APRIORI ALGORITHM

SYRACUSE UNIVERSITY
School of Information Studies

HOW TO MINE ASSOCIATION RULES?

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!

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 confidences.

Thus, we may decouple the support and confidence requirements.

MINING ASSOCIATION RULES

Two-step approach:

Frequent itemset generation

Generate all itemsets whose support ≥ *minsup*.

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.

SCALABLE METHODS FOR MINING FREQUENT PATTERNS

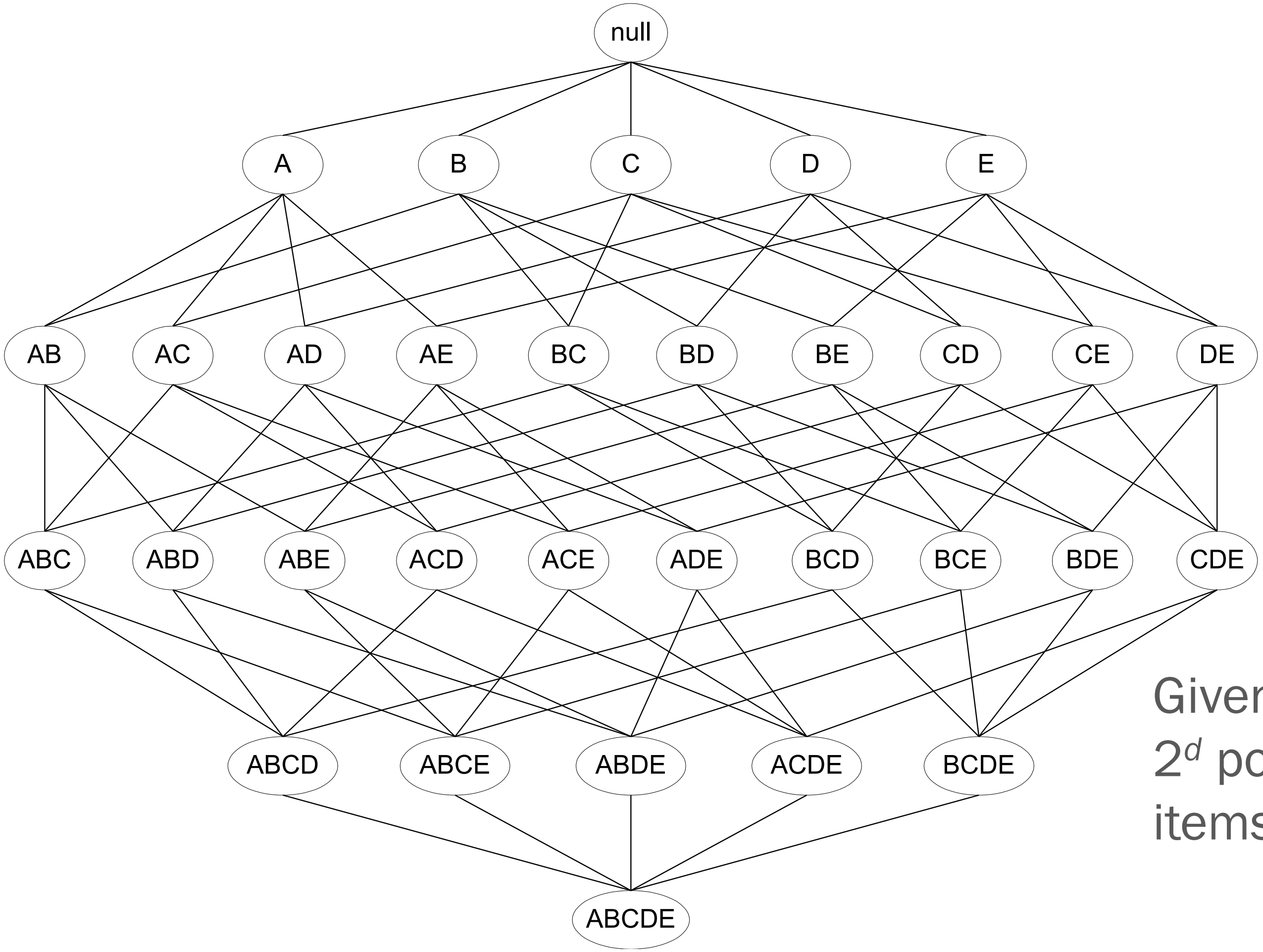
Scalable mining methods: Three major approaches

Apriori (Agrawal & Srikant@VLDB’94)

Frequent pattern growth (FPgrowth—Han, Pei, & Yin @SIGMOD’00)

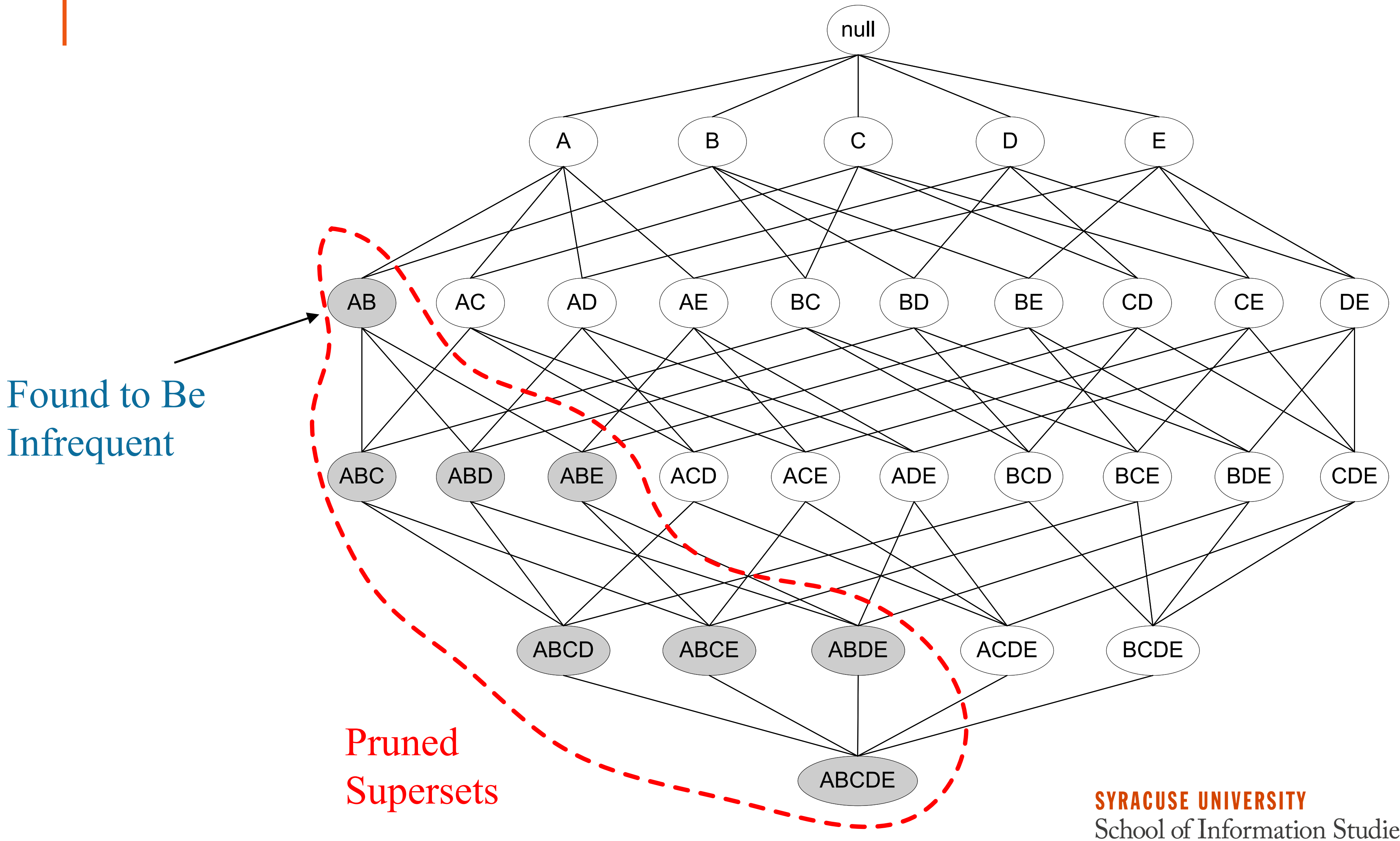
Vertical data format approach (Charm—Zaki & Hsiao @SDM’02)

FREQUENT ITEMSET GENERATION



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

ILLUSTRATING APRIORI PRINCIPLE



SYRACUSE UNIVERSITY
School of Information Studies

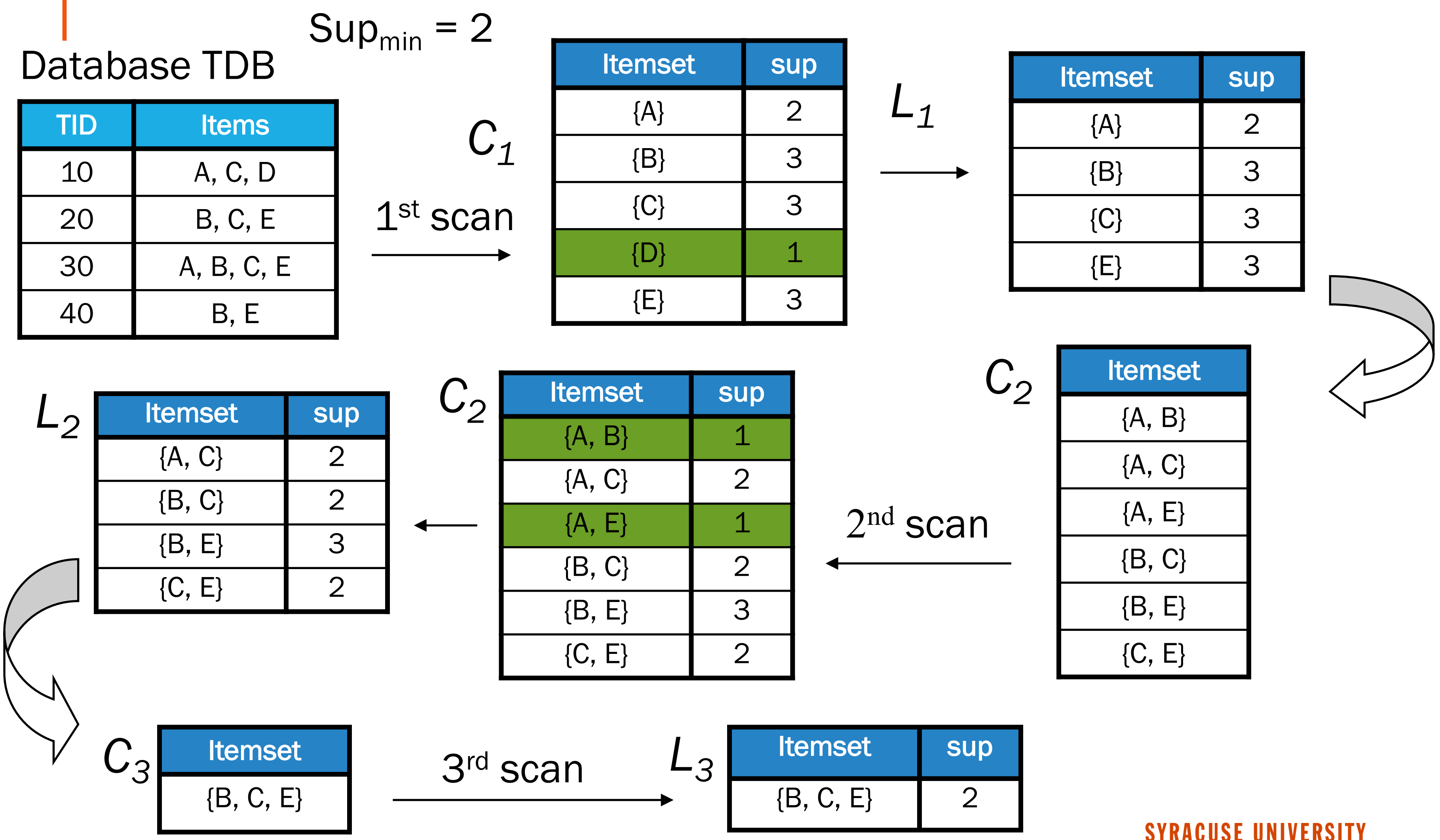
APRIORI: A CANDIDATE GENERATION-AND-TEST APPROACH

Apriori pruning principle: If there is any itemset that is infrequent, its superset should not be generated or tested!

Method:

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

THE APRIORI ALGORITHM: GENERATE FREQUENT ITEMSET



$\text{Sup}_{\text{min}} = 2$

Database TDB

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

1st 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}

2nd 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}

3rd scan

L_3

Itemset	sup
{B, C, E}	2

RULE GENERATION

Given a frequent itemset L, find all nonempty subsets f, such that f → (L – f) satisfies the minimum confidence requirement.

If {A, B, C, D} is a frequent itemset, candidate rules:

ABC → D, ABD → C, ACD → B, BCD → A

AB → CD, AC → BD, …

A → BCD, B → ACD, C→ ABD, D → ABC

Compute the confidence for each rule, and keep the ones that are greater than min_conf.

RULE GENERATION

How to efficiently generate rules from frequent itemsets?

Start from long LHS:

For itemset {ABCD}, c(x) means confidence of rule x
c(ABC → D) ≥ c(AB → CD) ≥ c(A → BCD)

Proof:

C(ABC->D) = support(ABCD)/support(ABC)

C(AB->CD) = support (ABCD)/support (AB)

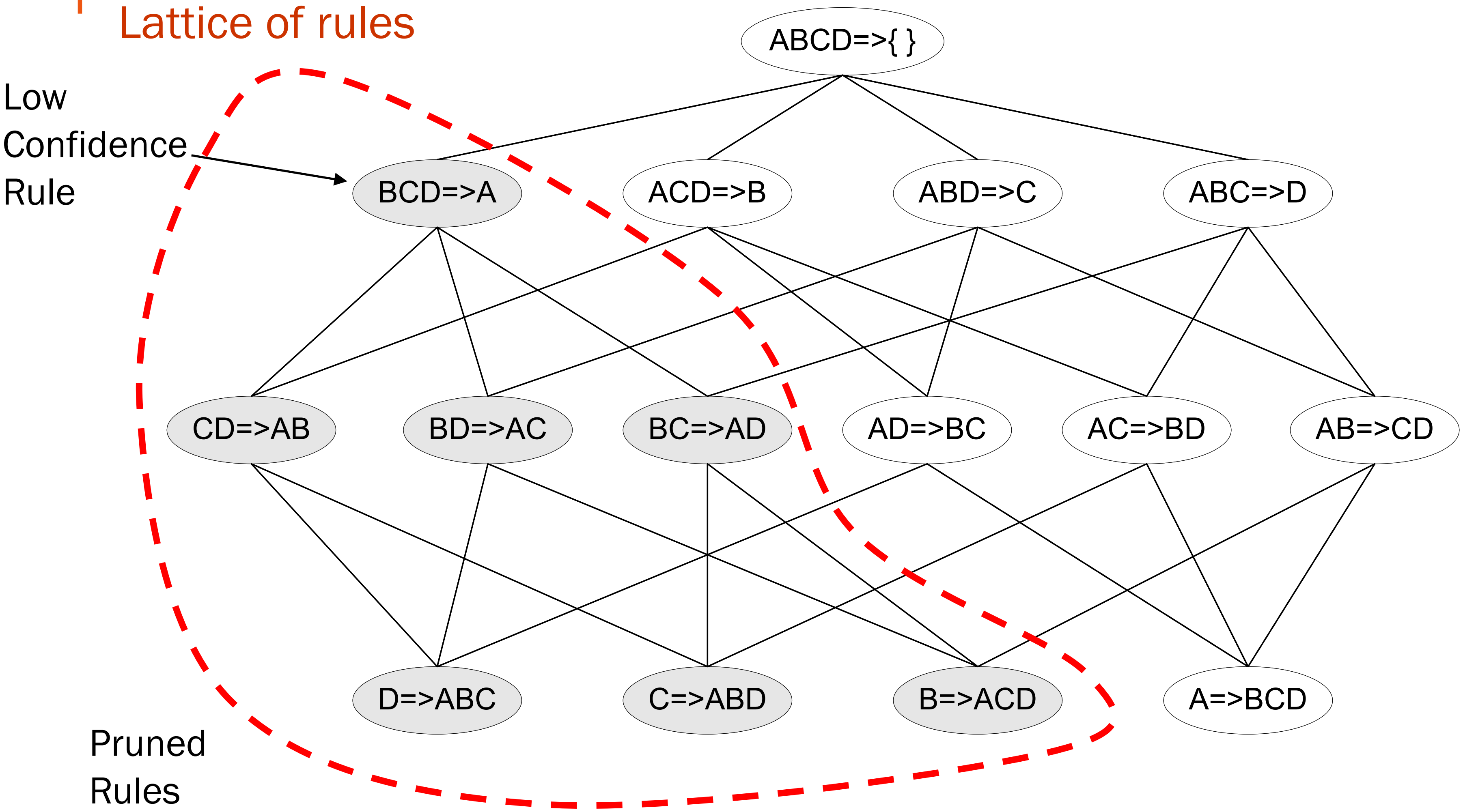
support(AB) ≥ support (ABC)

So C(ABC->D) ≥ C(AB->CD)

If min_conf is not satisfied, no need to generate rules with larger right-hand side (RHS).

THE APRIORI ALGORITHM: RULE PRUNING

Lattice of rules


ABCD=>{ }
BCD=>A
ACD=>B
ABD=>C
ABC=>D
CD=>AB
BD=>AC
BC=>AD
AD=>BC
AC=>BD
AB=>CD
D=>ABC
C=>ABD
B=>ACD
A=>BCD
Low Confidence Rule
Pruned Rules