# Frequent Pattern Mining

Toon Calders
University of Antwerp

# Summary

- Frequent Itemset Mining
- Algorithms
- Constraint Based Mining
- Condensed Representations

# Frequent Itemset Mining

- Market-Basket Analysis

transaction identifier

items

transaction

| TID | A | B | C | D |
|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 |
| 2 | 0 | 1 | 1 | 0 |
| 3 | 1 | 0 | 1 | 1 |
| 4 | 1 | 1 | 1 | 1 |
| 5 | 0 | 1 | 0 | 1 |

# Frequent Itemset Mining

- support(I): number of transactions "containing I"

| TID | A | B | C | D |
| --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 1 | 0 |
| 2 | 0 | 1 | 1 | 0 |
| 3 | 1 | 0 | 1 | 1 |
| 4 | 1 | 1 | 1 | 1 |
| 5 | 0 | 1 | 0 | 1 |

Support(BC) = 3

Support(ACD) = 2

# Frequent Itemset Mining Problem

Given D, minsup

Find all sets I with support(I) ≥ minsup

| TID | A | B | C | D |
|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 |
| 2 | 0 | 1 | 1 | 0 |
| 3 | 1 | 0 | 1 | 1 |
| 4 | 1 | 1 | 1 | 1 |
| 5 | 0 | 1 | 0 | 1 |

minsup=2

{}, A, B, C, D,
AC, AD, BC, BD, CD,
ACD

# Why?

- Important component in mining algorithms
- Sufficient statistics for interestingness measures
  - Confidence $X \rightarrow Y$ : Support(XY)/Support(X)
  - Contingency tables (correlation, $X^2$)

| | X | ¬X |
|---|---|---|
| Y | s(XY) | s(Y) - s(XY) |
| ¬Y | s(X) - s(XY) | s({}) - s(X) -s(Y) + s(XY) |

# Summary

- Frequent Itemset Mining
- Algorithms
- Constraint Based Mining
- Condensed Representations

# Algorithms

- There exist hundreds of algorithms that solve FIM (or related problems)
  - AIS, Apriori, AprioriTID, AprioriHybrid, FPGrowth, FPGrowth*, Eclat, dEclat, Pincer-search, ABS, DCI, kDCI, LCM, AIM, PIE, ARMOR, AFOPT, COFI, Patricia, MAXMINER, MAFIA, NDI-ALL, …

# Algorithms

- There exist hundreds of algorithms that solve FIM (or related problems)
- Concentrate on the most important pruning principle:
  - Monotonicity

and the two main search strategies:

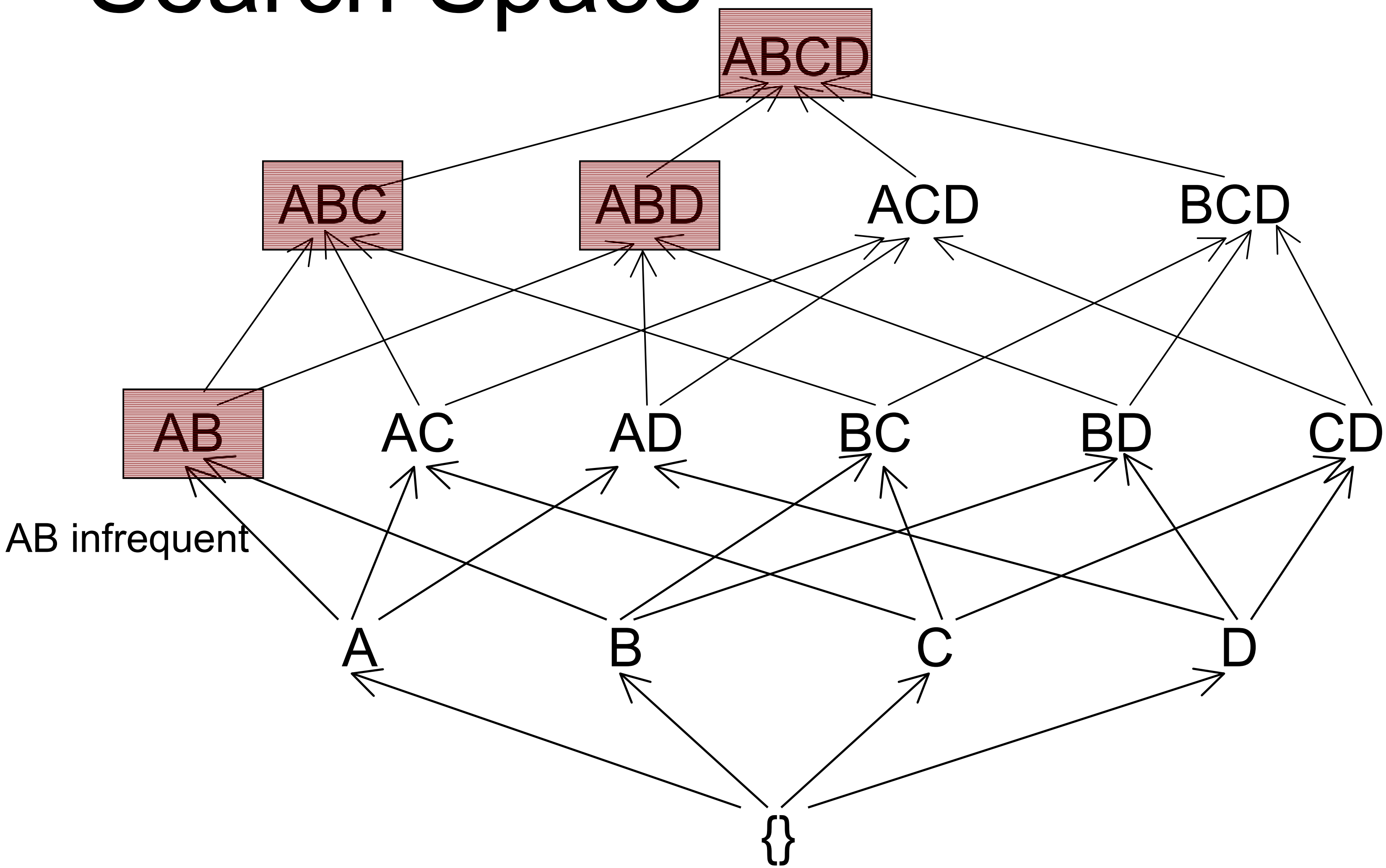
  - Breadth-first
  - Depth-first

# Monotonicity Principle

- If $I \subseteq J$, then $support(I) \geq support(J)$
- Therefore, if I is infrequent, then all its supersets are infrequent as well.

- All FIM algorithms rely heavily on this principle to prune large parts of the search space.

# Search Space



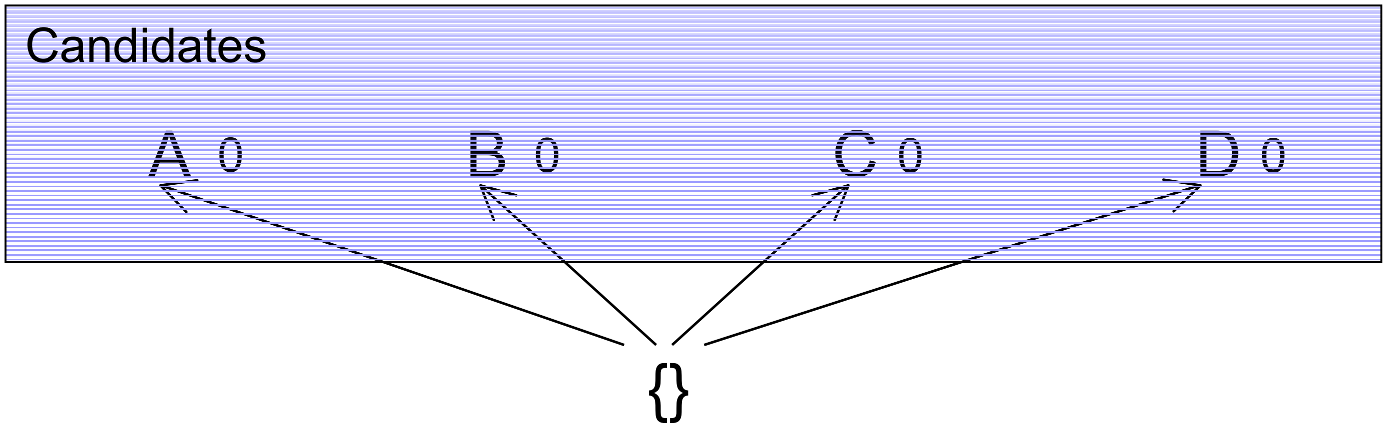
# Levelwise Algorithm

- Exploits monotonicity as much as possible.
- Search Space is traversed bottom-up, level by level
- Support of an itemset is only counted in the database if all its subsets were frequent.

# Apriori

| TID | A | B | C | D |
|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 |
| 2 | 0 | 1 | 1 | 0 |
| 3 | 1 | 0 | 1 | 1 |
| 4 | 1 | 1 | 1 | 1 |
| 5 | 0 | 1 | 0 | 1 |

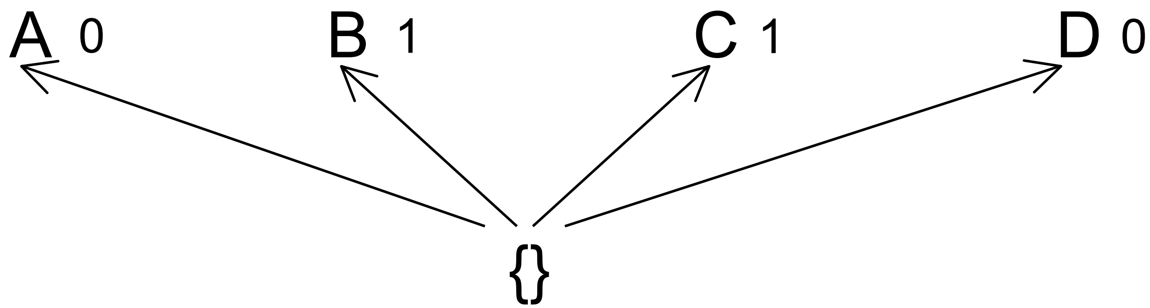
minsup=2



# Apriori

| TID | A | B | C | D |
|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 |
| 2 | 0 | 1 | 1 | 0 |
| 3 | 1 | 0 | 1 | 1 |
| 4 | 1 | 1 | 1 | 1 |
| 5 | 0 | 1 | 0 | 1 |

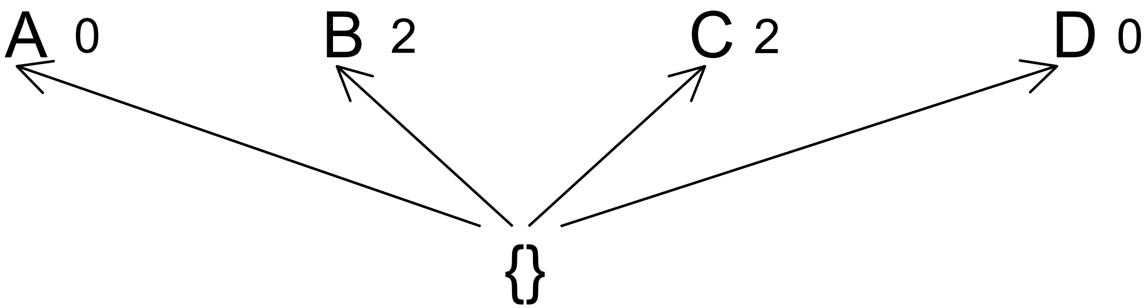
minsup=2



| TID | A | B | C | D |
|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 |
| 2 | 0 | 1 | 1 | 0 |
| 3 | 1 | 0 | 1 | 1 |
| 4 | 1 | 1 | 1 | 1 |
| 5 | 0 | 1 | 0 | 1 |

# Apriori

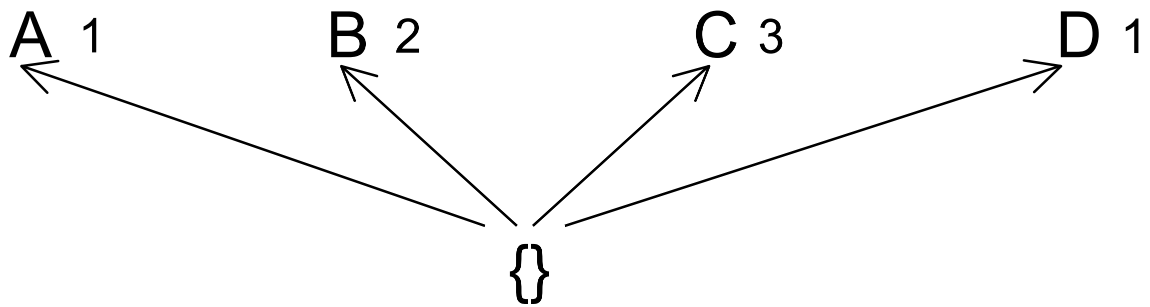
minsup=2



| TID | A | B | C | D |
|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 |
| 2 | 0 | 1 | 1 | 0 |
| 3 | 1 | 0 | 1 | 1 |
| 4 | 1 | 1 | 1 | 1 |
| 5 | 0 | 1 | 0 | 1 |

# Apriori

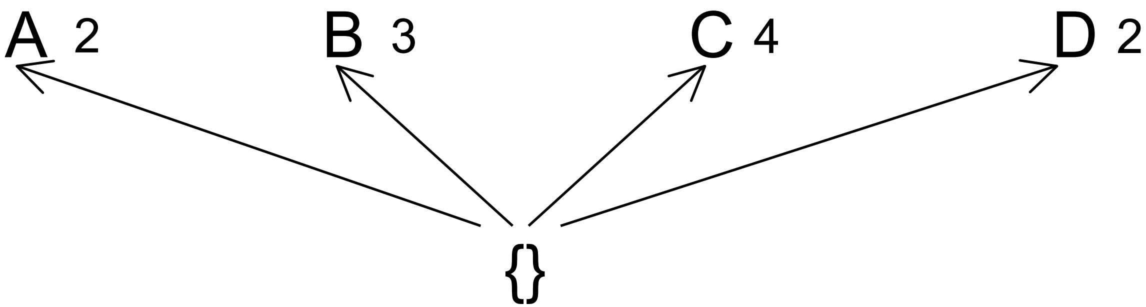
minsup=2



# Apriori

| TID | A | B | C | D |
|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 |
| 2 | 0 | 1 | 1 | 0 |
| 3 | 1 | 0 | 1 | 1 |
| 4 | 1 | 1 | 1 | 1 |
| 5 | 0 | 1 | 0 | 1 |

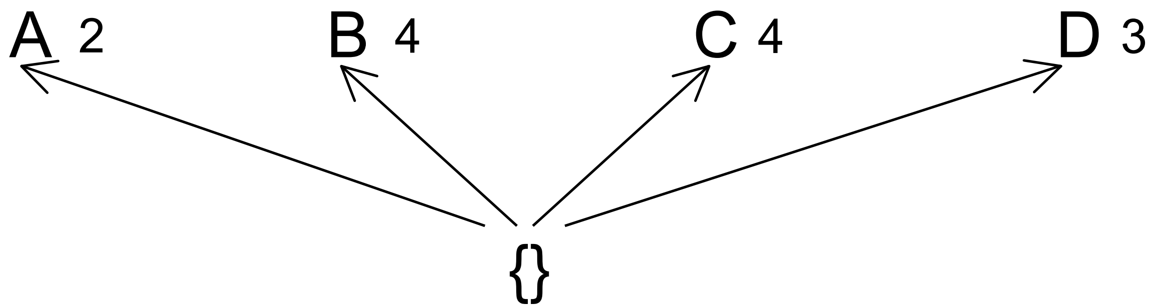
minsup=2



# Apriori

| TID | A | B | C | D |
| --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 1 | 0 |
| 2 | 0 | 1 | 1 | 0 |
| 3 | 1 | 0 | 1 | 1 |
| 4 | 1 | 1 | 1 | 1 |
| 5 | 0 | 1 | 0 | 1 |

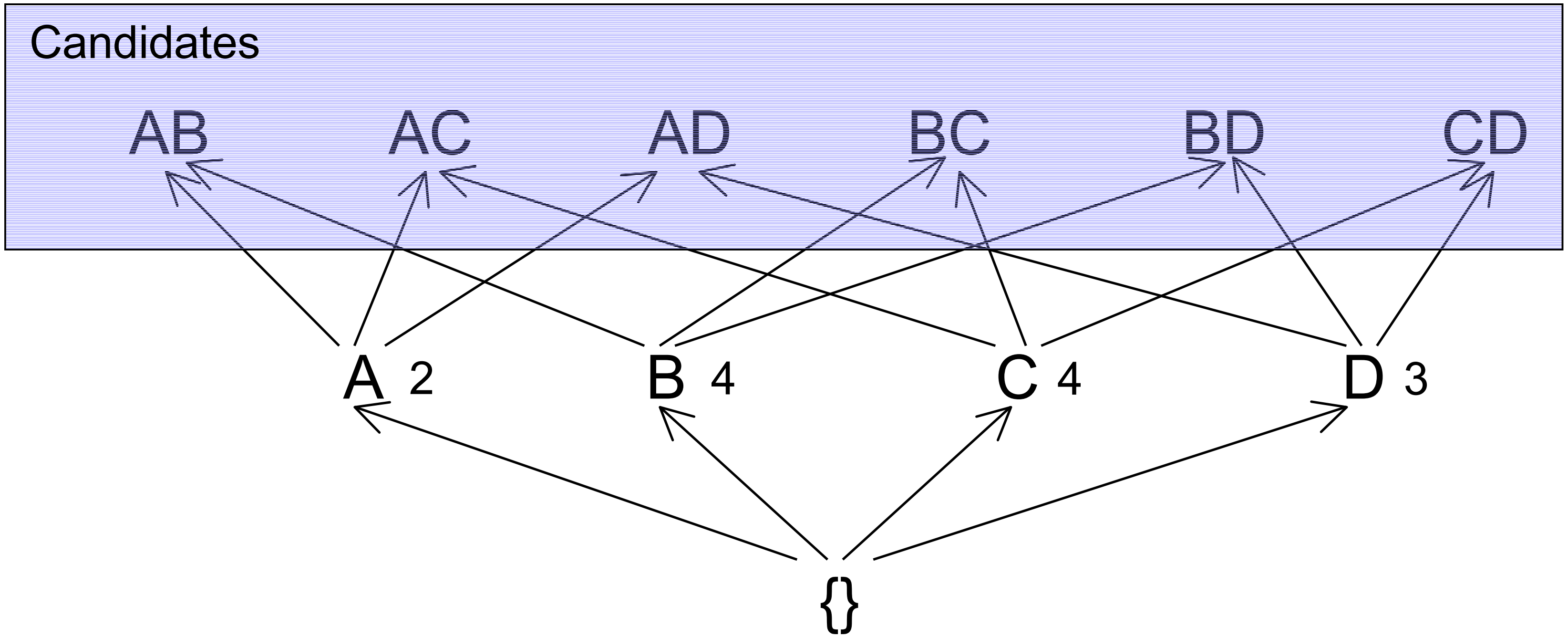
minsup=2



| TID | A | B | C | D |
|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 |
| 2 | 0 | 1 | 1 | 0 |
| 3 | 1 | 0 | 1 | 1 |
| 4 | 1 | 1 | 1 | 1 |
| 5 | 0 | 1 | 0 | 1 |

# Apriori

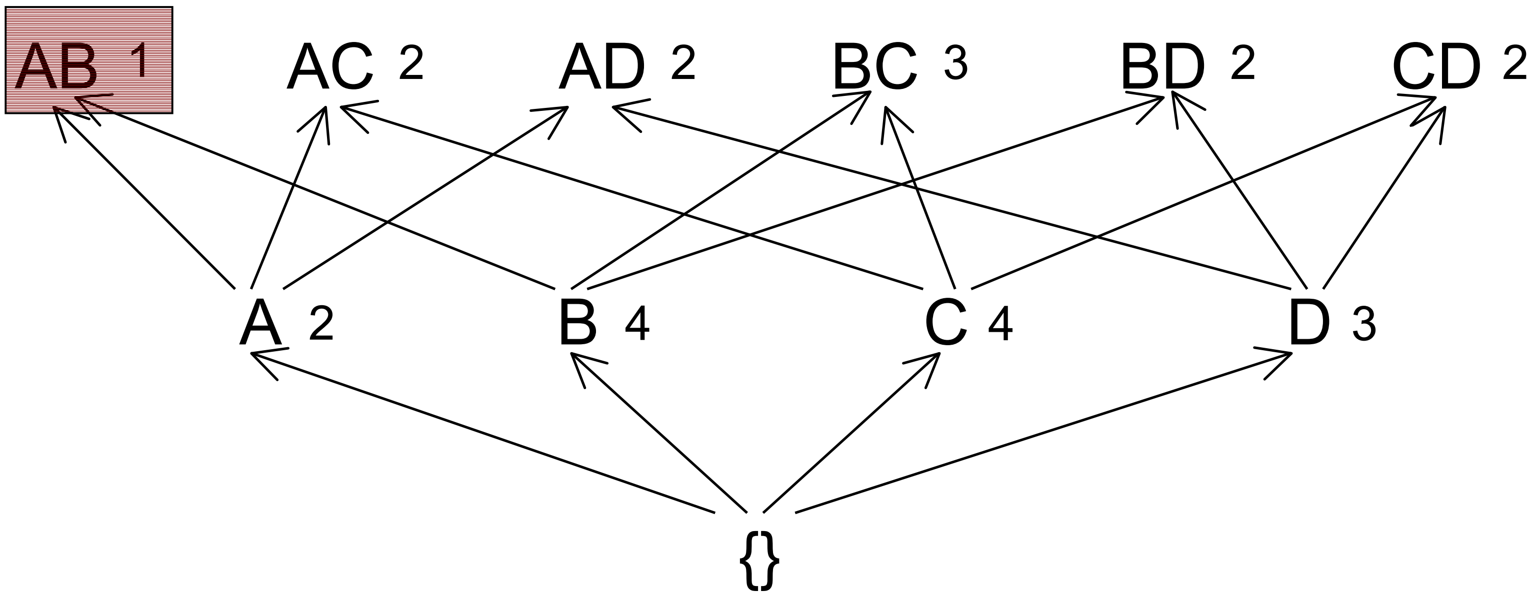
minsup=2



# Apriori

| TID | A | B | C | D |
|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 |
| 2 | 0 | 1 | 1 | 0 |
| 3 | 1 | 0 | 1 | 1 |
| 4 | 1 | 1 | 1 | 1 |
| 5 | 0 | 1 | 0 | 1 |

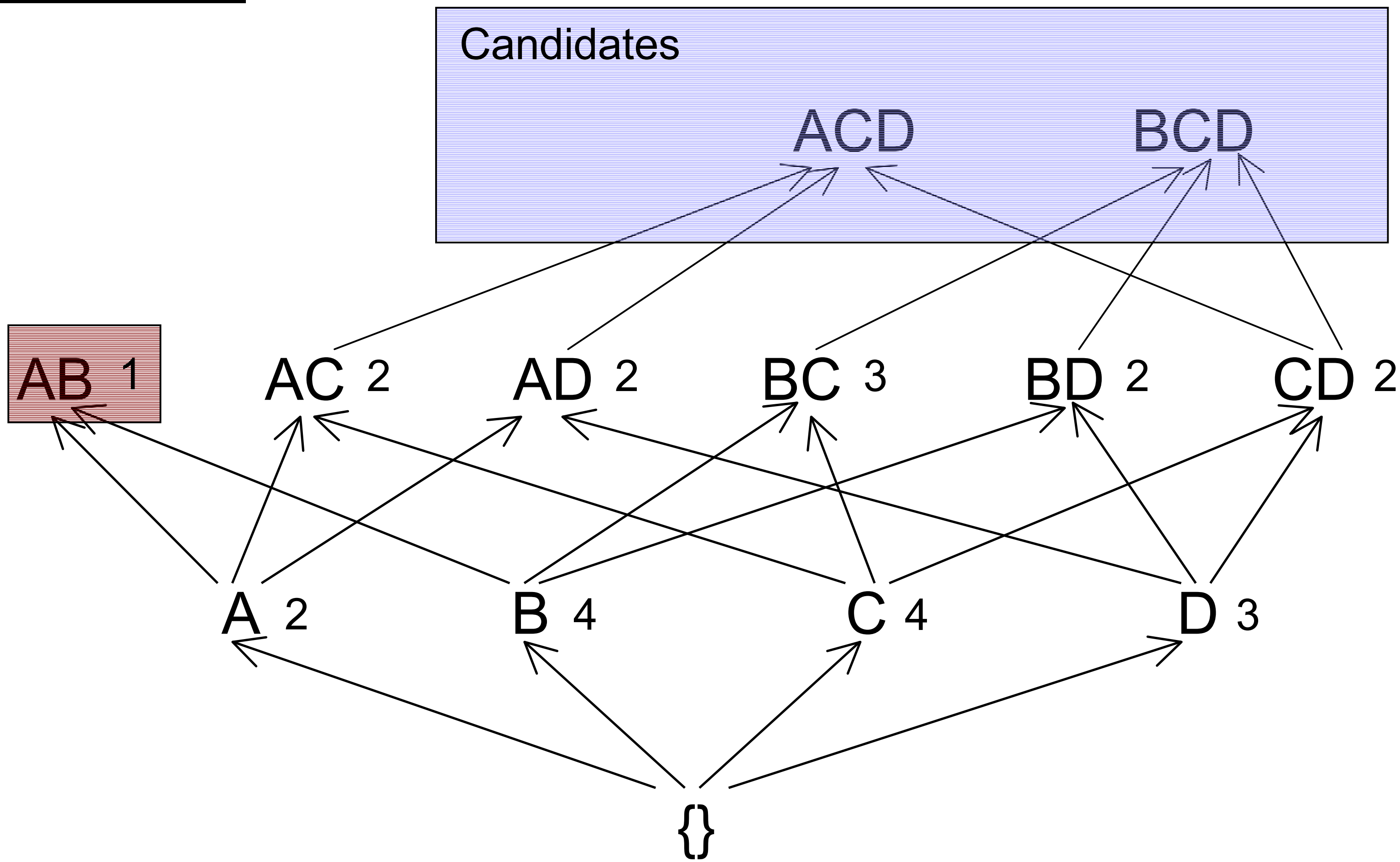
minsup=2



# Apriori

| TID | A | B | C | D |
|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 |
| 2 | 0 | 1 | 1 | 0 |
| 3 | 1 | 0 | 1 | 1 |
| 4 | 1 | 1 | 1 | 1 |
| 5 | 0 | 1 | 0 | 1 |

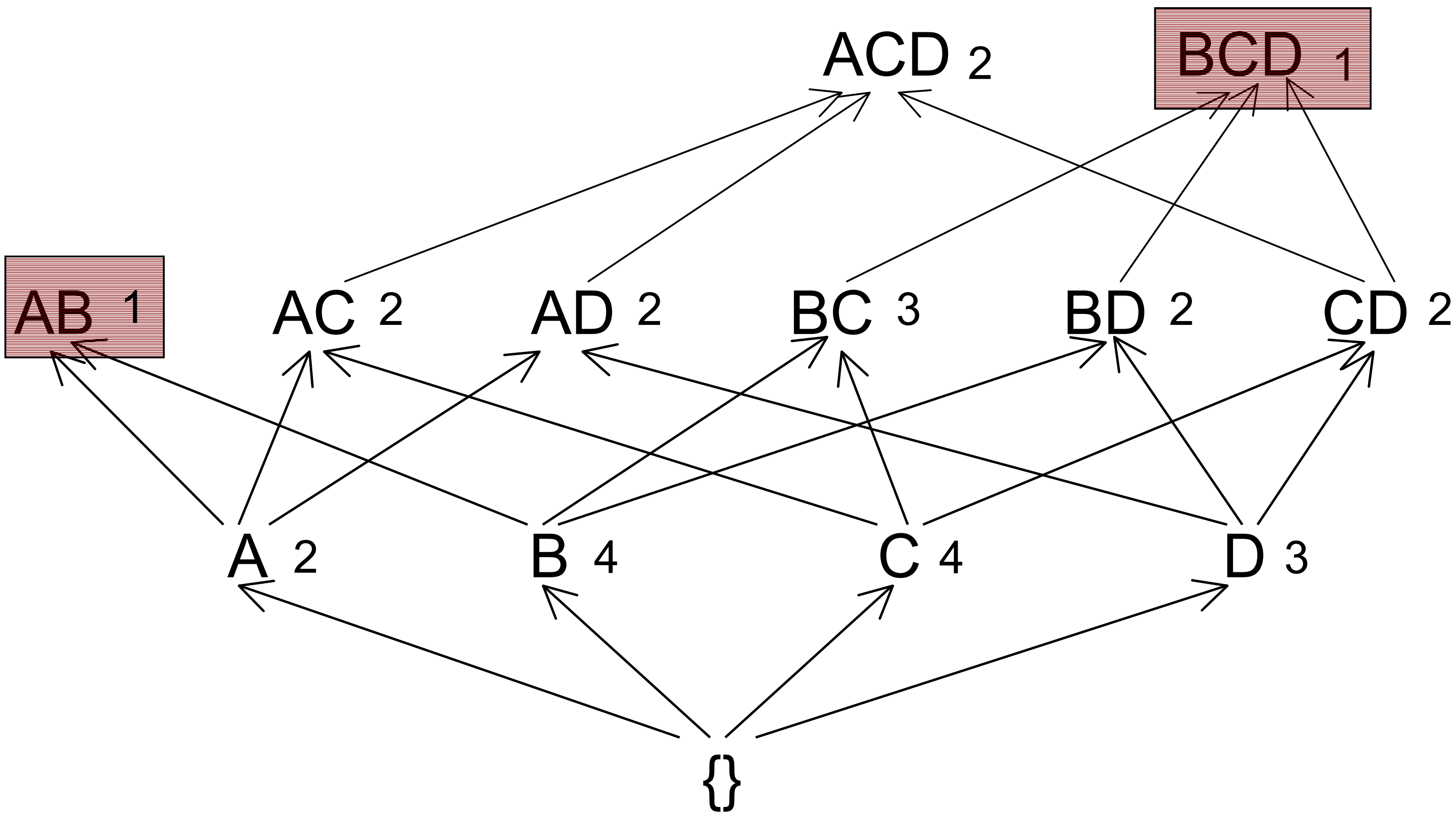
minsup=2



# Apriori

| TID | A | B | C | D |
|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 |
| 2 | 0 | 1 | 1 | 0 |
| 3 | 1 | 0 | 1 | 1 |
| 4 | 1 | 1 | 1 | 1 |
| 5 | 0 | 1 | 0 | 1 |

minsup=2



# Depth-First Algorithms

Find all frequent itemsets

| TID | A | B | C | D |
|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 |
| 2 | 0 | 1 | 1 | 0 |
| 3 | 1 | 0 | 1 | 1 |
| 4 | 1 | 1 | 1 | 1 |
| 5 | 0 | 1 | 0 | 1 |

Find all frequent itemsets, with D

| TID | A | B | C |
|---|---|---|---|
| 3 | 1 | 0 | 1 |
| 4 | 1 | 1 | 1 |
| 5 | 0 | 1 | 0 |

Find all frequent itemsets, without D

| TID | A | B | C |
|---|---|---|---|
| 1 | 0 | 1 | 1 |
| 2 | 0 | 1 | 1 |
| 3 | 1 | 0 | 1 |
| 4 | 1 | 1 | 1 |
| 5 | 0 | 1 | 0 |

# Depth-First Algorithm

A,B,C,D
AD, BD, CD
ACD
AC, BC

| TID | A | B | C | D |
|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 |
| 2 | 0 | 1 | 1 | 0 |
| 3 | 1 | 0 | 1 | 1 |
| 4 | 1 | 1 | 1 | 1 |
| 5 | 0 | 1 | 0 | 1 |

DB[D]

| TID | A | B | C |
|---|---|---|---|
| 3 | 1 | 0 | 1 |
| 4 | 1 | 1 | 1 |
| 5 | 0 | 1 | 0 |

DB[CD]

| TID | A |
|---|---|
| 3 | 1 |
| 4 | 1 |

DB[BD]

| TID |
|---|

DB[C]

| TID | A | B |
|---|---|---|
| 1 | 0 | 1 |
| 2 | 0 | 1 |
| 3 | 1 | 0 |
| 4 | 1 | 1 |

DB[BC]

| TID |
|---|

DB[B]

| TID |
|---|

# Breadth-First vs Depth-First

- Depth-first outperformes breadth-first
  - Number of frequent itemsets is very high
  - Database is relatively small
- Breadth-first outperformes depth-first
  - Number of frequent sets is small
  - Database is large
- Differences usually very small

# Summary

- Frequent Itemset Mining
- Algorithms
- Constraint Based Mining
- Condensed Representations

# Mining With Constraints

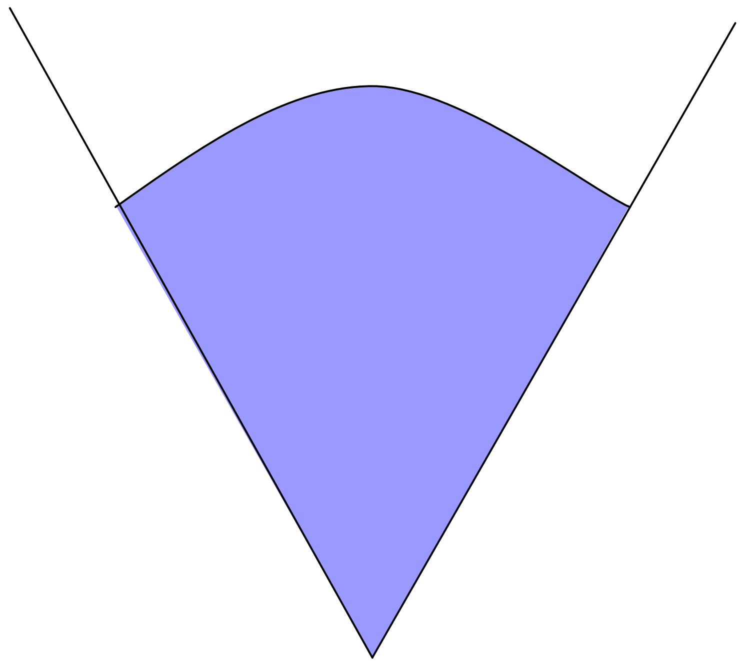
- Reduce output size, user sets focus
  - itemsets of size > 5
  - sets of products with cost less than 10 EUR
  - sets that contain A, B, or C.
  - sets that are frequent in dataset $D_1$, but infrequent in $D_2$

# Mining With Constraints

- Types of constraints
  - (Anti-)Monotone,
  - Succinct
  - Convertible
- Two Approaches
  - Pushing constraints into the mining algorithm
  - Changing the Database

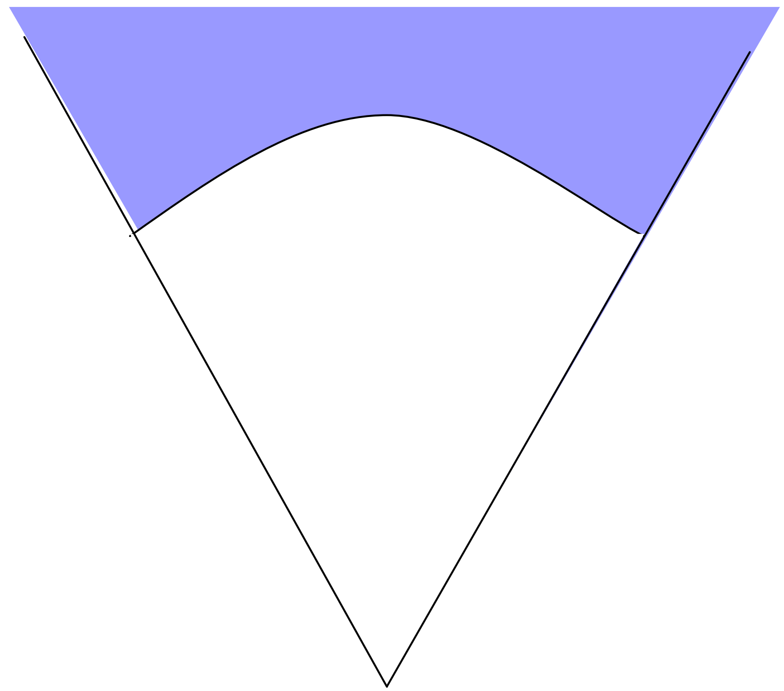
# Types of Constraints

- Anti-monotone
  - Support, size < 10, …



# Types of Constraints

- Monotone
  - Cost >10EUR, Contains A, B, or C, …



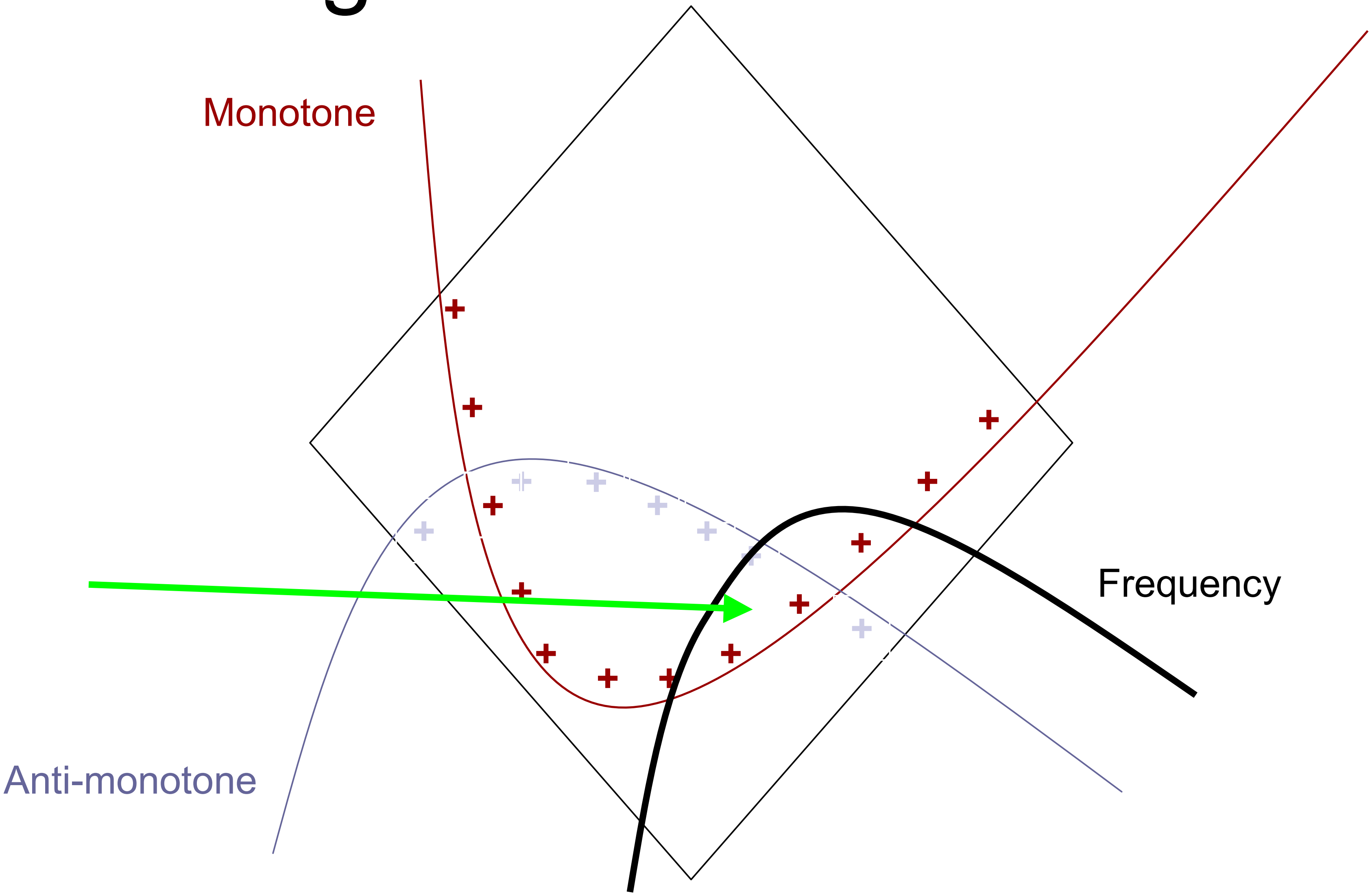
# Types of Constraints

- Succinct
  - Can be expressed using minus and union on a fixed number of powersets
    - E.g., Contains A or B, but not C: $2^{I-C} – 2^{I-AB}$
  - Can be generated efficiently
- Convertible anti-monotone
  - Anti-Monotone w.r.t. prefix-order
    - E.g. avg(I.price)<10 EUR when ordered ascending by price.

# Mining With Constraints

- Two approaches:
  - Pushing constraints deep in data mining algorithm
  - Changing database such that
    - Support of itemsets satisfying the constraint does not change
    - The support of itemsets that do not satisfy the constraint decreases

# Pushing Constraints



# Pushing Constraints

- Trade-off
  - Pushing monotone constraints
  - vs. anti-monotone pruning
- Not always better to push monotone constraints
  - E.g. Size > 10 …

# Changing the Database

- ExAnte Algorithm
  - Exploit Monotone and Anti-monotone constraints
  - A transaction that does not satisfy a monotone constraint will not contribute to any itemset satisfying the constraints
    - E.g. constraint “size > 10”: every transaction of size < 10 can be thrown away!

# Changing the Database

minsup = 3 anti-mon.
size ≥ 4 monotone

| ID | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 2 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 3 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| 4 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 5 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 6 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 7 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |

# Summary

- Frequent Itemset Mining
- Algorithms
- Constraint Based Mining
- Condensed Representations

# Condensed Representations

- Sometimes, the output of frequent set mining remains too large:
  - Huge number of items
  - Highly correlated
  - High support items
- Hence, instead of mining all itemsets
  - Condensed representation

# Condensed Representations

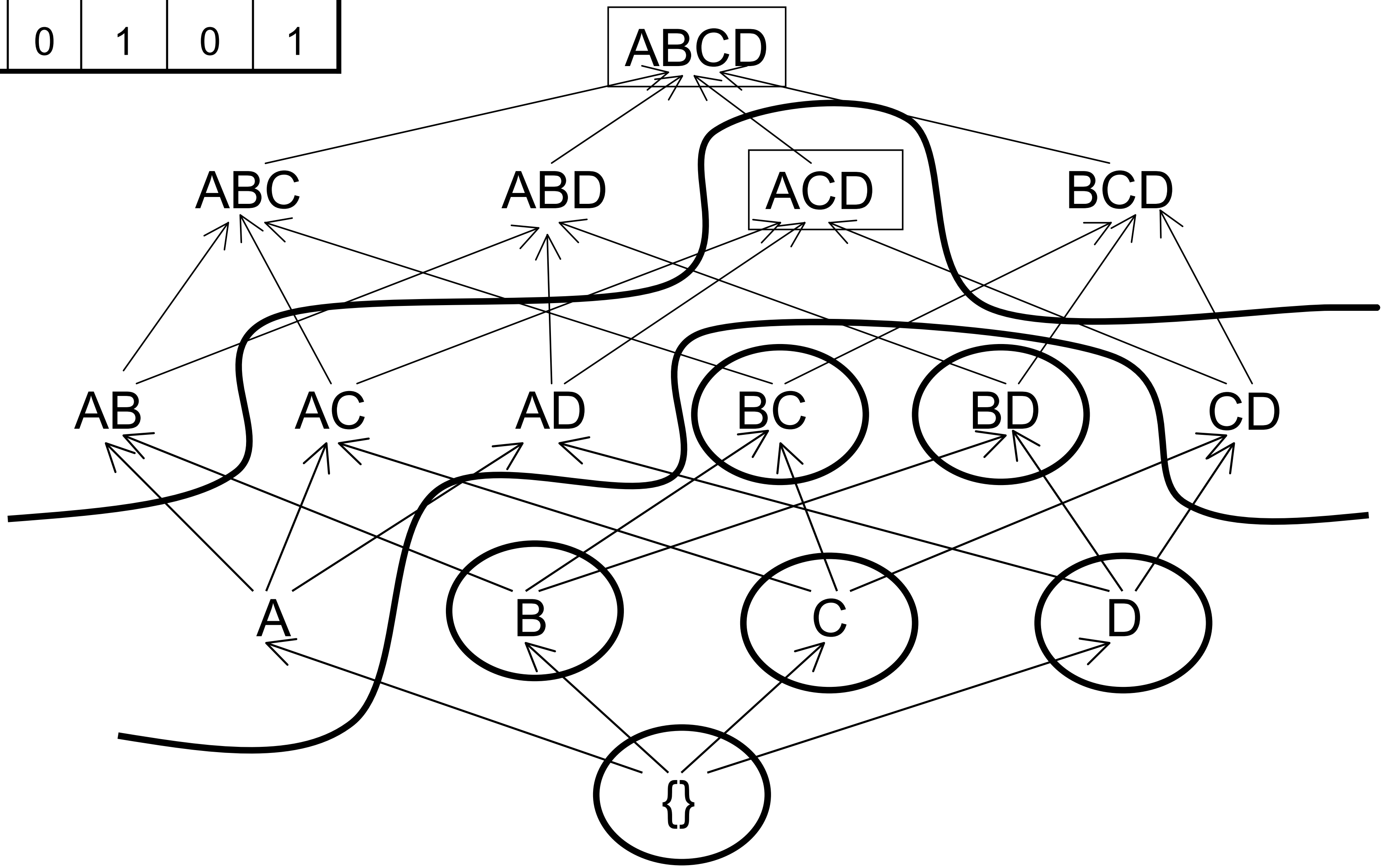
- Closed sets
  - Divide frequent itemsets into equivalence classes
  - Two itemsets are equivalent if they occur in the same transactions
  - Closed set: maximal element in an equivalence class

# Closed Itemsets

- All sets in the same equivalence class have the same support
  - Occur in the same transactions
- Maximal element in an equivalence class is unique
  - If two itemsets occur in the same transactions, then so does their union

# Closed Itemsets

| TID | A | B | C | D |
|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 |
| 2 | 0 | 1 | 1 | 0 |
| 3 | 1 | 0 | 1 | 1 |
| 4 | 1 | 1 | 1 | 1 |
| 5 | 0 | 1 | 0 | 1 |



# Closed Itemsets

- Has nice mathematical properties
  - Closed sets form a lattice
  - Galois connection
- Efficient algorithms to find them
- Based on the closed sets, it is easy to find the support of the other itemsets.

# Closed Itemsets

- Interesting class of patterns
  - Maximal frequent itemsets are closed sets
  - Highest correlation between items
  - Strongest association rules
- Significant reduction of number of itemsets
  - Especially with small number of large transactions

# Non-Derivable Itemsets

- Based on redundancies
  - How do supports interact?

- What information about unknown supports can we derive from known supports?
  - Concise representation: only store relevant part of the supports

# Redundancies

- Agrawal et al. (Monotonicity)
  - $\text{Supp}(AX) \leq \text{Supp}(A)$
- Boulicaut et al., Lakhal et al. (Free sets)
  - **If** $\text{Supp}(A) = \text{Supp}(AB)$ (Closed sets)

    **Then** $\text{Supp}(AX) = \text{Supp}(AXB)$

# Redundancies

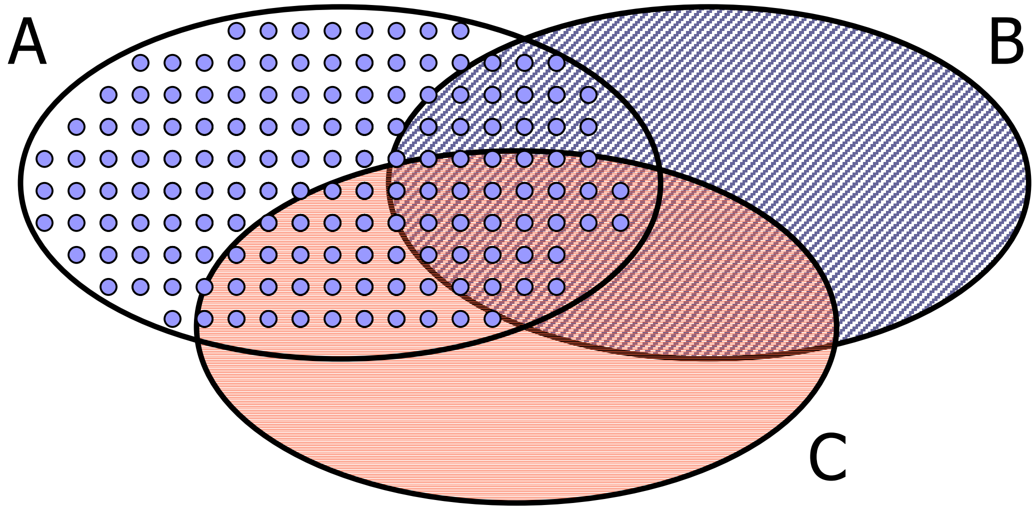
- Bayardo (MAXMINER)
  - $Supp(ABX) \geq Supp(AX) - (\underline{Supp(X)-Supp(BX)})$

    drop (X, B)

- Bykowski, Rigotti (Disjunction-free sets)

  if Supp(ABC) = Supp(AB) + Supp(AC) – Supp(A), then Supp(ABCX) can be derived from ABX, ACX, AX

# The Inclusion – Exclusion Principle




$$|A \cup B \cup C| = |A| + |B| + |C|$$
$$- |A \cap B| - |A \cap C| - |B \cap C|$$
$$+ |A \cap B \cap C|$$

# Deduction Rules via Inclusion-Exclusion

- Let A, B, C, … be items
- Let A' correspond with the set

  { transaction t | t contains A }
- $AB' = A' \cap B'$

- Then: Supp(ABC) = | ABC' |

# Deduction Rules via Inclusion-Exclusion

- Inclusion-exclusion principle:

$$| A' \cup B' \cup C' | = |A'| + |B'| + |C'| - |AB'| - |AC'| - |BC'| + |ABC'|$$

Thus, since $| A' \cup B' \cup C' | \leq n$,

$$\mathbf{Supp(ABC) \leq s(AB) + s(AC) + s(BC) - s(A) - s(B) - s(C) + n}$$

# Complete Set for Supp(ABC)

| | | |
|---|---|---|
| 0 | $s_{ABC} \geq 0$ | |
| 1 | $s_{ABC} \leq s_{AB}$<br>$s_{ABC} \leq s_{AC}$<br>$s_{ABC} \leq s_{BC}$ | Monotonicity |
| | | Free, Closed |
| 2 | $s_{ABC} \geq s_{AB} + s_{AC} - s_A$<br>$s_{ABC} \geq s_{AB} + s_{BC} - s_B$<br>$s_{ABC} \geq s_{AC} + s_{BC} - s_C$ | Disjunction-Free |
| 3 | $s_{ABC} \leq s_{AB} + s_{AC} + s_{BC} - s_A - s_B - s_C + n$ | |

# Derivable Itemsets

Given: Supp(I) for all I $\subset$ J

Lower bound on Supp(J) = l

Upper bound on Supp(J) = u

- Without counting : Supp(J) $\in$ [l,u]
- J is a ***derivable itemset*** (DI) iff l = u
  - We **know** Supp(J) **exactly** without counting!

# Derivable Itemsets

J derivable itemset:

- No need to ***count*** Supp(J)
- No need to ***store*** Supp(J)
  - We can use the deduction rules

- Concise representation:

  $$C = \{ ( J, \mathrm{Supp}(J) ) \mid J \text{ not derivable from } \mathrm{Supp}(I), I \subset J \}$$

# Derivable Itemsets

Theorem (Monotonicity)

If $J \subset K$, $J$ derivable, then $K$ derivable.

Moreover:

The width of the interval for $J \cup \{A\}$ is at most ***half*** the size of the interval for $J$

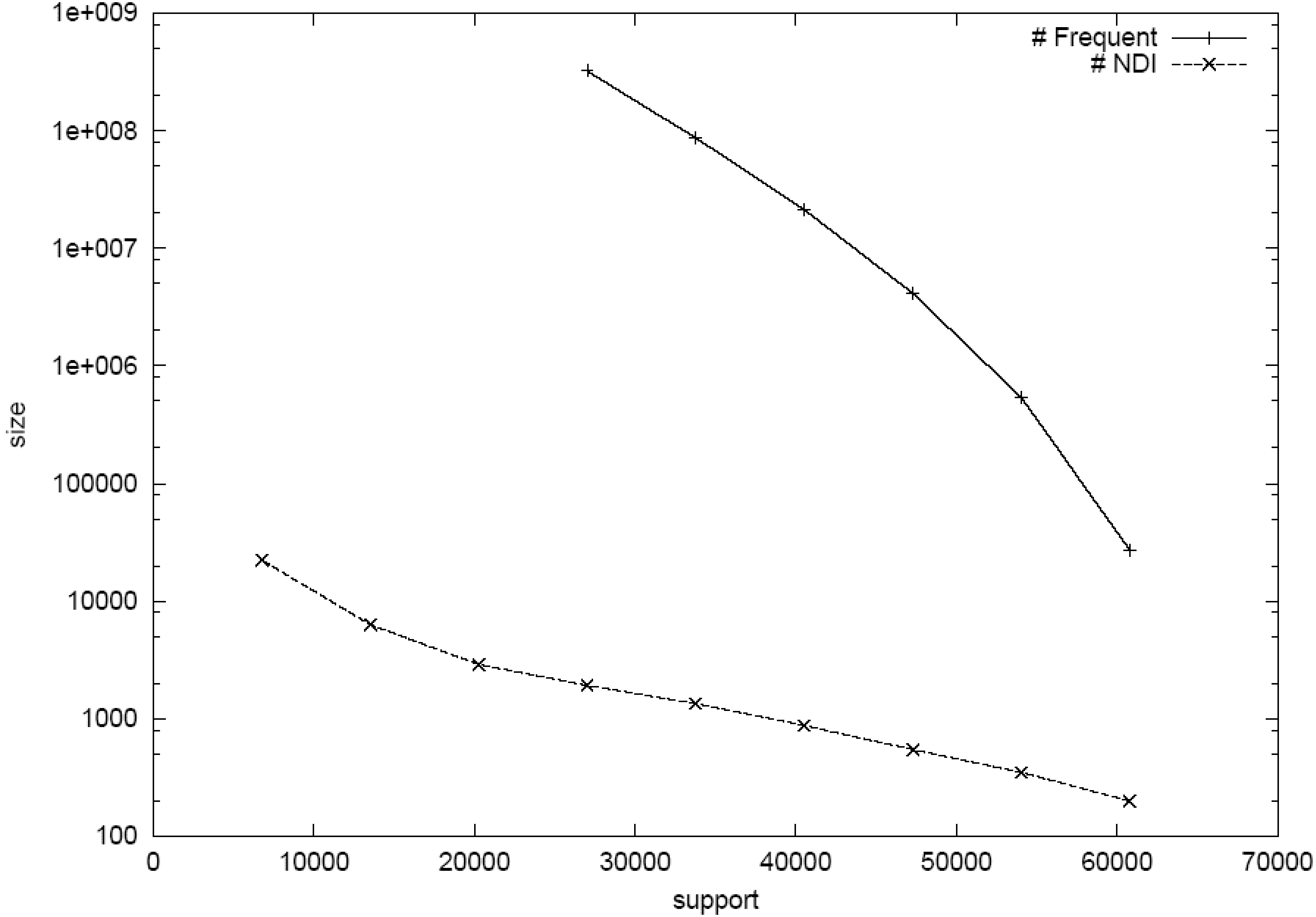
# IV. Evaluation --- Theoretical

- Interval widths decrease exponentially
  - Half each step

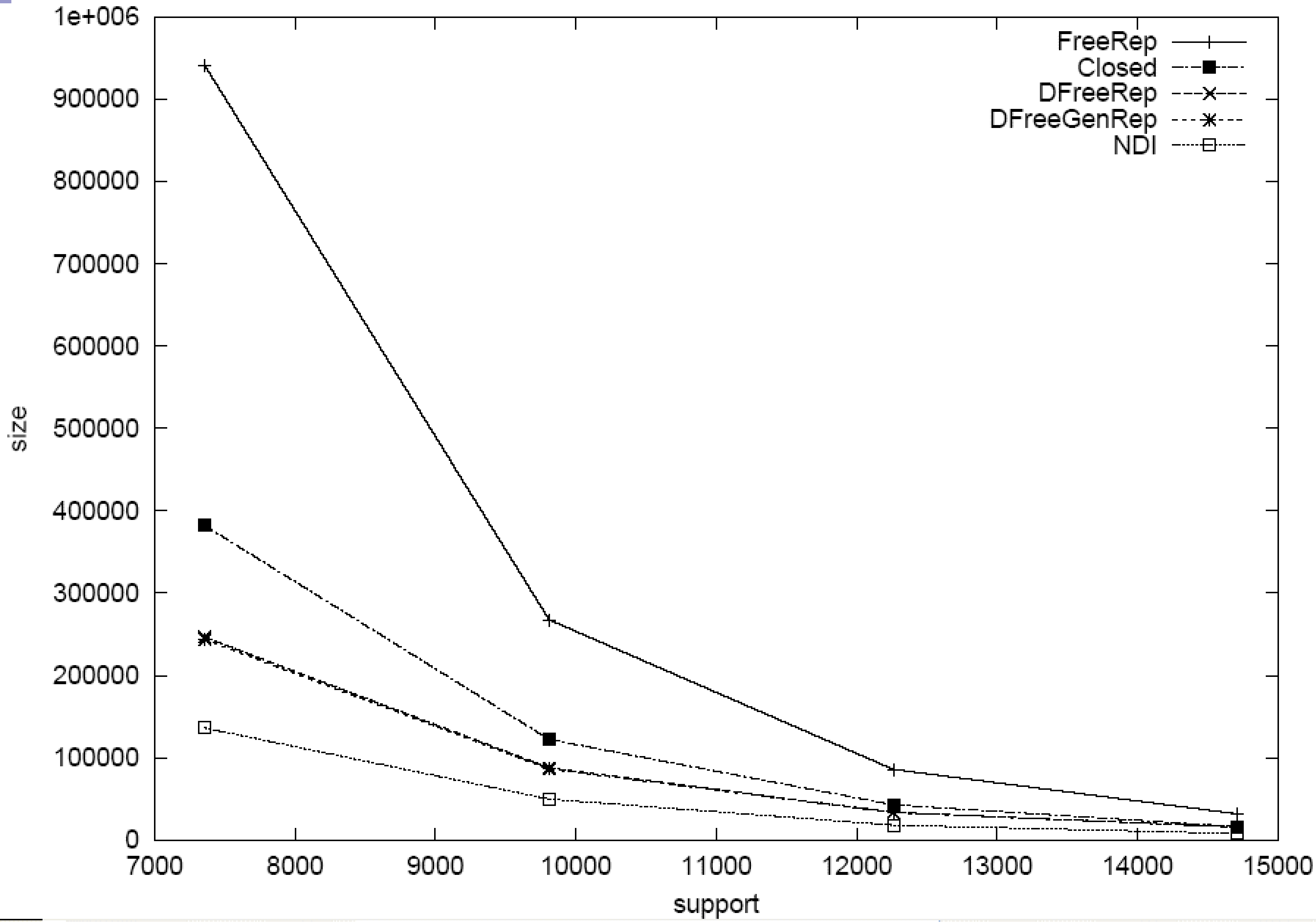
- Non-derivable itemset can never be larger than log(|Database|)
  - Independent of sparse, dense, ...

# Evaluation --- Empirically

- Size NDI vs. frequent itemsets
- Comparison with Other Concise Reps


1e+009
1e+008
1e+007
1e+006
100000
10000
1000
100
size
0
10000
20000
30000
40000
50000
60000
70000
support
# Frequent
# NDI

PUMSB



1e+006
900000
800000
700000
600000
500000
400000
300000
200000
100000
0
size
7000
8000
9000
10000
11000
12000
13000
14000
15000
support
FreeRep
Closed
DFreeRep
DFreeGenRep
NDI


PUMSB

# Evaluation

- Number of frequent NDIs considerable smaller than number of frequent itemsets
- Algorithm is efficient
  - Calculating NDI + deducing DIs often outperforms Apriori

# Condensed Representations

- Many other representations
  - Free sets
  - Disjunction-free sets
  - Generalized disjunction-free sets
  - …
- Closed sets and NDIs provable the smallest ones

# Conclusion

- Depth-first vs Breadth-first algorithms for FIM
- Constraint mining to incorporate user focus
  - Pushing constraints vs changing database
- Condensed Representations
  - Closed sets
  - Non-Derivable Itemsets

# Topics Not Covered …

Parallel algorithms for FIM

Incremental FIM

Generalized, Quantitative, Multi-level, Fuzzy ARs

Coupling FIM with RDBMS

Privacy Preserving ARM

Computational Complexity Results

Inverse mining problem

Emerging Patterns, jumping emerging patterns

Dependency value, $X^2$

Lift, gain

Block support, tilings,

…