Inf-KDDM:
Knowledge Discovery and Data Mining

Winter Term 2019/20

Lecture 3: Frequent Itemsets and Association Rule Mining

Lectures: Prof. Dr. Matthias Renz

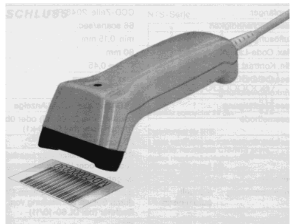
Exercises: Christian Beth

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/

- 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: *{"Diapers"} → {"Beer} (0.5%, 60%)*
- Similarly in an online shop, e.g. Amazon
  - Example: *{"Computer"} → {"MS office"} (50%, 80%)*
- University library
  - Items are the books, transactions are the books borrowed by a student during the semester
  - Example: *{"Kumar book"} →{"Weka book"} (60%, 70%)*
- University
  - Items are the courses, transactions are the courses that are chosen by a student
  - Example: *{"CS"} ∧ {"DB"} →{"Grade A"} (1%, 75%)*
- … and many other applications.
- Also, frequent patter 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 $I = \{i_1, ..., i_m\}$
  - 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 $X = (x_1, x_2, ..., x_k)$, such as $x_1 \leq x_2 \leq ... \leq x_k$

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

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

Let X be an itemset.

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

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

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

  $supportCount(X) = |cover(X)|$

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

  $support(X) = P(X) = supportCount(X) / |DB|$

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

  $support(X) \geq s$

- L_k: the set of frequent *k*-itemsets
  - *L* comes from "Large" ("large itemsets"), another term for "frequent itemsets"

Example: Itemsets

- I = {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

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

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.:

$$\{X \subseteq I \mid support(X) \geq s\}$$

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=∅ .

- Association rules: rules of the form

$$\boxed{X} \rightarrow \boxed{Y}$$

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

support(X→Y) = 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*

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

E_X := Event that itemset X appears in a transaction

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

Explain the rules:

- {Diapers} → {Beer} (0.5%, 60%)
- {Toast bread} → {Toast cheese} (50%, 90%)

Example: association rules

- I = {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:

- {Butter}→{Bread} (20%, 25%)
  - support(Butter ∪Bread)=1/5=20%
  - support(Butter)=4/5=80%
  - Confidence = 20%/80%=1/4=25%
- {Butter, Milk} → Sugar (60%, 75%)
  - support(Butter, Milk ∪ Sugar) = 3/5=60%
  - Support(Butter,Milk) = 4/5=80%
  - 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.:

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

  - These rules are called *strong*.

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

Association rules:

$A \Rightarrow C$ (Support = 50%, Confidence= 66.6%)

$C \Rightarrow A$ (Support = 50%, Confidence= 100%)

…

Solving the problems

- Problem 1 (FIM): Find all frequent itemsets in *DB*, i.e.: $\{X \subseteq I \mid support(X) \geq s\}$
- Problem 2 (ARM): Find all association rules $X \rightarrow Y$ in *DB*, w.r.t. min support *s* and min confidence *c*, i.e.,: $\{X \rightarrow Y \mid support(X \cup Y) \geq s, confidence(X \rightarrow Y) \geq c, X,Y \subseteq I \text{ and } X \cap Y = \emptyset\}$

- Problem 1 is part of Problem 2:
  - Once we have support(X ∪Y) and support(X), we can check if X→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 $O(\binom{|I|}{k})$ such subsets
      - Total cost: $O(2^{|I|})$

      => Apriori-algorithm and variants

    FIM problem
  - Step 2: Generate the association rules w.r.t. min confidence *c*:
    from frequent itemsets *X*, generate $Y \rightarrow (X - Y)$, $Y \subset X$, $Y \neq \emptyset$, $Y \neq X$

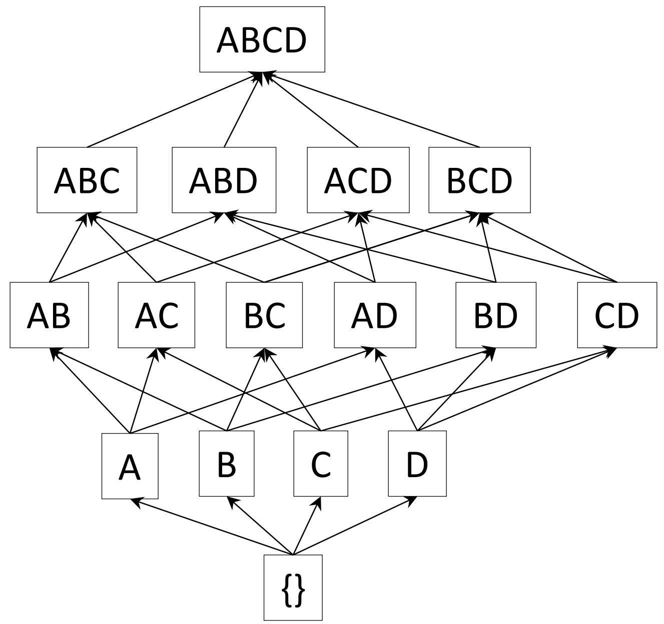
Step 1(FIM) is the most costly, so the overall performance of an association rules mining algorithm is determined by this step.

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:

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

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

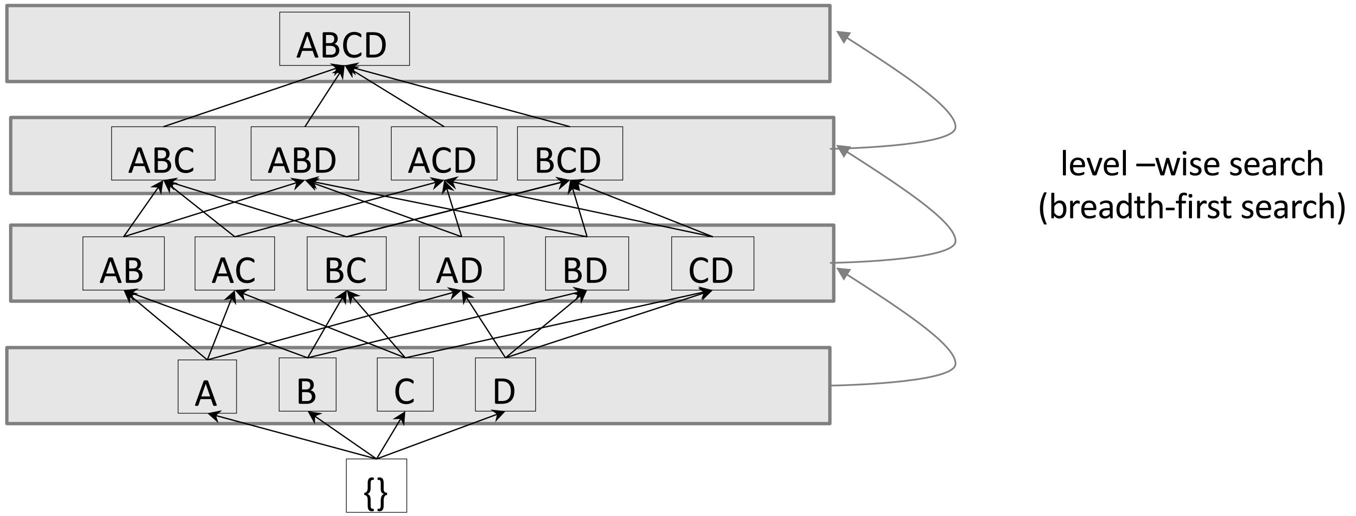


Outline

- 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}

{beer, diaper, nuts} 2

{beer, diaper} ≥2

minSupport =2

  - 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

{beer, diaper, nuts} <=1

{beer, diaper} 1

minSupport =2

Illustration of the Apriori property

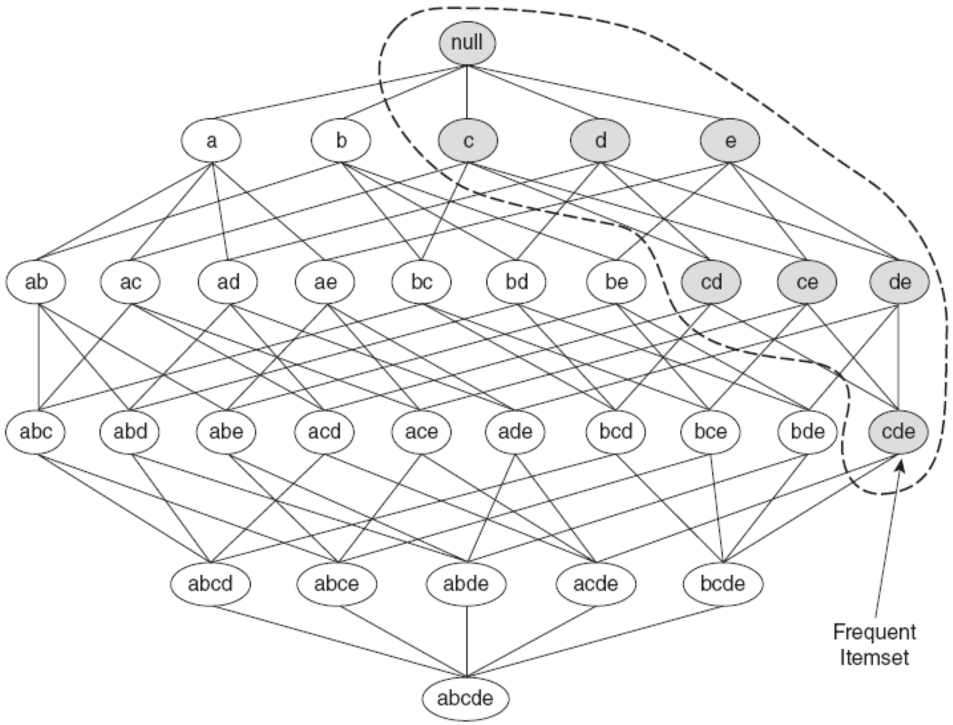


Figure 6.3. An illustration of the *Apriori* principle. If $\{c, d, e\}$ is frequent, then all subsets of this itemset are frequent.

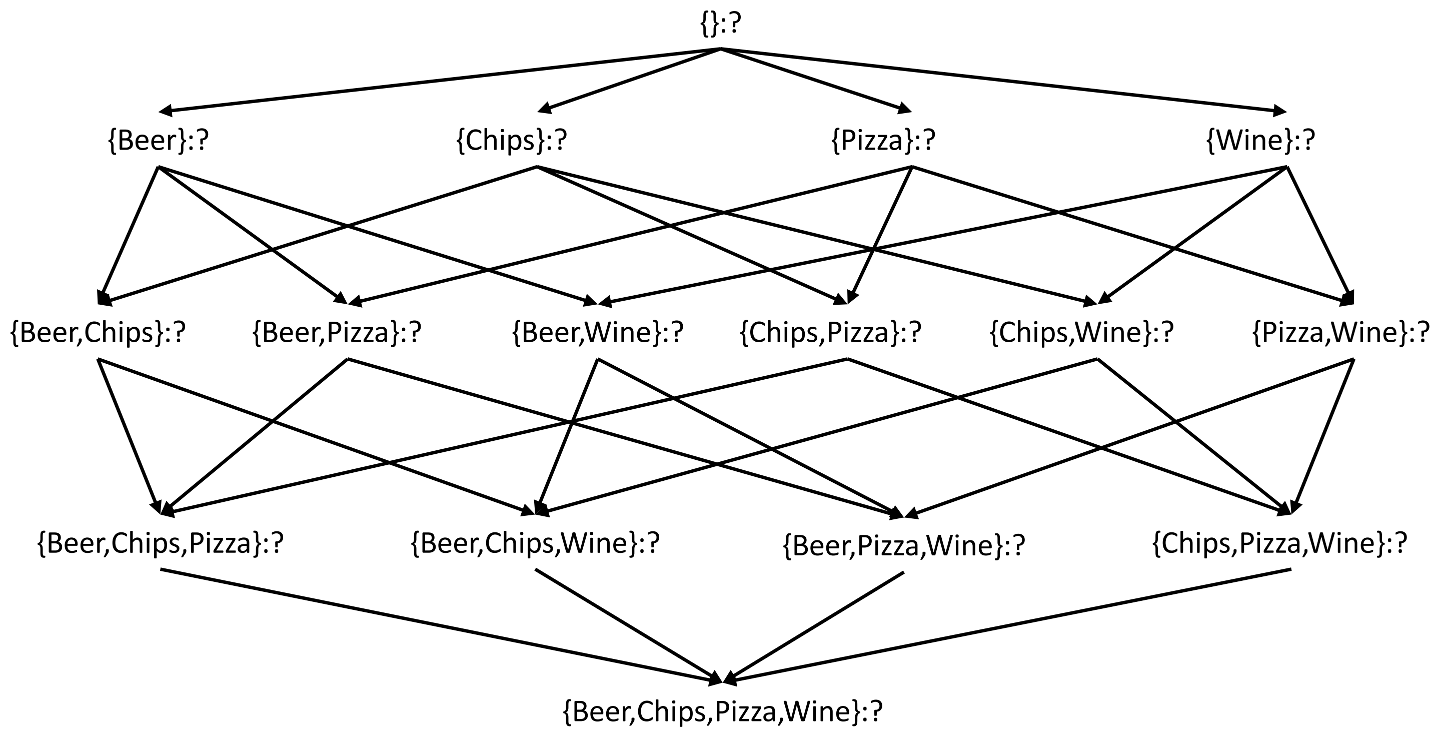
Search space and pruning

- 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

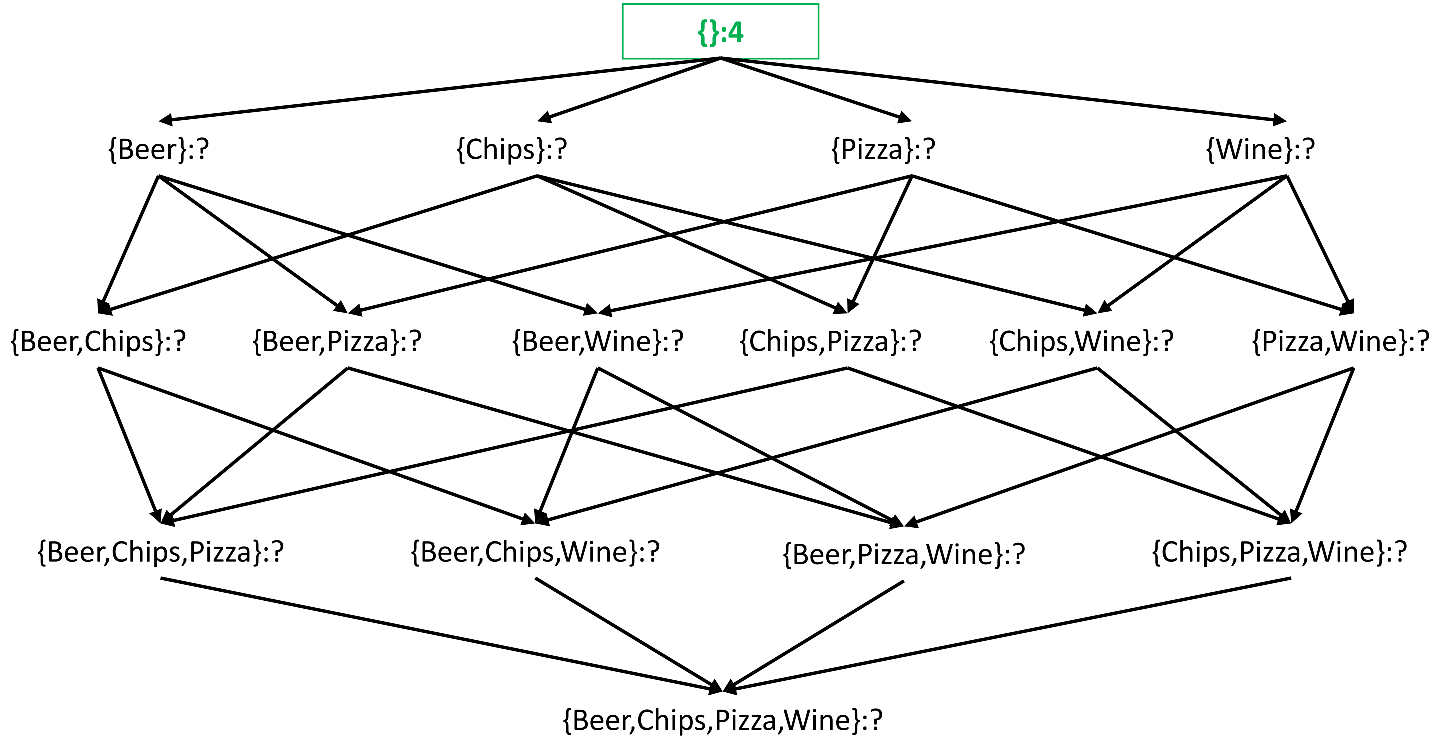


Transaction Database

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

minSupp = 2

Search space and pruning

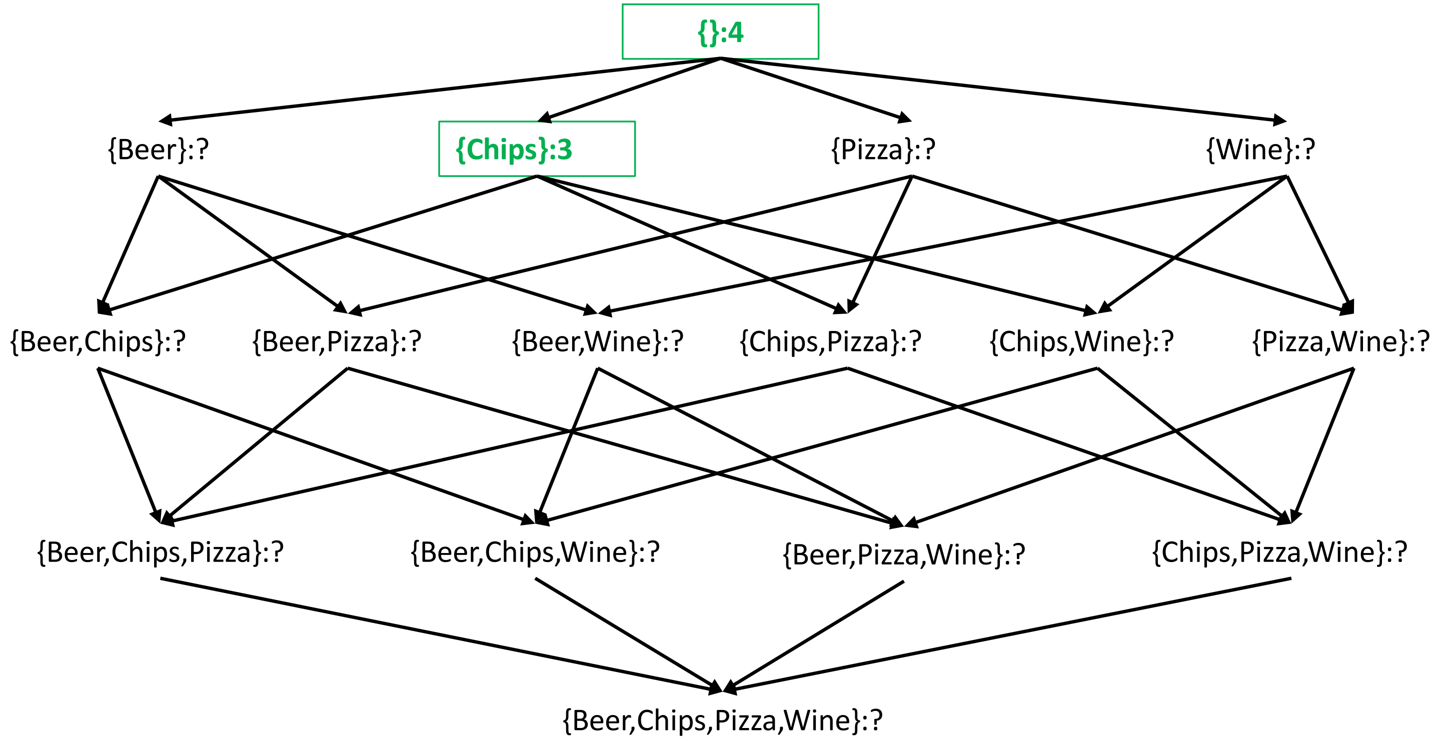


Transaction Database

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

minSupp = 2

Search space and pruning

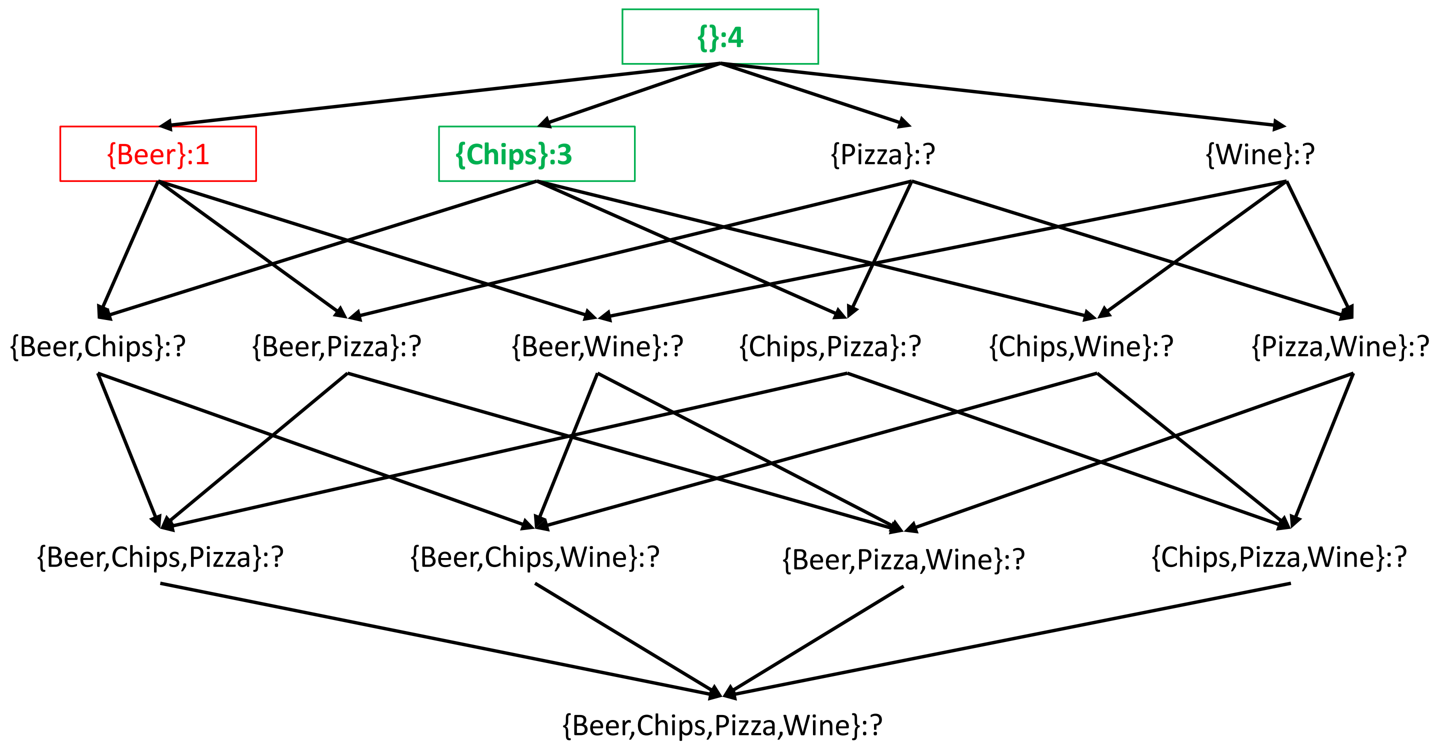


Transaction Database

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

minSupp = 2

Search space and pruning

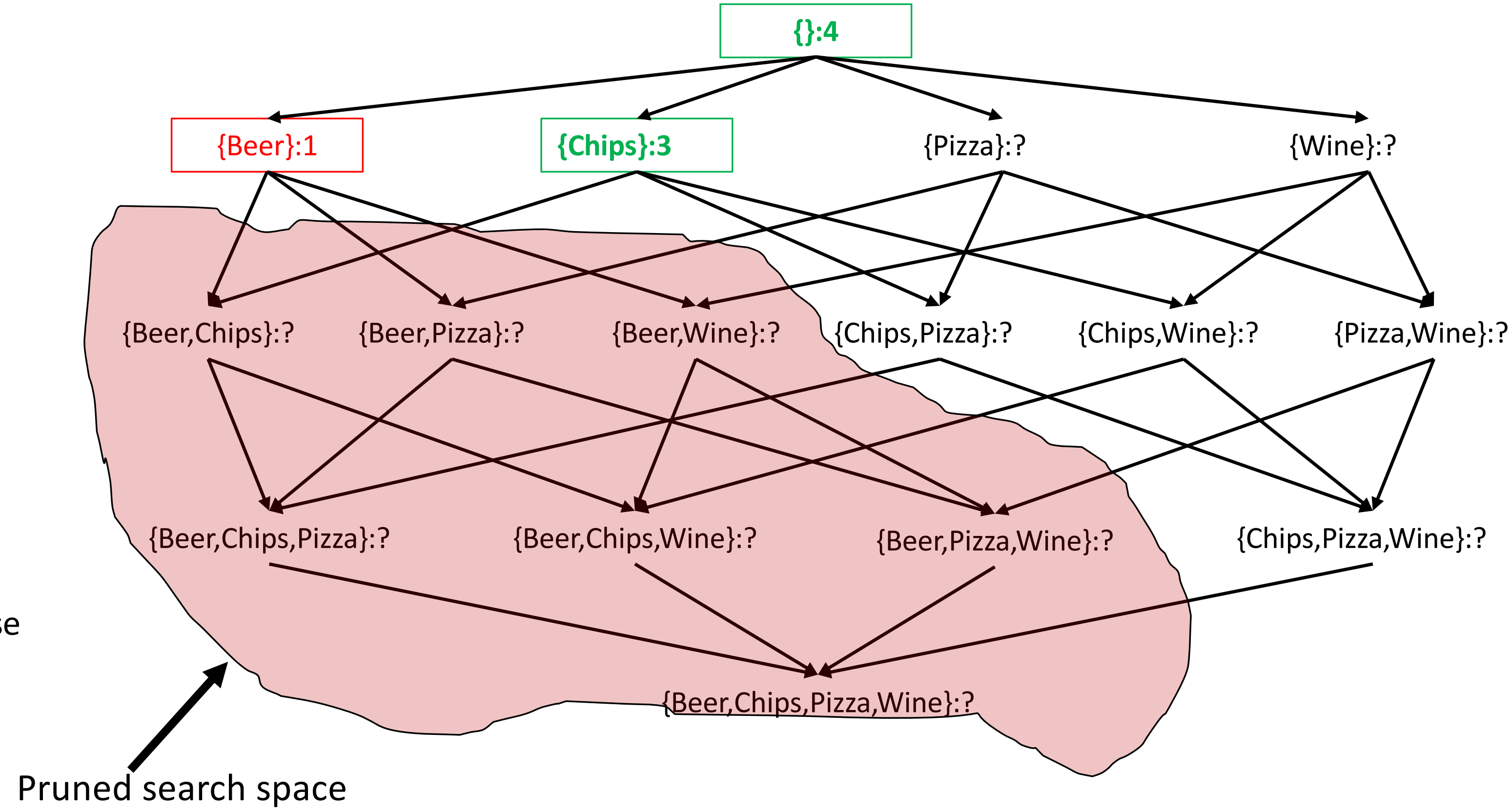


Transaction Database

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

minSupp = 2

Search space and pruning

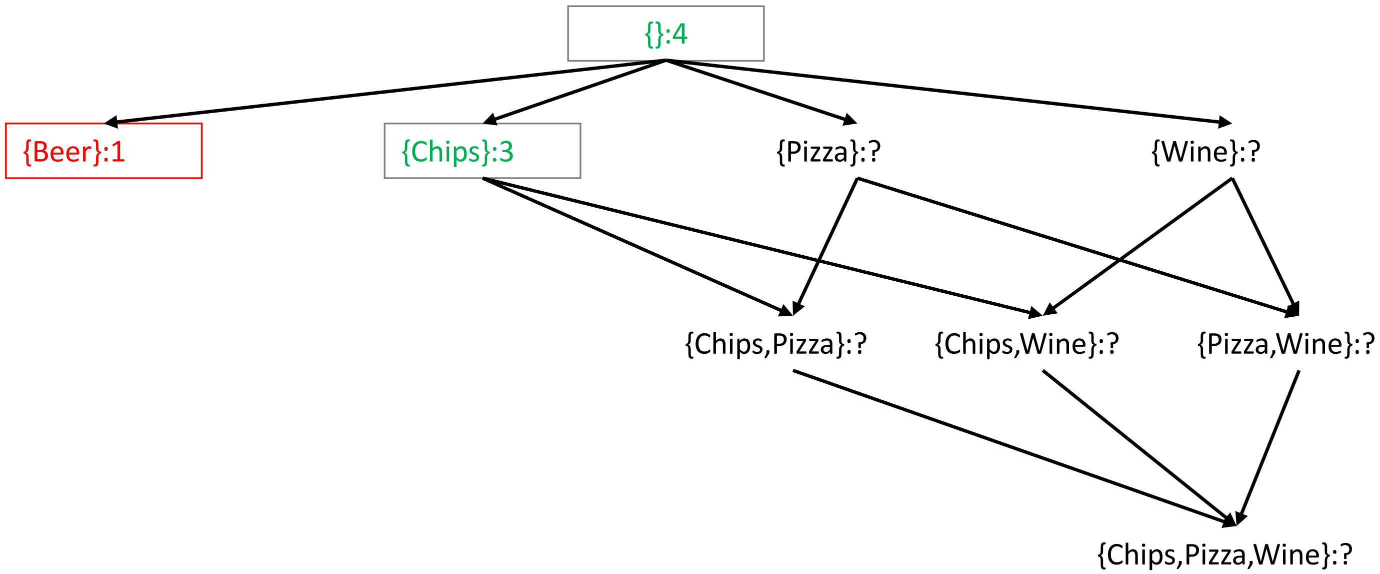


Transaction Database

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

minSupp = 2

Search space and pruning

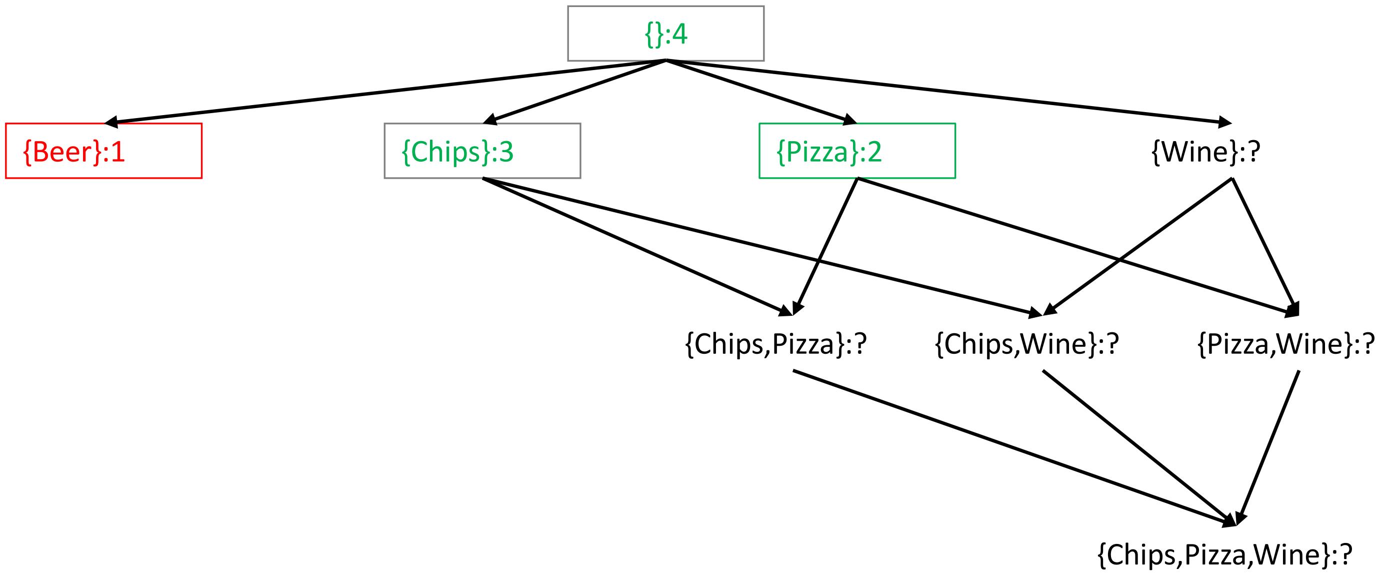


Transaction Database

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

minSupp = 2

Search space and pruning

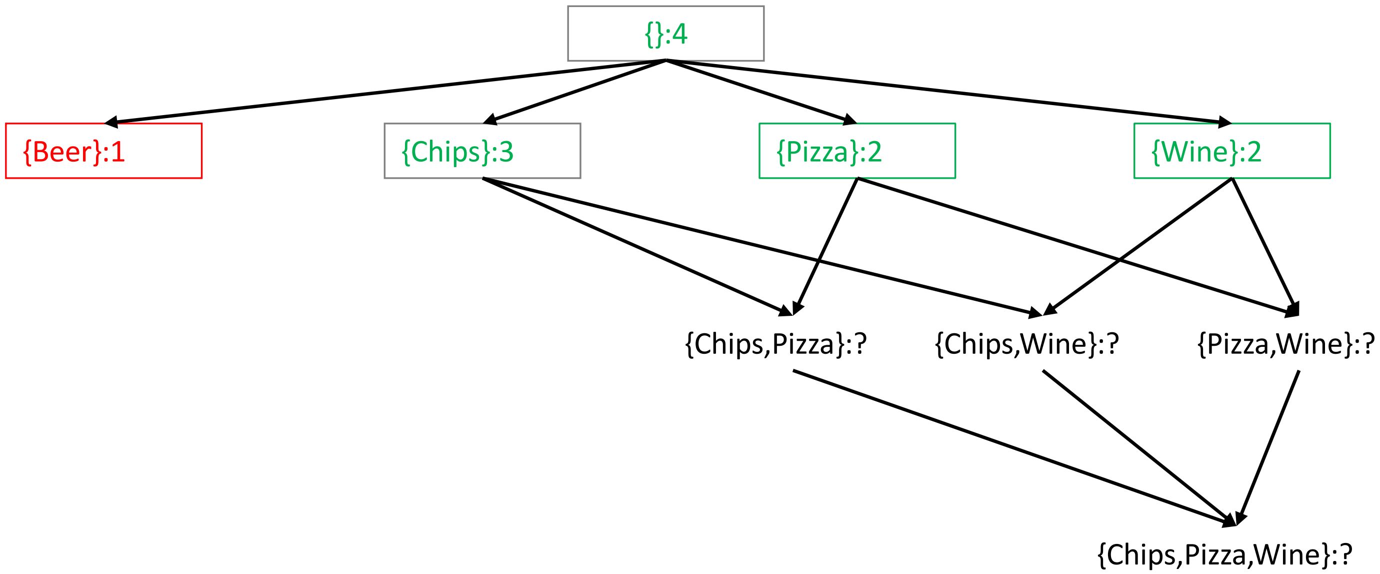


Transaction Database

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

minSupp = 2

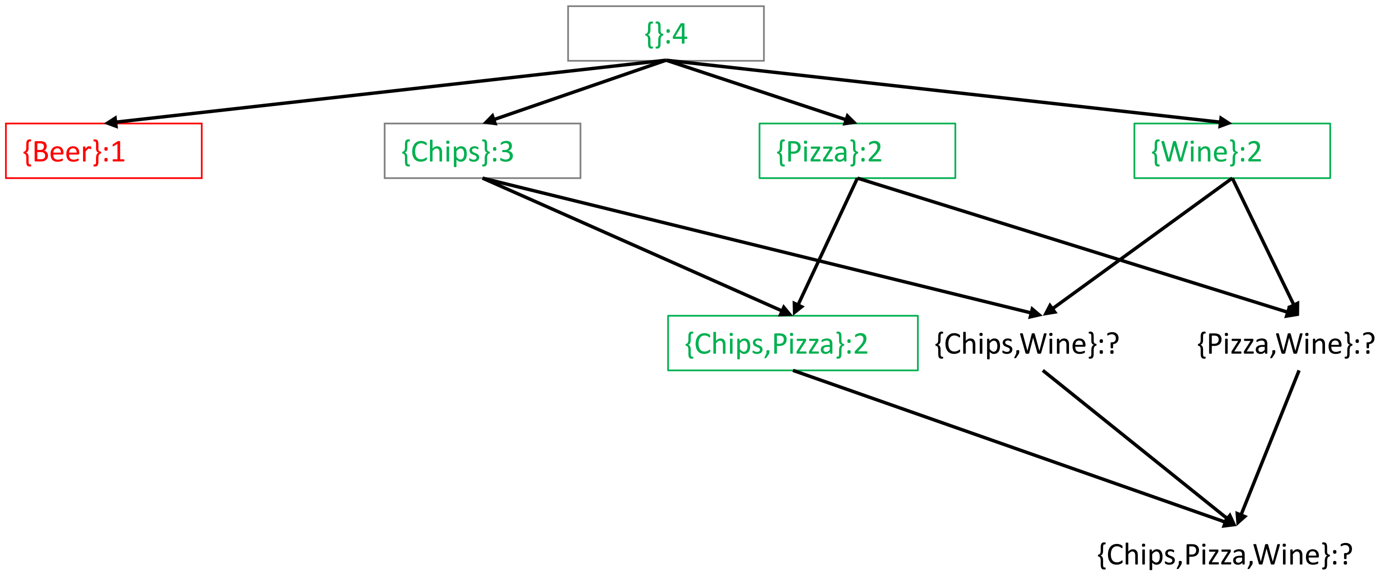
Search space and pruning



Transaction Database
{Chips, Pizza}
{Beer, Chips}
{Chips, Pizza, Wine}
{Wine}

minSupp = 2

Search space and pruning

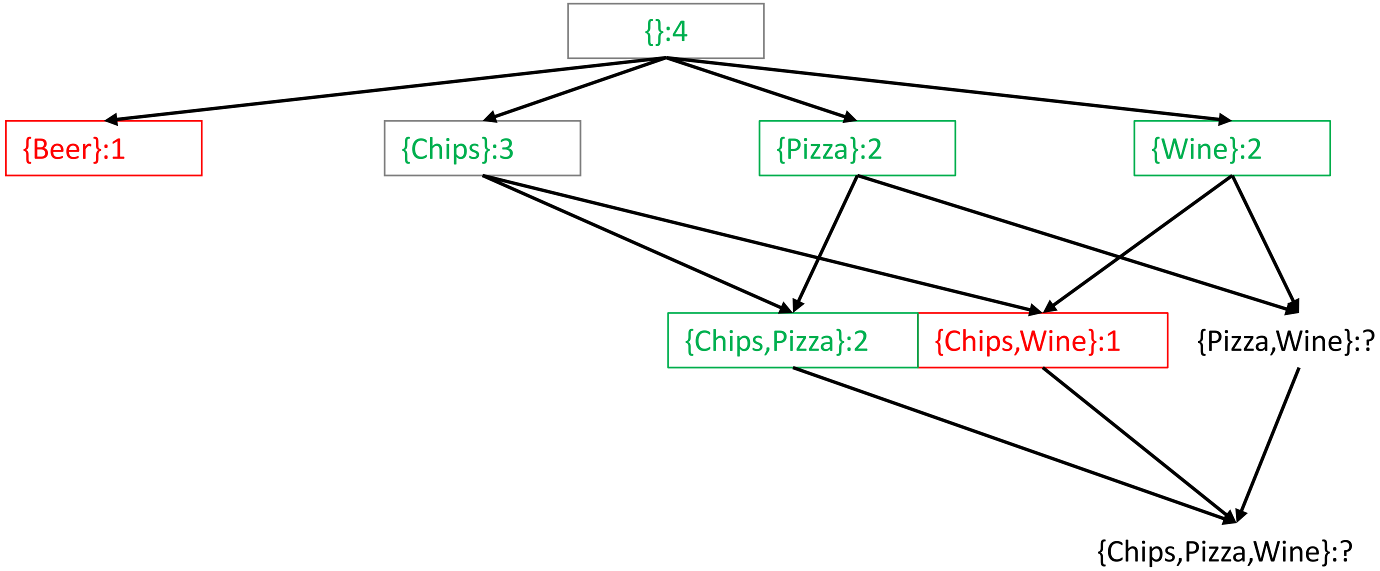


Transaction Database

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

minSupp = 2

Search space and pruning

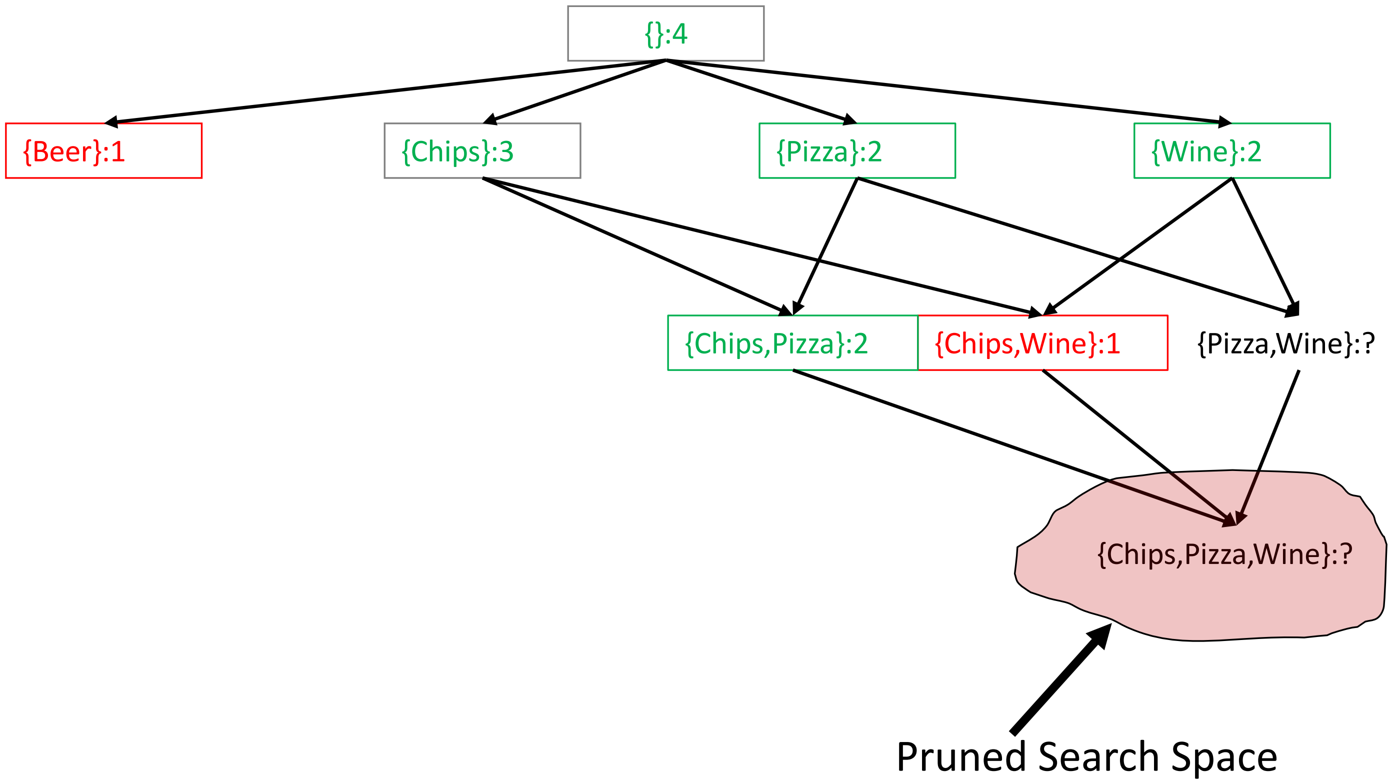


Transaction Database

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

minSupp = 2

Search space and pruning

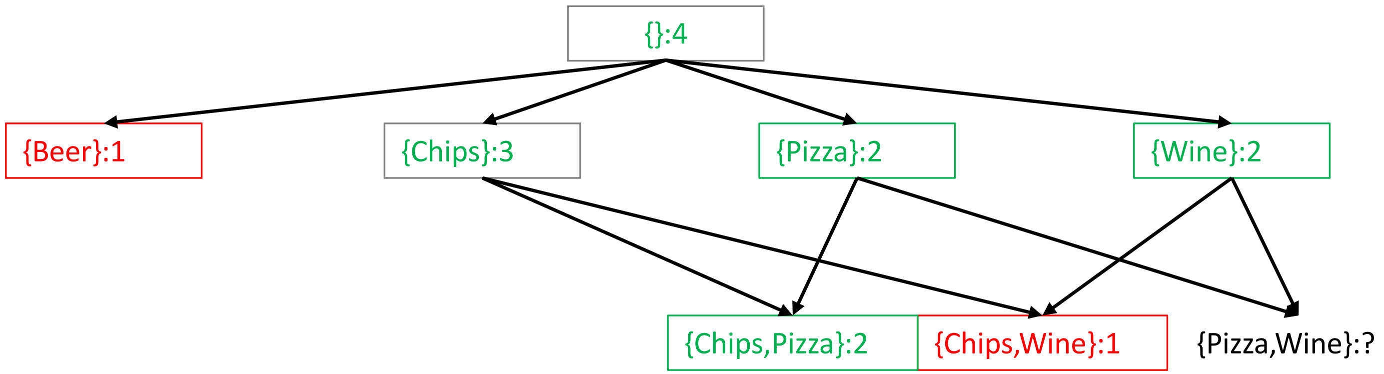


Transaction Database

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

minSupp = 2

Search space and pruning

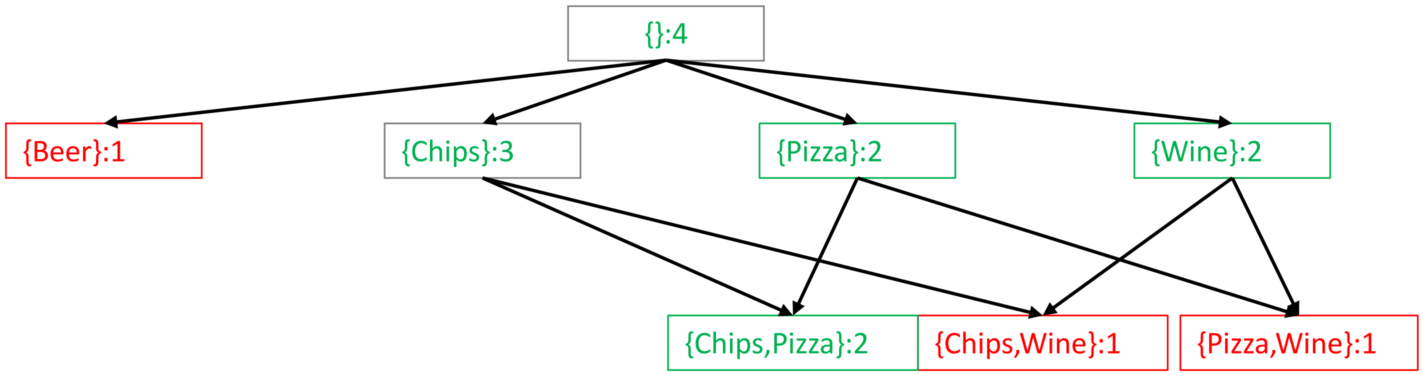


Transaction Database

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

minSupp = 2

Search space and pruning



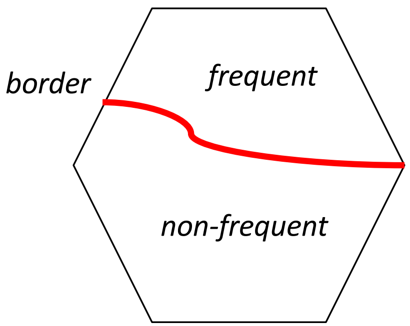
Transaction Database

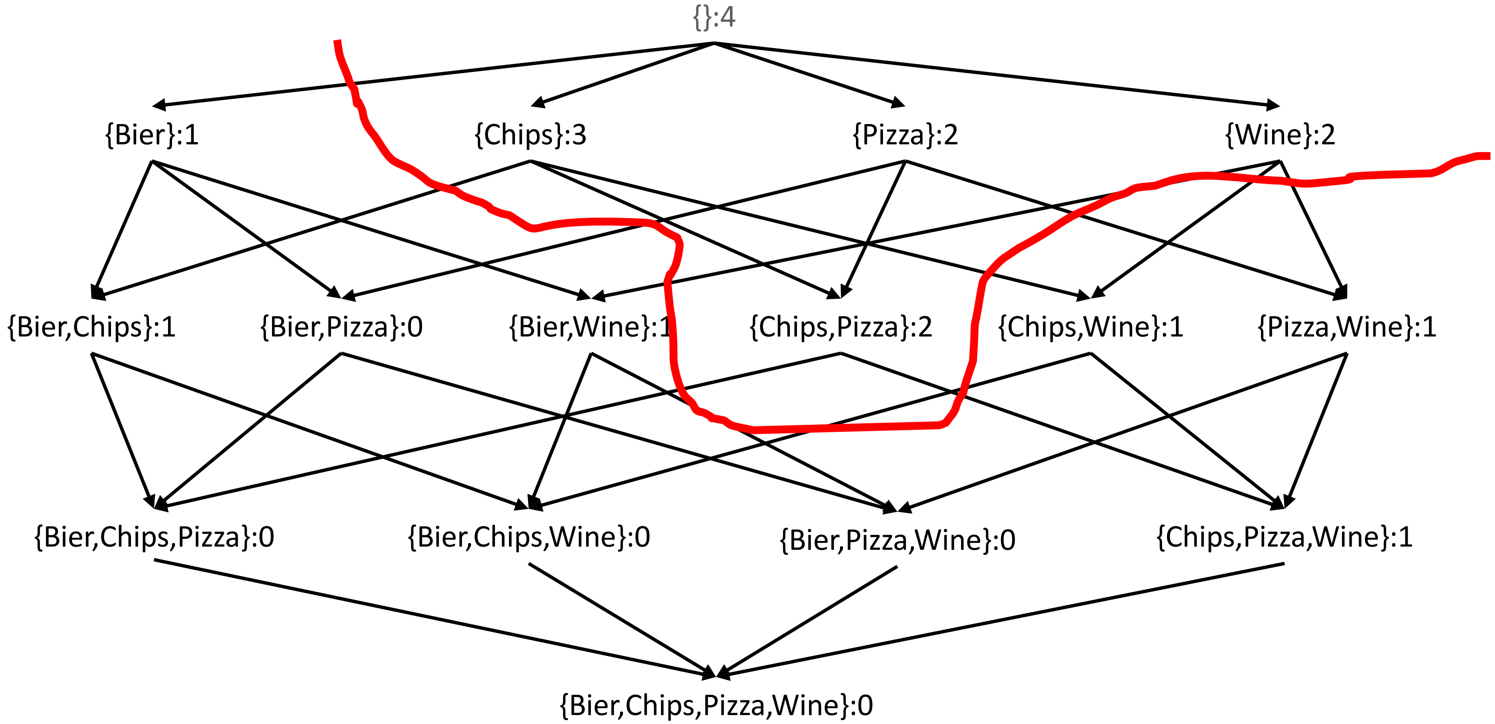
{Chips, Pizza}
{Beer, Chips}
{Chips, Pizza, Wine}
{Wine}

minSupp = 2

Search space and pruning

- Border itemsets *X*: all subsets Y ⊂ X are frequent, all supersets Z ⊃ X are not frequent





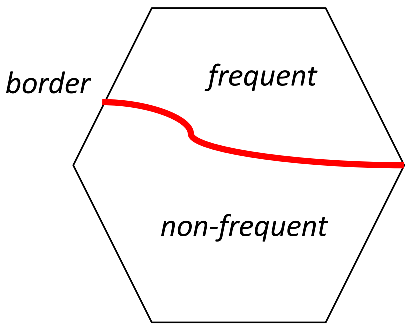
Transaction Database

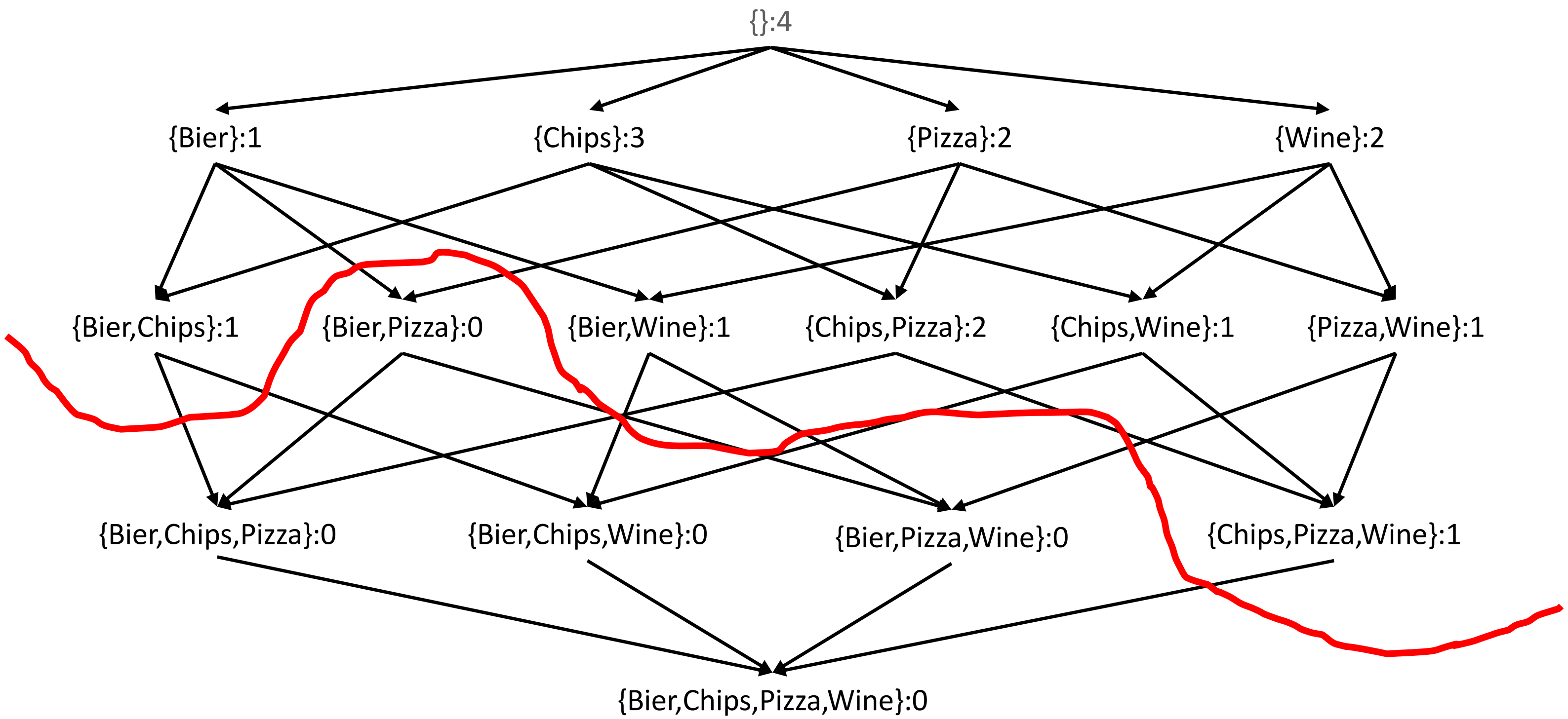
{Chips, Pizza}
{Beer, Chips}
{Chips, Pizza, Wine}
{Wine}

minSupport *s* = 2

Search space and pruning

- Border itemsets *X*: all subsets Y ⊂ X are frequent, all supersets Z ⊃ X are not frequent



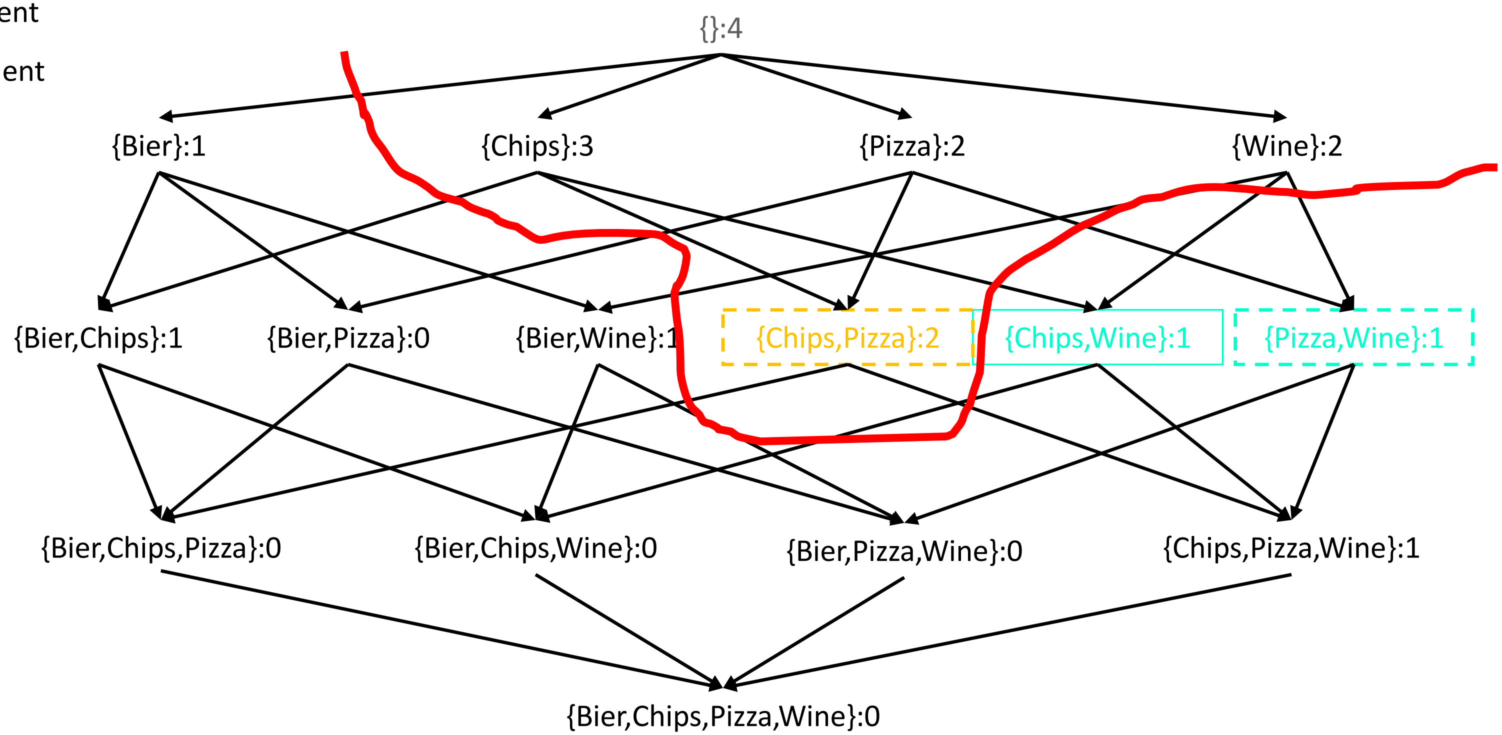


Transaction Database
{Chips, Pizza}
{Beer, Chips}
{Chips, Pizza, Wine}
{Wine}

minSupport *s* = **1**

Search space and pruning

- Border itemsets *X*: all subsets Y ⊂ X are frequent, all supersets Z ⊃ 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 = {frequent items};
for (k = 1; L_k !=$\emptyset$; 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

$Sup_{min} = 2$

Database TDB

Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

1^{st} scan

C_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

L_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{E}	3

C_2

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

2^{nd} scan

C_2

Itemset	sup
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

L_2

Itemset	sup
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

C_3

Itemset
{B, C, E}

3^{rd} scan

L_3

Itemset	sup
{B, C, E}	2

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

Outline

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

Association Rules Mining

- (Recall the) 2-step method to extract the association rules:
  - Determine the frequent itemsets w.r.t. min support s ⟵ FIM problem (Apriori)
  - Generate the association rules w.r.t. min confidence *c*.

- Regarding step 2, the following method is followed:
  - For every frequent itemset X
  - for every subset *Y* of *X*: *Y*≠Ø, *Y* ≠*X*, the rule *Y* → (*X* - *Y*) is formed
  - Remove rules that violate min confidence c

$$confidence(Y \rightarrow (X - Y)) = \frac{support\_count(X)}{support\_count(Y)}$$

  - Store the frequent itemsets and their supports in a hash table
    - no database access!

Let *X*={1,2,3} be frequent
There are 6 candidate rules that can be generated from *X*:

- {1,2}→3
- {1,3}→2
- {2,3}→1
- {1}→{2,3}
- {2}→{1,3}
- {3}→{1,2}

To identify strong rules, we can use the support counts (already computed during the FIM step)

Pseudocode

Input:

D //**Database of transactions**
I //**Items**
L //**Large itemsets**
s //**Support**
α //**Confidence**

Output:

R //**Association Rules satisfying** $s\ and\ \alpha$

ARGen Algorithm:

$R = \emptyset$;
for each $l \in L$ **do**
  for each $x \subset l$ **such that** $x \neq \emptyset$ **and** $x \neq l$ **do**
    if $\frac{support(l)}{support(x)} \geq \alpha$ **then**
      $R = R \cup \{x \Rightarrow (l - x)\}$;

Confidence-based pruning

- How to efficiently generate rules from frequent itemsets?
- Confidence does not follow the monotonicity property
  - i.e., confidence ($X \rightarrow Y$) can be >,<,= to confidence ($X' \rightarrow Y'$), $X' \subseteq X$, $Y' \subseteq Y$
  - e.g., confidence($ABC \rightarrow D$) can be larger or smaller than confidence($AB \rightarrow D$)

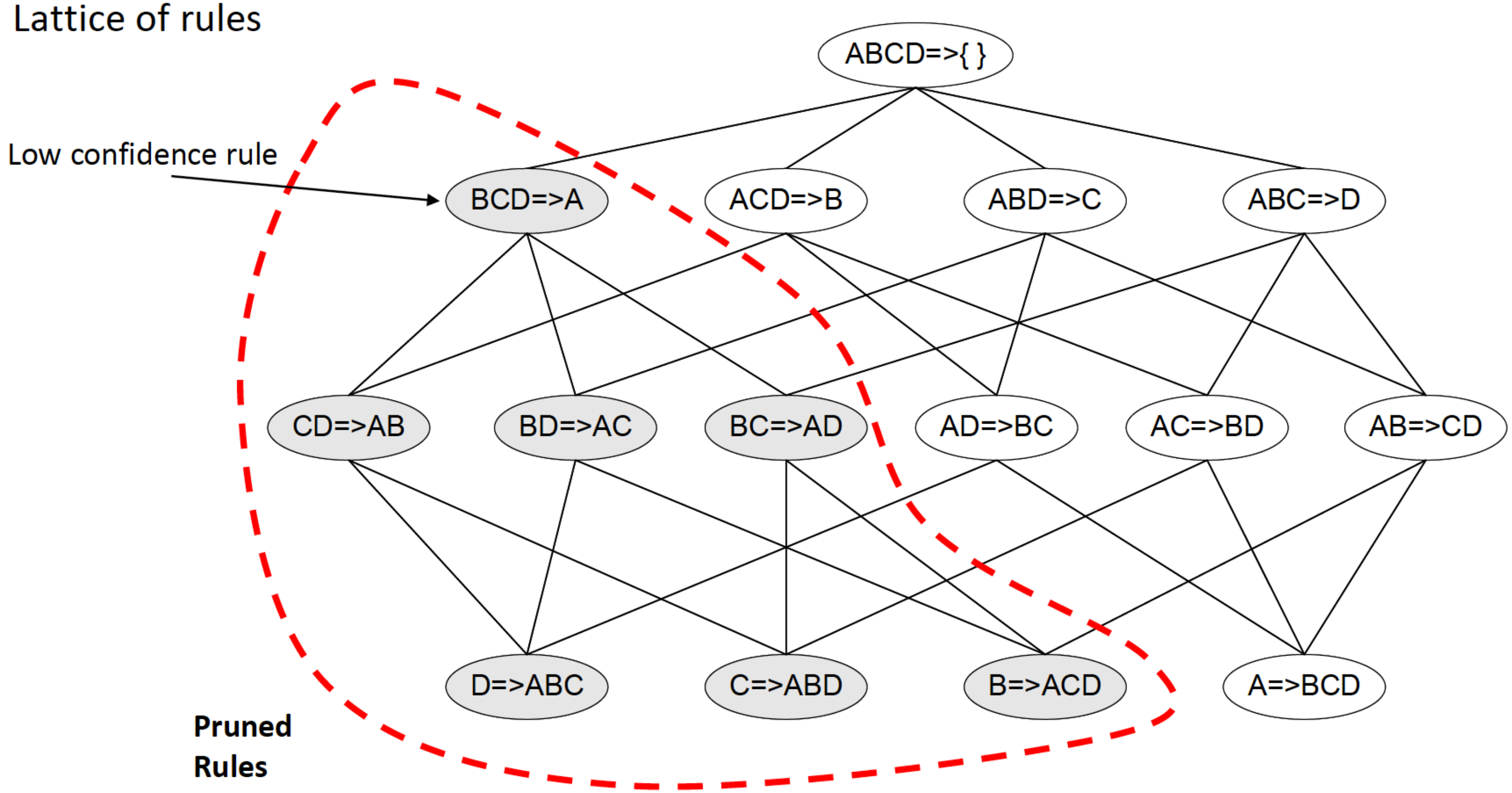
- But the confidence of rules generated from the same itemset does

If rule $X \rightarrow Y\text{-}X$ does not satisfy the minConfidence threshold, then any rule $X' \rightarrow Y\text{-}X'$, where $X' \subseteq X$, must not satisfy the minConfidence threshold as well.

  - For example, for X={ABCD}, then
    - confidence($ABC \rightarrow D$) $\geq$ confidence($AB \rightarrow CD$) $\geq$ confidence($A \rightarrow BCD$)

Confidence-based pruning

- Lattice of rules



Example

tid	X_T
1	{Bier, Chips, Wine}
2	{Bier, Chips}
3	{Pizza, Wine}
4	{Chips, Pizza}

Transaction database

I = {Bier, Chips, Pizza, Wine}

Itemset	Cover	Sup.	Freq.
{}	{1,2,3,4}	4	100 %
{Bier}	{1,2}	2	50 %
{Chips}	{1,2,4}	3	75 %
{Pizza}	{3,4}	2	50 %
{Wine}	{1,3}	2	50 %
{Bier, Chips}	{1,2}	2	50 %
{Bier, Wine}	{1}	1	25 %
{Chips, Pizza}	{4}	1	25 %
{Chips, Wine}	{1}	1	25 %
{Pizza, Wine}	{3}	1	25 %
{Bier, Chips, Wine}	{1}	1	25 %

Rule	Sup.	Freq.	Conf.
{Bier} $\Rightarrow$ {Chips}	2	50 %	100 %
{Bier} $\Rightarrow$ {Wine}	1	25 %	50 %
{Chips} $\Rightarrow$ {Bier}	2	50 %	66 %
{Pizza} $\Rightarrow$ {Chips}	1	25 %	50 %
{Pizza} $\Rightarrow$ {Wine}	1	25 %	50 %
{Wine} $\Rightarrow$ {Bier}	1	25 %	50 %
{Wine} $\Rightarrow$ {Chips}	1	25 %	50 %
{Wine} $\Rightarrow$ {Pizza}	1	25 %	50 %
{Bier, Chips} $\Rightarrow$ {Wine}	1	25 %	50 %
{Bier, Wine} $\Rightarrow$ {Chips}	1	25 %	100 %
{Chips, Wine} $\Rightarrow$ {Bier}	1	25 %	100 %
{Bier} $\Rightarrow$ {Chips, Wine}	1	25 %	50 %
{Wine} $\Rightarrow$ {Bier, Chips}	1	25 %	50 %

Evaluating Association Rules 1/2

Interesting and misleading association rules

Example:

- Database on the behavior of students in a school with 5.000 students
- Itemsets:
  - 60% of the students play Soccer,
  - 75% of the students eat chocolate bars
  - 40% of the students play Soccer and eat chocolate bars
- Association rules: *{"Play Soccer"}* → *{"Eat chocolate bars"}*, confidence = 40%/60%= 67%
  - The rule has a high confidence, however:

    {"Eat chocolate bars"}, support= 75%, regardless of whether they play soccer.
  - Thus, knowing that one is playing soccer decreases his/her probability of eating chocolate (from 75%→67%)
  - Therefore, the rule *{"Play Soccer"}* → *{"Eat chocolate bars"}* is misleading despite its high confidence

Evaluating Association Rules 2/2

Task: Filter out misleading rules

Let {*A*} → {*B*}

- Measure of "interestingness"-score of a rule:

$$interest = \frac{support(A \cup B)}{support(A)} - support(B)$$

  - the higher the value the more interesting the rule is

- Measure of dependent/correlated events:

$$lift = \frac{support(A \cup B)}{support(A)support(B)}$$

  - the ratio of the *observed* support to that *expected* if X and Y were independent.
  - Lift > 1 means that the rule is interesting, lift < 1 means that the presence of one item has negative effect on presence of other item and vice versa.

Measuring Quality of Association Rules

For a rule A→ B

- Support $support(A \cup B)$ $P(E_A \cap E_B)$ E_X := Event that itemset X appears in a transaction
  - e.g. support(milk, bread, butter)=20%, i.e. 20% of the transactions contain these
- Confidence $\frac{support(A \cup B)}{support(A)}$ $\frac{P(E_A \cap E_B)}{P(E_A)}$
  - e.g. confidence(milk, bread→ butter)=50%, i.e. 50% of the times a customer buys milk and bread, butter is bought as well.
- Lift $\frac{support(A \cup B)}{support(A)support(B)}$ $\frac{P(EA \cap E_B)}{P(E_A)P(EB)}$
  - e.g. lift(milk, bread→ butter)=20%/(40%*40%)=1.25. the observed support is 20%, the expected (if they were independent) is 16%.