Chapter 5
Association Analysis: Basic Concepts

Introduction to Data Mining, 2nd Edition

by

Tan, Steinbach, Karpatne, Kumar

Association Rule Mining

association analysis:
computationaly expensive

- Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction

Market-Basket transactions

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Example of Association Rules

$\{Diaper\} \rightarrow \{Beer\}$,
$\{Milk, Bread\} \rightarrow \{Eggs,Coke\}$,
$\{Beer, Bread\} \rightarrow \{Milk\}$,

Implication means co-occurrence, not causality!

Definition: Frequent Itemset

- **Itemset** unique set
  - A collection of zero or more items
    - Example: {Milk, Bread, Diaper} 每个transaction都含有一部分的Itemset(subset of itemset)
  - k-itemset
    - An itemset that contains k items
- **Support count** (σ) 一组itemset的出现频率
  - Frequency of occurrence of an itemset
  - E.g. σ({Milk, Bread,Diaper}) = 2
- **Support** 一组itemset出现的概率
  - Fraction of transactions that contain an itemset
  - E.g. s({Milk, Bread, Diaper}) = 2/5
- **Frequent Itemset** 出现概率最高的itemset组合
  - An itemset whose support is greater than or equal to a *minsup* threshold

Itemset: {I1,I2,I3…In}
Transaction:{t1,t2,t3…tn}

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Definition: Association Rule

- **Association Rule**
  - An implication expression of the form X → Y, where X and Y are itemsets
  - Example:
    {Milk, Diaper} → {Beer}

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

- **Rule Evaluation Metrics**

X-antecedent 前因
Y-consequent 后果

  - Support (s)
    - Fraction of transactions that contain both X and Y

在所有transaction中的出现概率

  - Confidence (c)
    - Measures how often items in Y appear in transactions that contain X

Y中的item在X中出现的概率

Example:

$$\{\text{Milk}, \text{Diaper}\} \Rightarrow \{\text{Beer}\}$$

$$s = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{|T|} = \frac{2}{5} = 0.4$$

$$c = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{\sigma(\text{Milk}, \text{Diaper})} = \frac{2}{3} = 0.67$$

Association Rule Mining Task

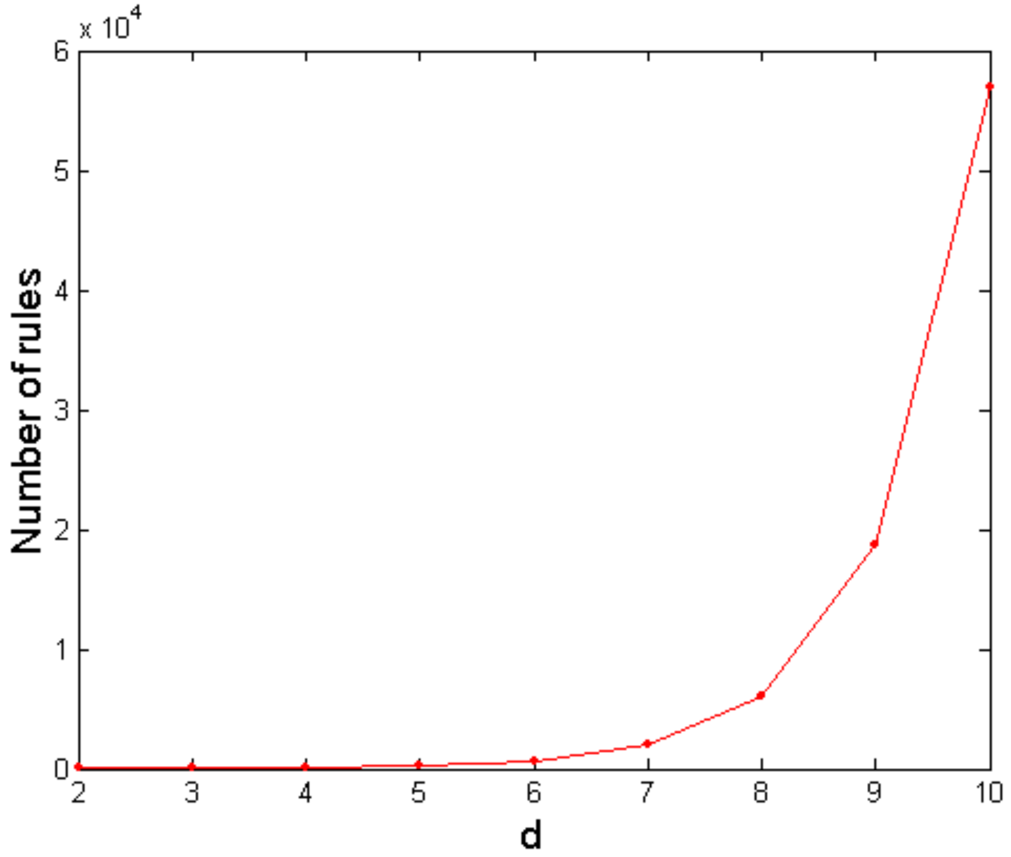
- Given a set of transactions T, the goal of association rule mining is to find all rules having
  - support ≥ *minsup* threshold
  - confidence ≥ *minconf* threshold

- Brute-force approach:
  - List all possible association rules
  - Compute the support and confidence for each rule
  - Prune rules that fail the *minsup* and *minconf* thresholds

  ⇒ Computationally prohibitive!

Computational Complexity

- Given d unique items:
  - Total number of itemsets = 2^d
  - Total number of possible association rules:



$$R = \sum_{k=1}^{d-1}\left[\binom{d}{k} \times \sum_{j=1}^{d-k}\binom{d-k}{j}\right]$$
$$= 3^d - 2^{d+1} + 1$$

If d=6, R = 602 rules

Mining Association Rules

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Example of Rules:

{Milk,Diaper} → {Beer} (s=0.4, c=0.67)
{Milk,Beer} → {Diaper} (s=0.4, c=1.0)
{Diaper,Beer} → {Milk} (s=0.4, c=0.67)
{Beer} → {Milk,Diaper} (s=0.4, c=0.67)
{Diaper} → {Milk,Beer} (s=0.4, c=0.5)
{Milk} → {Diaper,Beer} (s=0.4, c=0.5)

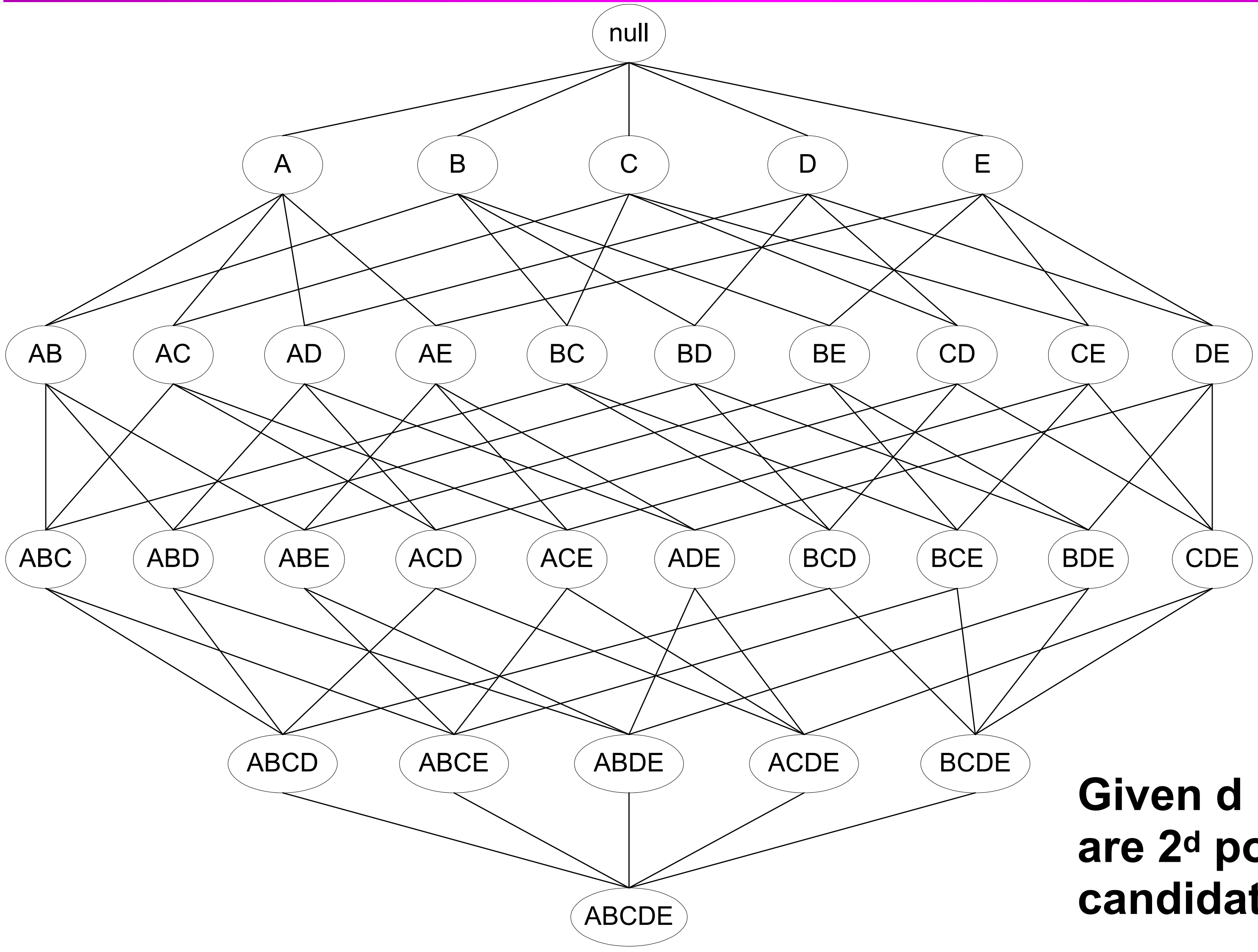
Observations:

- All the above rules are binary partitions of the same itemset:
  {Milk, Diaper, Beer}
- Rules originating from the same itemset have identical support but can have different confidence
- Thus, we may decouple the support and confidence requirements

Mining Association Rules

- Two-step approach:
  1. Frequent Itemset Generation
     - Generate all itemsets whose support ≥ minsup
  2. Rule Generation
     - Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset
- Frequent itemset generation is still computationally expensive

Frequent Itemset Generation

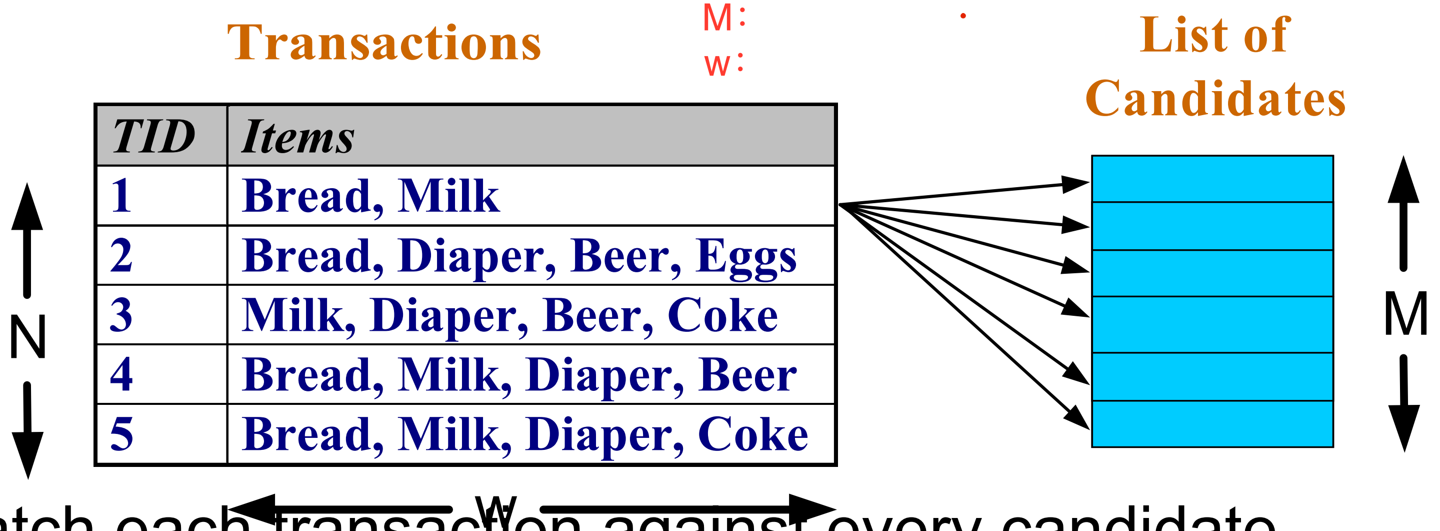


Given d items, there are 2^d possible candidate itemsets

Frequent Itemset Generation

- Brute-force approach:
  - Each itemset in the lattice is a candidate frequent itemset
  - Count the support of each candidate by scanning the database

N:transaction 数量
M：
w：



TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

  - Match each transaction against every candidate
  - Complexity ~ O(NMw) => Expensive since $M = 2^d$!!!

Frequent Itemset Generation Strategies

- Reduce the number of candidates (M)
  - Complete search: $M=2^d$
  - Use pruning techniques to reduce M
- Reduce the number of transactions (N)
  - Reduce size of N as the size of itemset increases
  - Used by DHP and vertical-based mining algorithms
- Reduce the number of comparisons (NM)
  - Use efficient data structures to store the candidates or transactions
  - No need to match every candidate against every transaction

Reducing Number of Candidates

ACE：frequent

A,C,E,AC,AE,CE,ACE都是frequent

- Apriori principle: 如果一个itemset是frequent，那么所有子集都是frequent
  - If an itemset is frequent, then all of its subsets must also be frequent

itemset frequent—subset frequent

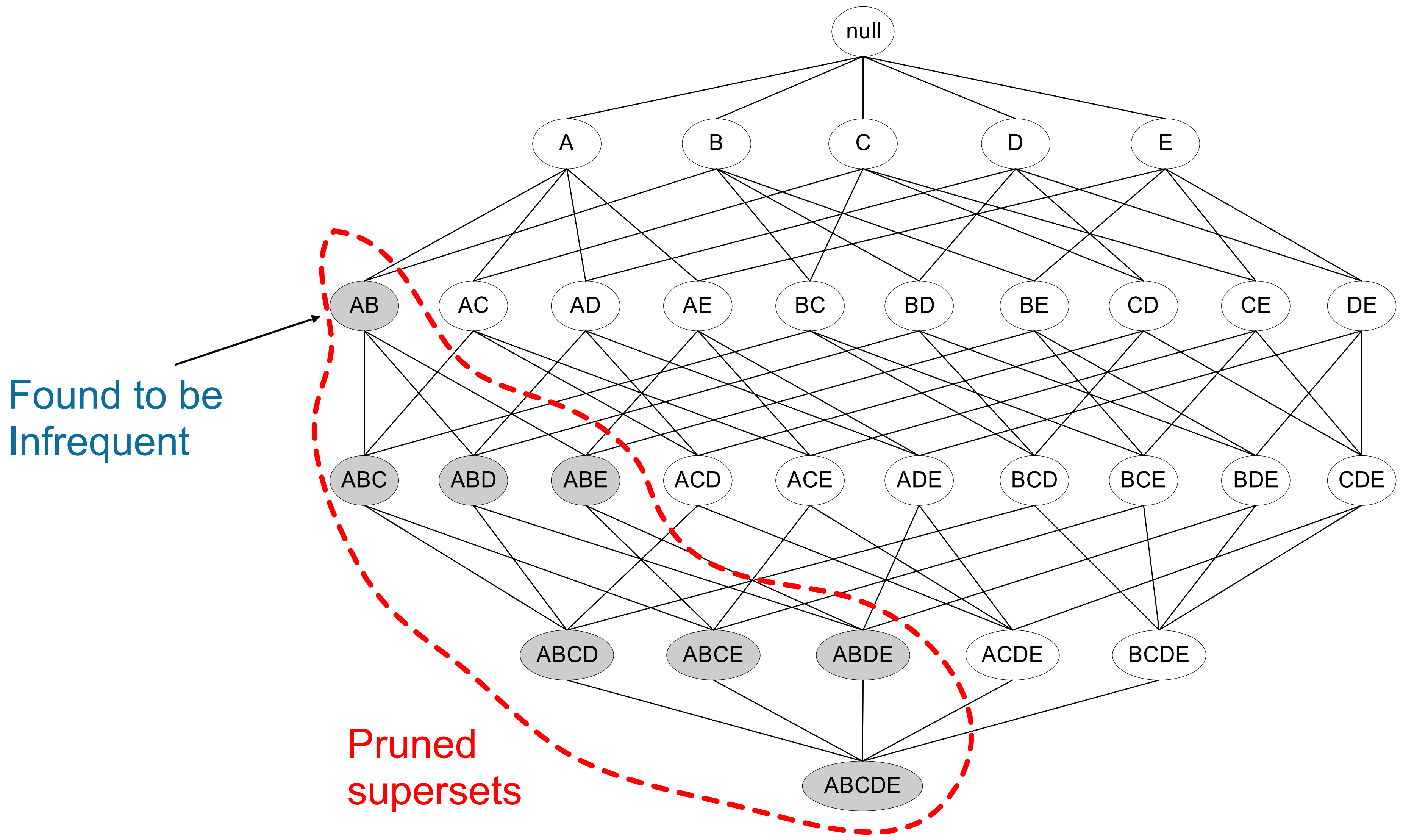
itemset infrequent—superset infrequent 因此这里可以剪枝

- Apriori principle holds due to the following property of the support measure:

$$\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

  - Support of an itemset never exceeds the support of its subsets support值— 子集的support永远会比本身大
  - This is known as the anti-monotone property of support

Illustrating Apriori Principle



Illustrating Apriori Principle

TID	*Items*
1	Bread, Milk
2	Beer, Bread, Diaper, Eggs
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Bread, Coke, Diaper, Milk

$\Rightarrow$

Items (1-itemsets)

Item	Count
Bread	4
~~Coke~~	~~2~~
Milk	4
Beer	3
Diaper	4
~~Eggs~~	~~1~~

必须得要3个item

Minimum Support Count = 3

If every subset is considered,

$$^6C_1 + {}^6C_2 + {}^6C_3$$

$$6 + 15 + 20 = 41$$

With support-based pruning,

$$^6C_1 + {}^4C_2 + 1$$

$$6 + 6 + 1 = 13$$

如果brute froce-要41种
如果剪枝：
可以删除比min support count小的item，表中也就是coke/egg

Illustrating Apriori Principle

TID	*Items*
1	Bread, Milk
2	Beer, Bread, Diaper, Eggs
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Bread, Coke, Diaper, Milk

$\Longrightarrow$

Items (1-itemsets)

Item	Count	
Bread	**4**	
Coke	**2**	✗
Milk	**4**	
Beer	**3**	
Diaper	**4**	
Eggs	**1**	✗

Minimum Support Count = 3

If every subset is considered,

$$^6C_1 + {}^6C_2 + {}^6C_3$$

$$6 + 15 + 20 = 41$$

With support-based pruning,

$$^6C_1 + {}^4C_2 + 1$$

$$6 + 6 + 1 = 13$$

Illustrating Apriori Principle

Item	Count
Bread	**4**
Coke	**2**
Milk	**4**
Beer	**3**
Diaper	**4**
Eggs	**1**

Items (1-itemsets)

重新组合，把剩下的bread，milk，beer，diaper重新组合再进行count

Itemset
{Bread,Milk}
{Bread, Beer }
{Bread,Diaper}
{Beer, Milk}
{Diaper, Milk}
{Beer,Diaper}

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

Minimum Support Count = 3

If every subset is considered,

$^6C_1 + {}^6C_2 + {}^6C_3$

$6 + 15 + 20 = 41$

With support-based pruning,

$^6C_1 + {}^4C_2 + 1$

$6 + 6 + 1 = 13$

Illustrating Apriori Principle

Item	Count
Bread	**4**
Coke	**2**
Milk	**4**
Beer	**3**
Diaper	**4**
Eggs	**1**

Items (1-itemsets)

Itemset	Count
{Bread,Milk}	**3**
{Beer, Bread}	**2**
{Bread,Diaper}	**3**
{Beer,Milk}	**2**
{Diaper,Milk}	**3**
{Beer,Diaper}	**3**

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

这两个依然<3,所以删除

Minimum Support Count = 3

If every subset is considered,

$^6C_1 + {}^6C_2 + {}^6C_3$

$6 + 15 + 20 = 41$

With support-based pruning,

$^6C_1 + {}^4C_2 + 1$

$6 + 6 + 1 = 13$

Illustrating Apriori Principle

Item	Count
Bread	4
Coke	2
Milk	4
Beer	3
Diaper	4
Eggs	1

Items (1-itemsets)

Itemset	Count
{Bread,Milk}	3
{Beer, Bread}	2
{Bread,Diaper}	3
{Beer,Milk}	2
{Diaper,Milk}	3
{Beer,Diaper}	3

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

根据上面的principle，因为{beer,bread},{beer,milk}是infrequent，所以它的superset都是infrequent

Triplets (3-itemsets)

上面的组合可以组成
bread,milk,diaper
bread, milk,beer,diaper
bread,diaper,beer
diaper,milk,beer
根据infrequent的问题，只剩下最后一个

Itemset
{Bread, Diaper, Milk}

(No need to generate candidates with infrequent subsets)

Minimum Support Count = 3

If every subset is considered,

$${}^6C_1 + {}^6C_2 + {}^6C_3$$

$$6 + 15 + 20 = 41$$

With support-based pruning,

$${}^6C_1 + {}^4C_2 + 1$$

$$6 + 6 + 1 = 13$$

Illustrating Apriori Principle

Item	Count
Bread	**4**
Coke	**2**
Milk	**4**
Beer	**3**
Diaper	**4**
Eggs	**1**

Items (1-itemsets)

Itemset	Count
{Bread,Milk}	**3**
{Beer, Bread}	**2**
{Bread,Diaper}	**3**
{Beer,Milk}	**2**
{Diaper,Milk}	**3**
{Beer,Diaper}	**3**

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

Minimum Support Count = 3

If every subset is considered,

$${}^6C_1 + {}^6C_2 + {}^6C_3$$

$$6 + 15 + 20 = 41$$

With support-based pruning,

$${}^6C_1 + {}^4C_2 + 1$$

$$6 + 6 + 1 = 13$$

Triplets (3-itemsets)

Itemset	Count
{Bread, Diaper, Milk}	**2**

(No need to generate candidates with infrequent subsets)

Apriori Algorithm

- F_k: frequent k-itemsets
- L_k: candidate k-itemsets

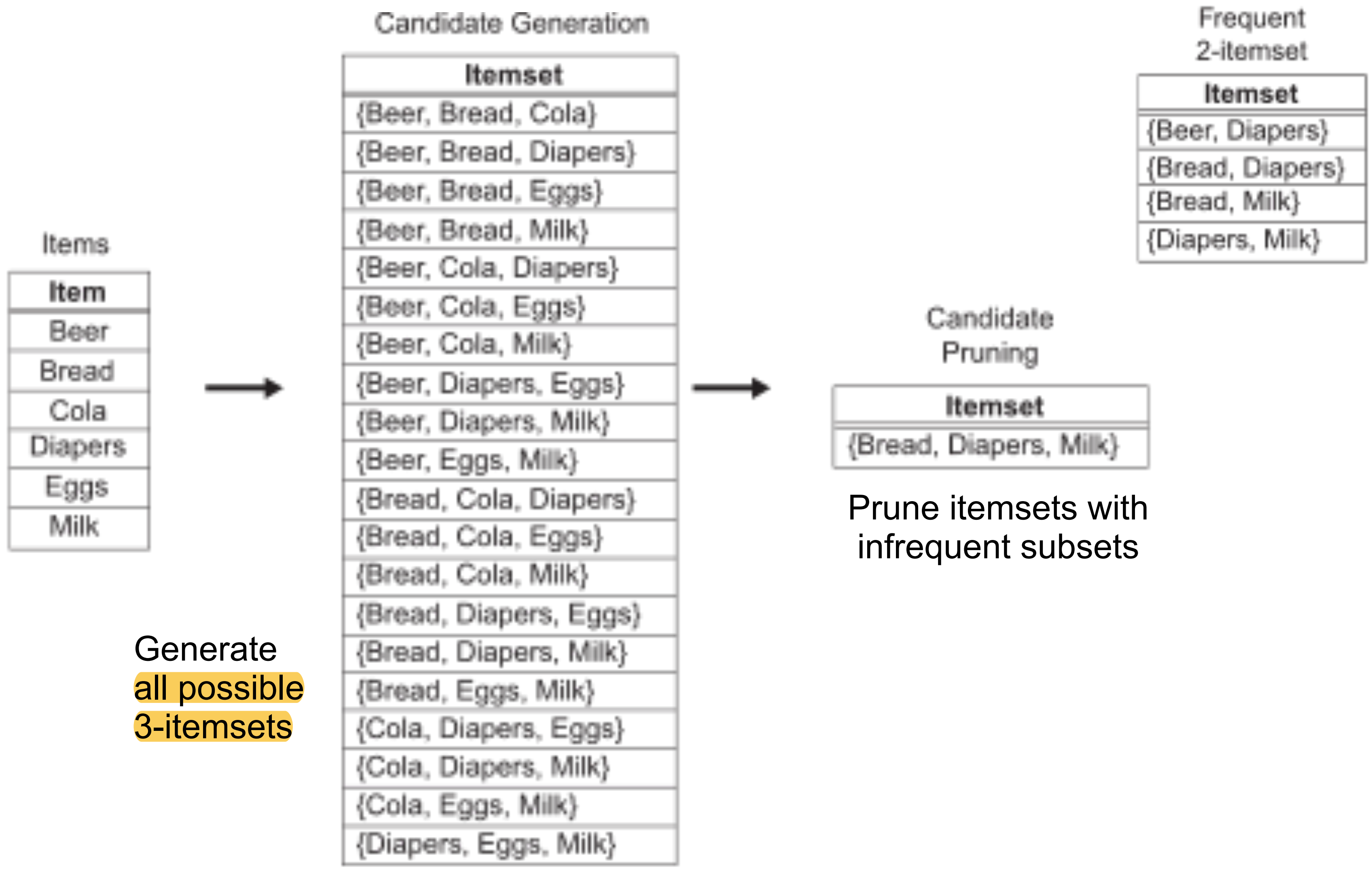
- Algorithm
  - Let k=1
  - Generate F_1 = {frequent 1-itemsets}
  - Repeat until F_k is empty
    - **Candidate Generation**: Generate L_{k+1} from F_k
    - **Candidate Pruning**: Prune candidate itemsets in L_{k+1} containing subsets of length k that are infrequent
    - **Support Counting**: Count the support of each candidate in L_{k+1} by scanning the transaction database
    - **Candidate Elimination**: Eliminate candidates in L_{k+1} that are infrequent, leaving only those that are frequent => F_{k+1}

Candidate Generation Requirements

- Candidate itemsets must include all frequent itemsets
- Algorithm should not produce an itemset more than once
  - Example: {a,b,c,d} can be obtained by merging {a,b,c} with {d}, {a,b,d} with {c}, {a,b,c} with {b,c,d}, and so on
  - To avoid duplicates, keep all items in an itemset sorted in **lexicographic order** 按字母排序
    - Satisfies order: {a,b,c,d}, {a,b,c}, {a,b,d}, …
    - Violates order: {a,c,b,d}, {a,c,b}, {d,a,b}, …
  - Merge two itemsets P and Q only if items in Q are not lexicographically smaller than items in P
    - Merge {a,b,c} with {d}, {a,b,c} with {a,b,d}, …
    - Do not merge {a,b,d} with {c}, {a,c,d} with {b,c,d}

Q必须大于P！！是里面的所有item都必须比P大

Candidate Generation: Brute-force method



Items

Item
Beer
Bread
Cola
Diapers
Eggs
Milk

Candidate Generation

Itemset
{Beer, Bread, Cola}
{Beer, Bread, Diapers}
{Beer, Bread, Eggs}
{Beer, Bread, Milk}
{Beer, Cola, Diapers}
{Beer, Cola, Eggs}
{Beer, Cola, Milk}
{Beer, Diapers, Eggs}
{Beer, Diapers, Milk}
{Beer, Eggs, Milk}
{Bread, Cola, Diapers}
{Bread, Cola, Eggs}
{Bread, Cola, Milk}
{Bread, Diapers, Eggs}
{Bread, Diapers, Milk}
{Bread, Eggs, Milk}
{Cola, Diapers, Eggs}
{Cola, Diapers, Milk}
{Cola, Eggs, Milk}
{Diapers, Eggs, Milk}

Frequent 2-itemset

Itemset
{Beer, Diapers}
{Bread, Diapers}
{Bread, Milk}
{Diapers, Milk}

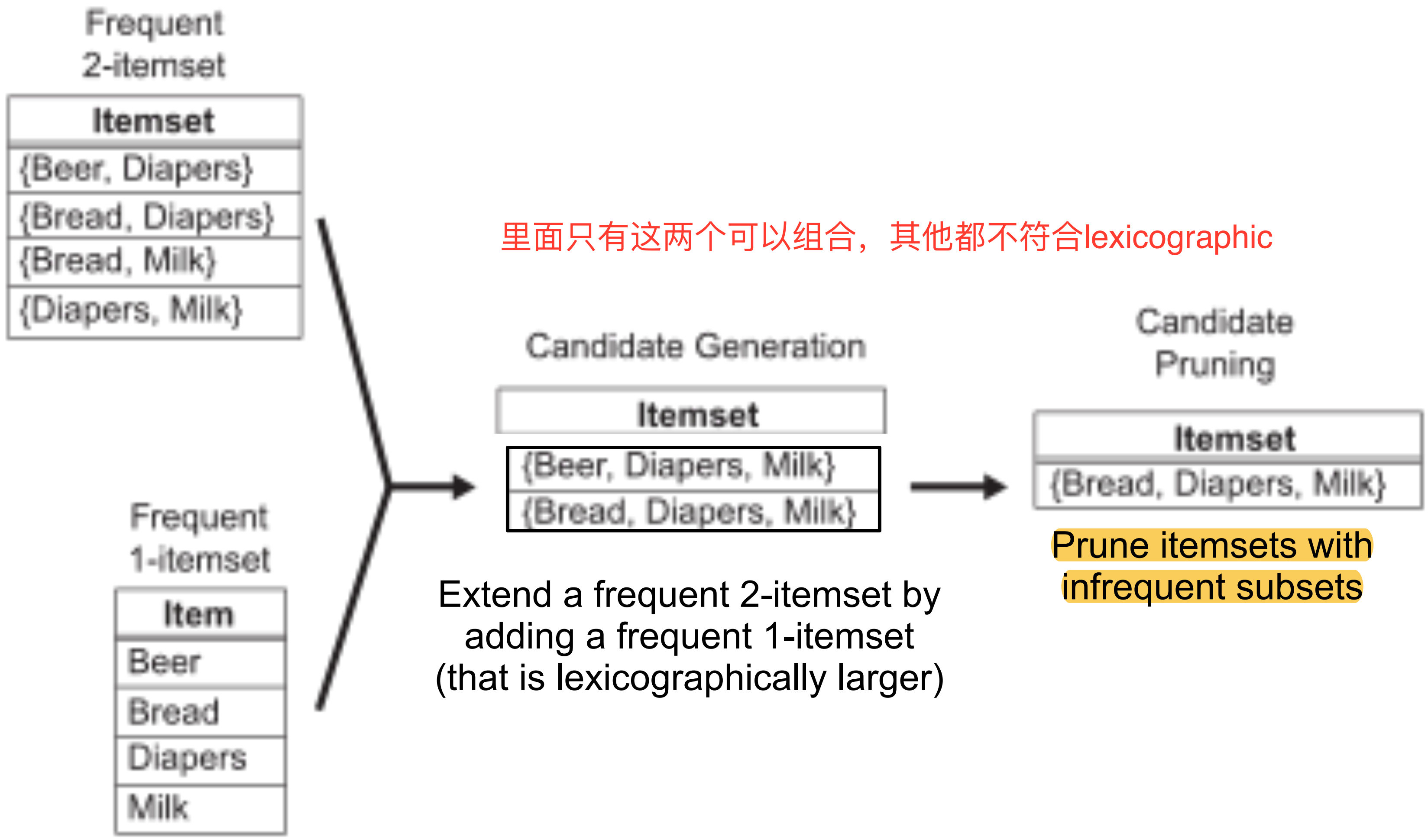
Candidate Pruning

Itemset
{Bread, Diapers, Milk}

Generate all possible 3-itemsets

Prune itemsets with infrequent subsets

Candidate Generation: Fk-1 X F1 Method



Frequent 2-itemset

Itemset
{Beer, Diapers}
{Bread, Diapers}
{Bread, Milk}
{Diapers, Milk}

Frequent 1-itemset

Item
Beer
Bread
Diapers
Milk

里面只有这两个可以组合，其他都不符合lexicographic

Candidate Generation

Itemset
{Beer, Diapers, Milk}
{Bread, Diapers, Milk}

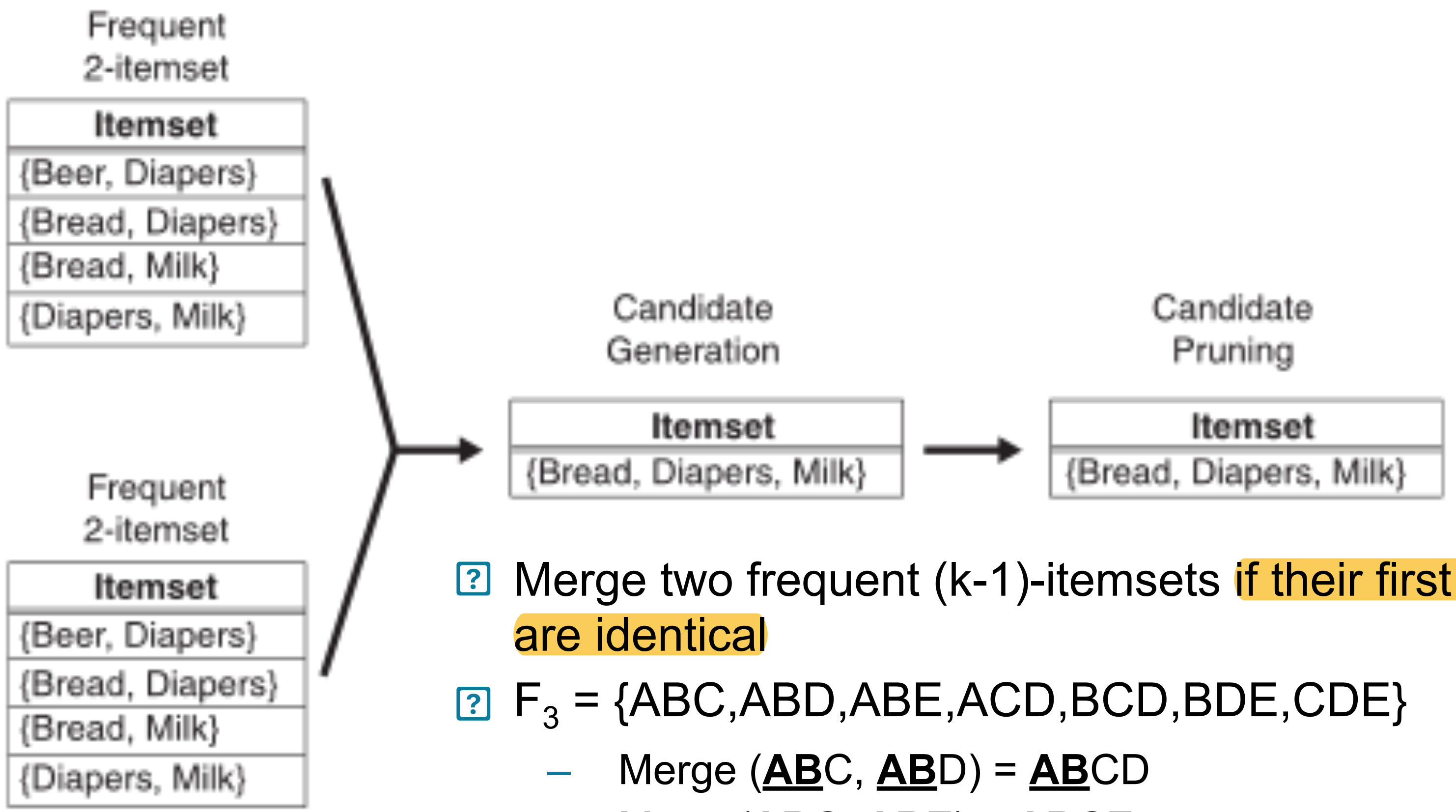
Candidate Pruning

Itemset
{Bread, Diapers, Milk}

Prune itemsets with infrequent subsets

Extend a frequent 2-itemset by adding a frequent 1-itemset (that is lexicographically larger)

Candidate Generation: Fk-1 x Fk-1 Method



Frequent 2-itemset

Itemset
{Beer, Diapers}
{Bread, Diapers}
{Bread, Milk}
{Diapers, Milk}

Frequent 2-itemset

Itemset
{Beer, Diapers}
{Bread, Diapers}
{Bread, Milk}
{Diapers, Milk}

Candidate Generation

Itemset
{Bread, Diapers, Milk}

Candidate Pruning

Itemset
{Bread, Diapers, Milk}

- Merge two frequent (k-1)-itemsets if their first (k-2) items are identical
- F_3 = {ABC,ABD,ABE,ACD,BCD,BDE,CDE}
  - Merge (**AB**C, **AB**D) = **AB**CD
  - Merge(**AB**C, **AB**E) = **AB**CE
  - Merge(**AB**D, **AB**E) = **AB**DE
  - Do not merge (**A**BD,**A**CD) because they share only prefix of length 1 instead of length 2

Candidate Pruning

- Let F_3 = {ABC,ABD,ABE,ACD,BCD,BDE,CDE} be the set of frequent 3-itemsets
- L_4 = {ABCD,ABCE,ABDE} is the set of candidate 4-itemsets generated
- Candidate pruning
  - Prune a k-itemset if any of its (k-1)-size subsets is infrequent
  - Prune ABCE because ACE and BCE are infrequent
  - Prune ABDE because ADE is infrequent
- After candidate pruning: L_4 = {ABCD}

Support Counting of Candidate Itemsets

- Scan the database of transactions to determine the support of each candidate itemset
  - Must match every candidate itemset against every transaction, which is an expensive operation

- Number of comparisons can be reduced by storing the candidates in a data structure such as a hash tree (see book for details)

TID	*Items*
1	Bread, Milk
2	Beer, Bread, Diaper, Eggs
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Bread, Coke, Diaper, Milk

Itemset
{ Beer, Diaper, Milk}
{ Beer,Bread,Diaper}
{Bread, Diaper, Milk}
{ Beer, Bread, Milk}

Rule Generation

x属于L
y属于L
x=L-y
y=L-x
互斥

- Given a frequent itemset L, find all non-empty subsets $f \subset L$ such that $f \rightarrow L - f$ satisfies the minimum confidence requirement
  - If {A,B,C,D} is a frequent itemset, candidate rules:

$ABC \rightarrow D$,	$ABD \rightarrow C$,	$ACD \rightarrow B$,	$BCD \rightarrow A$,
$A \rightarrow BCD$,	$B \rightarrow ACD$,	$C \rightarrow ABD$,	$D \rightarrow ABC$
$AB \rightarrow CD$,	$AC \rightarrow BD$,	$AD \rightarrow BC$,	$BC \rightarrow AD$,
$BD \rightarrow AC$,	$CD \rightarrow AB$,		

- If $|L| = k$, then there are $2^k - 2$ candidate association rules (ignoring $L \rightarrow \varnothing$ and $\varnothing \rightarrow L$)

Rule Generation

- In general, confidence does not have an anti-monotone property

  $c(ABC \rightarrow D)$ can be larger or smaller than $c(AB \rightarrow D)$

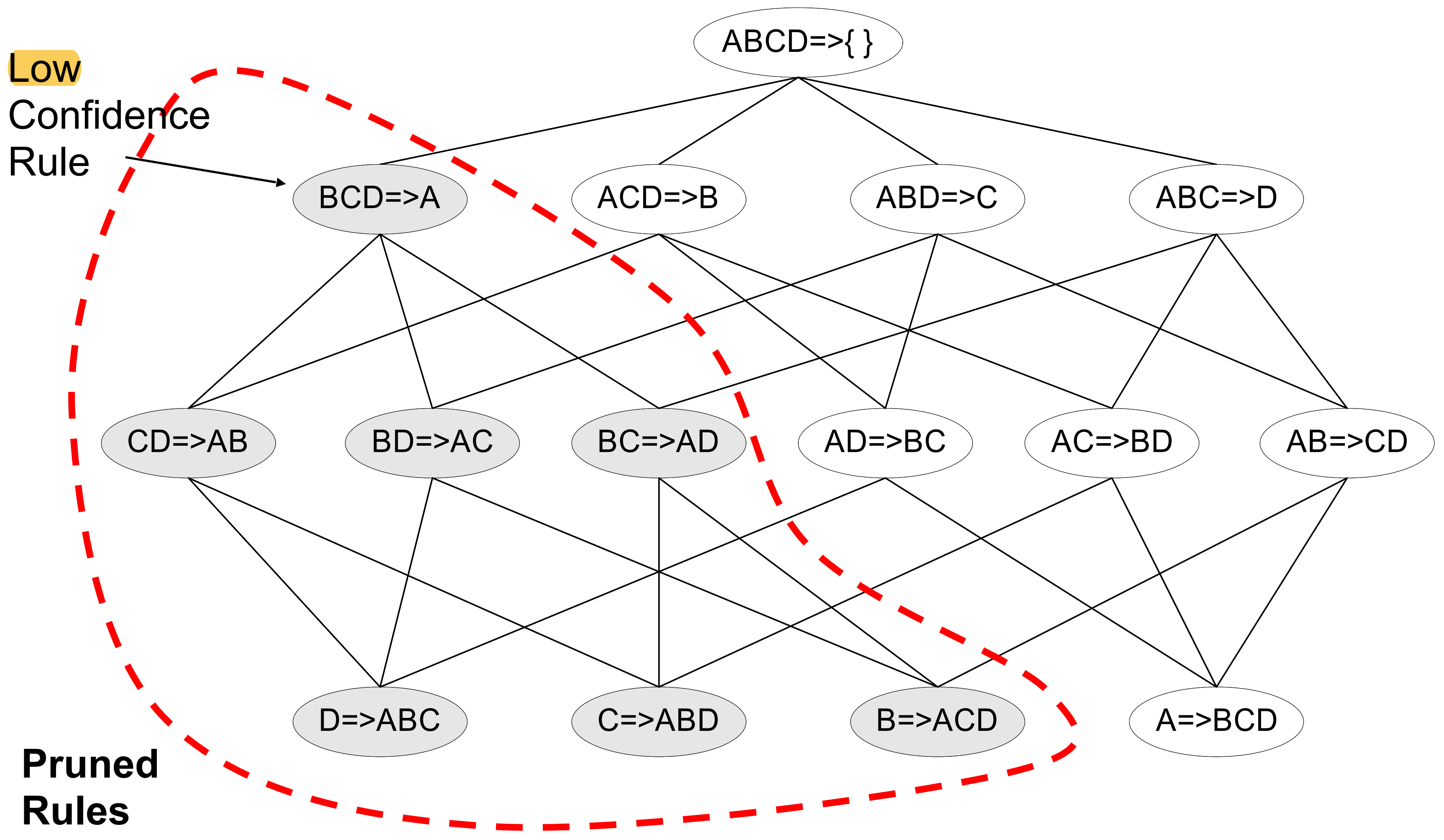
- But confidence of rules generated from the same itemset has an anti-monotone property
  - E.g., Suppose {A,B,C,D} is a frequent 4-itemset:

    $$c(ABC \rightarrow D) \geq c(AB \rightarrow CD) \geq c(A \rightarrow BCD)$$

  - Confidence is anti-monotonic w.r.t. number of items on the RHS (consequent) of the rule

Rule Generation for Apriori Algorithm

Lattice of rules



Factors Affecting Complexity of Apriori

- Choice of minimum support threshold min越低，需要分析的越多
  - lowering support threshold results in more frequent itemsets
  - this may increase number of candidates and max length of frequent itemsets
- Dimensionality (number of items) of the data set
  - more space is needed to store support count of each item
  - if number of frequent items also increases, both computation and I/O costs may also increase
- Number of transactions
  - since Apriori makes multiple passes, run time of algorithm may increase with number of transactions
- Average transaction width
  - transaction width increases with denser data sets
  - This may increase max length of frequent itemsets (number of subsets supported by a transaction increases with its width)