Frequent itemsets, association rules

Jiří Kléma

Department of Cybernetics,
Czech Technical University in Prague

http://ida.felk.cvut.cz

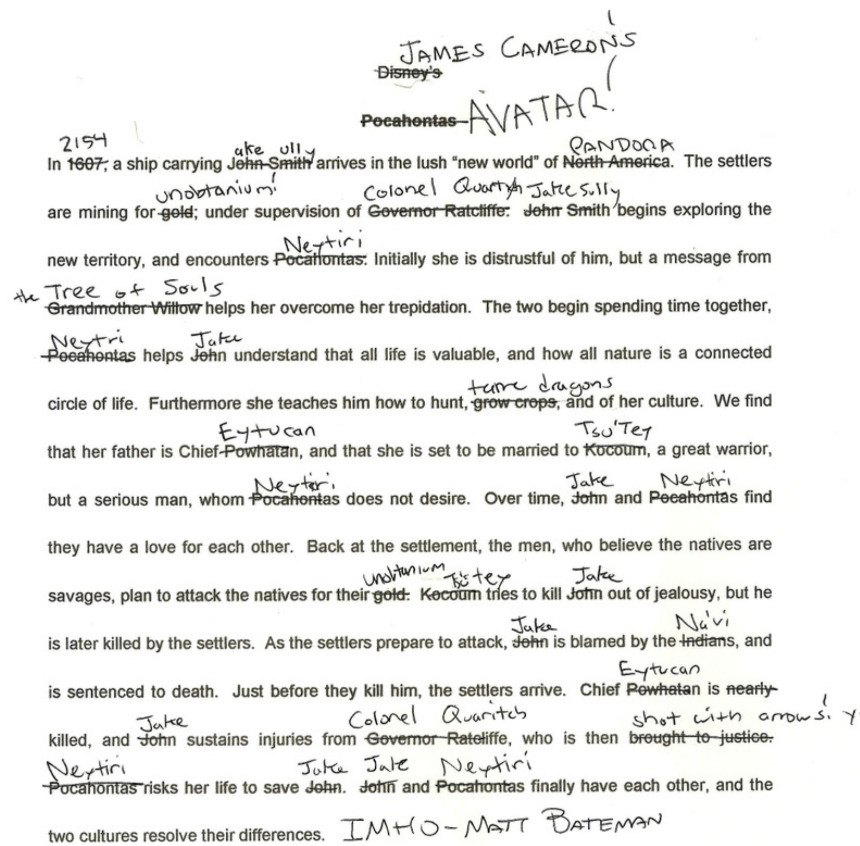
Outline

- Motivation to search for event co-occurrence
  - origin – market basket analysis,
  - another example, general aims,
  - local models,
- association rule formalization
  - definitions – support, confidence,
  - (sub)problem – frequent itemset mining,
- frequent itemset mining algorithms
  - APRIORI algorithm,
  - ECLAT and (FPGrowth) algorithm,
  - reduction of the set of frequent itemsets,
- generate rules from frequent itemsets
  - AR-Gen algorithm,
- equivalence quantifiers and rules
  - alternatives to implication rules based on confidence.

General motivation

- a data mining legend
  - beer and nappies.

- plagiarism detection
  - is AVATAR's story original?



JAMES CAMERON'S ~~Disney's~~

~~Pocahontas~~ AVATAR!

In ~~1607~~, 2154 a ship carrying ~~John Smith~~ Jake Sully arrives in the lush "new world" of ~~North America~~ PANDORA. The settlers are mining for ~~gold~~ unobtanium!; under supervision of ~~Governor Ratcliffe~~ Colonel Quaritch. ~~John Smith~~ Jake Sully begins exploring the new territory, and encounters ~~Pocahontas~~ Neytiri. Initially she is distrustful of him, but a message from ~~Grandmother Willow~~ the Tree of Souls helps her overcome her trepidation. The two begin spending time together, ~~Pocahontas~~ Neytiri helps ~~John~~ Jake understand that all life is valuable, and how all nature is a connected circle of life. Furthermore she teaches him how to hunt, ~~grow crops~~ tame dragons, and of her culture. We find that her father is Chief ~~Powhatan~~ Eytucan, and that she is set to be married to ~~Kocoum~~ Tsu'Tey, a great warrior, but a serious man, whom ~~Pocahontas~~ Neytiri does not desire. Over time, ~~John~~ Jake and ~~Pocahontas~~ Neytiri find they have a love for each other. Back at the settlement, the men, who believe the natives are savages, plan to attack the natives for their ~~gold~~ unobtanium. ~~Kocoum~~ Tsu'tey tries to kill ~~John~~ Jake out of jealousy, but he is later killed by the settlers. As the settlers prepare to attack, ~~John~~ Jake is blamed by the ~~Indians~~ Na'vi, and is sentenced to death. Just before they kill him, the settlers arrive. Chief ~~Powhatan~~ Eytucan is ~~nearly~~ killed, and ~~John~~ Jake sustains injuries from ~~Governor Ratcliffe~~ Colonel Quaritch, who is then ~~brought to justice~~ shot with arrows! yo. ~~Pocahontas~~ Neytiri risks her life to save ~~John~~ Jake. ~~John~~ Jake and ~~Pocahontas~~ Neytiri finally have each other, and the two cultures resolve their differences. IMHO - MATT BATEMAN

Association rules

- Association Rules (ARs)
- Definition
  - simple event co-occurrence assertions based on data,
  - probabilistic character – co-occurrence is not strict,
- Notation and meaning
  - if **Ant** then **Suc**,
  - another notation: Ant $\Rightarrow$ Suc,
  - antecedent (Ant) and succedent/consequent (Suc) define general events observable in data,
  - event – a binary phenomenon, it either occurs or not,
  - an extensive representation (data) transformed into a concised and understandable description (knowledge).
- Association rules, examples
  - book store recommendations (Amazon):
    {Castaneda: **Teachings of Don Juan**}
    $\Rightarrow$ {Hesse: **Der Steppenwolf** & Ruiz: **The Four Agreements**}
  - relation among risk factors and diseases in medicine (Stulong):
    {**beer** $\geq$ 1litre/day & **liquors**=0} $\Rightarrow$ {not(**heart disease**)}

Association rules – basic terms

- Items: $I = \{i_1, i_2, \ldots, i_m\}$
  - binary features or outcomes of relational operators,
- Transaction database: $D = \{t_1, t_2, \ldots, t_n\}$, $t_i \subseteq I$
  - transactions = examples, objects, instances,
- Itemsets: $\{i_{i_1}, i_{i_2}, \ldots, i_{i_k}\} \subseteq I$
  - a condition analogy, concurrent occurrence of multiple items,
- Itemset cover: $K_D(J) = \{k \in \{1, 2, \ldots, n\} | J \subseteq t_k\}$
  - a transaction t covers an itemset J iff $J \subseteq t$ (J is contained in t),
  - the cover of J wrt D is a set of transaction indices in D covering J,
- Itemset support: $s_D(J) = |K_D(J)|$, resp. $s_D(J) = \frac{1}{n}|K_D(J)|$
  - a (relative) number of transactions covering the given itemset,
- Frequent itemset:
  - itemset frequency exceeds a threshold,

Association rules – basic terms

- Association rule (AR):
  - the implication Ant $\Rightarrow$ Suc, where Ant, Suc $\subseteq$ I and Ant $\cap$ Suc $= \emptyset$,
- AR support: $s_{Ant \Rightarrow Suc}(D) = s_D(Ant \cup Suc)$
  - the ratio (or number of) transactions covering both Ant and Suc,
  - note: support of Ant $\Rightarrow$ Suc equals to the support of Ant $\cup$ Suc,
- AR confidence: $\alpha_{Ant \Rightarrow Suc}(D) = \frac{s_D(Ant \cup Suc)}{s_D(Ant)}$
  - the ratio between AR support (Ant $\cup$ Suc) and its antecedent support (Ant),
  - can be viewed as $Pr(Suc|Ant)$ estimate,
  - always lower or equal to 1.
- If both AR and D are obvious, we will shorten the notation and use s, α etc.

AR mining

- Given:
  - an itemset $I = \{i_1, i_2, \ldots, i_m\}$,
  - a transaction database $D = \{t_1, t_2, \ldots, t_n\}$
    * where $t_i = \{i_{i_1}, i_{i_2}, \ldots, i_{i_k}\}$, a $i_{i_j} \in I$,
  - minimum threshold support s_{min},
  - minimum threshold confidence α_{min}.
- Association rule mining task:
  - find all the rules Ant $\Rightarrow$ Suc with support $s \geq s_{min}$ and confidence $\alpha \geq \alpha_{min}$.
- Implementation can be split into 2 phases:
  - identify all the frequent (sub)itemsets,
  - take frequent itemset and generate rules out of them.

Example: market basket analysis

- Aim: increase sales and minimize costs, i.e. find items often both together in one transaction,

Transaction	Items
t_1	Bread, Jelly, Butter
t_2	Bread, Butter
t_3	Bread, Milk, Butter
t_4	Beer, Bread
t_5	Beer, Milk

- I = {Beer, Bread, Jelly, Milk, Butter},
- A rule: Bread $\Rightarrow$ Butter,
  - Ant={Bread}$\in \{t_1, t_2, t_3, t_4\}$, s_{ant}=4/5=80%,
  - Ant $\cup$ Suc={Bread,Butter} $\in \{t_1, t_2, t_3\}$, support AR is s=3/5=60%,
  - Confidence AR is $\alpha = s/s_{ant}$=75%.

- Other rules and their parameters:

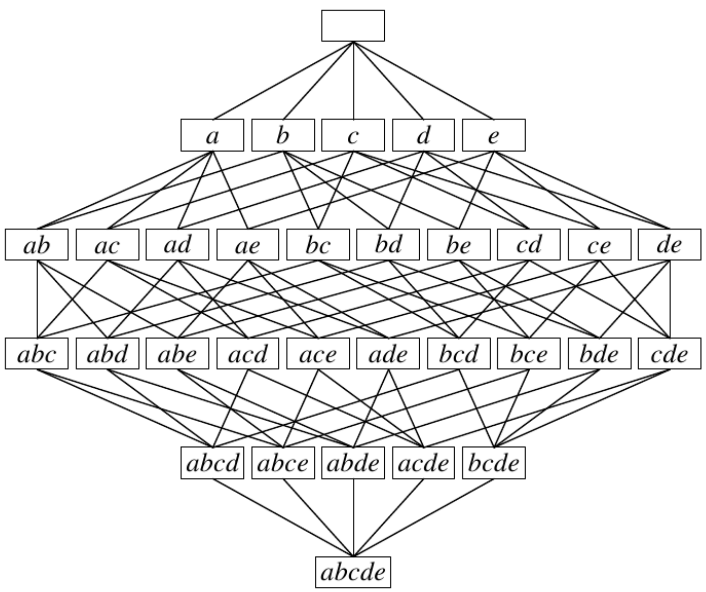
Ant $\Rightarrow$ Suc	s [%]	α [%]
Bread $\Rightarrow$ Butter	60	75
Butter $\Rightarrow$ Bread	60	100
Beer $\Rightarrow$ Bread	20	50
Butter $\Rightarrow$ Jelly	20	33
Jelly $\Rightarrow$ Butter	20	100
Jelly $\Rightarrow$ Milk	0	0

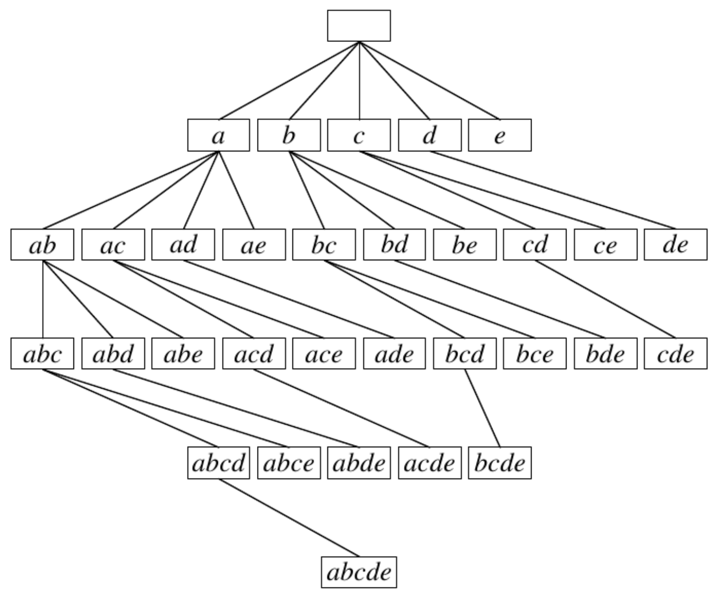
Frequent itemset mining

- phase 1 in association rule mining,
- often makes a stand-alone task
  - find product families bough together,
  - in general, **find events that frequently co-occur**,
    - text analysis: transactions = documents, items = words,
    - plagiarism detection: transactions = sentence occurrences, items = documents,
- exhaustive search of the itemset space
  - having m binary items, there are $2^m - 1$ itemsets,
  - having N nominal features, each with K categories, there are $(1+K)^N - 1$ itemsets,
  - complexity increases **exponentially** with the number of items (features),
- for large tasks assumptions
  - sparse data – everything does not relate to everything else,
  - early and efficient search space pruning.

Frequent itemset mining – method categorization (1)

- a set of all the itemsets is **partially ordered** (makes a poset)
  - can be depicted as an acyclic graph – Hasse diagram,
  - nodes = itemsets, an edge $I \rightarrow J$ iff $I < J$ and there is no $K: \; I < K < J$,
  - when depicting all the subsets it also makes a **lattice**,
  - efficient to reduce on a **tree** (each node needs to be visited and tested only once),
- methods for the itemset lattice/tree search
  - breath-first – level-wise, each level concerns itemsets of a certain length,
  - depth-first – traverse the itemsets with an identical prefix.





Frequent itemset mining – method categorization (2)

- transaction set/database representation
  - horizontal – transactions as the main units, transaction ≈ a list/array of items
    - a natural way,
  - vertical – items as the main units, a transaction list is stored for each item
    - advantage: efficient (recursive) access to the transaction list of an itemset,
    - the transaction list for a pair of items is the intersection of the transaction lists of the individual items.

Transactions	Items
t_1	a, d, e
t_2	b, c, d
t_3	a, c, e
t_4	a, c, d, e
t_5	a, e
t_6	a, c, d
t_7	b, c
t_8	a, c, d, e
t_9	b, c, e
t_{10}	a, d, e

a	b	c	d	e
1	2	2	1	1
3	7	3	2	3
4	9	4	4	4
5		6	6	5
6		7	8	8
8		8	10	9
10		9		10

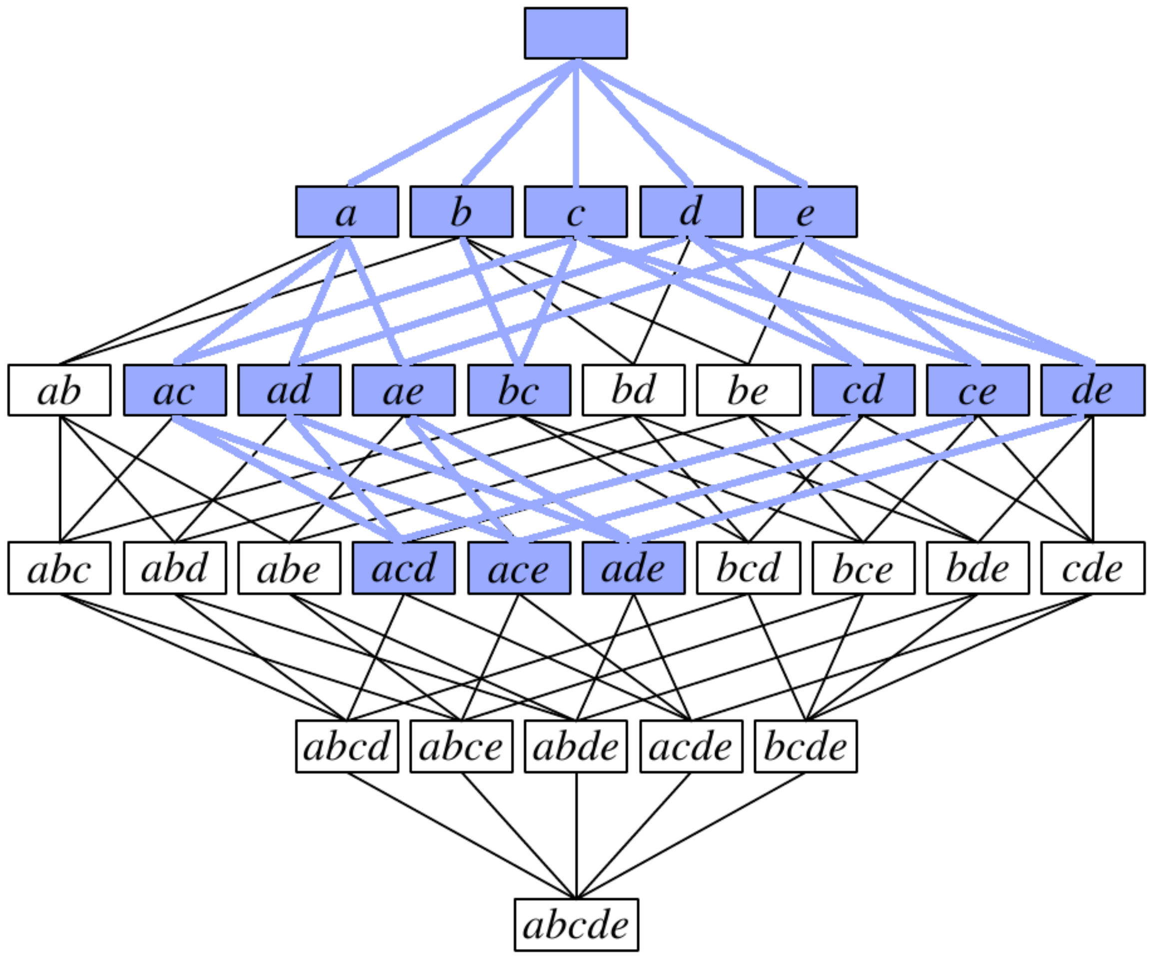
APRIORI algorithm – the basic idea

- pioneering, the most well-known, but not the most efficient,
- based on the elemental characteristic of any frequent itemset:
  Each subset of a frequent itemset is frequent.
- as we proceed bottom-up from subsets to supersets
  the logical **contraposition** principle
  $(p \Rightarrow q) \Leftrightarrow (\neg q \Rightarrow \neg p)$
- the anti-monotone property transformed to a monotone property, consequence:
  No superset of an infrequent itemset can be frequent.
- candidate itemsets
  - potentially frequent – all the subsets are known to be frequent.
- APRIORI categories: breath-first search, horizontal transaction representation.

Frequent itemset mining – example

Transakce	Položky
t_1	a, d, e
t_2	b, c, d
t_3	a, c, e
t_4	a, c, d, e
t_5	a, e
t_6	a, c, d
t_7	b, c
t_8	a, c, d, e
t_9	b, c, e
t_{10}	a, d, e

the lattice with frequent itemsets for $s_{min} = 3$



APRIORI algorithm [Agrawal et al., 1996]

```
Apriori:
  C_1 = ∀ candidate itemsets of size 1 in I;
  L_1 = ∀ frequent itemsets of size 1 (support ≥ s_min);
  i = 1;
  repeat
    i = i + 1;
    C_i = Apriori-Gen(L_{i-1});
    Get support C_i and create L_i;
  until no frequent itemset found (L_i = ∅);
  L = ⋃ L_i, ∀ i

Apriori-Gen(L_{i-1}):
  C_i = ∅
  for ∀ itemset pairs Comb_p, Comb_q ∈ L_{i-1}:
    if they match in i-2 items then add Comb_p ∪ Comb_q do C_i
  for ∀ itemsets Comb from C_i:
    if any Comb subset of size i-1 ∉ L_{i-1} then remove Comb.
```

APRIORI – pros and cons

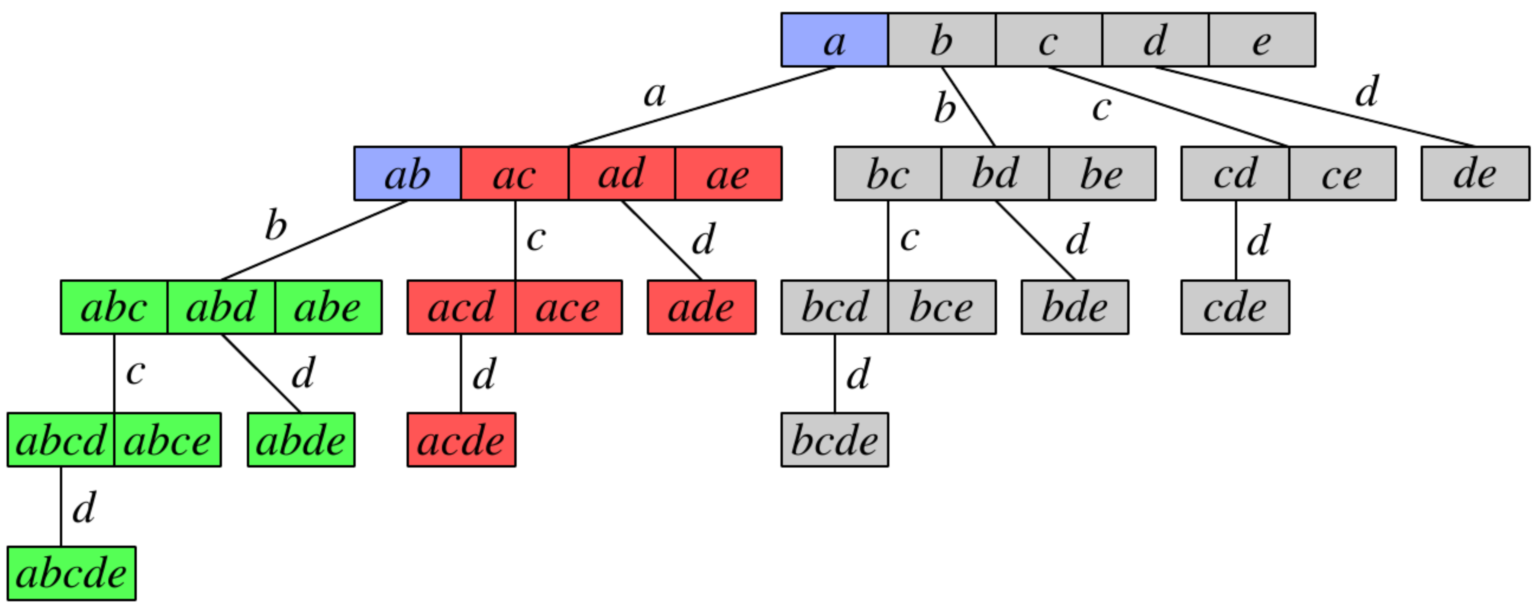
- Advantages
  - efficient due to monotone property of large itemsets,
  - still worst-case exponential time complexity, feasible provided:
    * a proper (high enough) s_{min} and α_{min},
    * sparse data (in practice rather holds).
  - straightforward implementation including parallelization,
  - for highly correlated data with a prohibitive number of frequent itemsets
    * needs improvements, e.g. with a condensed representation.
- Disadvantages
  - all frequent itemsets are represented, it can take a lot of memory .
  - support counting can take long time for large transactions,
  - assumes permanent access to the transaction database (in memory),
  - needs up to m (the number of items) database scans
    * the speed improved with hash trees,
    * the number of scans can be reduced by merging two consecutive steps into one,
    * compensated by larger sets of candidate itemsets, but . . .

ECLAT algorithm (Zaki et al., 1997) – the basic idea

- sorts items lexicographically $\rightarrow$ canonical itemset representation
  - $\{a, b, c\} \approx abc < bac < bca < \dots < cba$,
  - an itemset encoded by its lexicographically smallest (largest) code word,
- the tree is searched in a **depth-first** way
  - owing to the canonical representation it is a prefix tree,
- uses purely **vertical** transaction set representation,
- can generate more candidate itemsets than APRIORI
  - decides when support of any subset is not necessarily available.

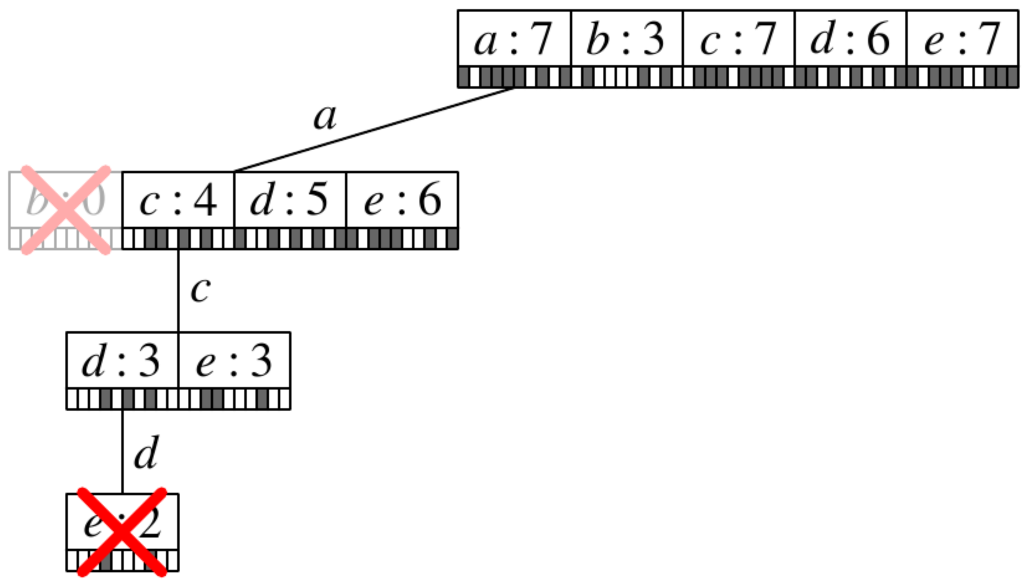
Conditional transaction database – depth-first search

- depth-first search the prefix tree, **divide and conquer** strategy
  - find all the frequent itemsets with the given prefix first,
  - do the same for the rest of itemsets,
  - leads to transaction set splits (transactions with/without the given prefix),
- node colors
  - the prefix in blue, itemsets having the prefix in green, itemsets without the prefix in red,
  - a recursive procedure, the previous a-step needs also to be concerned.



ECLAT example, $s_{min} = 3$ (Borgelt: Frequent Pattern Mining)

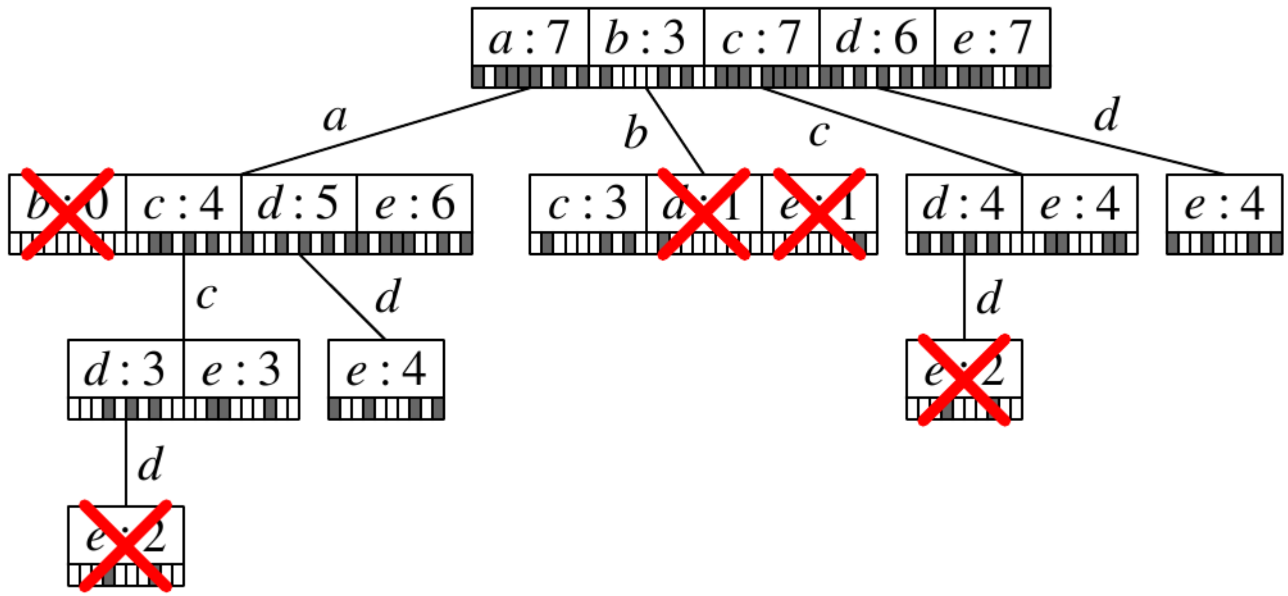
t_1	a, d, e
t_2	b, c, d
t_3	a, c, e
t_4	a, c, d, e
t_5	a, e
t_6	a, c, d
t_7	b, c
t_8	a, c, d, e
t_9	b, c, e
t_{10}	a, d, e



- preprocessing: vertical representation by the bit vector (grey/white – item in/out of transaction)
  - the only transaction database scan, intersections follow exclusively,
- step 1: the conditional transaction database for a item,
- step 2: $\{a, b\}$ infrequent – prune,
- step 3: the conditional transaction database for $\{a, c\}$ itemset,
- step 4: the conditional transaction database for $\{a, c, d\}$ itemset and prune $\{a, c, d, e\}$.

ECLAT example, $s_{min} = 3$ (Borgelt: Frequent Pattern Mining)

t_1	a, d, e
t_2	b, c, d
t_3	a, c, e
t_4	a, c, d, e
t_5	a, e
t_6	a, c, d
t_7	b, c
t_8	a, c, d, e
t_9	b, c, e
t_{10}	a, d, e



- the whole tree shown, the outcome (certainly) identical with APRIORI,
- APRIORI might prune $\{a, c, d, e\}$ without counting its support,
  - knowing that $s_{\{c,d,e\}} = 2 \leq s_{min} = 3$,
- in contrary, APRIORI needs more transaction database scans.

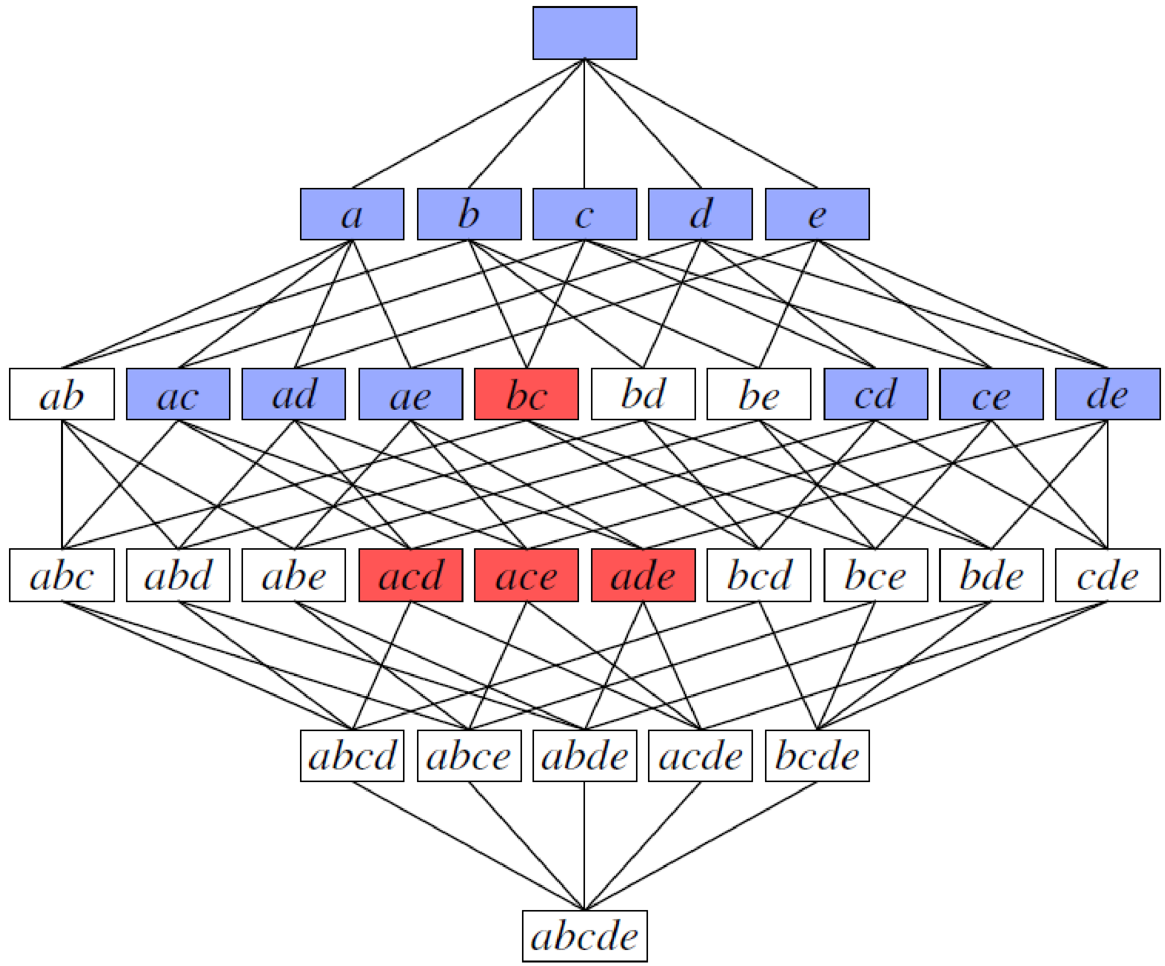
Reducing the output – the pruned sets of frequent itemsets

- the number of frequent itemsets can be prohibitive
  - the output is not comprehensible, a user can be interested in long patterns only,
  - leads to a notion of **maximal** itemset
    * frequent and none of its proper supersets is frequent,
    * the set of maximal itemsets:
      $M_D(s_{\min}) = \{J \subseteq I \mid s_D(J) \geq s_{\min} \wedge \forall K \supset J : s_D(K) < s_{\min}\}$,
- the set of frequent itemsets is redundant
  - all the information about it can be preserved in a smaller set (subset),
  - leads to a notion of **closed** itemset
    * frequent and none of its proper supersets has the same support,
    * the set of closed itemsets:
      $C_D(s_{\min}) = \{J \subseteq I \mid s_D(J) \geq s_{\min} \wedge \forall K \supset J : s_D(K) < s_D(J)\}$,
- obvious relations
  - all maximal itemsets and all closed itemsets are frequent,
  - any maximal itemset is necessarily closed.

Reducing the output – illustration

- the frequent itemsets for $s_{min} = 3$ (blue), the maximal itemsets (red),
- how many frequent itemsets are not closed?, which itemsets?

Transactions	Items
t_1	a, d, e
t_2	b, c, d
t_3	a, c, e
t_4	a, c, d, e
t_5	a, e
t_6	a, c, d
t_7	b, c
t_8	a, c, d, e
t_9	b, c, e
t_{10}	a, d, e



Searching for closed and maximal itemsets

- principal ways to find close and maximal itemsets
  - filter the set of frequent itemsets
    - reasonable when the set of frequent itemsets is needed anyway,
  - direct search with earlier and more efficient pruning
    - a compact representation accelerates search,
    - specialized algorithms derived from classical ones – MaxMiner, Closet, Charm, GenMax,
    - among other properties, for any closed itemset it holds
      - the closed itemset matches the intersection of all the transactions that contain it,
      - it also explains why $\{d, e\}$ is not closed:

Transactions	Items
t_1	a, d, e
t_4	a, c, d, e
t_8	a, c, d, e
t_{10}	a, d, e
$\cap$	a, d, e

Generate rules from frequent itemsets – step 2

```
Inputs:
  I, D, L, α_min;

Output:
  R; % pravidla splňující s_min a α_min

AR-Gen:
  R = ∅;
  for ∀ l ∈ L do:
    for ∀ x ⊂ l such that x ≠ ∅ and x ≠ l do:
      if s(l)/s(x) ≥ α_min, then R = R ∪ {x ⇒ (l-x)}
      (apply the property: s(l)/s(x) < α_min ⇒ ∀ x'⊂ x s(l)/s(x') < α_min)
```

- Example: market basket analysis
  - Inputs: L={Bread, Butter} (generated for s_{min}=30%), α_{min}=50%
  - Output: R={Bread ⇒ Butter: s=60%, α=75%, Butter ⇒ Bread: s=60%, α=100%}

Example: the study plan

- Aim: find out whether the real study plans correspond with recommendations/study programs
- Courses: RZN (Knowledge representation), PAH (Planning and games), VI (Computational intelligence), MAS (Multi-agent systems), SAD (Machine learning and data analysis), AU (Automatic reasoning)

Transactions	Items
t_1	RZN
t_2	VI, SAD, AU
t_3	PAH, AU
t_4	PAH, VI, AU
t_5	PAH, MAS
t_6	VI, AU
t_7	PAH, SAD
t_8	PAH, VI, MAS, AU
t_9	PAH
t_{10}	PAH, VI, AU

Transactions	Items
t_{11}	AU
t_{12}	RZN, PAH, VI, SAD, AU
t_{13}	PAH, VI, MAS, AU
t_{14}	VI, SAD, AU
t_{15}	PAH, AU
t_{16}	SAD, AU
t_{17}	RZN, PAH, SAD
t_{18}	PAH, VI, MAS, AU
t_{19}	PAH
t_{20}	PAH, VI, MAS, AU

APRIORI step – s_{min}=20%, resp. 4

i	C_i	L_i
1	{RZN}, {PAH}, {VI} {MAS}, {SAD}, {AU}	{PAH}, {VI}, {MAS} {SAD}, {AU}
2	{PAH, VI}, {PAH, MAS}, {PAH, SAD} {PAH, AU}, {VI, MAS}, {VI, SAD} {VI, AU}, {MAS, SAD}, {MAS, AU} {SAD, AU}	{PAH, VI}, {PAH, MAS} {PAH, AU}, {VI, MAS} {VI, AU}, {MAS, AU} {SAD, AU}
3	{PAH, VI, MAS}, {PAH, VI, AU} {PAH, MAS, AU}, {PAH, SAD, AU} {VI, MAS, AU}, {VI, SAD, AU} {MAS, SAD, AU}	{PAH, VI, MAS} {PAH, VI, AU} {PAH, MAS, AU} {VI, MAS, AU}
4	{PAH, VI, MAS, AU}	{PAH, VI, MAS, AU}
5	$\emptyset$	$\emptyset$

AR-Gen step – α_{min}=80%, selected frequent itemsets

$\mathbf{L_2}$

PAH, VI: PAH $\Rightarrow$ VI α=50%, VI $\Rightarrow$ PAH α=70%

(PAH & VI concurrently 7times, PAH 14times, VI 10times)

PAH, MAS: PAH $\Rightarrow$ MAS 36%, **MAS $\Rightarrow$ PAH** 100%

(PAH & MAS concurrently 5times, PAH 14times, MAS 5times)

$\mathbf{L_3}$

PAH, VI, MAS: PAH & VI $\Rightarrow$ MAS 57%, **PAH & MAS $\Rightarrow$ VI** 80%, **VI & MAS $\Rightarrow$ PAH** 100%

(PAH nor VI cannot make an antecedent, test MAS only)

MAS $\Rightarrow$ PAH & VI 80%

$\mathbf{L_4}$

PAH, VI, MAS, AU: **PAH & VI & MAS $\Rightarrow$ AU** 100%, PAH & VI & AU $\Rightarrow$ MAS 57%,

PAH & MAS & AU $\Rightarrow$ VI 100%, **VI & MAS & AU $\Rightarrow$ PAH** 100%

(the antecedents PAH & VI, PAH & AU, VI & AU without testing)

PAH & MAS $\Rightarrow$ VI & AU 80%, **VI & MAS $\Rightarrow$ PAH & AU** 100%,

MAS & AU $\Rightarrow$ PAH & VI 100%

(the antecedents PAH, VI a AU without testing)

MAS $\Rightarrow$ PAH & VI & AU 80%

Four-fold table, quantifiers for the relation between Ant and Suc

- 4-fold table (4FT),
  - a, b, c, d $\longrightarrow$ the numbers of transactions meeting conditions.

4FT	Suc	$\neg$Suc	$\sum$
Ant	a	b	r=a+b
$\neg$Ant	c	d	s=c+d
$\sum$	k=a+c	l=b+d	n=a+b+c+d

- Confidence is not the only/always best quantifier
  - its **implicative** nature is misleading for frequent succedents,
  - independent itemsets can show a high confidence,
  - 4-fold table example (s=45%,α=90%, Ant and Suc independent):

450	50	500
450	50	500
900	100	1000

Alternative quantifiers

- Confidence can be replaced by an arbitrary 4ft function in the AR-Gen step:
  - **lift** (above-average) how many times more often Ant and Suc occur together than expected under independence assumption
    - lift=an/rk
  - **leverage** the difference in the real Ant and Suc co-occurrence and the co-occurrence expected under independence assumption
    - leverage=1/n(a-rk/n)
  - **conviction** measures the effect of the right-hand-side of the rule not being true
    - conviction=rl/bn

450	50	500
450	50	500
900	100	1000

s=0.45, α=0.9, **lift=1, leverage=0, conviction=1**

10	1	11
90	899	989
100	900	1000

s=0.01, α=0.91, lift=9.09, **leverage=0.01**, conviction=9.9

450	50	500
50	450	500
500	500	1000

s=0.45, α=0.9, lift=1.8, leverage=0.2, conviction=5

Association rules – summary

- One of the basic descriptive data mining procedures
  - identify frequent co-occurrences of events in data,
  - detecting hidden dependencies, subgroup discovery, knowledge discovery.
- Practical applications
  - not only market basket analysis!!!
  - generally applicable to any attribute-valued data
    - medicine, industrial measurements, temporal and spatial data, . . . ,
  - the necessary preprocessing step – binarization
    - dichotomization ((gradual) division into two sharply different categories),
    - for continuous features discretization,
    - coding could also be concerned (minizes the number of items, human understandability usually decreases),
    - example: temperature in Celsius degrees
      - discretization: $\{(-\infty,0\rangle \equiv$ low, $(0,15\rangle \equiv$ medium, $(15, \infty) \equiv$ high$\}$,
      - dichotomization: $\{i_1 \equiv$ t=low, $i_2 \equiv$ t=medium, $i_3 \equiv$ t=high$\}$,
- Demo
  - census data, relations between social factors and salary.

Recommended reading, lecture resources

:: Reading

- Agrawal, Srikant: **Fast Algorithms for Mining Association Rules.**
  - the article that introduced the task and proposed APRIORI algorithm,
  - http://rakesh.agrawal-family.com/papers/vldb94apriori.pdf,
- Borgelt: **Frequent Pattern Mining.**
  - slides, a detailed course, including a formal notation,
  - http://www.borgelt.net/teach/fpm/slides.html,
- Hájek, Havránek: **Mechanizing Hypothesis Formation.**
  - a pioneering theory from 1966, decades before Agrawal,
  - http://www.cs.cas.cz/hajek/guhabook/.