Algorithms that Search for Association Rules

Paulo Trigo Silva

What are Association Rules?

- Association Rules have some similarity with Classification Rules
  - … similar idea: `left-hand-side` ⇒ `right-hand-side`
  - the difference between Association and Classification rules
  - … concerns the information in the: `right-hand-side`
- … `right-hand-side` of an **Association Rule**
  - predicts the **value of one or more attributes** (any attribute)
  - … algorithm searches for those attributes; *without user intervention*
- … `right-hand-side` of a **Classification Rule**
  - predicts the **value of a single attribute** (*defined by the user*)
- A classification algorithm can search for association rules
  - given that it executes once for every combination of attributes, with every possible combination of values, on the right-hand side!
  - … an enormous number of association rules to be pruned…

When to Search for Association Rules?

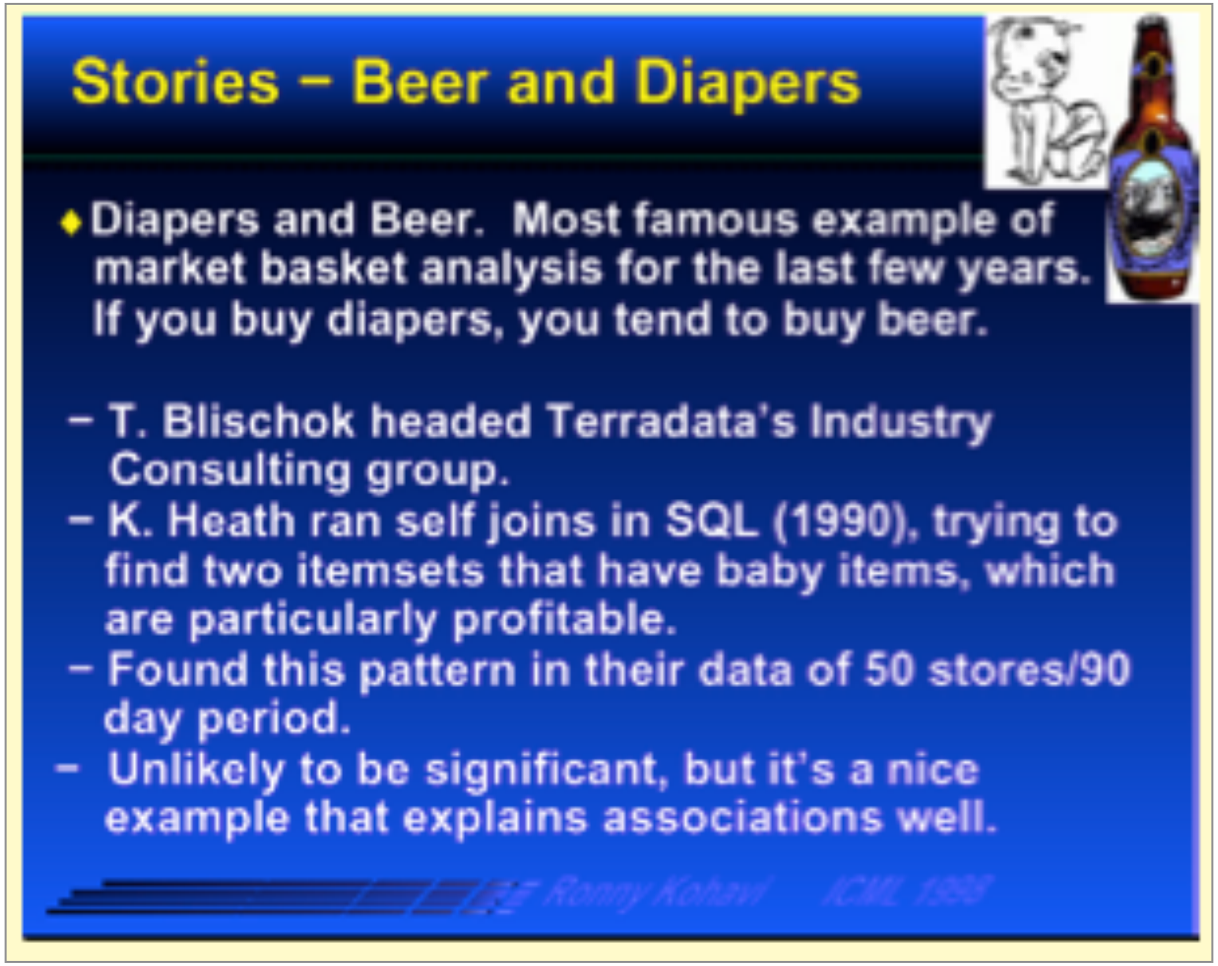
- Whenever our problem can be formulated within the scope of
  - **market-basket** analysis
- The **market-basket analysis** goal is to
  - "**find groups of items that tend to occur together in transactions**"
  - e.g., supermarket checkout records are analyzed to detect associations among items that people purchase
- … in e-commerce it may be used for Web page personalization
  - find that 40% of users visit pages A, B and C, and that 75% (of those 40%) have similar behavior pattern of always visiting C after A and B
  - … based on that rule, a dynamic link could be created for users who are likely to be interested in page C
- … the previous association rule could be expressed as,
  - `(A and B)` $\Rightarrow$ `C | support=40% | confidence=75% |`

Paulo Trigo Silva

… search for Association Rules is Unsupervised

- The goal is not predict a target value
  - because there is no predefined target (or class attribute)
  - … no class values are provided
- The goal is to find the intrinsic relations among data
  - algorithms explore the interconnectedness of the data
  - … as if all attributes were considered as “possible classes”
- Each rule associates two sets (LHS and RHS) of attribute values
  - i.e., `left-hand-side` ⇒ `right-hand-side`
  - both the LHS and the RHS are extracted without user intervention
- The association rules also provide classification support?
  - in fact the set of rules having the same attribute (e.g., attrA) in the RHS can be regarded as classification rules (and attrA as a class)
  - … but classification methods should be used for classification problems!

“a classical story”



Whether a real story or a “fabrication” it is as effective illustrative example

Paulo Trigo Silva

Problem Formulation

The **market-basket analysis** goal is to,

"find groups of **items** that tend to occur together in **transactions**"

- $I = \{ i_1, i_2, \ldots, i_M \}$, where
  - the I is a set of M **items**
  - each i_j is an item
- $T = \{ t_1, t_2, \ldots, t_N \}$
  - the T is a set of N **transactions**
  - each t_j is a transaction
  - and $t_j \subseteq I$ (i.e., t_j is a set of items)

Paulo Trigo Silva

An example – transactions in supermarket

the recorded supermarket transactions

TID	ITEM
t1	beer
t1	milk
t2	bread
t2	butter
t3	bread
t3	butter
t3	jelly
t4	bread
t4	butter
t4	bread
t5	beer
t5	bread

- *I* = { beer, bread, butter, jelly, milk}
  - the set of **items** sold in the store
- *T* = { t1, t2, t3, t4, t5 }
  - the set of **baskets recorded** at checkout
- each **basket and the items purchased**
  - t1 = { beer, milk }
  - t2 = { bread, butter }
  - t3 = { bread, butter, jelly }
  - t4 = { bread**=2**, butter } // **describe quantity**
  - t5 = { beer, bread }

Paulo Trigo Silva

… transactions in supermarket – different representation

the recorded supermarket transactions

TID	ITEM
t1	beer
t1	milk
t2	bread
t2	butter
t3	bread
t3	butter
t3	jelly
t4	bread
t4	butter
t4	bread
t5	beer
t5	bread

a tabular **sparse** representation of the same information

	beer	bread	butter	jelly	milk
t1	1				1
t2		1	1		
t3		1	1	1	
t4		2	1		
t5	1	1			

`.tab` (Orange file)

a **compact** representation of the same information

t1	bear milk
t2	bread butter
t3	bread butter jelly
t4	bread=2 butter
t5	beer bread

`.basket` (Orange file)

Paulo Trigo Silva

… how to generate the different representations?

the recorded
supermarket transactions

TID	ITEM
t1	beer
t1	milk
t2	bread
t2	butter
t3	bread
t3	butter
t3	jelly
t4	bread
t4	butter
t4	bread
t5	beer
t5	bread

apply the
pivot table
operator

a tabular **sparse** representation
of the same information

Count (	Colu Labels				
Row	beer	bread	butter	jelly	milk
t1	1				1
t2		1	1		
t3		1	1	1	
t4		2	1		
t5	1	1			

implement (e.g., in Pytohn) a script
to **transform table** into this format

a **compact** representation
of the same information

beer milk
bread butter
bread butter jelly
bread=2 butter
beer bread

Paulo Trigo Silva

… another example – documents in a collection

the collection
of documents

doc1: student, teach, school

doc2: student, school

doc3: teach, soccer, city, game

doc4: tennis, soccer

doc5: game, school, tennis

doc6: student, tennis

- I = { student, teach, school, city, game, tennis, soccer}
  - the set of **keywords** in the documents' collection
- T = { t1, t2, t3, t4, t5, t6 }
  - the set of **documents recorded** in the collection
- each document is treated as a bag (**basket**) of words
  - t1 = { student, teach, school }
  - t2 = { student, school }
  - t3 = { tennis, soccer, city, game }
  - t4 = { tennis, soccer }
  - t5 = { game, school, tennis }
  - t6 = { student, tennis }

Paulo Trigo Silva

… recall – What is Market Basket Analysis?

- Understanding behavioral patterns (e.g., of customers)
  - “what items were bought together?”
  - … “what is in each shopping car/basket?”
- The basket data is a collection of items bought in a transaction
  - an “itemset”
- How does this data differ from a transaction database?
  - apply a “pivot table” operator to the transaction (operational) table
- The overall goal is to generate qualified decisions and strategies
  - what to put on sale?
  - how to place merchandise on shelves for maximizing profit?
  - how to segment customers based on their buying patterns?

Paulo Trigo Silva

Association Rule and Itemset – Definitions

A transaction $\boldsymbol{t}$ ***contains*** $\boldsymbol{X}$, a set of items (itemset) in $\boldsymbol{I}$

if $\boldsymbol{X} \subseteq \boldsymbol{t}$

An ***association rule*** is an implication of the form

$$\boldsymbol{LHS} \Rightarrow \boldsymbol{RHS}$$

where $\boldsymbol{LHS}, \boldsymbol{RHS} \subset \boldsymbol{I}$ and $\boldsymbol{LHS} \cap \boldsymbol{RHS} = \varnothing$

An ***itemset*** is a set of items

e.g., X = { milk, bread, cereal } is an itemset

A ***k-itemset*** is an itemset with ***k*** items

e.g., X = { milk, bread, cereal } is a 3-itemset

Paulo Trigo Silva

Two (Main) Measures for "Rule Evaluation"

- **Support** of rule ***LHS* ⇒ *RHS***
  - measures how often the collection of items in an association, i.e., ***LHS*** and ***RHS***, occur together as a percentage of all the transactions

support(***LHS* ⇒ *RHS*** **)** =

= P(*LHS*, *RHS*) =

= **#tuples-with-both-*LHS*-and-*RHS* / #total-of-tuples**

- **Confidence** of rule ***LHS* ⇒ *RHS***
  - measures how likely it is that ***RHS*** occurs when ***LHS*** has occurred

confidence(***LHS* ⇒ *RHS*** **)** =

= P(*RHS* | *LHS*) =

= **#tuples-with-both-*LHS*-and-*RHS* / #tuples-with-*LHS***

Paulo Trigo Silva

The Goal of the Association Rules' Mining

Generate **all association rules**, from the dataset, that **satisfy**, the user defined, **minimum support** (*min_sup*), and **minimum confidence** (*min_conf*)

- An itemset satisfies minimum support if the occurrence frequency
  - of the itemset is greater or equal to ***min_sup***
- The "**frequent itemset**" concept
  - designates an itemset that **satisfies minimum support**
- A "**strong rule**" is one that satisfies, both
  - a minimum **support threshold** and a minimum **confidence threshold**
- Rules originating from the same itemset have the same support
  - but can have different confidence

Paulo Trigo Silva

… example – transaction, itemset, support, confidence

$$T = \begin{array}{l|l} t_1 & \mathbf{ABCD} \\ t_2 & \mathbf{BC} \\ t_3 & \mathbf{AC} \\ t_4 & \mathbf{AC} \\ t_5 & \mathbf{ABCD} \\ t_6 & \mathbf{ABC} \end{array}$$

Itemset	Support	Confidence
A	?	?
B	?	?
C	?	?
D	?	?
AB	?	?
AC	?	?
AD	?	?
CD	?	?
ABC	?	?
ACD	?	?
ABCD	?	?
...		

Paulo Trigo Silva

… support and confidence

$$T = \begin{array}{l|l} t_1 & \mathrm{ABCD} \\ t_2 & \mathrm{BC} \\ t_3 & \mathrm{AC} \\ t_4 & \mathrm{AC} \\ t_5 & \mathrm{ABCD} \\ t_6 & \mathrm{ABC} \end{array}$$

Itemset	Support	Confidence
A	5/6	-
B	4/6	-
C	6/6	-
D	2/6	-
AB	3/6	A⇒B: 3/5 B⇒A: 3/4
AC	5/6	...
AD	2/6	...
CD	2/6	...
ABC	3/6	AB⇒C: 3/3 | A⇒BC: 3/5 CB⇒A: 3/3 | B⇒AC: 3/4 AC⇒B: 3/5 | C⇒AB: 3/6
ACD	2/6	...
ABCD	2/6	...
...		

Paulo Trigo Silva

Are the "support" and "confidence" enough?

- Support for A ⇒ B = P(A, B)
  - "how frequently the items in the rule occur together"
- Confidence for A ⇒ B = P(B | A) = P(A, B) / P(A)
  - "probability of both the antecedent and the consequent appearing in the same transaction"
- … support and confidence are used to determine if a rule is valid
  - however, there are times when both of these measures may be high, and yet still produce a rule that is not useful…
- e.g., orange-juice ⇒ milk | support 30% | confidence 75%
  - sounds like an excellent rule, and in most cases, it would be!
  - but, *what if customers in general buy milk 90% of the time*?
  - … *then, orange juice customers are actually less likely to buy milk than customers in general*!

Additional measure – Lift (Improvement)

- The “lift” (improvement) indicates the strength of a rule
  - over the random co-occurrence of the antecedent and the consequent, given their individual support
- … “lift” provides information about the improvement,
  - i.e., the increase in probability of the consequent given the antecedent
- the “lift” is defined as

$$\text{lift}(LHS \Rightarrow RHS) =$$
$$= \text{support}(LHS \Rightarrow RHS) \,/\, (\text{support}(LHS) * \text{support}(RHS))$$
$$= \text{confidence}(LHS \Rightarrow RHS) \,/\, \text{support}(RHS)$$
$$= P(RHS \mid LHS) \,/\, P(RHS)$$

if Lift >= 1, then the *LHS* and the *RHS* are positively correlated,
otherwise, the *LHS* and the *RHS* are negatively correlated

Paulo Trigo Silva

… the “Lift” (Improvement) – an example

- … there are times when both of the “support” and “confidence” measures are high, and yet still produce a rule that is not useful

- e.g., orange-juice ⇒ milk | support 30% | confidence 75%
  - sounds like an excellent rule, and in most cases, it would be!
  - but, *what if customers in general buy milk 90% of the time*?
  - … *then, orange juice customers are actually less likely to buy milk than customers in general*!

- The “Lift” for the previous rule is given by

`lift( orange-juice ⇒ milk )` `=`

`= confidence(orange-juice⇒milk) / support(milk)`

`= 75% / 90% =` **`0.83 < 1`**

- … the increase in probability of the consequent given antecedent is ***lower than 1***, so those ***items are negatively correlated***

Paulo Trigo Silva

… example – support, confidence and lift

$$T = \begin{array}{l|l} t_1 & \text{ABCD} \\ t_2 & \text{BC} \\ t_3 & \text{AC} \\ t_4 & \text{AC} \\ t_5 & \text{ABCD} \\ t_6 & \text{ABC} \end{array}$$

Itemset	Support	Confidence	Lift
A	5/6	-	-
B	4/6	-	-
C	6/6	-	-
D	2/6	-	-
AB	3/6	A⇒B: 3/5 B⇒A: 3/4	?
AC	5/6	...	...
AD	2/6	...	...
CD	2/6	...	...
ABC	3/6	AB⇒C: 3/3 | A⇒BC: 3/5 CB⇒A: 3/3 | B⇒AC: 3/4 AC⇒B: 3/5 | C⇒AB: 3/6	?
ACD	2/6	...	...
ABCD	2/6	...	...
...			...

Paulo Trigo Silva

… example – support, confidence and lift

$$T = \begin{array}{l|l} t_1 & \text{ABCD} \\ t_2 & \text{BC} \\ t_3 & \text{AC} \\ t_4 & \text{AC} \\ t_5 & \text{ABCD} \\ t_6 & \text{ABC} \end{array}$$

Itemset	Support	Confidence	Lift
A	5/6	-	-
B	4/6	-	-
C	6/6	-	-
D	2/6	-	-
AB	3/6	A⇒B: 3/5 B⇒A: 3/4	A⇒B: (3/5)/(4/6)=9/10 B⇒A: (3/4)/(5/6)=9/10
AC	5/6	...	...
AD	2/6	...	...
CD	2/6	...	...
ABC	3/6	AB⇒C: 3/3 | A⇒BC: 3/5 CB⇒A: 3/3 | B⇒AC: 3/4 AC⇒B: 3/5 | C⇒AB: 3/6	AB⇒C: (3/3)/(6/6)=1 ...
ACD	2/6	...	...
ABCD	2/6	...	...
...			...

Paulo Trigo Silva

The “Lift” measure – characteristics

- If Lift > 1 then
  - the rule if better at predicting the result than “just guessing”
- If Lift < 1 then
  - the rule is doing worse than “just guessing”
- If Lift = 1 then
  - the rule is “similar to guessing”

Synthesis

Any rule with an improvement (lift) of less than 1 does not indicate a real cross-selling opportunity, no matter how high its support and confidence, because it actually offers less ability to predict a purchase than does random chance.

… "be careful" with the "Lift" measure

Rules that hold nearly 100% of the time may not get the highest possible lift.

An example:

5% of the people (in the database) are retired.

90% of the people (in the database) are more than 2 years old.

now, consider the rule:

`retired ⇒ more-than-2-years-old`

then, the lift for that rule is: $0.05 / (0.05*0.9) = 1.11$

which is only slightly above 1 for that rule (that holds nearly 100% of the time)!

Paulo Trigo Silva

Some other measures

- **Leverage** of rule ***LHS*** ⇒ ***RHS***
  - measures the proportion of additional elements covered by both the premise (left side) and consequence (right side) above the expected

  leverage(*LHS* ⇒ *RHS*) =

  = support(*LHS*, *RHS*) - support(*LHS*)*support(*RHS*)

  = P(*LHS*, *RHS*) - P(*LHS*)*P(*RHS*)

- **Coverage** of rule ***LHS*** ⇒ ***RHS***
  - measures how likely it is that the ***LHS*** occurs

  coverage(*LHS* ⇒ *RHS*) =

  = support(*LHS*) =

  = P(*LHS*)

Paulo Trigo Silva

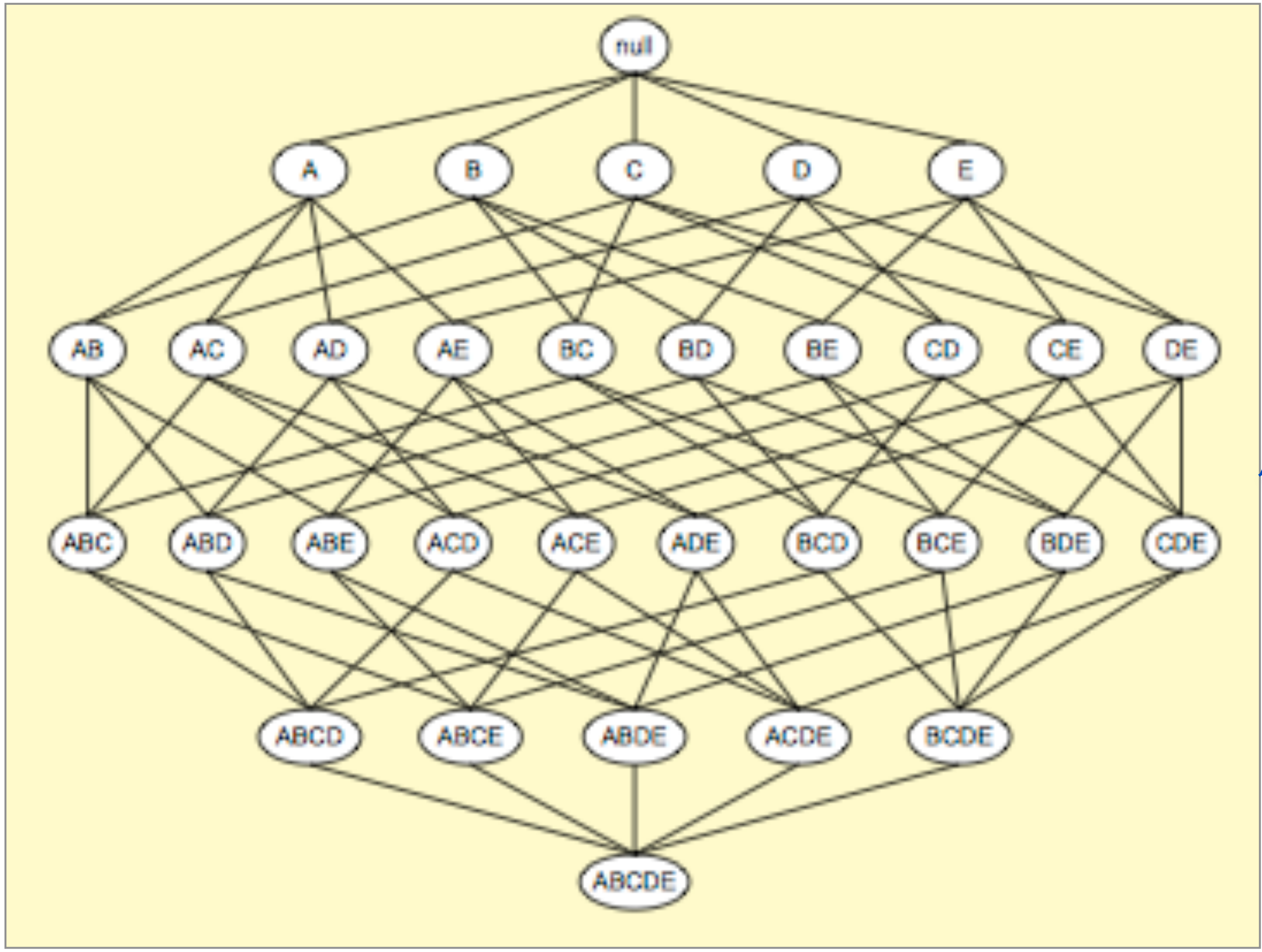
How to search for (i.e., mining of) association rules?

- Step 1: “frequent itemset generation”
  - generate all itemsets with support ≥ *min_sup* (e.g., provided by user)
- Step 2: “rule generation”
  - generate rules, from each frequent itemset, with confidence ≥ *min_conf*
  - … each rule is a binary partition (into LHS, RHS) of a frequent itemset

The “frequent itemset generation is computationally expensive.

Paulo Trigo Silva

The itemset space (an example with 5 items)



Given M items there are 2^M possible candidate itemsets.

e.g., for 5 items we have a total of $2^5=32$ itemsets

Paulo Trigo Silva

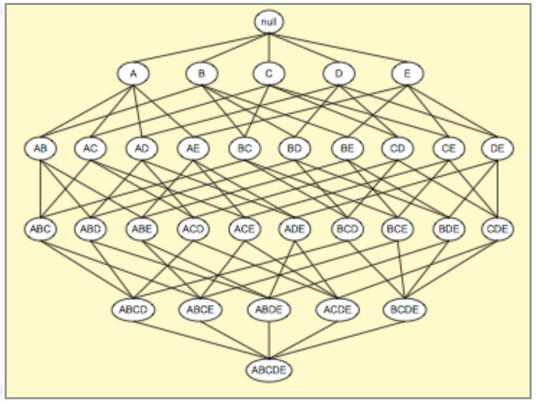
The “frequent itemset generation”

	ITEMS				
TID	A	B	C	D	E
t1		1		1	
t2		1	1	1	
t3			1		1
t4	1	1			
t5		1			

N

M

space of (itemset) candidates



2^M

a “brute force” approach:

Each itemset is a candidate frequent itemset

Count the support of each candidate by scanning the database

i.e., match each (of the 2^M) candidate itemset against all the (N) transactions!

complexity is O(N * 2^M); exponentially expensive with M

Paulo Trigo Silva

Strategies for the “frequent itemset generation”

- Reduce the number (2^M) of candidate itemsets to analyze
  - use pruning techniques
  - … e.g., as in the APRIORI algorithm
- Reduce the number (N) of transactions to scan
  - filter the number of transactions as the size of itemset increases
- Reduce the number ($N * 2^M$) of comparisons
  - use efficient data structures to store candidates or transactions
  - no need to match every candidate against every transaction
  - … e.g., as in the APRIORI algorithm

Paulo Trigo Silva

Algorithms to search for association rules

- The APRIORI algorithm
  - by Agrawal, R., T. Imielinski, and A. Swami
  - "Mining Association Rules between Sets of Items in Large Databases"; ACM SIGMOD International Conference on Management of Data"
  - … available in Orange DataMining
  - *"(…) two algorithms for induction of association rules, a standard APRIORI algorithm for sparse (basket) data analysis and a variant for attribute-value data sets*"; cf., http://orange.biolab.si/doc/reference/Orange.associate/
- Other algorithms
  - DHP: Dynamic Hash and Pruning
  - FP-Growth: Frequent Pattern Growth
  - H-Mine: Hyper-structure mining of frequent-patterns

Paulo Trigo Silva

The APRIORI principle

*If an **itemset is frequent**, then all of its **subsets** must also be **frequent***

or

*If an **itemset is infrequent**, then all of its **supersets** must also be **infrequent***

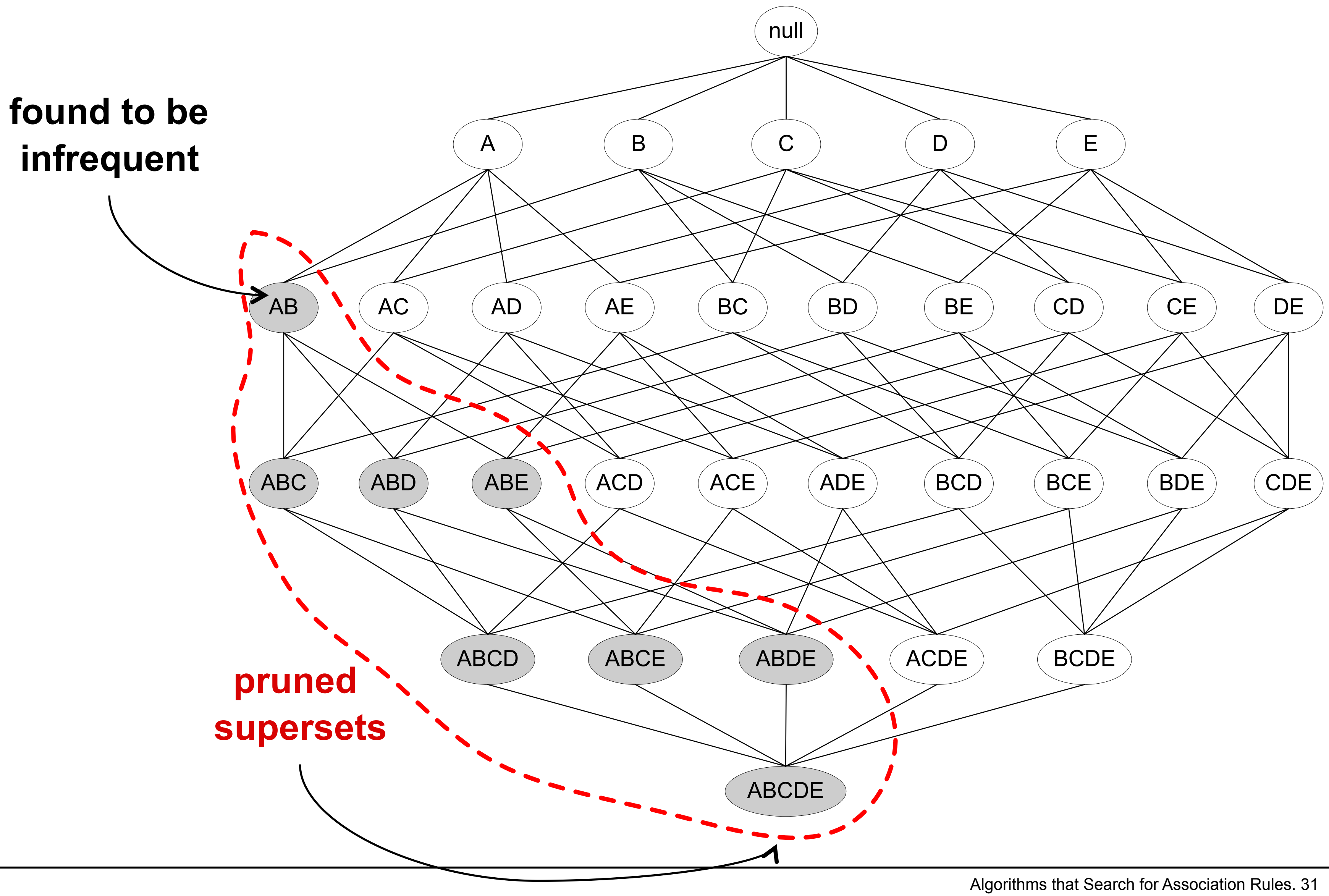
- The APRIORI principle holds due to this property of “support”:

$$\forall\ X, Y: (X \subseteq Y) \Rightarrow (\text{support}(X) \geq \text{support}(Y))$$

- … support of an itemset never exceeds the support of its subsets
  - this is known as the **anti-monotone** property of support
  - … increase itemset dimension decreases, or maintains, its support

Paulo Trigo Silva

Illustration of the APRIORI principle



Paulo Trigo Silva

The APRIORI algorithm – example

Consider the following “recorded database transactions”.

Given that “**`support = 2`**”, **generate the “frequent itemset”**

(*to simplify, “support” is presented before dividing it by the total number of transactions*)

Hint: apply the “APRIORI principle”

the recorded
database transactions

TID	ITEM
t1	A
t1	C
t1	D
t2	B
t2	C
t2	E
t3	A
t3	B
t3	C
t3	E
t4	B
t4	E

… example – a representation step

build a sparse representation of the transaction database

makes it easier to analyze the data

the recorded database transactions

TID	ITEM
t1	A
t1	C
t1	D
t2	B
t2	C
t2	E
t3	A
t3	B
t3	C
t3	E
t4	B
t4	E

apply the **pivot table** operator

a tabular **sparse** representation of the same information

	ITEMS				
TID	A	B	C	D	E
t1	1		1	1	
t2		1	1		1
t3	1	1	1		1
t4		1			1

Paulo Trigo Silva

… example – apply the “APRIORI principle”

Given that “**`support = 2`**”, **generate the “frequent itemset”**

Hint: apply the “APRIORI principle”

	ITEMS				
TID	A	B	C	D	E
t1	1		1	1	
t2		1	1		1
t3	1	1	1		1
t4		1			1

remove “**`support < 2`**”

ITEMSET	SUPPORT
{ A }	2
{ B }	3
{ C }	3
~~{ D }~~	~~1~~
{ E }	3

apply APRIORI principle

ITEMSET	SUPPORT
{ A }	2
{ B }	3
{ C }	3
{ E }	3

$\underbrace{\qquad}_{F_1}$

ITEMSET
{ A, B }
{ A, C }
{ A, E }
{ B, C }
{ B, E }
{ C, E }

$\underbrace{\qquad}_{C_2}$

F_{k-1} ≡ input for the “k-itemset candidate generation”
C_k ≡ output of the “k-itemset candidate generation”

see next slide…

… example – apply the "APRIORI principle" (cont.)

	ITEMS				
TID	A	B	C	D	E
t1	1		1	1	
t2		1	1		1
t3	1	1	1		1
t4		1			1

Given that "**support = 2**", **generate the "frequent itemset"**

Hint: apply the "APRIORI principle"

ITEMSET
{ A, B }
{ A, C }
{ A, E }
{ B, C }
{ B, E }
{ C, E }

ITEMSET	SUPPORT
~~{ A, B }~~	~~1~~
{ A, C }	2
~~{ A, E }~~	~~1~~
{ B, C }	2
{ B, E }	3
{ C, E }	2

ITEMSET	SUPPORT
{ A, C }	2
{ B, C }	2
{ B, E }	3
{ C, E }	2

***F*$_2$**

ITEMSET
{ A, B, C }
{ A, B, E }
{ A, C, E }
{ B, C, E }

***C*$_3$**

ITEMSET	SUPPORT
~~{ A, B, C }~~	~~1~~
~~{ A, B, E }~~	~~1~~
~~{ A, C, E }~~	~~1~~
{ B, C, E }	2

ITEMSET	SUPPORT
{ B, C, E }	2

Notice that {A, B, C} and {A, B, E} could never belong to "frequent itemset" because {A, B} was already excluded.

The same for {A, C, E} and {A, E}.

F_{k-1} ≡ input for the "k-itemset candidate generation"
C_k ≡ output of the "k-itemset candidate generation"

Paulo Trigo Silva

Details on how to generate "frequent itemset" candidates

- An important **assumption** – the "**ordering of items**"
  - meaning that **items** are stored in **lexicographical order**
  - … which is a **total order relation**
- The items' ordering is used by the algorithm in each item set

  w = (w[1], w[2], …, w[3]) is a tuple representing a k-itemset w, where $w[1] < w[2] < \ldots < w[3]$ according to the total order
- … for example, in the 3-itemset w = (B, C, E) we have
  - we have $w[1]=B < w[2]=C < w[3]=E$
  - according to the lexicographical total ordering of items

Paulo Trigo Silva

… how to generate "frequent itemset" candidates?

The "**generate-candidate**" function:

input: F_{k-1}; a *(k-1)*-itemset

output: C_k; a superset, called "candidates", of the set of all frequent *k*-itemsets.

The "**generate-candidate**" function has 2 steps.

- *step1*: **self-joining**
  - generate all possible candidate itemsets C_k of length *k*.
- *step2*: **pruning**
  - remove all candidates C_k that can't be frequent (i.e., are non-frequent)
  - … a superset of a non-frequent itemset is also non-frequent!

… the "generate-candidate" function

function generate-candidate($F_{k\text{-}1}$)

$C_k \leftarrow \varnothing$;

forall $f_1, f_2 \in F_{k\text{-}1}$

with $f_1 = \{i_1, \ldots, i_{k\text{-}2}, i_{k\text{-}1}\}$

and $f_2 = \{i_1, \ldots, i_{k\text{-}2}, i'_{k\text{-}1}\}$

and $i_{k\text{-}1} < i'_{k\text{-}1}$ **do**

$c \leftarrow \{i_1, \ldots, i_{k\text{-}1}, i'_{k\text{-}1}\}$; ***// join the f_1 and f_2 itemsets***

$C_k \leftarrow C_k \cup \{c\}$;

for each (k-1)-subset ***s*** of ***c*** **do**

if ($s \notin F_{k\text{-}1}$) **then**

delete ***c*** from C_k; ***// prune superset, c, of non-frequent s***

end

end

return C_k;

Paulo Trigo Silva

… example of “generate-candidate” function

The “**generate-candidate**” function:

input: F_{k-1}; a *(k-1)*-itemset

output: C_k; a superset, called “candidates”, of the set of all frequent *k*-itemsets.

Apply the “**generate-candidate**” function to F_3 and generate C_4.

input: F_{4-1} = { {1, 2, 3}, {1, 2, 4}, {1, 3, 4}, {1, 3, 5}, {2, 3, 4} }

output: C_4 = **?**

Paulo Trigo Silva

… example ("generate-candidate")

input: $F_{4\text{-}1}$ = { {1, 2, 3}, {1, 2, 4}, {1, 3, 4}, {1, 3, 5}, {2, 3, 5} }

output: C_4 = **?**

```
function generate-candidate( F_{k-1} )
    C_k ← ∅;
    forall f_1, f_2 ∈ F_{k-1}
        with f_1 = {i_1, … , i_{k-2}, i_{k-1}}
        and f_2 = {i_1, … , i_{k-2}, i'_{k-1}}
        and i_{k-1} < i'_{k-1} do
      c ← {i_1, …, i_{k-1}, i'_{k-1}};        // join f_1 and f_2
      C_k ← C_k ∪ {c};
      for each (k-1)-subset s of c do
        if ( s ∉ F_{k-1} ) then
          delete c from C_k;     // prune
      end
    end
    return C_k;
```

input → $F_{4\text{-}1}$ = {
{ 1, 2, 3 },
{ 1, 2, 4 },
{ 1, 3, 4 },
{ 1, 3, 5 },
{ 2, 3, 4 } }

after join → C_4 = {
{ 1, 2, 3, 4 },
{ 1, 3, 4, 5 } }

after prune → C_4 = {
{ 1, 2, 3, 4 } }

{ 1, 3, 4, 5 } is removed because { 1, 4, 5 } is not in F_3.

Paulo Trigo Silva

… how to "generate-rules" from the "frequent itemsets"?

"frequent itemsets" ≠ "association rules"

we need **one more step** to generate **association rules**

for each "frequent itemset" $\boldsymbol{X}$

  for each proper nonempty subset $\boldsymbol{A}$ of X (i.e., $\forall\ A \subset X$ and $X \neq \emptyset$)

    let $\boldsymbol{B} = X - A$

    if `confidence(` $\boldsymbol{A \Rightarrow B}$ `)` $\geq$ **`min_conf`**

    then $A \Rightarrow B$ is an association rule

recall that

`support(` $\boldsymbol{A \Rightarrow B}$ **`)`** = `support(` $A \cup B$ `)` = **`support(`** $\boldsymbol{X}$ **`)`**

`confidence(` $\boldsymbol{A \Rightarrow B}$ **`) = support(`** $\boldsymbol{A \cup B}$ **`) / support(`** $\boldsymbol{A}$ **`)`**

Paulo Trigo Silva

… example of “generate-rules”

suppose {2, 3, 4} is a “frequent itemset”
with support = 50%

Which rules can be generated from that itemset and what is the confidence values of each rule?

for the purpose of this example assume that:

- each 1-itemset has confidence=80%
- each 2-itemset has confidence=50%

Paulo Trigo Silva

… example (“generate-rules”)

suppose {2, 3, 4} is a “frequent itemset” with support = 50%

Proper non-empty subsets and *assumed* confidence values:

{2}	confidence=80%
{3}	confidence=80%
{4}	confidence=80%
{2, 3}	confidence=50%
{2, 4}	confidence=50%
{3, 4}	confidence=50%

The list of the corresponding association rules:

2⇒3,4	confidence=50%/80%=62.5%
3⇒2,4	confidence=50%/80%=62.5%
4⇒2,3	confidence=50%/80%=62.5%
2,3⇒4	confidence=50%/50%=100%
2,4⇒3	confidence=50%/50%=100%
3,4⇒2	confidence=50%/50%=100%

Paulo Trigo Silva

The “generate-rules” process – a synthesis

- In order to generate $A \Rightarrow B$ we need to have
  - the value of `confidence`($A \Rightarrow B$), that can be computed
  - … from the values of `support`($A \cup B$) and `support`(A)
- So, all the required information for confidence computation
  - was already recorded during itemset “generate-candidate” process
  - … no need to scan the data anymore for “generate-rule” process
- … the “generate-rules” process
  - is less time consuming than the itemset “generate-candidates” process

Paulo Trigo Silva

… example – the "generate-rule" process

the "generate-rule" process:

for each "frequent itemset" ***X***

for each proper nonempty subset ***A*** of X (i.e., $\forall\ A \subset X$ and $X \neq \varnothing$)

let ***B*** = $X - A$

if `confidence`(***A*** ⇒ ***B***) ≥ **`min_conf`**

then $A \Rightarrow B$ is an association rule

If {A, B, C, D} is a "frequent itemset" then what is the entire set of candidate rules?

Apply the above "generate-rule" process

Paulo Trigo Silva

… example – the result of the "generate-rule" process

for each "frequent itemset" ***X***

for each proper nonempty subset ***A*** of X (i.e., $\forall\ A \subset X$ and $X \neq \emptyset$)

let ***B*** = $X - A$

if `confidence`($\boldsymbol{A \Rightarrow B}$) ≥ **`min_conf`**

then $A \Rightarrow B$ is an association rule

If {A, B, C, D} is a "frequent itemset" then the candidate rules are:

A,B,C ⇒ D	A,B,D ⇒ C	A,C,D ⇒ B	B,C,D ⇒ A
A,B ⇒ C,D	A,C ⇒ B,D	A,D ⇒ B,C	
B,C ⇒ A,D	B,D ⇒ A,C		
C,D ⇒ A,B			
A ⇒ B,C,D	B ⇒ A,C,D	C ⇒ A,B,D	D ⇒ A,B,C

If $|X| = k$, then there are $2^k - 2$ candidate association rules

(ignoring $\emptyset \Rightarrow X$ and $X \Rightarrow \emptyset$)

Paulo Trigo Silva

… "confidence" and the anti-monotone property

In general "confidence" does not have an anti-monotone property:

e.g., `confidence`(*ABC* ⇒ *D*) can be higher or lower than `confidence`(*AB* ⇒ *DE*)

But, "**confidence**" of **rules generated from the same itemset, *X*,**
have an anti-monotone property:

e.g., *X* = {A, B, C, D}

`confidence`(*ABC* ⇒ ***D***) ≥ `confidence`(*AB* ⇒ ***CD***) ≥ `confidence`(*A* ⇒ ***BCD***)

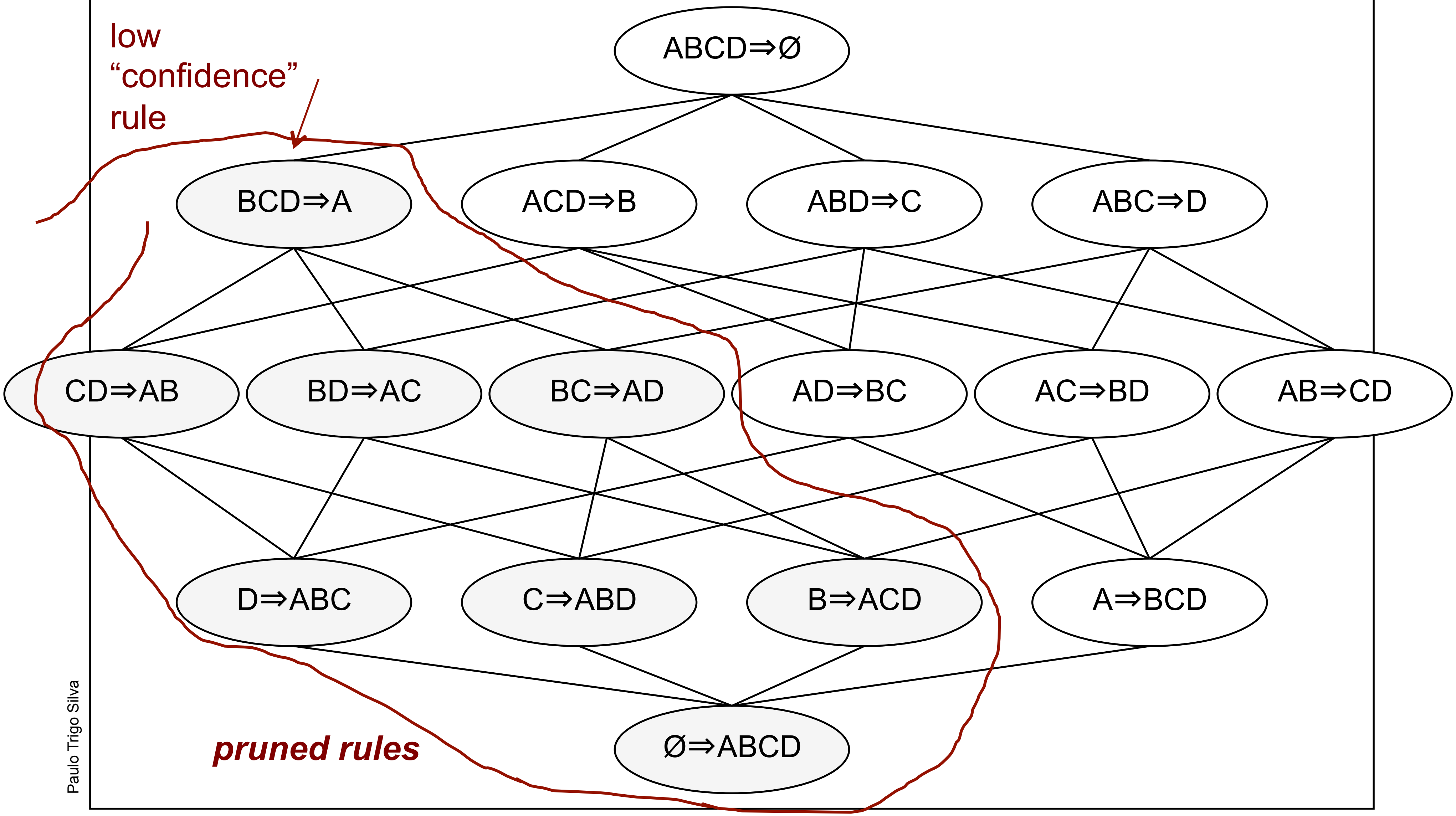
***The "confidence" anti-monotone property*:**

"confidence" is anti-monotone with respect to the number of items on the RHS of the rule.

i.e., increase RHS dimension decreases, or maintains, its confidence

Paulo Trigo Silva

… the “generate-rule” space



Paulo Trigo Silva

A more efficient way to "generate-rule"

- A candidate rule is generated by merging two rules
  - that share the same prefix in the rule consequent
  - and have a non disjoint antecedent
  - … i.e., **same prefix in RHS** of both rules
  - … and **intersection of both LHS is not empty**
- … `join`($CD \Rightarrow \boldsymbol{A}B$, $BD \Rightarrow \boldsymbol{A}C$) would produce the candidate rule:
  $D \Rightarrow \boldsymbol{A}BC$
  (LHS with common items of both rules; RHS includes both last items)
- … if rule $D \Rightarrow \boldsymbol{A}BC$ does not have the required confidence
  - then prune this rule
- recall that support counts have already been computed
  - during the "frequent itemset" generation step

Paulo Trigo Silva

Characteristics of the APRIORI method and an extension

- Uses the same minimum support constraint for all items
  - i.e., a single `min_sup` threshold value
  - in many scenarios some items appear more frequently than others
  - e.g., people by "cooking pan" and "coffee machine" much less frequently than they by bread and milk!
- If frequency of items vary a lot, then
  - if `min_sup` is set too high, rules involving rare items will not be found
  - if `min_sup` is set too low, may cause combinatorial explosion as those frequent items will be associated with each other in all possible ways
- The original algorithm can be extended to
  - deal with "multiple `min_sup` thresholds"
  - … where each item can have a "minimum item support" (MIS)
  - … the APRIORI principle must consider the total order relation of MIS

Paulo Trigo Silva