Unsupervised Data Mining: From Batch to Stream Mining Algorithms

Prof. Dr. Stefan Kramer

Johannes Gutenberg-Universität Mainz

Outline

- Itemsets and APriori

Itemsets and APriori

Example Microarray Data

	ARG1	ARG4	ARO3	….	LYS1
1	1	1	1	…	0
2	1	1	1	…	1
3	0	1	1	…	1
4	0	1	0	…	1
5	1	1	1	…	0
6	0	0	0	…	0
7	…	….	…	…	…

Before data mining step: data cleaning, sampling, discretization, feature selection, etc.

Another Representation

	ARG1	ARG4	ARO3	...	LYS1
1	1	1	1	...	0
2	1	1	1	...	1
3	0	1	1	...	1
4	0	1	0	...	1
5	1	1	1	...	0
6	0	0	0	...	0
7	...	....	...	...	...

D = {{ARG1, ARG4, ARO3},
{ARG1, ARG4, ARO3, LYS1},
{ARG4, ARO3, LYS1},
{ARG4, LYS1},
{ARG1, ARG4, ARO3},
{},
...}

Multiset of itemsets

Association Rule Mining

Table in relational database

	ARG1	ARG4	ARO3	...	LYS1
1	1	1	1	...	0
2	1	1	1	...	1
3	0	1	1	...	1
4	0	1	0	...	1
5	1	1	1	...	0
6	0	0	0	...	0
7	...	....	...	...	...

Association rules

“**IF** ARG1 and HIS5
THEN LYS1”

support: 54 %
confidence: 93 %

“**IF** YOL118C
THEN ARG1”

support: 53 %
confidence: 88 %

Frequent Itemsets and Association Rules

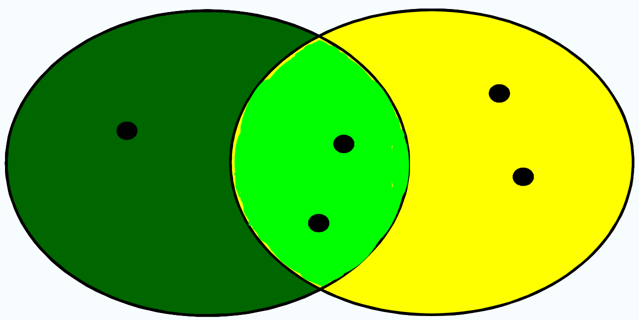
60 % of observations: ARO3 and LYS1 upregulated

80 % of observations: ARG1 upregulated

40 % of observations: ARO3, LYS1 and ARG1 upregulated

"**IF** ARO3 and LYS1
THEN ARG1"

support: 40 %
confidence: 67 %



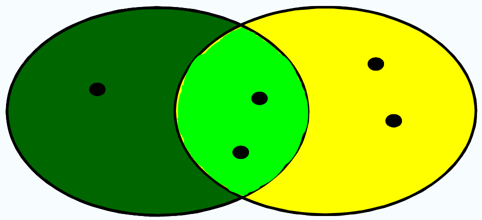
Two-Phased Algorithm

- *First phase*: find frequent itemsets (e.g., {ARO3, LYS1} , {ARG1} , {ARO3, LYS1, ARG1})
- *Second phase*: construct association rules (e.g., if {ARO3, LYS1} then {ARG1})

“**IF** ARO3 and LYS1
THEN ARG1”

support: 40 %
confidence: 67 %

{ARO3, LYS1} vs. {ARG1}

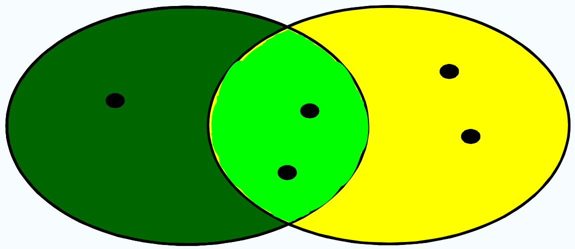


Support and Confidence

"**IF** ARO3 and LYS1
THEN ARG1"

support: 40 %
confidence: 67 %

{ARO3, LYS1} vs.
{ARG1}



"**IF** Y **THEN** X"

Support: $p(X,Y)$

Confidence: $p(X|Y) = \dfrac{p(X,Y)}{p(Y)}$

Frequent Pattern Discovery

Input:

- table D in relational database
- minimum support threshold: minSupport

Output:

- all patterns (here: itemsets) p for which
  freq(p, D) $\geq$ minSupport

How?

APriori Algorithm (Agrawal et al., 1993)

$i := 1$
$C_i := \{\{A\} | A \text{ is an item}\}$
while $C_i \neq \{\}$ **do**
    % candidate testing (database scan)
    for each set in C_i test whether it is frequent
    let F_i be the collection of frequent sets from C_i
    % candidate formation
    let C_{i+1} be those sets of size i+1 such that all subsets are in F_i (frequent)
    $i := i + 1$
return $\cup F_j$

Candidate Formation

- By *joining*: union of pairs of frequent itemsets from the previous level
- e.g., {A,B} and {B,C} gives {A,B,C}
- However, {A, C} might still be infrequent
- Thus, additional pruning step checking whether all subsets are known to be frequent

Main Ideas of APriori

- Each iteration consists of two phases
  - candidate formation
  - candidate testing (database scan)
- Minimize database scans

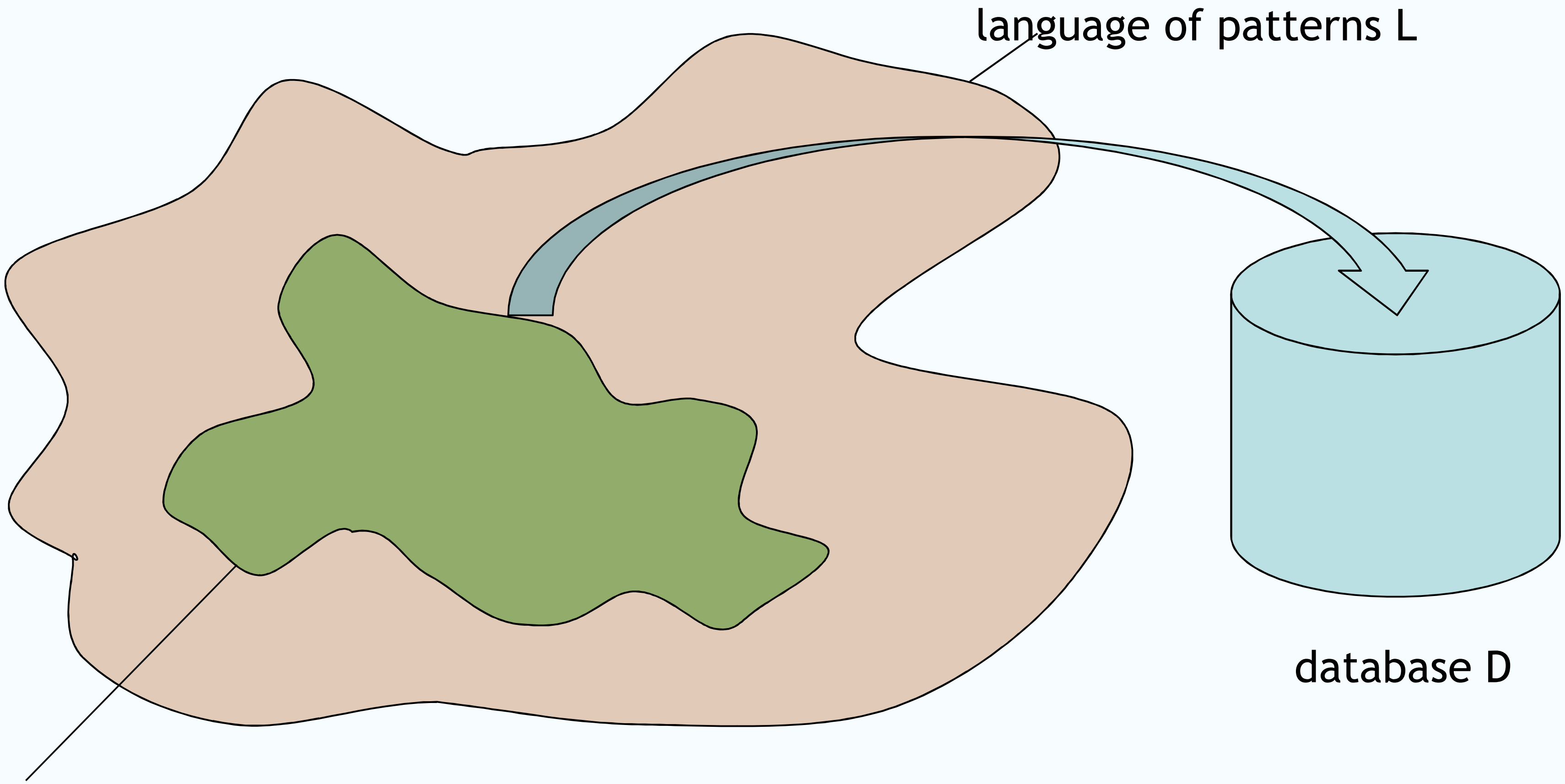
```
for each tuple t do
  for each candidate itemset i do
    …
```

- Avoid unnecessary tests on the database (test only those patterns that can, knowing the previous levels, be frequent)

Patterns (Itemsets) and (Association) Rules

- From frequent itemsets c and $c \cup \{i\}$ derive if c then {i}
- Start with the maximally specific frequent itemsets
- Variants possible: only one item in the RHS (very common assumption), only one item in the LHS (not very common)
- Generally: patterns and rules
  frequent patterns p, q such that $p \leq q$
  if p then q (with some confidence)

Formalization of Data Mining



q(p, D) … interestingness predicate: a pattern p from L is interesting wrt. database D

what is interesting? *frequent*, *non-redundant*, *class correlated*, *structurally diverse*, …

Formalization of Data Mining

- Simple formalization/definition of data mining (Mannila & Toivonen, 1997)
- Language L of patterns p
- Database D
- Interestingness predicate q
- Find a theory of the data:
  $Th(L, D, q) = \{p \in L \mid q(p,D) \text{ is true}\}$

Anti-Monotonicity and Monotonicity

L: language of patterns

Constraint is *anti-monotonic* iff

$\forall \phi, \gamma \in L: \phi < \gamma \wedge \gamma \in S \rightarrow \phi \in S$

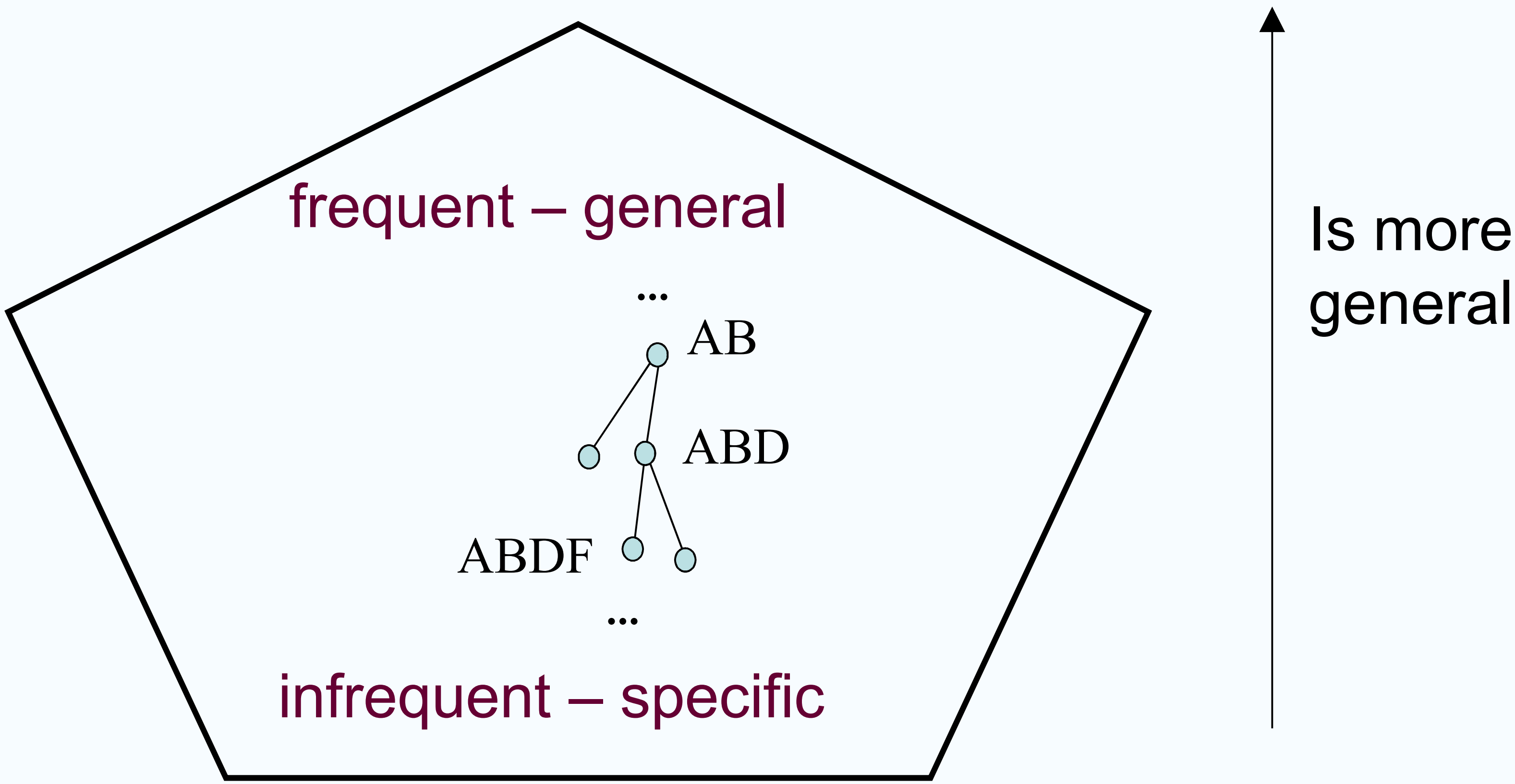
e.g., minimum frequency, $p \leq \{ABDF\}$

Constraint is *monotonic* iff

$\forall \phi, \gamma \in L: \phi < \gamma \wedge \phi \in S \rightarrow \gamma \in S$

e.g., maximum frequency, $p \geq \{AB\}$

Monotonicity and Anti-Monotonicity

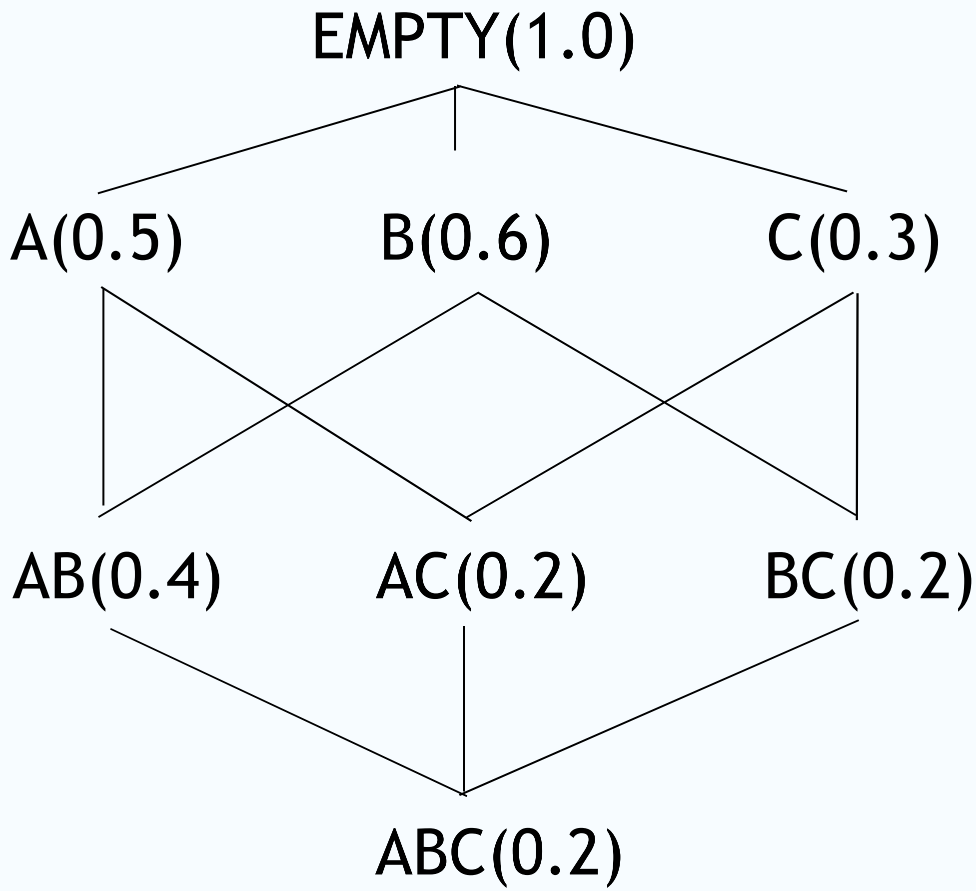


Basic Components of Pattern Mining Algorithms

Generality Order

- Many pattern languages are ordered according to the „is-more-general-than“ relation
- $p \leq q$ „p is more general than q“
- „Whenever q occurs,
  it is also the case that p occurs“
  (in all *conceivable* examples)
- *Lattice* of patterns
- Example: lattice of itemsets

Example



Subsumption Operator

- Generality ordering between patterns: „Does a pattern p subsume a pattern q?“
- Itemsets:
  $p \leq q$ iff $p \subseteq q$
- Trivial for itemsets, but computationally hard, e.g., for graphs

Example Run APriori

freq(p, D) $\geq$ 0.3

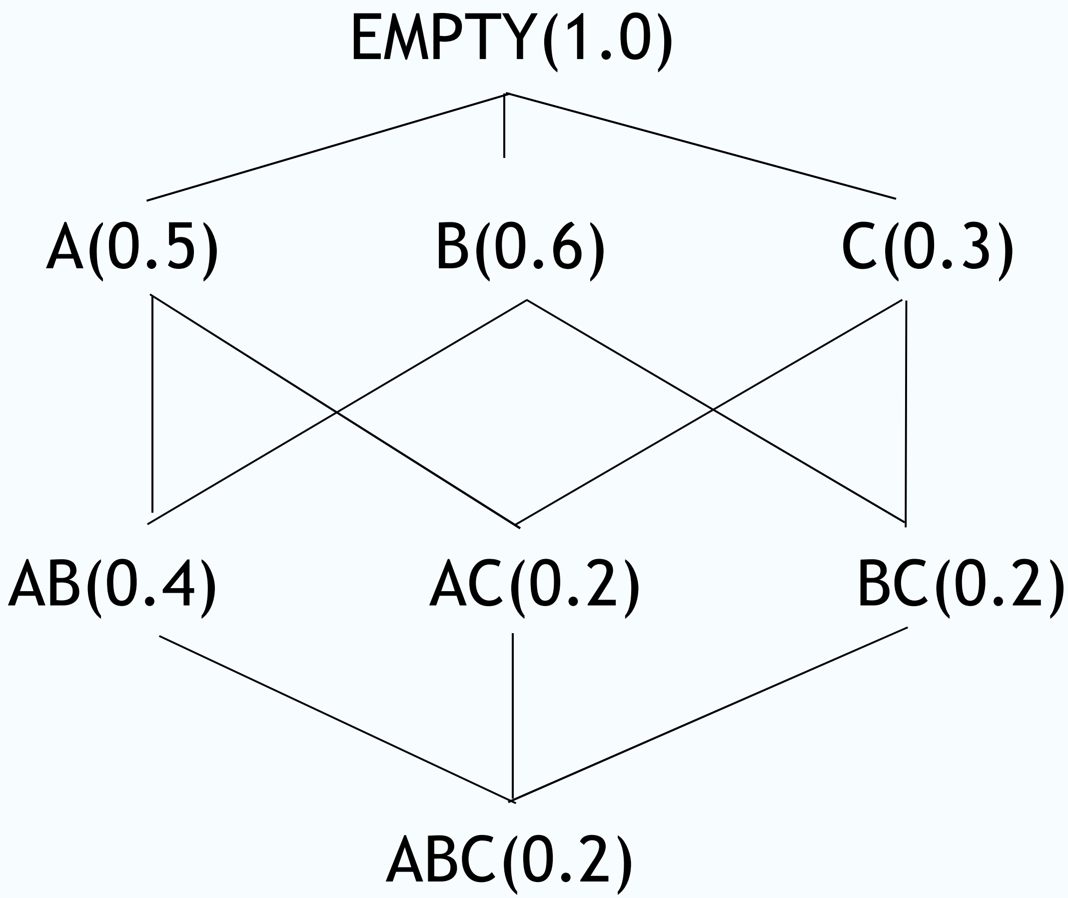
C1 = {A, B, C}

F1 = {A, B, C}

C2 = {AB, AC, BC}

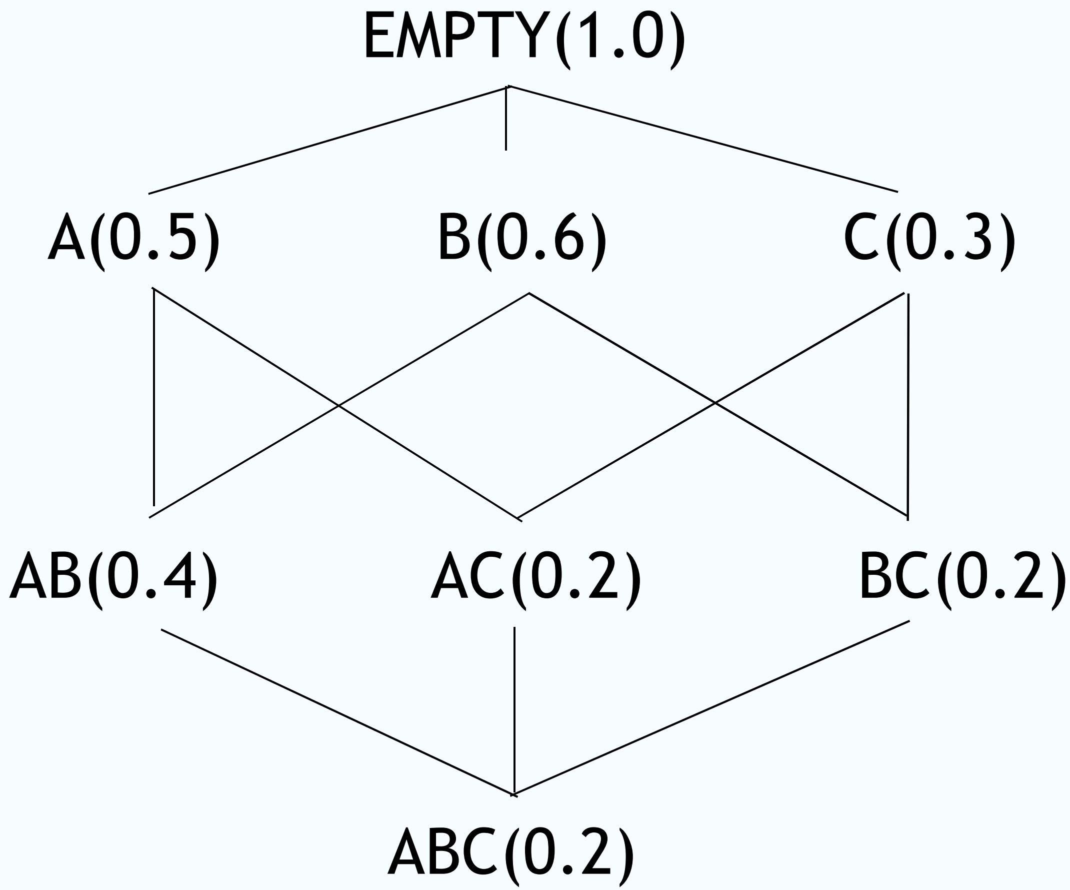
F2 = {AB}

C3 = {}



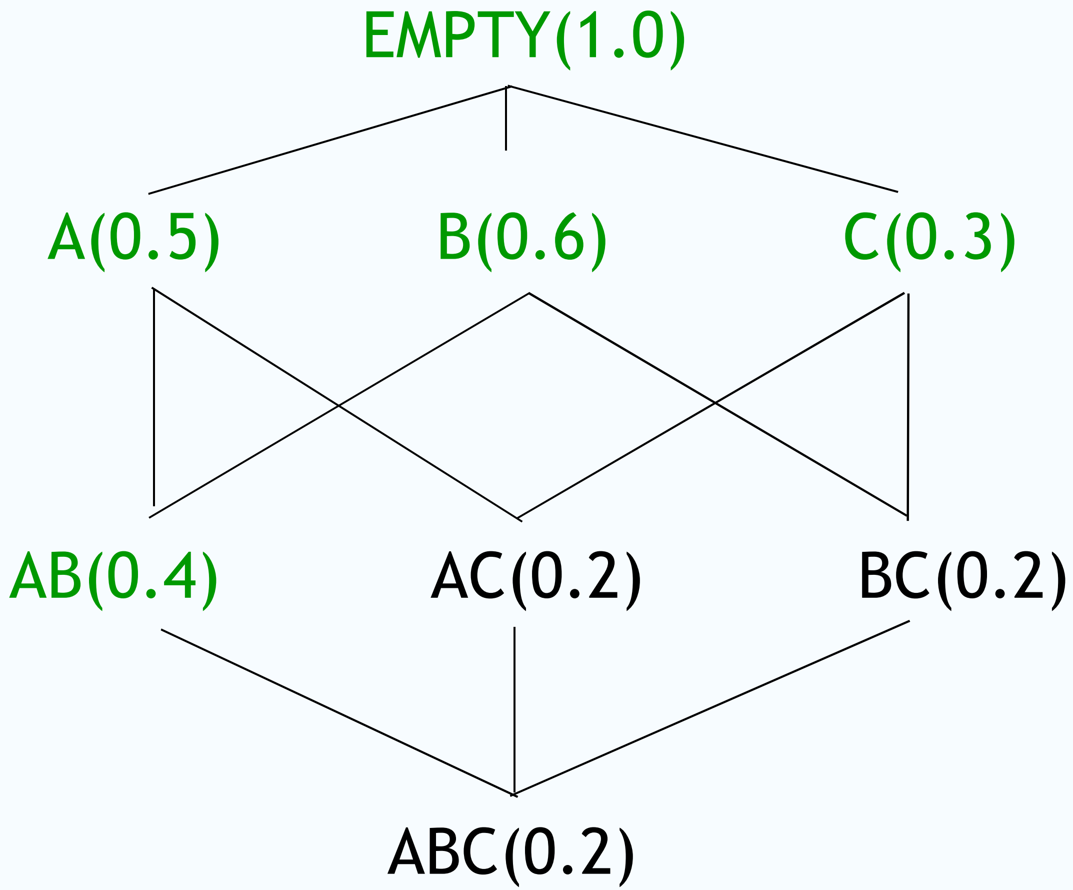
Example Borders

freq(p, D) ≥ 0.3



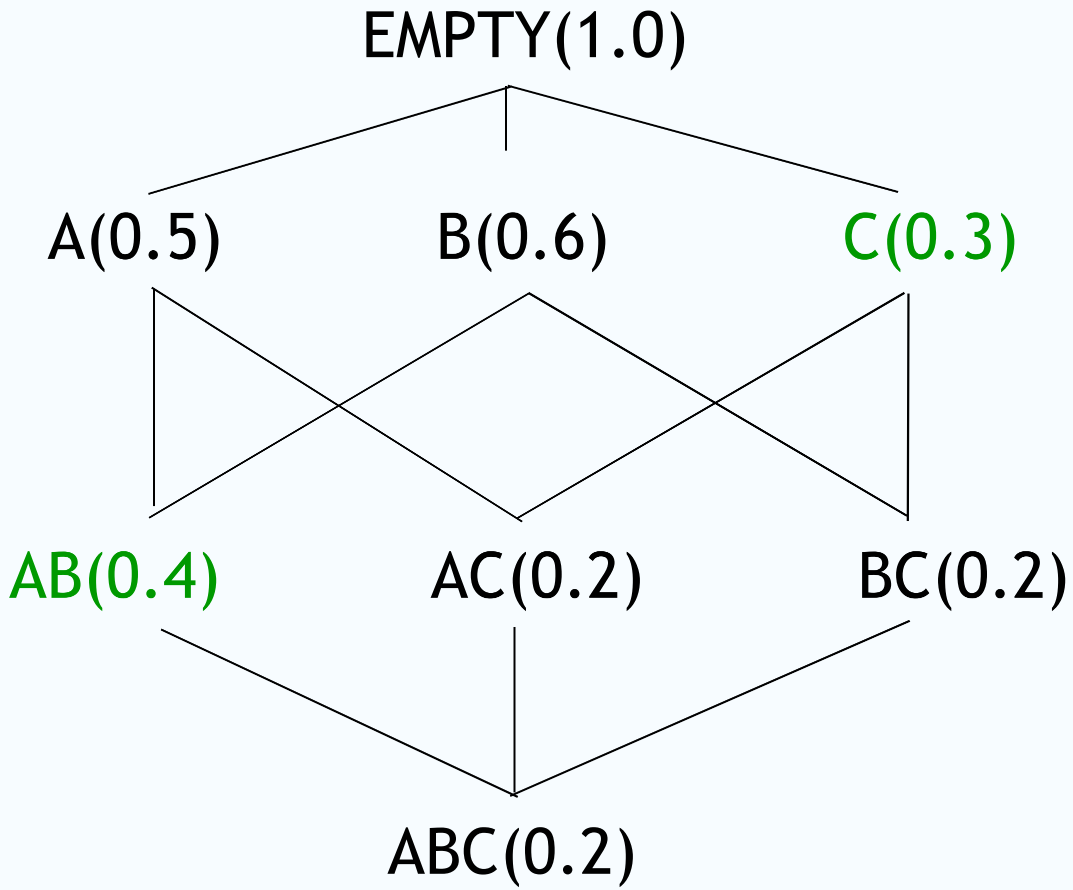
Example Borders

freq(p, D) $\geq$ 0.3



Example Borders

$freq(p, D) \geq 0.3$



$Bd^{+} = \{AB, C\}$

Borders
(Mannila & Toivonen, 1997)

- **Positive Border** for minimum frequency constraint:
  most specific solution patterns in L
- S: set of solution patterns

$$Bd^{+}(S) = \{\varphi \in S \mid \forall \gamma \in L : \varphi \prec \gamma \rightarrow \gamma \notin S\}$$

- **Negative Border:**
  most general non-solution patterns in L

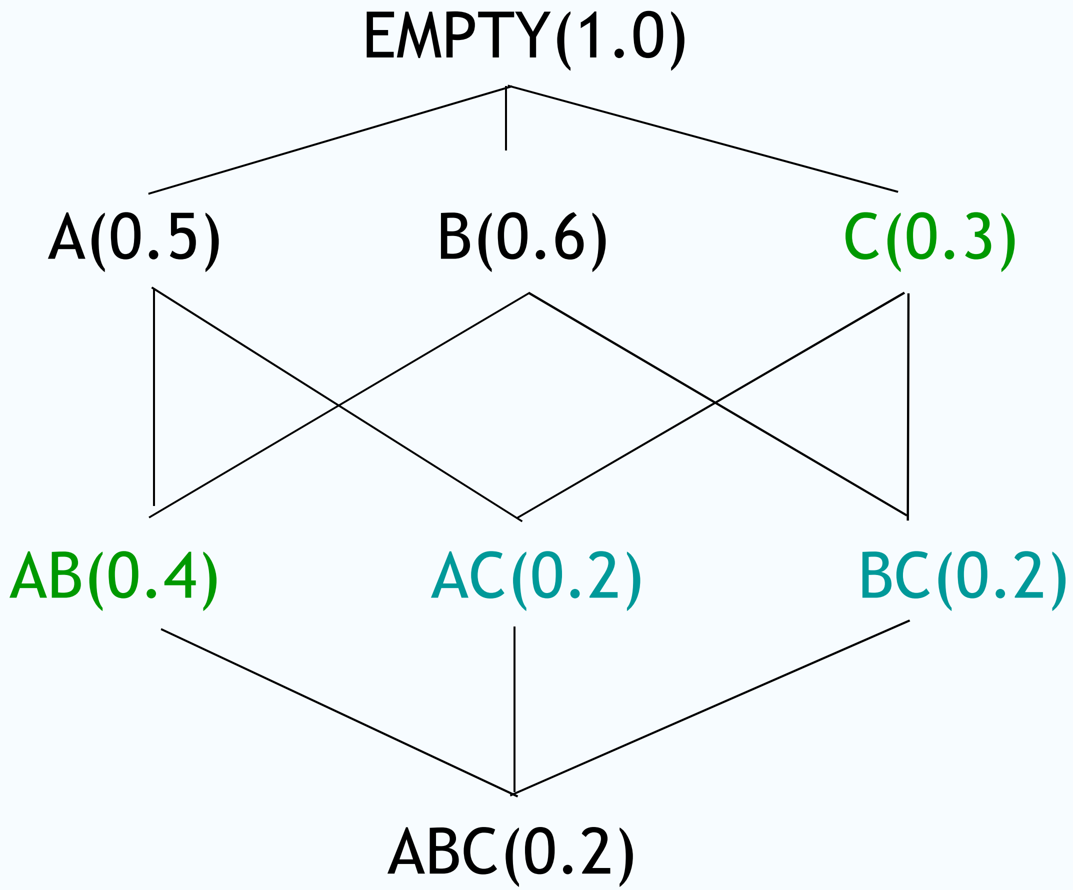
$$Bd^{-}(S) = \{\varphi \in L \setminus S \mid \forall \gamma \in L : \gamma \prec \varphi \rightarrow \gamma \in S\}$$

Example Borders

freq(p, D) ≥ 0.3

Bd⁺ = {AB, C}
Bd⁻ = {AC, BC}

$$Bd^{-}(S) = \{\varphi \in L \setminus S \mid \forall \gamma \in L : \gamma \prec \varphi \rightarrow \gamma \in S\}$$



From APriori Output to Borders

Positive border:

- either: collect *all* frequent patterns in F and then maximize:

  $Bd^{+} = \{\phi \in F \mid \neg\exists\gamma \in F\colon \phi < \gamma\}$

- or: collect *only those from the transition* from frequent to infrequent and then maximize

Negative border:

- just keep track of the *candidates* that turn out to be infrequent

From APriori Output to Borders

Example:

- F1= {A, B, **C**, **D**}
- C2 = {AB, AC, AD, BC, BD, **CD**}
- F2 = {**AB**, AC, **AD**, BC, **BD**}
- C3 = {ABC, **ABD**}
- F3 = {**ABC**}

Consequently:

- Bd^- = {CD, ABD}
- max({C, D, AB, AD, BD, ABC}) = Bd^+ = {AD, BD, ABC}

Coverage

- covers(p, e)
  pattern p covers example e =
  pattern p is contained in example e
- Itemsets:
  simply $p \subseteq e$
- *Non-trivial for more complex pattern languages!*

Canonization of Patterns

- Unique representation of patterns (and examples): *canonical form*
- If syntactic variants exist, then the search space may explode even more dramatically
- Itemsets {A, B} and {B, A} represented as lists?

Downward Refinement (Specialization) Operator

- Downward refinement operator (specialization operator)
  $Q := \rho_s(p)$
  set of all *minimal* specializations of p:
  all $q \in Q$ are subsumed by p, but there is no q´ more general than q also subsumed by p
- Itemsets:
  $\rho_s(p) := \{q \mid q = p \cup \{i \in I\} \wedge |q|=|p|+1\}$
  where I is the set of all itemsets
- Example: $\rho_s(A) = \{AB, AC\}$

Upward Refinement (Generalization) Operator

- Upward refinement operator (generalization operator)
  $Q := \rho_g(p)$
  set of all *minimal* generalizations of p:
  all $q \in Q$ subsume p, but there is no more specific q´ also subsuming p
- Itemsets:
  $\rho_g(p) := \{q \mid q = p \setminus \{i \in p\} \wedge |q|=|p|-1\}$
- Example: $\rho_g(ABCD) = \{ABC, ABD, ACD, BCD\}$

Desirable Properties of Downward Refinement Operators

1. **locally finite:** there exists n such that $|\rho_s(p)| \leq n$ for all $p \in L$
2. **complete for** L:
   all patterns are reachable within a finite number of steps $L = \rho^*(\text{most general pattern})$
3. **proper**: there is no $p' \in \rho_s(p)$ such that $p' \equiv \rho_s(p)$
4. **optimal:** there is only one path to each $p \in L$

APriori Revisited

$i := 1$

$C_i := r_s(\{\})$

while $C_i \neq \{\}$ **do**

% candidate testing (database scan)

for each set in C_i test whether it is frequent

let F_i be the collection of frequent sets from C_i

% candidate formation

$C_{i+1} := \{p | q \in F_i \wedge p \in \rho_s(q) \wedge \rho_g(p) \subseteq F_i\}$

$i := i + 1$

return $\cup F_j$

Pruning and Canonization

- Database scans are expensive
- Can be avoided by exploiting knowledge about previously encountered patterns (e.g., all patterns more general are known to be frequent) – *trade-off!*
- Canonization needed to prevent further combinatorial explosion (especially important for more complex pattern languages)