Association Rules Discovery

Lecture outline

- What is association rule mining?
- Frequent itemsets, support, and confidence
- Mining association rules
- The “Apriori” algorithm
- Rule generation

What is association mining?

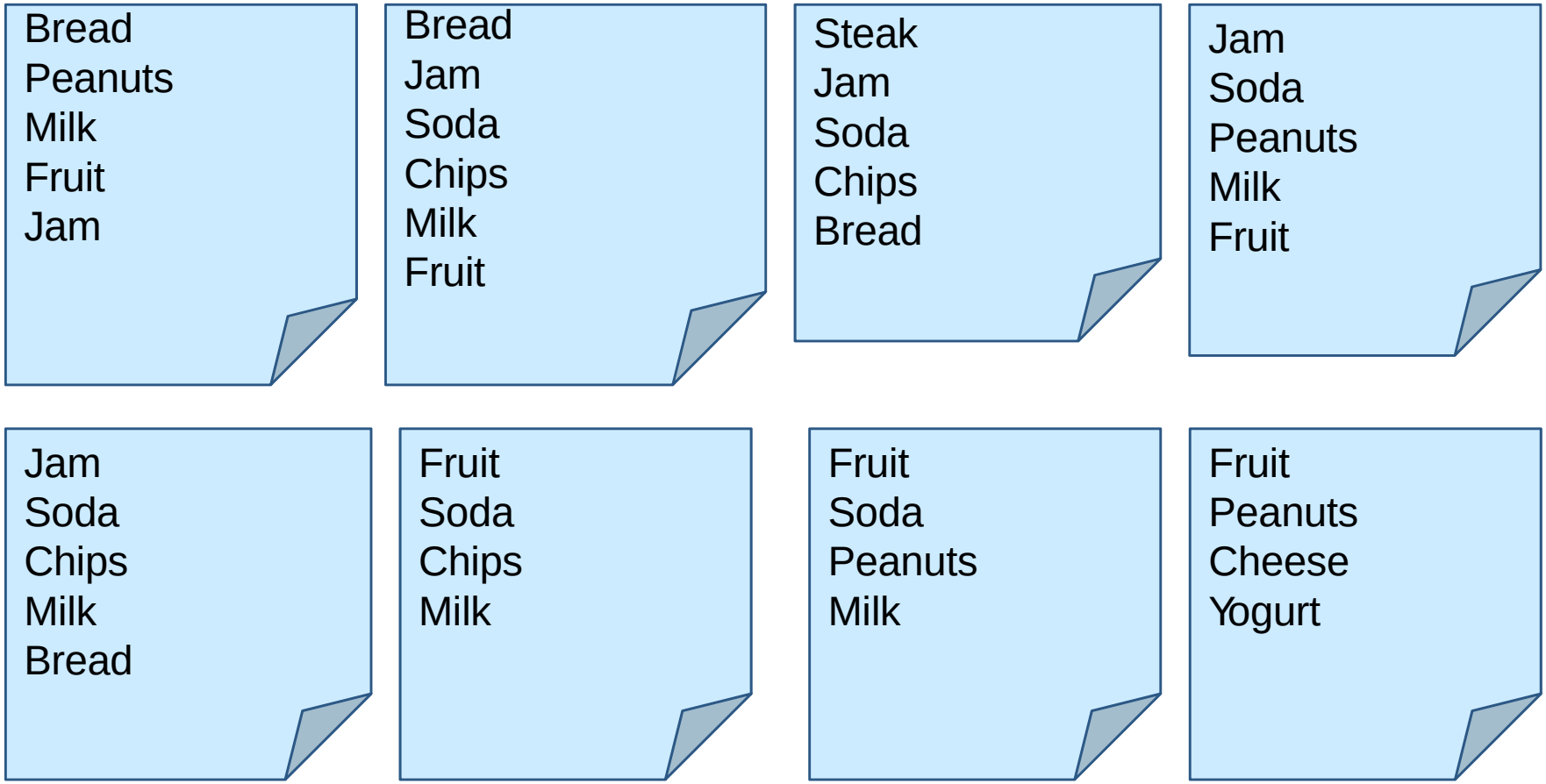
- Finding frequent patterns, associations, correlations, or causal structures among sets of items or objects in transaction databases, relational databases, and other information repositories

- Applications
  - Basket data analysis
  - Cross-marketing
  - Catalog design
  - …

Applications in more detail

- **Market Basket Analysis**: given a database of customer transactions, where each transaction is a set of items the goal is to find groups of items which are frequently purchased together.
- **Telecommunication:** (each customer is a transaction containing the set of phone calls)
- **Credit Cards/ Banking Services:** (each card/account is a transaction containing the set of customer's payments)
- **Medical Treatments:** (each patient is represented as a transaction containing the ordered set of diseases)
- **Basketball-Game Analysis:** (each game is represented as a transaction containing the ordered set of ball passes)

Example of market-basket transactions



What is association rule mining?

TID	Items
1	Bread, Peanuts, Milk, Fruit, Jam
2	Bread, Jam, Soda, Chips, Milk, Fruit
3	Steak, Jam, Soda, Chips, Bread
4	Jam, Soda, Peanuts, Milk, Fruit
5	Jam, Soda, Chips, Milk, Bread
6	Fruit, Soda, Chips, Milk
7	Fruit, Soda, Peanuts, Milk
8	Fruit, Peanuts, Cheese, Yogurt

Examples

$\{bread\} \Rightarrow \{milk\}$

$\{soda\} \Rightarrow \{chips\}$

$\{bread\} \Rightarrow \{jam\}$

- 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

Definition: Frequent Itemset

- Itemset
  - A collection of one or more items, e.g., {milk, bread, jam}
  - k-itemset, an itemset that contains k items
- Support count (σ)
  - Frequency of occurrence of an itemset
  - σ({Milk, Bread}) = 3
    σ({Soda, Chips}) = 4
- Support
  - Fraction of transactions that contain an itemset
  - s({Milk, Bread}) = 3/8
    s({Soda, Chips}) = 4/8
- Frequent Itemset
  - An itemset whose support is greater than or equal to a minsup threshold

TID	Items
1	Bread, Peanuts, Milk, Fruit, Jam
2	Bread, Jam, Soda, Chips, Milk, Fruit
3	Steak, Jam, Soda, Chips, Bread
4	Jam, Soda, Peanuts, Milk, Fruit
5	Jam, Soda, Chips, Milk, Bread
6	Fruit, Soda, Chips, Milk
7	Fruit, Soda, Peanuts, Milk
8	Fruit, Peanuts, Cheese, Yogurt

What is an association rule?

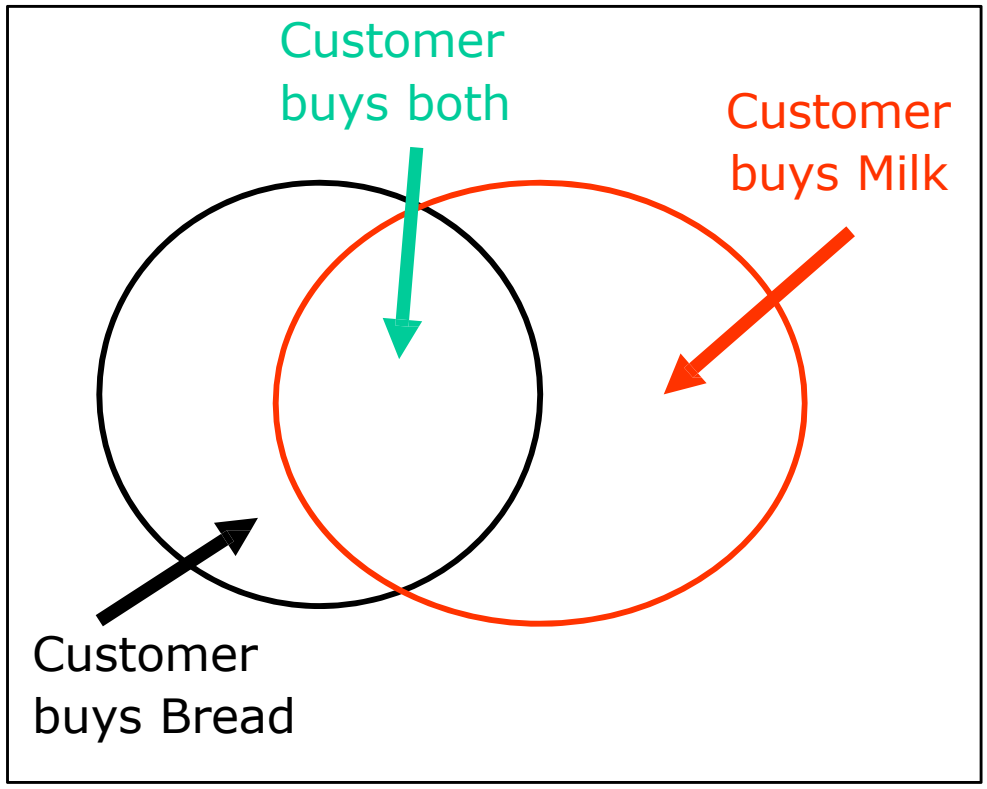
- Implication of the form X $\Rightarrow$ Y, where X and Y are itemsets
- Example, {bread} $\Rightarrow$ {milk}
- Rule Evaluation Metrics, Suppor & Confidence
- Support (s)
  - Fraction of transactions that contain both X and Y

$$s = \frac{\sigma(\{\text{Bread}, \text{Milk}\})}{\#\text{ of transactions}} = 0.38$$

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

$$c = \frac{\sigma(\{\text{Bread}, \text{Milk}\})}{\sigma(\{\text{Bread}\})} = 0.75$$

Support and Confidence



What is the goal?

- 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

- Brute-force approach is computationally prohibitive!

Mining Association Rules

{Bread, Jam} ⇒ {Milk} s=0.4 c=0.75
{Milk, Jam} ⇒ {Bread} s=0.4 c=0.75
{Bread} ⇒ {Milk, Jam} s=0.4 c=0.75
{Jam} ⇒ {Bread, Milk} s=0.4 c=0.6
{Milk} ⇒ {Bread, Jam} s=0.4 c=0.5

- All the above rules are binary partitions of the same itemset:

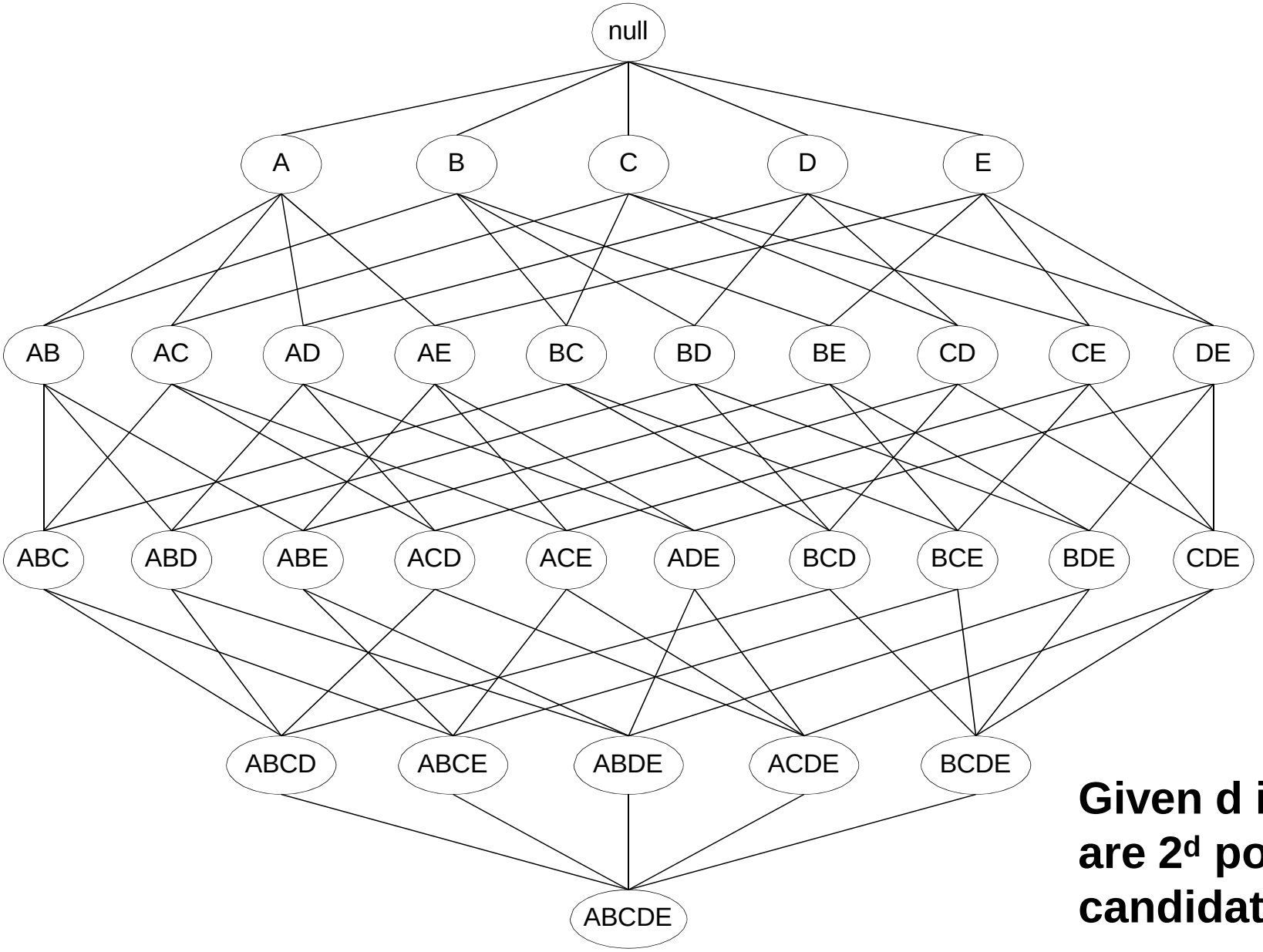
{Milk, Bread, Jam}

- Rules originating from the same itemset have identical support but can have different confidence
- We can decouple the support and confidence requirements!

Mining Association Rules: Two Step Approach

- Frequent Itemset Generation
  - Generate all itemsets whose support $\geq$ minsup
- Rule Generation
  - Generate high confidence rules from frequent itemset
  - Each rule is a binary partitioning of a frequent itemset
- Frequent itemset generation is computationally expensive

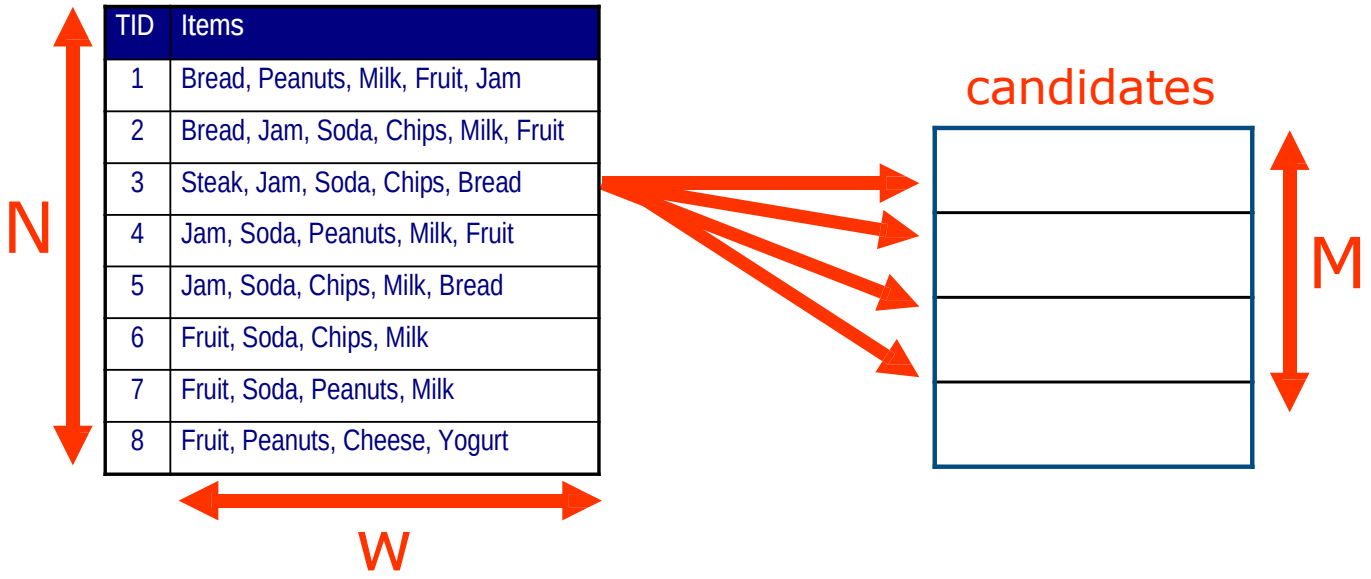
Frequent Itemset Generation



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

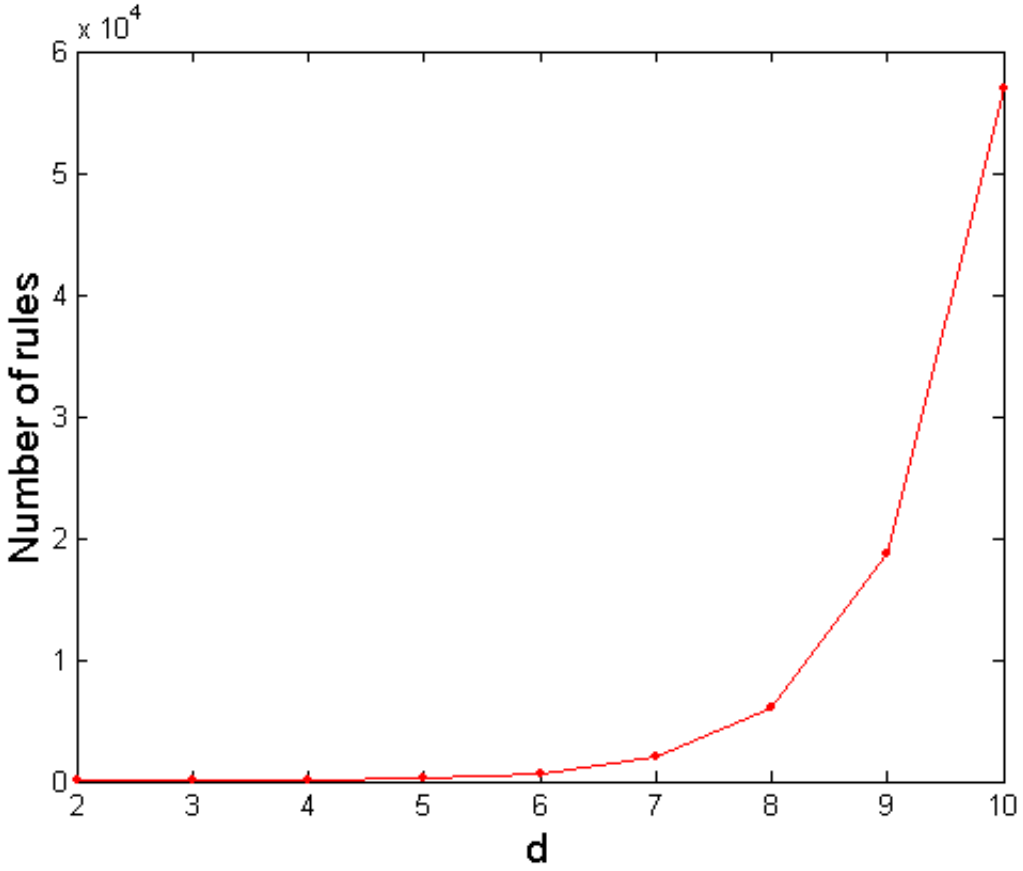
Frequent Itemset Generation

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



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

Computational Complexity



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

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

$$=3^d-2^{d+1}+1$$

  - For d=6, there are 602 rules

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
- 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 the Number of Candidates

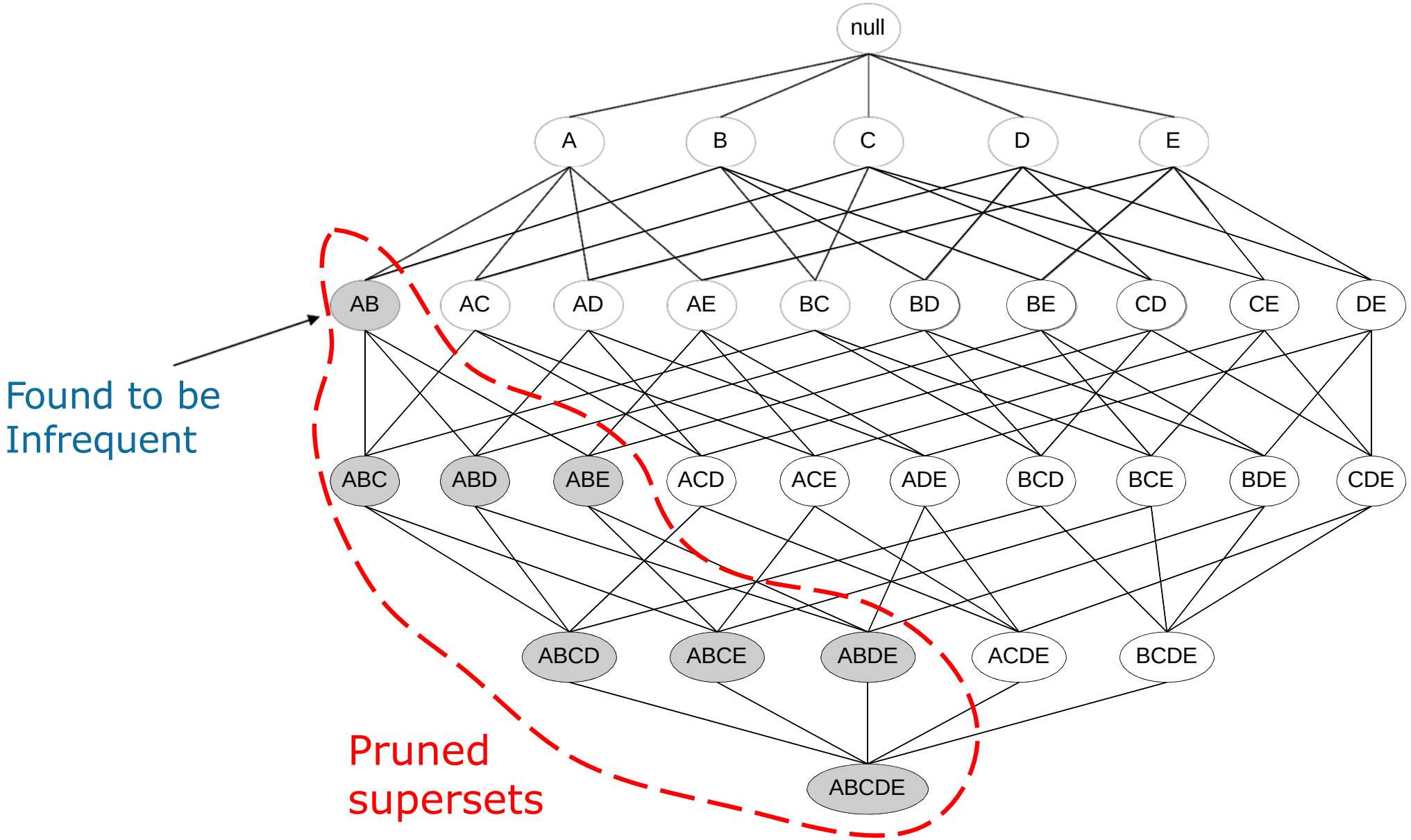
- Apriori principle
  - If an itemset is frequent, then all of its subsets must also be frequent

- 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
- This is known as the anti-monotone property of support

Illustrating Apriori Principle



How does the Apriori principle work?

Items (1-itemsets)

Item	Count
Bread	4
Peanuts	4
Milk	6
Fruit	6
Jam	5
Soda	6
Chips	4
Steak	1
Cheese	1
Yogurt	1

2-itemsets

2-Itemset	Count
Bread, Jam	4
Peanuts, Fruit	4
Milk, Fruit	5
Milk, Jam	4
Milk, Soda	5
Fruit, Soda	4
Jam, Soda	4
Soda, Chips	4

3-itemsets

3-Itemset	Count
Milk, Fruit, Soda	4

Minimum Support = 4

Apriori Algorithm

- Let k=1
- Generate frequent itemsets of length 1
- Repeat until no new frequent itemsets are identified
  - Generate length (k+1) candidate itemsets from length k frequent itemsets
  - Prune candidate itemsets containing subsets of length k that are infrequent
  - Count the support of each candidate by scanning the DB
  - Eliminate candidates that are infrequent, leaving only those that are frequent

The Apriori Algorithm

C_k: Candidate itemset of size k

L_k : frequent itemset of size k

L_1 = {frequent items};
for (k = 1; L_k !=∅; 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 $\cup_k L_k$;

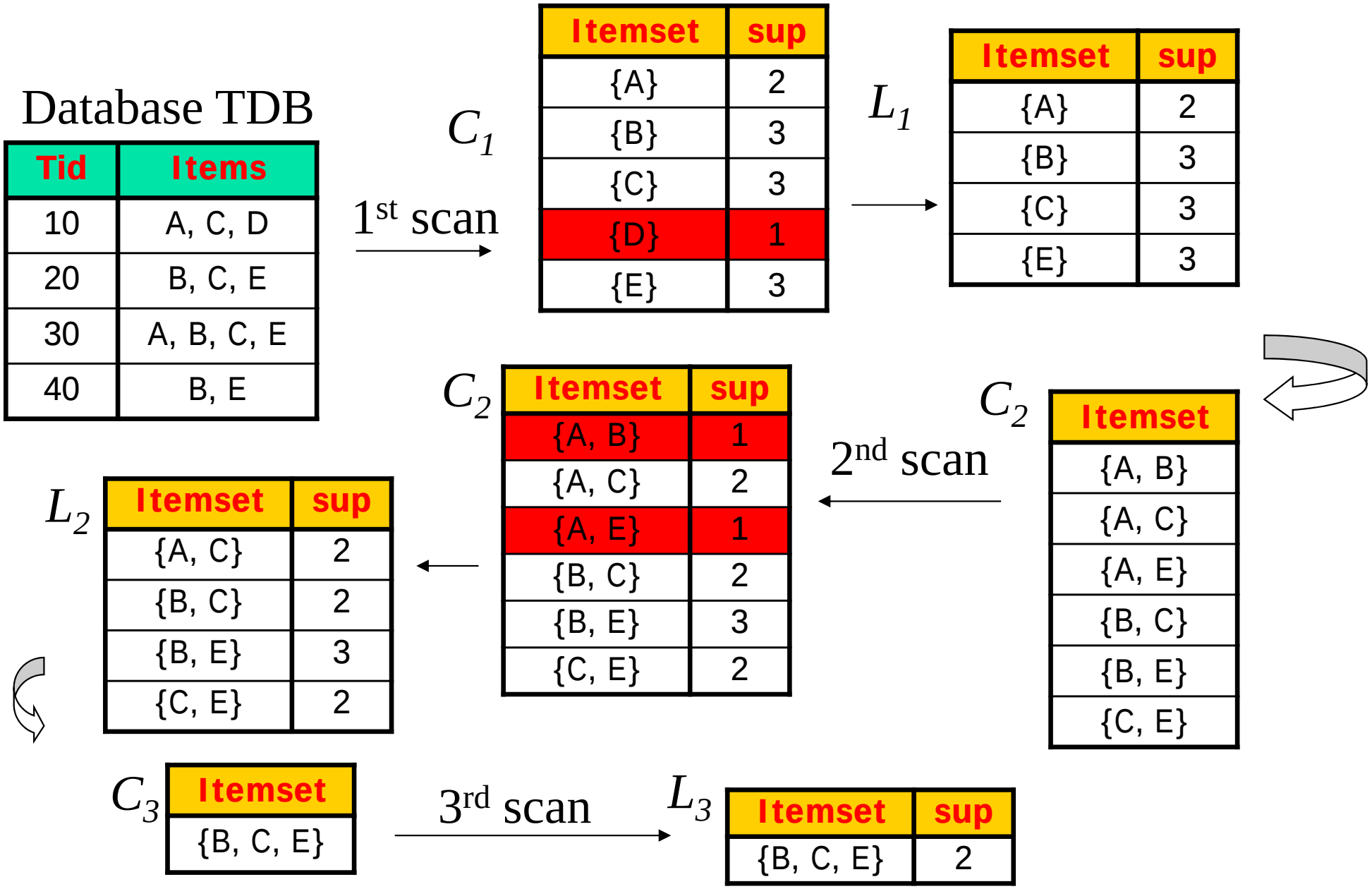
The Apriori Algorithm

- Join Step
  - C_k is generated by joining L_{k-1} with itself
- Prune Step
  - Any (k-1)-itemset that is not frequent cannot be a subset of a frequent k-itemset

Apriori: A Candidate Generation-and-test Approach

- Any subset of a frequent itemset must be frequent
  - if **{beer, diaper, nuts}** is frequent, so is **{beer, diaper}**
  - every transaction having {beer, diaper, nuts} also contains {beer, diaper}
- Apriori pruning principle: If there is any itemset which is infrequent, its superset should not be generated/tested!
- Method:
  - generate length (k+1) candidate itemsets from length k frequent itemsets, and
  - test the candidates against DB
- The performance studies show its efficiency and scalability

The Apriori Algorithm — An Example



Important Details of Apriori

- How to generate candidates?
  - Step 1: self-joining L_k
  - Step 2: pruning
- Example of Candidate-generation
  - L_3={*abc, abd, acd, ace, bcd*}
  - Self-joining: L_3*L_3

    abcd from *abc* and *abd*

    acde from *acd* and *ace*
  - Pruning:

    acde is removed because *ade* is not in L_3
  - C_4={*abcd*}

How to Generate Candidates?

- Suppose the items in L_{k-1} are listed in an order
- Step 1: self-joining L_{k-1}

  insert into C_k

  select $p.item_1, p.item_2, \ldots, p.item_{k-1}, q.item_{k-1}$

  from $L_{k-1}\ p, L_{k-1}\ q$

  where $p.item_1=q.item_1, \ldots, p.item_{k-2}=q.item_{k-2}, p.item_{k-1} < q.item_{k-1}$
- Step 2: pruning

  forall *itemsets c in* C_k do

    forall *(k-1)-subsets s of c* do

      if *(s is not in* L_{k-1}*)* then delete *c* from C_k

Rule Generation

- 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 $\varnothing \rightarrow L$)

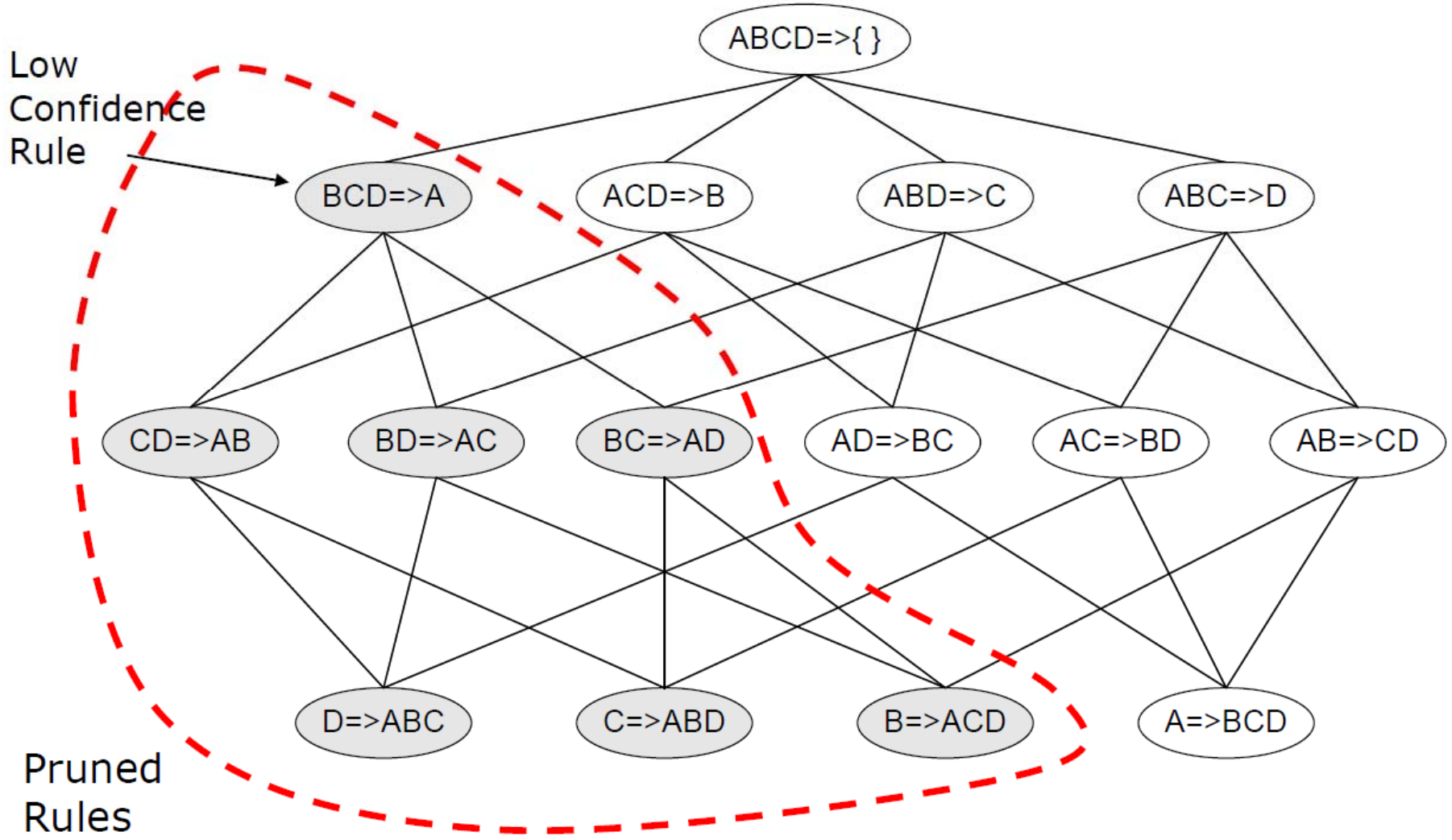
How to efficiently generate rules from frequent itemsets?

- Confidence does not have an anti-monotone property
- c(ABC→ D) can be larger or smaller than c(AB → D)
- But confidence of rules generated from the same itemset has an anti-monotone property
  - e.g., L = {A,B,C,D}:

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

  - Confidence is anti-monotone with respect to the number of items on the right hand side of the rule

Rule Generation for Apriori Algorithm

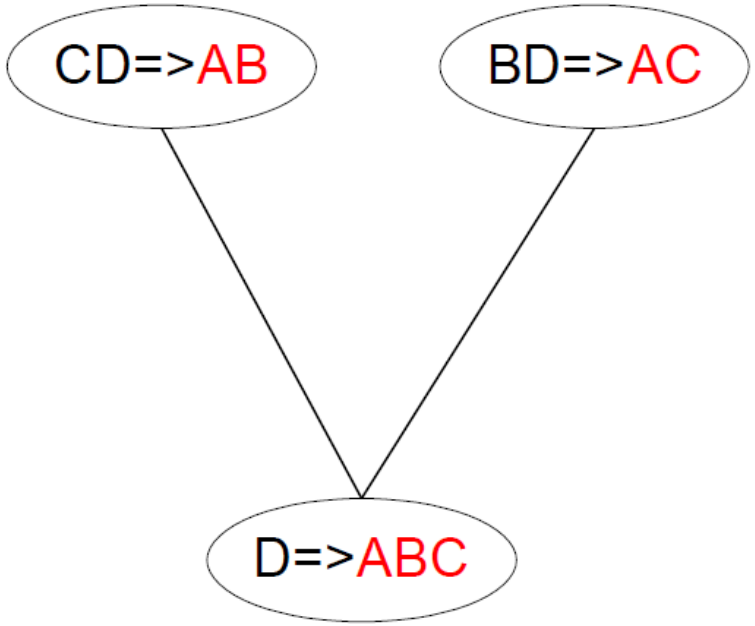


Rule Generation for Apriori Algorithm

- Candidate rule is generated by merging two rules that share the same prefix
  in the rule consequent

- join(CD=>AB,BD=>AC)
  would produce the candidate rule D => ABC

- Prune rule D=>ABC if its subset AD=>BC does not have high confidence



How to set the appropriate minsup?

- If minsup is set too high, we could miss itemsets involving interesting rare items (e.g., expensive products)
- If minsup is set too low, it is computationally expensive and the number of itemsets is very large
- A single minimum support threshold may not be effective

Filter Out Trivial And Misleading Associations

- Let's assume we have a Database about behavior of 5.000 students and that from the dataset, we know:
  - 3.000 of them play basketball (P) [60%]
  - 3.750 eat cereals for breakfast (E) [75%]
  - 2.000 play basketball and eat cereals ($P \wedge E$) [40%]

- Let's set thresholds: s=40%, c=60%

- Can we obtain the following rule?

$$P \rightarrow E$$

Filter Out Trivial And Misleading Associations

- Let's assume we have a Database about behavior of 5.000 students and that from the dataset, we know:
  - 3.000 of them play basketball (P) [60%]
  - 3.750 eat cereals for breakfast (E) [75%]
  - 2.000 play basketball and eat cereals ($P \wedge E$) [40%]

- Let's set thresholds: s=40%, c=60%

- Can we obtain the following rule?

$$P \rightarrow E$$

$S(P \wedge E) \geq 40\%$

$S(P \wedge E)/S(P) = 0,4/0,6 = 66\% > 60\%$

Filter Out Trivial And Misleading Associations

- Let's assume we have a Database about behavior of 5.000 students and that from the dataset, we know:
  - 3.000 of them play basketball (P) [60%]
  - 3.750 eat cereals for breakfast (E) [75%]
  - 2.000 play basketball and eat cereals ($P \wedge E$) [40%]

- Let's set thresholds: s=40%, c=60%

- Can we obtain the following rule?

$$P \rightarrow E$$

$S(P \wedge E) \geq 40\%$

$S(P \wedge E)/S(P) = 0,4/0,6 = 66\% > 60\%$

- So rule says: If the student plays basketball, then with confidence of 0.66 will eat cereals.

Filter Out Trivial And Misleading Associations

- Let's assume we have a Database about behavior of 5.000 students and that from the dataset, we know:
  - 3.000 of them play basketball (P) [60%]
  - 3.750 eat cereals for breakfast (E) [75%]
  - 2.000 play basketball and eat cereals ($P \wedge E$) [40%]

- Let's set thresholds: s=40%, c=60%

- Can we obtain the following rule?

$$P \rightarrow E$$

$S(P \wedge E) \geq 40\%$

$S(P \wedge E)/S(P) = 0,4/0,6 = 66\% > 60\%$

- So rule says: If the student plays basketball, then with confidence of 0.66 will eat cereals.

- But notice that we know S(E) is 75%... So rule does not contribute to our knowledge

Filter Out Trivial And Misleading Associations

- We should filter out trivial and misleading associations A→B:

$$\frac{S(A \wedge B)}{S(A)} - S(B) > 0$$

Filter Out Trivial And Misleading Associations

- We should filter out trivial and misleading associations A→B:

$$\frac{S(A \wedge B)}{S(A)} - S(B) > 0$$

- In the Basketball example:

S(P∧E)/S(P) – S(E) = 0,66 – 0,75 = -0,09 < 0


Heineken


IS THERE A CORRELATION
BETWEEN
DIAPERS & BEER


Budweiser
Budweiser