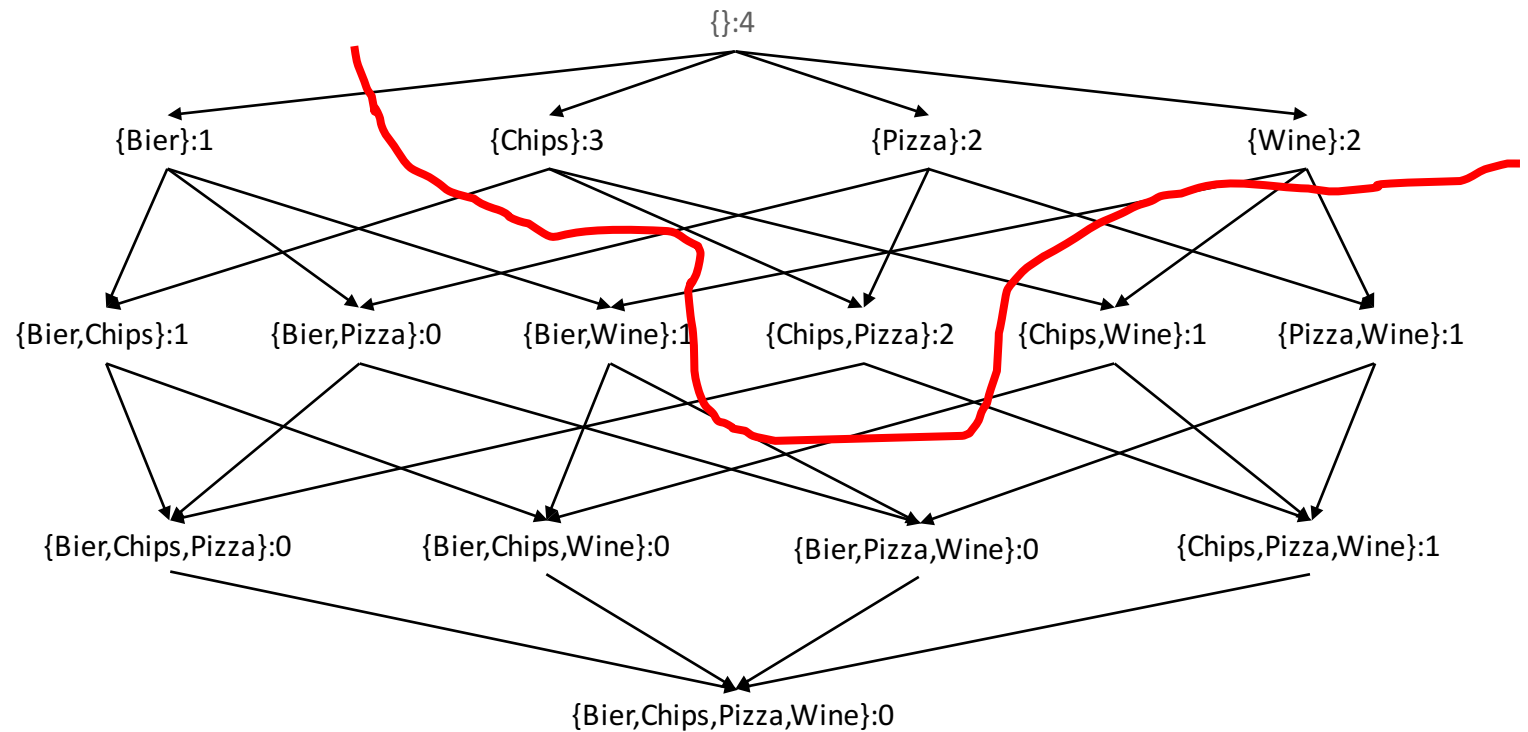
# Search space and pruning

- **Border itemsets**  $X$ : all subsets  $Y \subset X$  are frequent, all supersets  $Z \supset X$  are not frequent



## Transaction Database

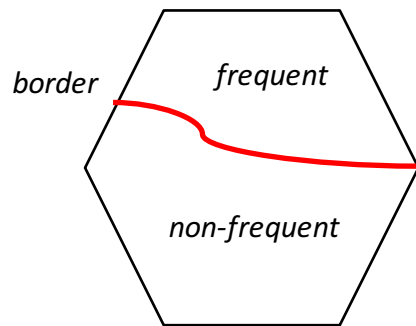
- {Chips, Pizza}
- {Beer, Chips}
- {Chips, Pizza, Wine}
- {Wine}



minSupport  $s = 2$

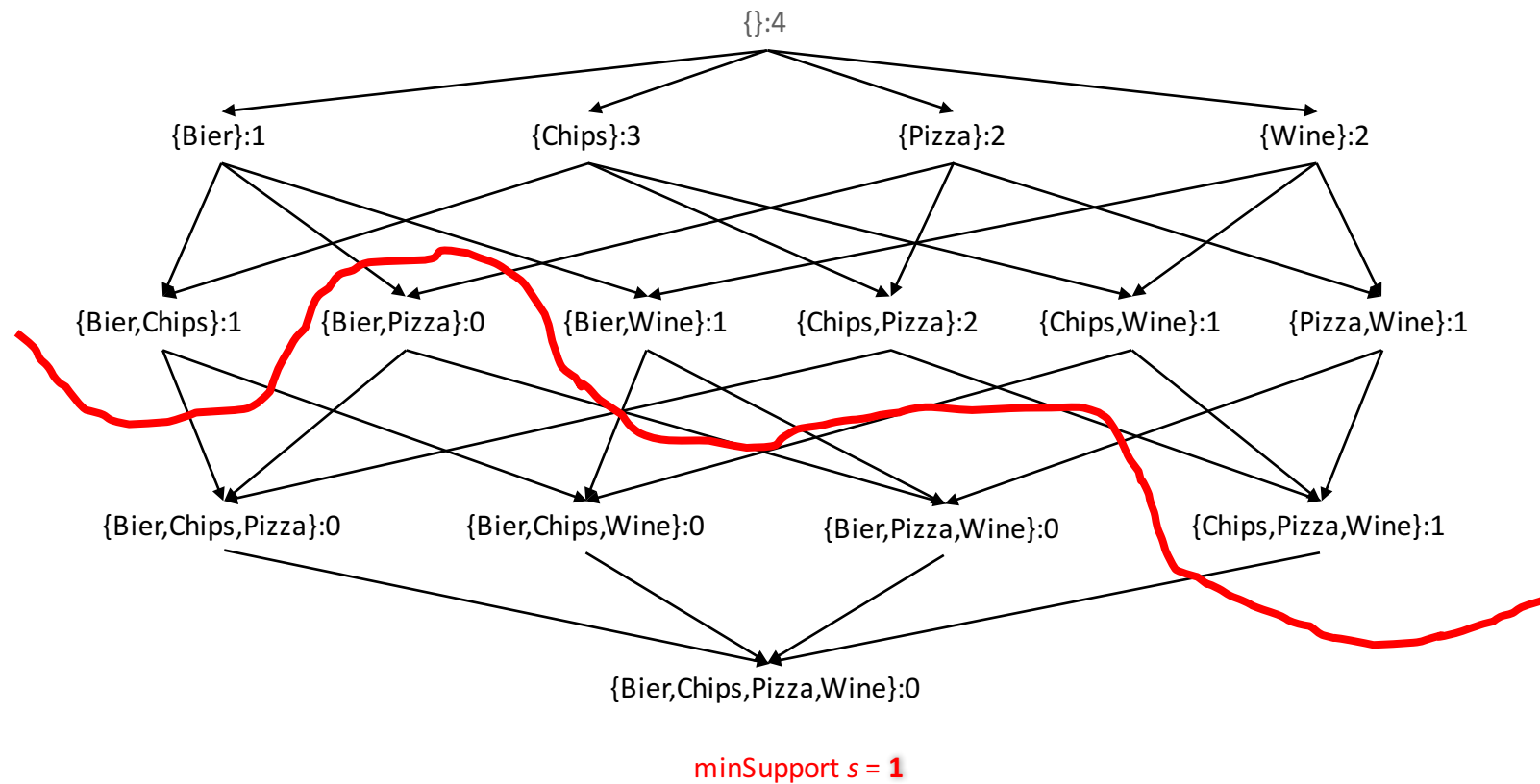
# Search space and pruning

- **Border itemsets**  $X$ : all subsets  $Y \subset X$  are frequent, all supersets  $Z \supset X$  are not frequent



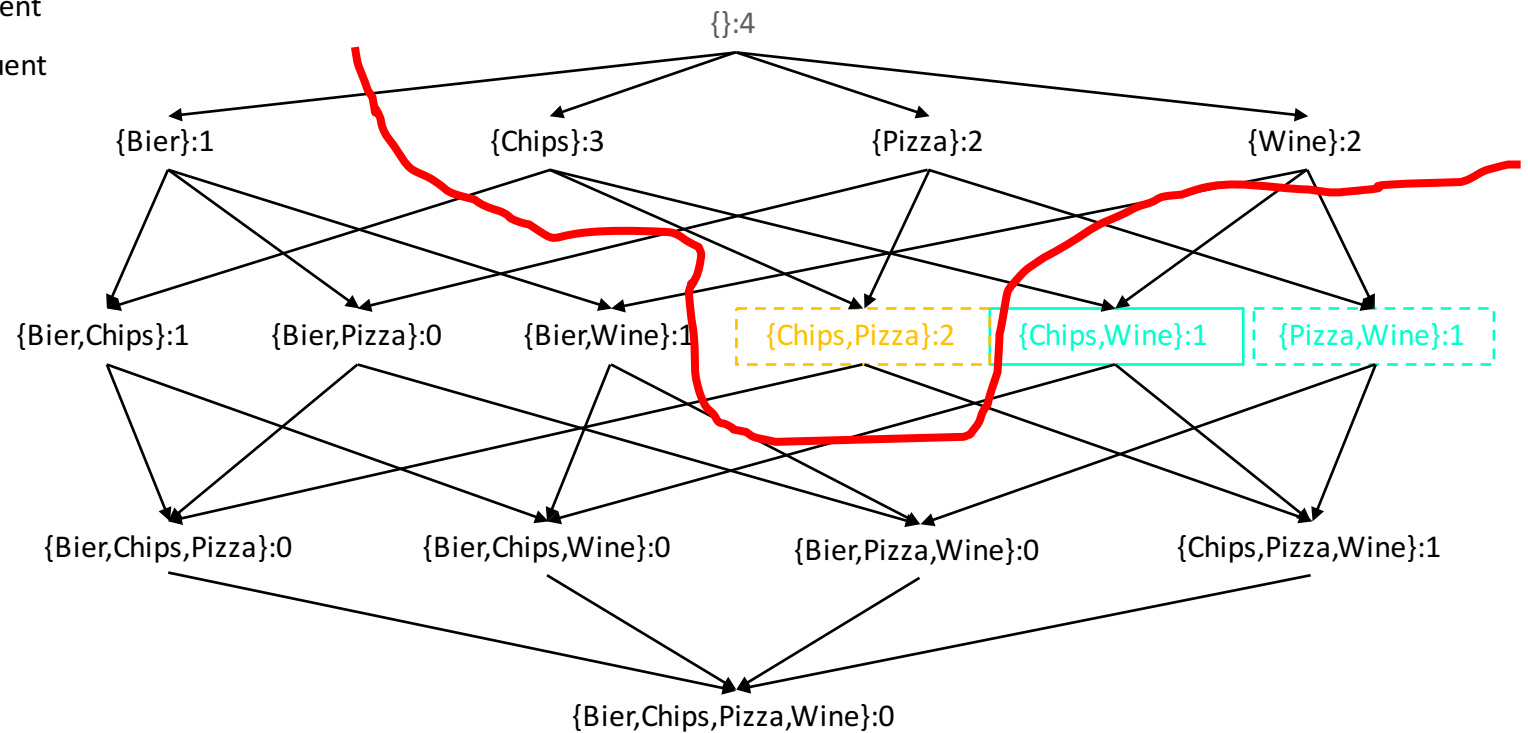
## Transaction Database

{Chips, Pizza}  
 {Beer, Chips}  
 {Chips, Pizza, Wine}  
 {Wine}



# Search space and pruning

- **Border itemsets**  $X$ : all subsets  $Y \subset X$  are frequent, all supersets  $Z \supset X$  are not frequent
  - **Positive border**:  $X$  is also frequent
  - **Negative border**:  $X$  is not frequent



## Transaction Database

- {Chips, Pizza}
- {Beer, Chips}
- {Chips, Pizza, Wine}
- {Wine}

$minSupp = 2$

Positive border-itemsets

$minSupport\ s = 2$

Negative border-itemsets

## Frequent itemsets generation: From $L_{k-1}$ to $C_k$ to $L_k$

$L_k$ : frequent itemsets of size  $k$ ;  $C_k$ : candidate itemsets of size  $k$

A 2-step process:

■ **Join step:** generate candidates  $C_k$

- $L_k$  is generated by self-joining  $C_k = L_{k-1} \bowtie L_{k-1}$ ,  $C_k :=$  Set of candidates in  $L_k$
- Two  $(k-1)$ -itemsets  $p, q$  are joined, if they agree in the first  $(k-2)$  items

■ **Prune step:** prune  $C_k$  and return  $L_k$

- $C_k$  is superset of  $L_k$
- Naïve idea: count the support for all candidate itemsets in  $C_k$  ....  $|C_k|$  might be large!
- Use Apriori property: a candidate  $k$ -itemset that has some non-frequent  $(k-1)$ -itemset cannot be frequent
  - Prune all those  $k$ -itemsets, that have some  $(k-1)$ -subset that is not frequent (i.e. does not belong to  $L_{k-1}$ )
  - Due to the level-wise approach of Apriori, we only need to check  $(k-1)$ -subsets
- For the remaining itemsets in  $C_k$ , prune by support count (DB)

Example:

Let  $L_3 = \{abc, abd, acd, ace, bcd\}$

- Join step:  $C_4 = L_3 * L_3$

$C_4 = \{abc*abd=abcd; acd*ace=acde\}$

- Prune step (apriori-based):

acde is pruned since cde is not frequent

- Prune step (DB-based):

check abcd support in the DB

## Apriori algorithm (pseudo-code)

$C_k$ : Candidate itemset of size  $k$

$L_k$ : frequent itemset of size  $k$

$L_1 = \{\text{frequent items}\};$

**for** ( $k = 1; L_k \neq \emptyset; k++$ ) **do begin**

$C_{k+1} =$  candidates generated from  $L_k$ ;

**for each** transaction  $t$  in database **do**

increment the count of all candidates in  $C_{k+1}$  that are contained in  $t$

$L_{k+1} =$  candidates in  $C_{k+1}$  with min\_support

**end**

**return**  $\cup_k L_k$ ;

*Candidate generation  
(self-join, apriori property)*

*DB scan*

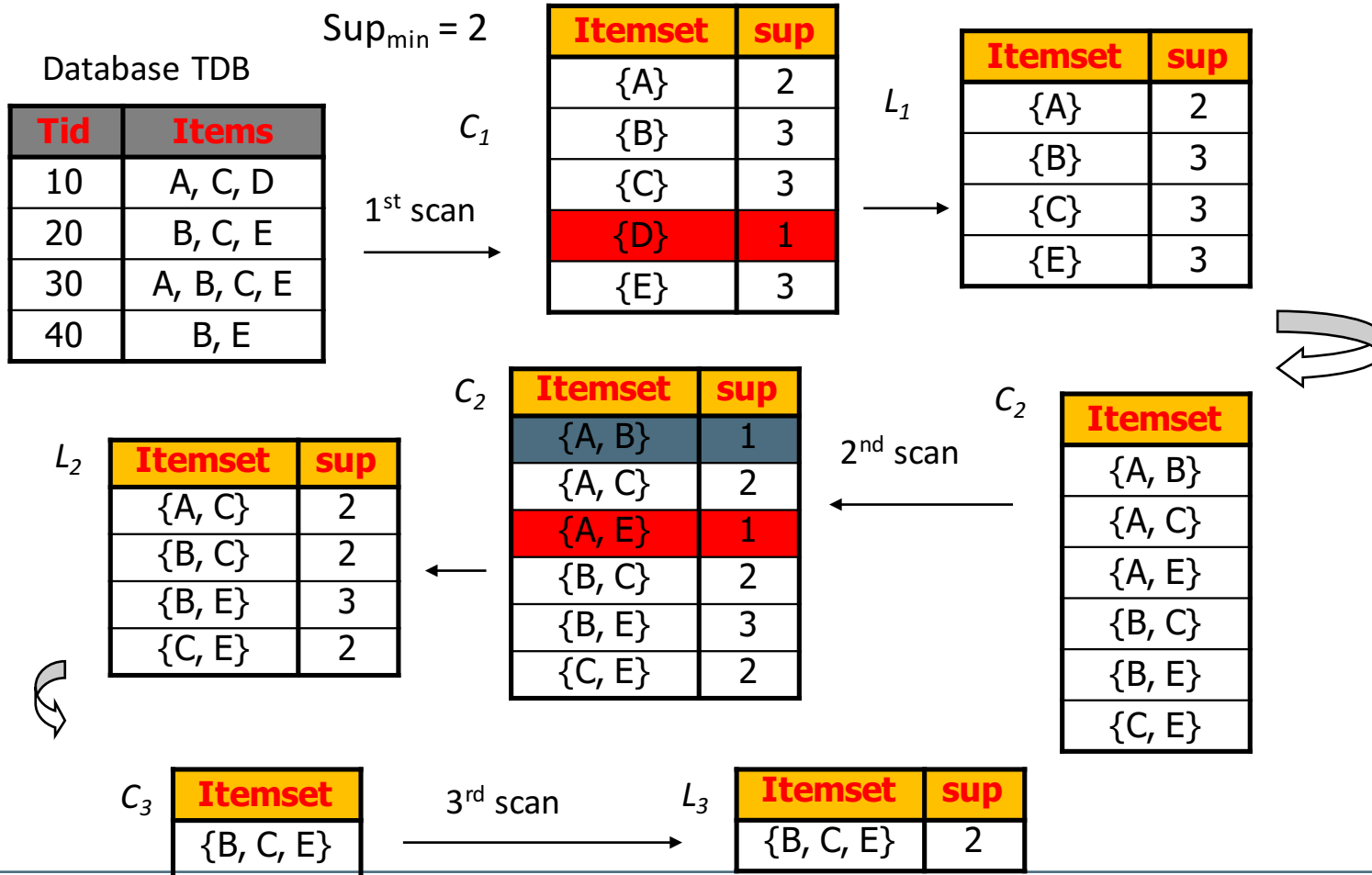
*subset function*

*Prune by support count (ask DB)*

*Subset function:*

- For each transaction  $T$  in DB, the subset function must check all candidates in the candidate set  $C_k$  whether they are part of the transaction  $T$
- Organize candidates  $C_k$  in a hash tree

# Example



## Apriori overview

---

- Advantages:
  - Apriori property
  - Easy implementation (in parallel also)
- Disadvantages:
  - It requires up to  $|I|$  database scans
  - It assumes that the DB is in memory
- Complexity depends on
  - minSupport threshold
  - Number of items (dimensionality)
  - Number of transactions
  - Average transaction length