Inductive Databases and Constraint-based Mining

Jean-François Boulicaut
INSA Lyon (F)

Luc De Raedt
Albert-Ludwigs-University Freiburg (D)

ECML-PKDD'02 Helsinki (FIN) - August 19, 2002

Inductive databases

- Data mining
  - search for interesting and understandable patterns in data
- State-of-the-art in data mining ~ databases in the early days
- A theory of data mining is lacking
- View by Mannila and Iemielinski (CACM 96)
  - Make first class citizens out of patterns
  - Query not only the data but also the patterns
  - Tightly integrate data mining and databases

Overview

1. Introduction to inductive databases
2. On query languages for data mining

    Design issues and examples

3. Underlying principles of inductive querying

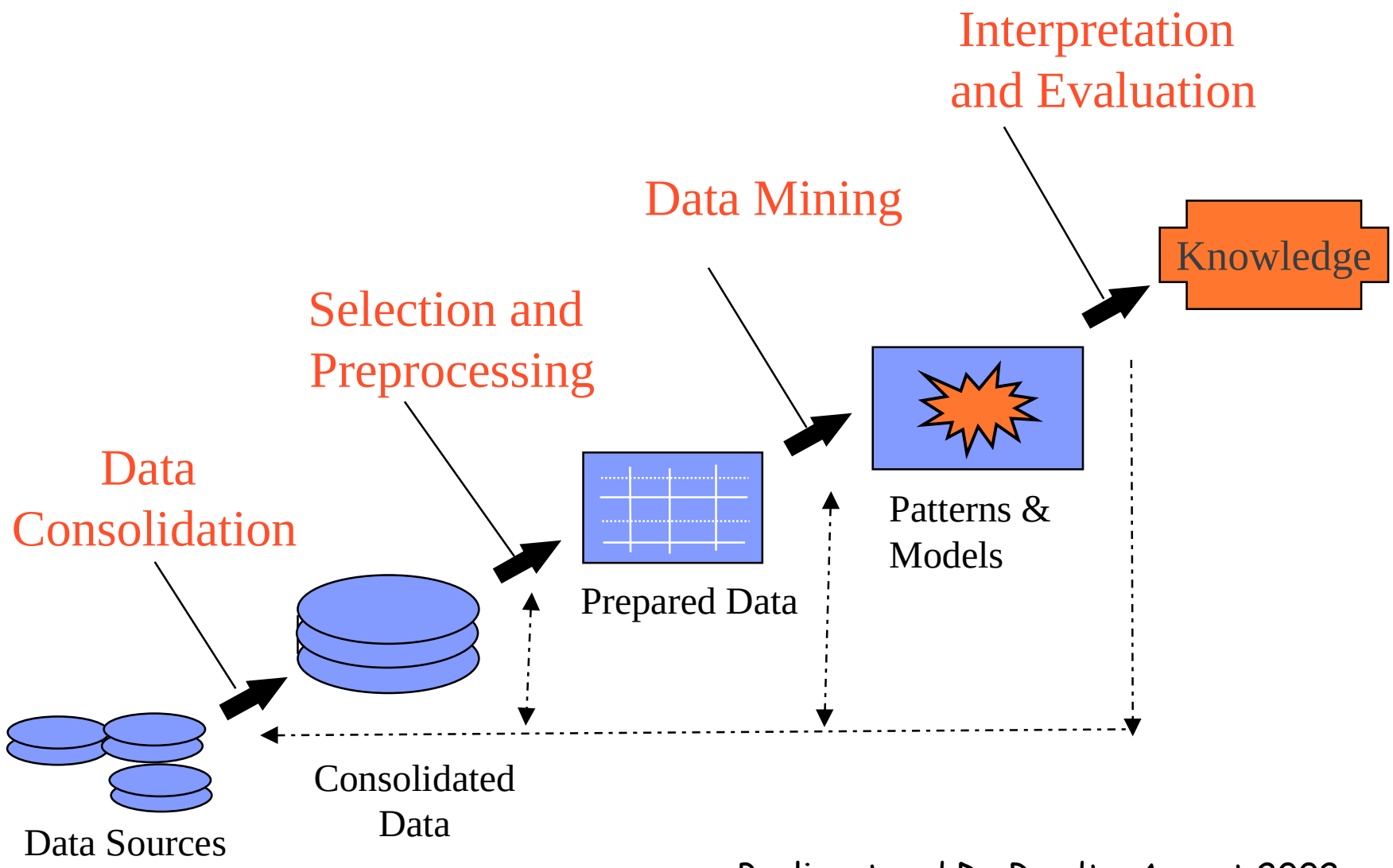
    Constraint-based Mining

4. Perspectives

1. Introduction to Inductive Databases

- Supporting complex and interactive knowledge discovery processes
  - Search for interesting patterns in data
    - Groups of customers, clusters of genes
    - Frequent sequential patterns in alarms
    - Molecular fragments that characterize toxicity
- From data to knowledge

The KDD process



A vision

- Supporting KDD processes by means of *queries*

  «There is no such thing as real discovery, just a matter of the expressive power of the query languages»

  Imielinski & Mannila, CACM Nov. 1996

  - Make first class citizens out of patterns

- Examples queries

  - Give a decision tree that tests upon at most 5 attributes including blood pressure and sex, and that has accuracy at least 90 % on the training data

  - Give all fragments of molecules that appear in at least 20% of the actives, and in at most 1% of the inactives, and that do not contain a benzene ring.

A long-term perspective

- Why is the relational model so succesful?
  - A general purpose query language with « nice » properties
    - simple theoretical foundations
    - declarative semantics
    - closure principle

  The same is needed for KDD applications

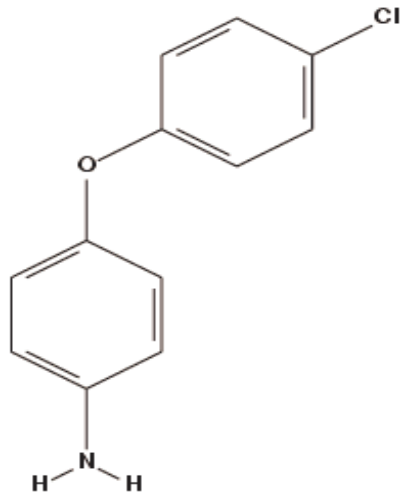
  The ultimate goal of IDBs is to find the equivalent of Codd's relational database model for use in data mining

Two Examples

- Molecular Fragments
  - A domain specific IDB
  - See Kramer et al KDD 01, De Raedt and Kramer IJCAI 01
- Association rules and Item sets
  - Main paradigm in existing IDBs and IDB extensions of SQL

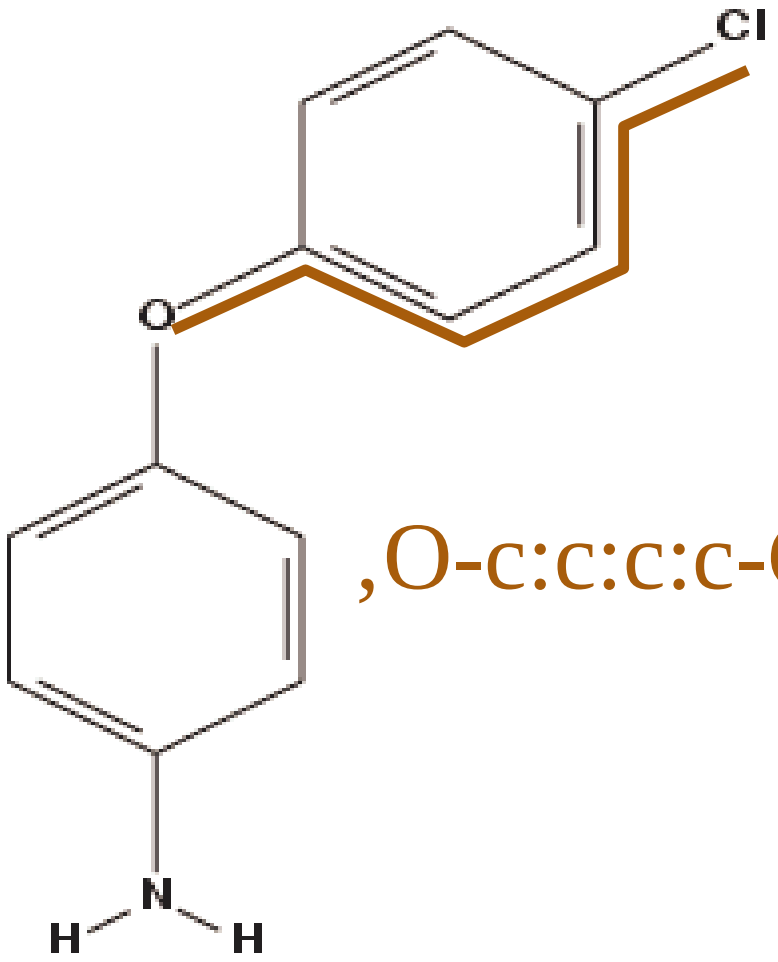
Molecular Feature Mining: Molfea

- What ?
  - Find fragments (substructures) of interest in sets of molecules



- Why ?
  - Discover new knowledge
  - Use in predictive models
    - SAR (Structure Activity Relationship)
    - De Raedt & Kramer 01 (ijcai)

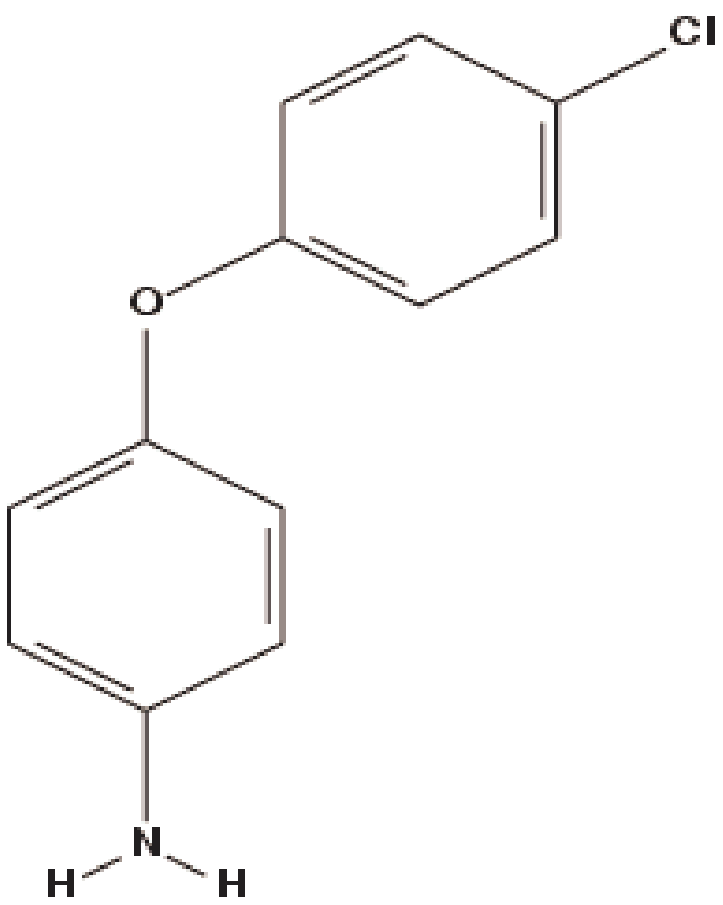
Molecules and Fragments



‚O-c:c:c:c-Cl‘

- 2D-structure
  - Essentially Graphs
- Fragments
  - Substructres
  - Linear fragments
  - Sequence of atoms and bonds
- Linear fragments
  - ‚o', ‚c', ‚cl', ‚n', 's',... denote elements
  - ‚-' ... single bond
    ‚=, ... double bond
    ‚#' ... triple bond
    ‚:' ... aromatic bond
  - (hydrogens implicit)
- Smarts encoding

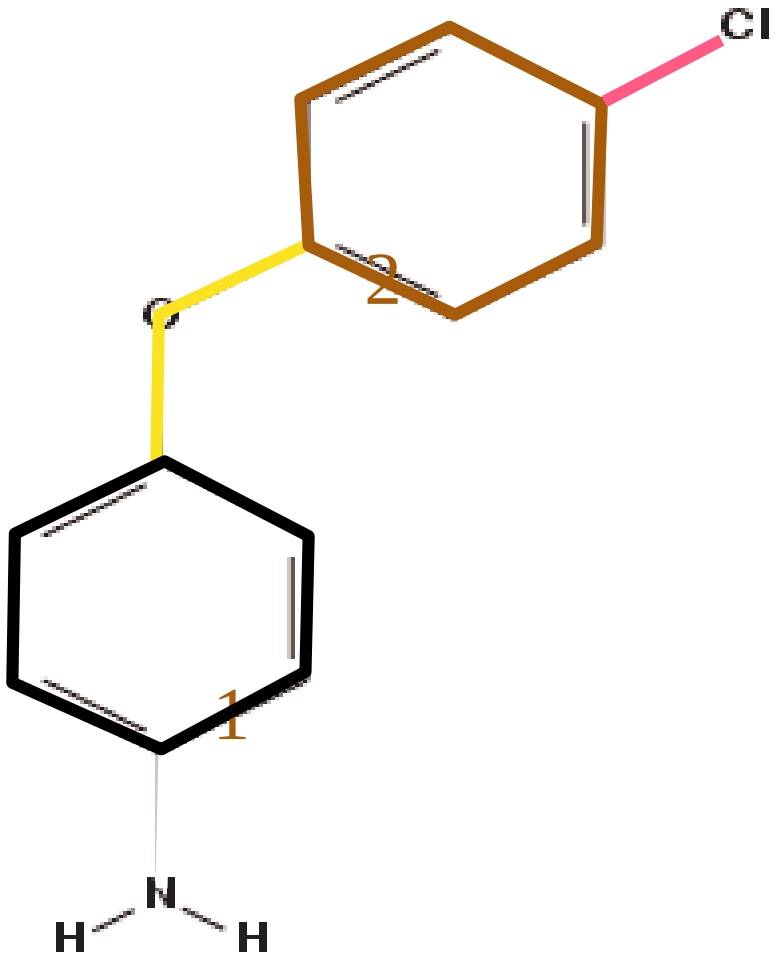
Smiles encoding



- Smiles
  - Compact encoding of molecular structure
  - Used by computational chemists
  - Supported by many tools (e.g. Daylight)
  - Very compact !
  - Very efficient matching

$$N-c1:c:c:c(-O-c2:c:c:c(-Cl):c:c2):c:c1$$

Smiles encoding



$N-$

$N-c1:c:c:c:c:c1$

$N-c1:c:c:c(-O-):c:c1$

$N-c1:c:c:c(-O-$

$c2:c:c:c:c:c2):c:c1$

$N-c1:c:c:c(-O-c2:c:c:c(-Cl):c:c2):c:c1$

Constraint-based mining (1)

- What ?
  - Use constraints to specify which fragments are interesting
    - The scientist/user controls the mining process
  - Evaluation functions (generality, frequency)
  - Primitive constraints (minimal/maximal frequency)
  - Boolean operators (conjunctions)
  - *Declarative* mining !

Constraint-based mining (2)

- Generality
  - One fragment is more general than another one if it is a substructure of the other one
  - Notation : $g \leq s$ *(g is more general than s, i.e., g will match a graph/string whenever s does)*
  - Graphs : ~ subgraph relationship
  - Strings : substring / subsequence relationship
    - E.g. *aabbcc is more general than ddaabbccee (substring), abc is more general than aabbcc (subsequence)*
  - Item sets : subset relation

Primitives

- MolFea Specific !
  - g is equivalent to s (*syntactic variants*) only when they are a reversal of one another

    E.g. ‚*C-O-S*' and ‚*S-O-C*' denote the same substructure

  - g is more general than s if and only if g is a subsequence of s or g is a subsequence of the reversal of s

    E.g. ‚*Cl-O-S*' ≤ ‚*Cl-O-S-c:c:c*', ‚*O-Cl*' ≤ ‚*Cl-O-S*'
- Frequency of a fragment f on a data set D
  - Percentage of data points in D that f occurs in

Primitive constraints

- $f \leq P$, $P \leq f$, $not\ (f \leq P)$ and $not\ (P \leq f)$

  f ... unknown target fragment
  P ... a specific fragment

  Assume $Freq(f,D)$ is the relative frequency of a fragment f on a data set D
  E.g., let f be aa and D={abaa,acc, caa}, freq(f,D) = .66=2/3

- $Freq(f,D1) \geq t$, $Freq(f,D2) \leq t$

  t ... positive real number between 0 and 1
  $D1$, $D2$ ... Data sets

  E.g. $Freq(f, Pos) \geq 0.20$

Example queries

- Queries are conjunctions of primitive constraints

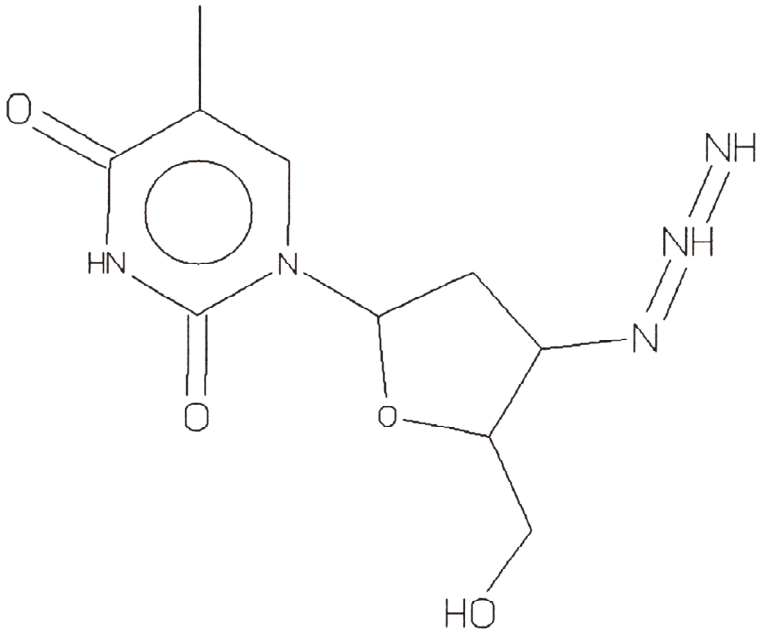
$$(\text{`}N\text{-}O\text{'} \leq f)$$
$$\wedge\ (Freq(f, Act) \geq 0.1)$$
$$\wedge\ (Freq(f, Inact) \leq 0.01)$$

$$not(,F\text{'} \leq f) \wedge not\ (,Cl\text{'} \leq f)$$
$$\wedge\ not\ (,Br\text{'} \leq f) \wedge not\ (,I\text{'} \leq f)$$
$$\wedge\ (Freq(f, Act) \geq 0.05)$$
$$\wedge\ (Freq(f, Inact) \leq 0.02)$$

The HIV Data Set De Raedt & al 01 (sigkdd)

- Developmental Therapeutics Program's AIDS Antiviral Screen Database (http://dtp.nci.nhi.gov)
  - One of the largest public domain databases of this type
- Measures protection of human CEM cells from HIV-1 infection using a soluble formazan assay
- 41768 compounds have been selected among the 43382 ones
  - 40282 Confirmed Inactive
  - 1069 Confirmed Moderately Active
  - 417 Confirmed Active

AZT (Azidothymidine)



$$N=N=N-C-C-C-n:c:c:c=O$$

$$N=N=N-C-C-C-n:c:n:c=O$$

The majority of these fragments are derivatives of AZT.

Gives insight into the structural requirements for anti-HIV activity.

A rediscovery that proves the principle

Post-processing

Combine fragments ?

Another Example : Item Sets

- Association rule mining Agrawal & al. 93 (sigmod)

A_1	A_2	A_3
1	0	0
1	1	1
1	0	1
0	1	1

baskets - products

documents - keywords

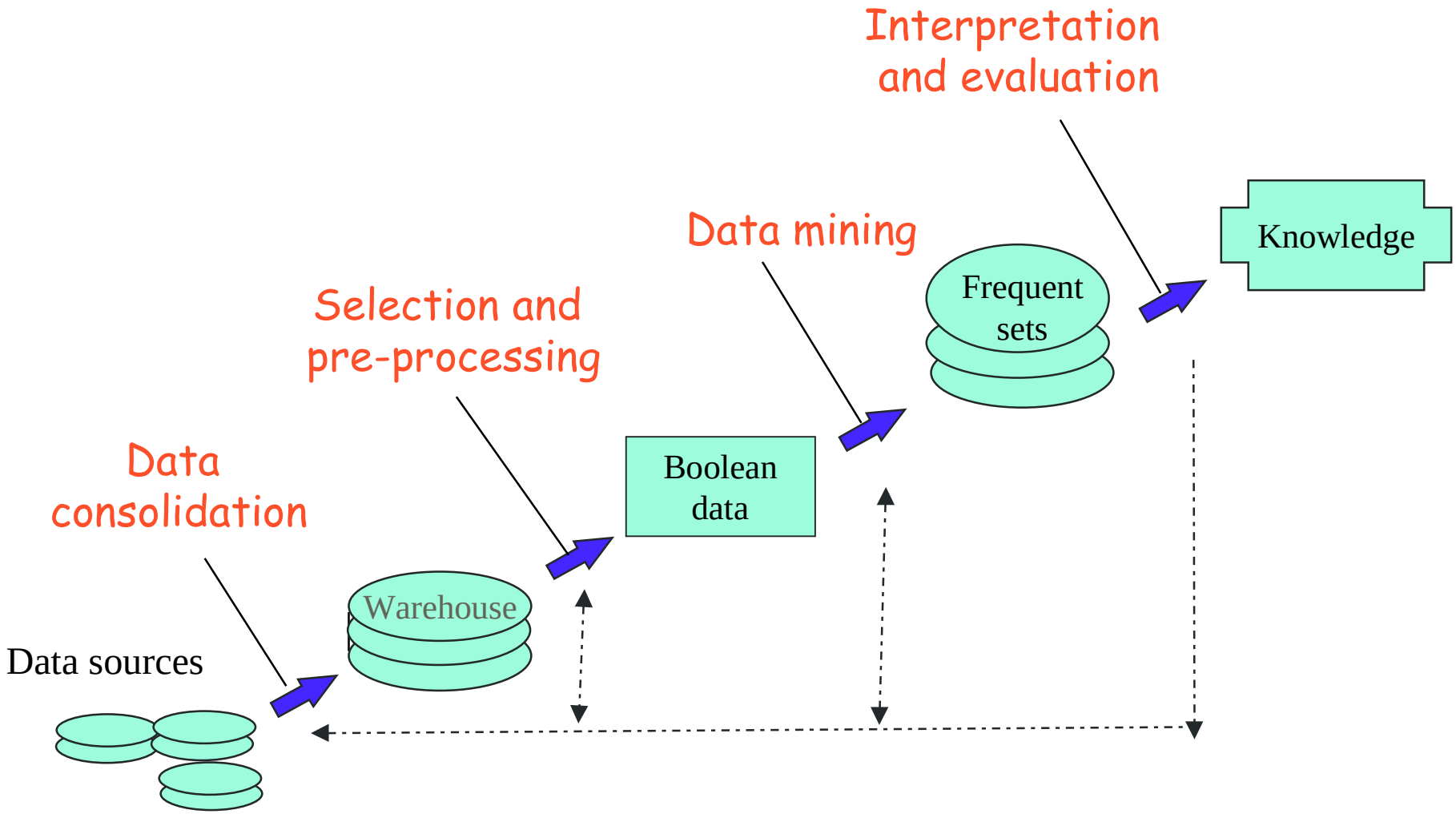
bacteria - properties

cells - genes

$A_1\ A_2 \Rightarrow A_3$ [1/4, 1]

$A_1 \Rightarrow A_2$ [1/4, 1/3]

Knowledge discovery from boolean contexts



Association rule mining process

- Standard process - Agrawal & al. 96 (aaai press)

  Mining every association rule for which support and confidence are greater than user-given thresholds

  - Computing frequent itemsets
  - Deriving interesting rules from frequent rules

  Objective vs. subjective measures of interestingness

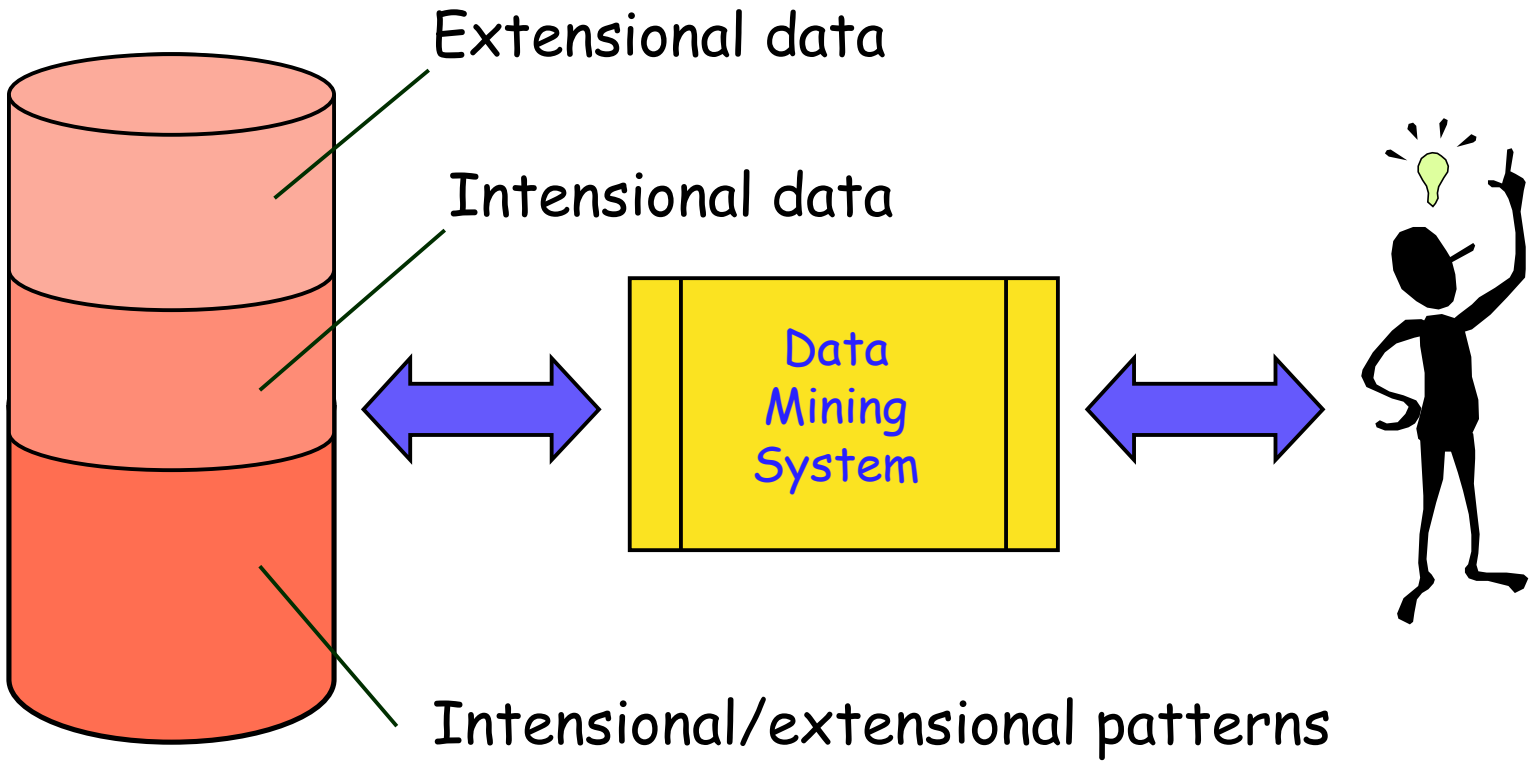
Supporting by means of queries (1)

- Pre-processing : manipulating data sets
  - E.g., compute a boolean context
    - Selections of relevant sources, agregations, sampling, discretizations, etc
- Data Mining : generating pattern sets
  - E.g., compute 5%-frequent association rules
    - A query as some « syntactic sugar » on top of an algorithm;
    - Declarative data mining using constraints

Supporting by means of queries (2)

- Post-processing : manipulating pattern sets
  - E.g., identify interesting rules among the frequent ones
    - Selections of relevant patterns or models, redundancy elimination, grouping, etc
  - Querying materialized collections of patterns
  - Crossing over the patterns and the data

The Inductive Database framework



Inductive database abstraction

- What is an inductive database ?
  - A set of data sets
  - A set of pattern sets
- IDB languages
  - A query language that generates data sets
  - An inductive query language that generates pattern sets
- Closure principle !
  - The result of a query should be a pattern set, a data set or a combination thereof
- An *abstract* set and logic oriented view
- Not a universal framework, though quite general

Manipulation

- `create data set D as query`
- `create view data set D as query`
- `create pattern set P as query`
- `create pattern view P as query`
- `Insert / Delete / Update statements`

- Data and Pattern sets can be extensional / intensional !

Illustration

```
create data set D1 as Q1
create pattern view P1 as Q2(D1)
   At this point assume P1 = PSet1
update data set D1 using Q2
   Update P1 too : P1 = update(PSet1)
```

- Incremental data mining !

- `Insert P2 into pattern view P1`
- Pattern view update problem

Abstraction

- Patterns domains specify
  - Language of patterns (e.g., itemsets, association rules, sequences, graphs, dependencies, decision trees, clusters)
  - Evaluation functions (e.g., frequency, closures, generality, validity, accuracy)
  - Primitive constraints (e.g., minimal and maximal frequency, freeness, syntactical constraints, minimal accuracy)
- Situation similar to constraint programming
  - Declarative aspects

2. On query languages for data mining

- MINE RULE — Meo & al. 96 (vldb), 98 (icde)
- MSQL — Imielinski & Virmani 96 (kdd), 99 (dmkd)
- LDL++ — Giannotti & Manco 99 (pkdd)
- RDM — de Raedt 00 (ilp)
- ...
- DMQL — Han & al. 96 (kdd)
- Molfea — De Raedt & Kramer 01 (ijcai)

Design issues

- Specification of the data part
  - Different data types
  - Pre-processing features
- Specification of the pattern part
  - Different pattern domains
  - Different constraints
  - Post-processing features
- The closure property

MINE RULE (1)

- A SQL-like operator on transactional DB

Table Purchase

Tid	Customer	Item	Date	Price	Qty
1	c1	ski-pants	12/1	55	1
1	c1	beer	12/1	4	2
2	c2	shirts	12/1	21	1
2	c2	jackets	12/1	115	1
3	c1	diapers	12/1	18	1
...	...	...	...	...	...

MINE RULE (2)

MINE RULE exemple as

SELECT DISTINCT 1..n Item as BODY, 1..1 Item as HEAD,
SUPPORT, CONFIDENCE

WHERE HEAD.Item=« umbrellas »

FROM Purchase

GROUP BY Tid

HAVING COUNT(*)<6

EXTRACTING RULES WITH SUPPORT: 0.06,
CONFIDENCE: 0.9

E.g., jacket flight_Dublin $\Rightarrow$ umbrellas (0.02,0.93)

MINE RULE (3)

```
MINE RULE WordOfMouth as
SELECT DISTINCT    1..1 Customer as BODY,
                   1..n Customer as HEAD,
                   SUPPORT, CONFIDENCE
WHERE BODY.Date <= HEAD.Date
FROM Purchase
GROUP BY Item
EXTRACTING RULES WITH SUPPORT: 0.01,
                      CONFIDENCE: 0.9
```

E.g., $c7 \Rightarrow c3\ c12$ (0.02,0.93)

MINE RULE (4)

++

- Data selection by means of « full » SQL
- Query evaluation can be effective

--

- Dedicated to association rules
- Poor possibilities for expressing background knowledge
- No specific mechanism for rule post-processing (results are stored in relational tables)

MSQL (1)

- Further integration within SQL

job=research $\wedge$ age = [26,38] $\Rightarrow$ position=AssProf
(0.31,0.95)

Emp(Id, Age, Job, Salary, Position)

```
GET_RULES (Emp)
INTO Rules
WHERE support > 0.1 and confidence > 0.8

SELECT_RULES (Rules)
WHERE    body has { (Age=*) (Job=*)}
         and head is { (Position=*)}
```

MSQL (2)

Emp(Id, Age, Job, Salary, Position)

```
SELECT *
FROM Emp
WHERE violates all ( GET_RULES (Emp)
                     WHERE   body is {(Age=*)}
                             and head is {(Salary=*)}
                             and confidence > 0.3 )
```

Connecting patterns to data

MSQL (3)

```
GET_RULES (Source) INTO R1
WHERE  body has {(Age=*)}
       and head has {(Salary=*)}
       and support > 0.1
       and confidence > 0.9
       and not exists (GET_RULES (Source) INTO R2
                       WHERE  body has {(Age=*)}
                              and head has {(Salary=*)}
                              and support > 0.1
                              and confidence > 0.9
                              and R2.body has R1.body)
```

A correlated query

MSQL (4)

++

- Query evaluation can be effective on data and persistently stored rules
- Useful operators for association rule mining (discretization, crossing over data and patterns)

--

- Dedicated to association rules
- Limits of the underlying relational framework (e.g., for the definition of background knowledge)

A first synthesis

- DMQL Han & al. 1996 (kdd) Han & Kamber 2001 (m-k)
  - A typical example of « syntactic sugar » for using many different data mining algorithms
  - But what are the fundamental primitives ?
- A critical evaluation of data mining query languages for association rule mining

  Deliverable D0 cInQ (01) - Botta & al. 02 (dawak)

  Pre and post-processing are poorly supported

Logic-based frameworks

- Data mining primitives embedded in logic programming / deductive databases
- Underlying idea :
  - Exploit similarity with constraint programming
- Two frameworks :
  - LDL++ Manco and Giannotti 99 (pkdd)
  - RDM De Raedt 00 (ilp)

LDL++

- Use LDL (deductive database language)
- Implement special « aggregate » primitives in LDL++ that can be used to implement data mining
- Various domains and tasks have been addressed

$q(Z_1, \ldots, Z_k, u\_d\_aggr\langle(X_1, \ldots, X_n)\rangle)$
$\leftarrow r(Y_1, \ldots, Y_m).$

E.g.,

$p(X_1, \ldots, X_n, patterns\langle(Y, m\_s, m\_c)\rangle)$
$\leftarrow r(Z_1, \ldots, Z_m).$

computes

$p(t_1, \ldots, t_n, lhs, rhs, f, c)$ See Ph.D. G. Manco (2001)

RDM

- From Inductive Logic Programming to Data mining primitives
  - Pattern language framework is based upon
  - Dehaspe's Warmr (dmkd 99)
  - Patterns : queries

    ?- customer(C),transaction(C,T1,D1,P1), transaction(C,T2,D2,P2), D1 > D2, P2 < P1.

    Frequent query framework
- Same constraints as in MolFea
- Not yet fully implemented, but see Lee and De Raedt (kdid 02)

LDL++/RDM ...

++

- Nice theoretical framework
- A number of data mining processes have been specified within that framework
- Representational issues: background knowledge, data but also patterns are expressed in the same formalism
- Power of embedding in logic programming language

--

- Efficiency (query optimisation issues)

3. Solving inductive queries

- Inductive Query Answering
- How to compute ?

  $Th(L \otimes E, r, q) = \{(\phi, e) \in L \otimes E \mid q(r, \varphi) \text{ is true}\}$

  - q is an inductive query
  - L a language of patterns
  - r an inductive database
  - e is a property of the pattern (e.g. frequency)
- « Generate and test » is generally impossible
- « Pushing constraints » can be difficult

Properties of constraints

- Anti-monotonicity of q w.r.t. $\leq$
  - q is anti-monotone w.r.t. $\leq$ if and only if
    - For all g,s : $g \leq s$ and s satisfies q implies g satisfies q
    - E.g., The minimal frequency is anti-monotone w.r.t. generality (molecular fragments, itemsets)

  The levelwise algorithm Mannila & Toivonen 97 (dmkd)

  - Many other examples (See, e.g., Ng & al. 98 (sigmod))

A String example

freq(f,D) ≥ 2 where D= $ABCD$ $BDEF$ $ABDF$ $ABCF$

ε

A B C D F

AB AC BD

ABC

Consider E

E is not frequent,

Therefore no string containing $E$is frequent

Consider ABC

ABC is frequent

Therefore all substrings of ABC are frequent

Characterized by $S = \{ABC, BD, F\}$

Another string example

Let $f \leq ABD$

$$\varepsilon$$

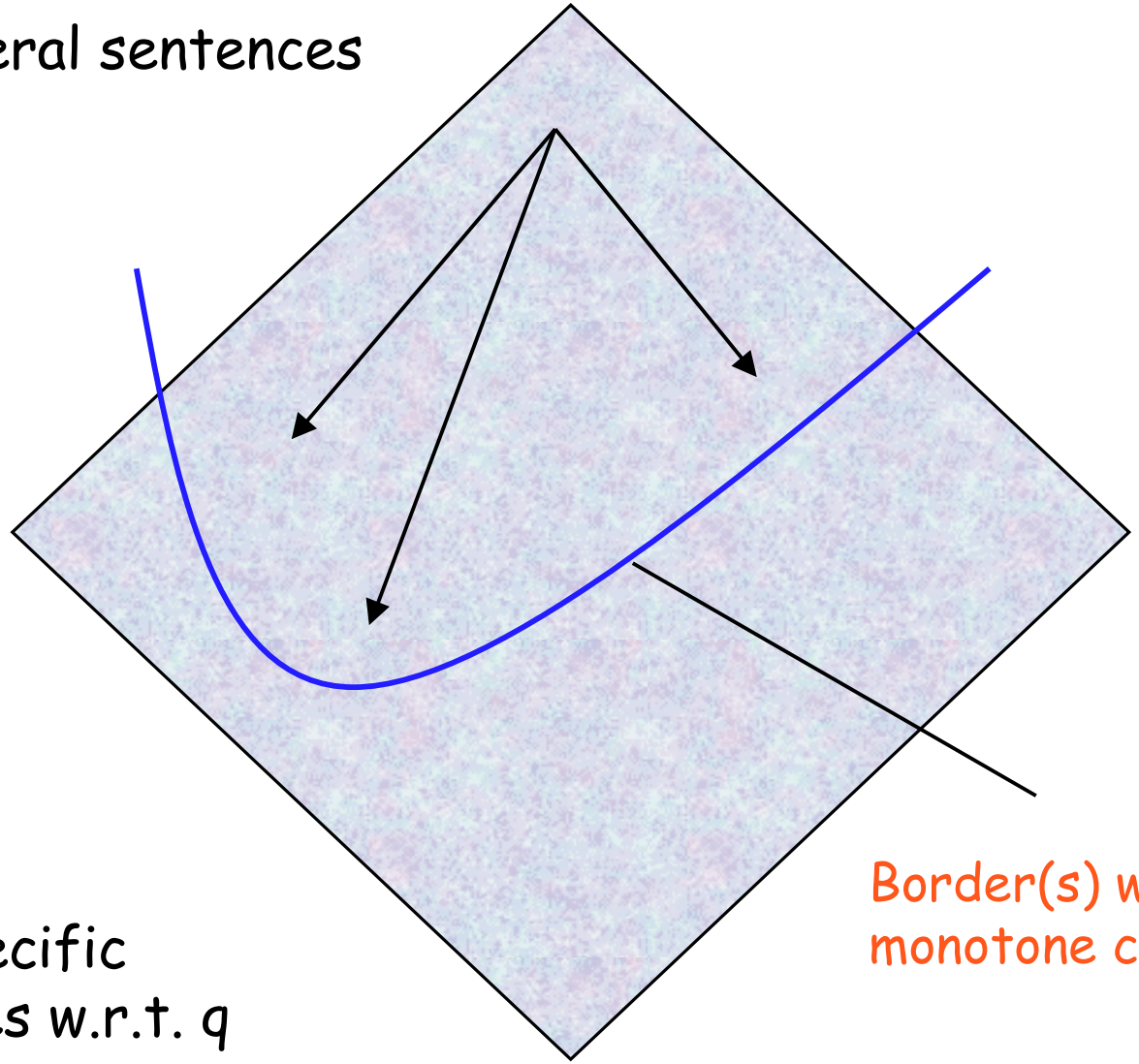
$$A \quad B \quad D$$

$$AB \quad BD$$

$$ABD$$

Characterized by $S = \{ABD\}$

Most general sentences w.r.t. q

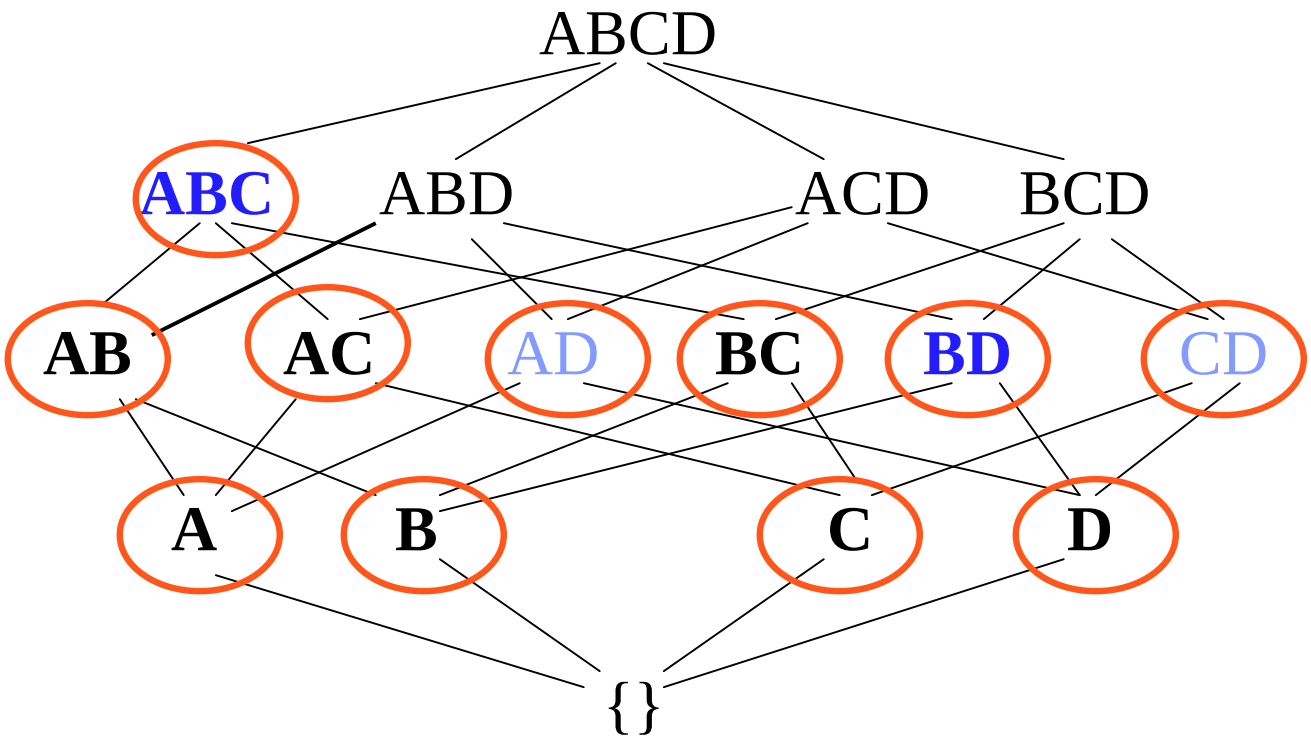


Most specific sentences w.r.t. q

Border(s) w.r.t. anti-monotone constraints

Application to frequent set mining (Apriori)

A	B	C	D
1	0	1	0
1	1	1	0
0	1	1	1
0	1	0	1
1	1	1	0



Frequency threshold 0.3

Borders of theories

- Positive border
  - The most specific interesting sentences

    E.g., the maximal frequent sets
  - In Machine Learning terminology : the S-set of the version space (Mitchell, Hirsh, Mellish)
- Negative border
  - The most general sentences that are not interesting

    E.g., the minimal infrequent sets
- Single border can represent the whole theory
  - Pro and Cons
- Borders are a *condensed* representation !
  - They store only a selection of the relevant solutions
  - Computing borders or theories ?

Example (Apriori type)

freq(f,D) ≥ 2 where D= $ABCD$ $BDEF$

$ABDF$ $ABCF$

ε

A B C D E F

AB AC AD AF BC BD BF CD CF DF

ABC ABD

Characterized by $S = \{ABC, BD, F\}$

« Guess and Correct » Mannila & Toivonen 97 (dmkd)

$C := Bd^+(S)$ — Clean the guess S

$E := \emptyset$

While C is not empty

  do $E := E \cup C$

    $S := S \setminus \{\varphi \in C \mid q(r,\varphi) \text{ is false}\}$

    $C := Bd^+(S) \setminus E$

  od

$C := Bd^-(S) \setminus E$ — Expand the corrected S

While C is not empty

  do $S := S \cup \{\varphi \in C \mid q(r,\varphi) \text{ is true}\}$

    $C := Bd^-(S) \setminus E$

  od

Output S — S = Th(L,r,q)

Computing Frequent Sets

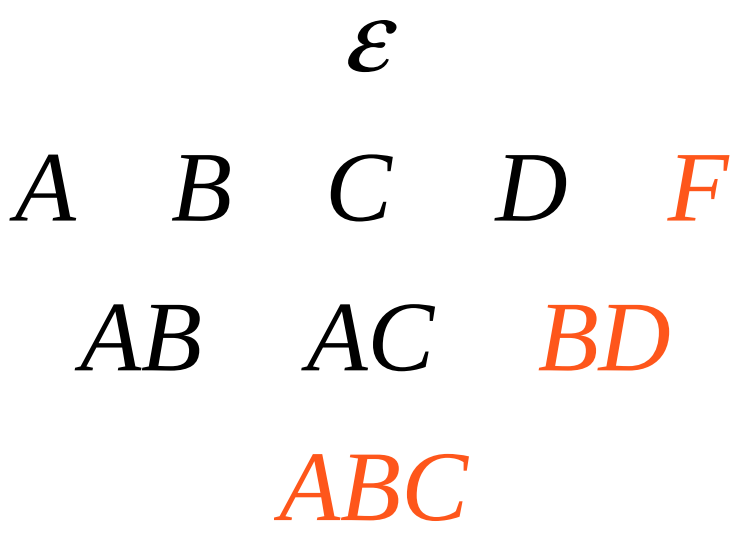
- Many variants exist, for theories
  - Fp Tree (Han et al.)
  - Apriori (Agrawal et al.)
- Borders, condensed
  - MaxMiner (Bayardo)

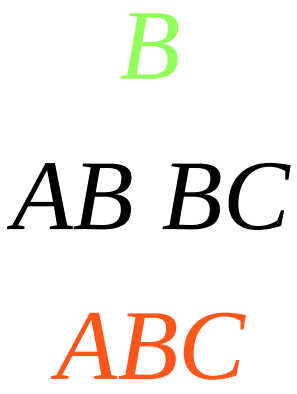
Representing solutions w.r.t. monotone constraints

- The maximal frequency constraint
  - Let c be freq(f, Act) < x, c is monotone w.r.t. ≤
    - If we have a fragment $g \leq s$, then if g is a solution then s is a solution as well
- Monotone constraints impose a border *G* on the space of solutions
  - q is monotone w.r.t. ≤ if and only if not(q) is anti-monotone w.r.t. ≤

A String example

Let "B" $\leq f$ and $Freq(f,D) \leq 2$ with D= $ABCD$ $BDEF$ $ABDF$ $ABCF$





Characterized by $S = \{ABC\}$ and $G = \{C\}$

Characterized by $S = \{ABC, BD, F\}$

Mitchell's Version Spaces (1)

- Consider now two constraints :

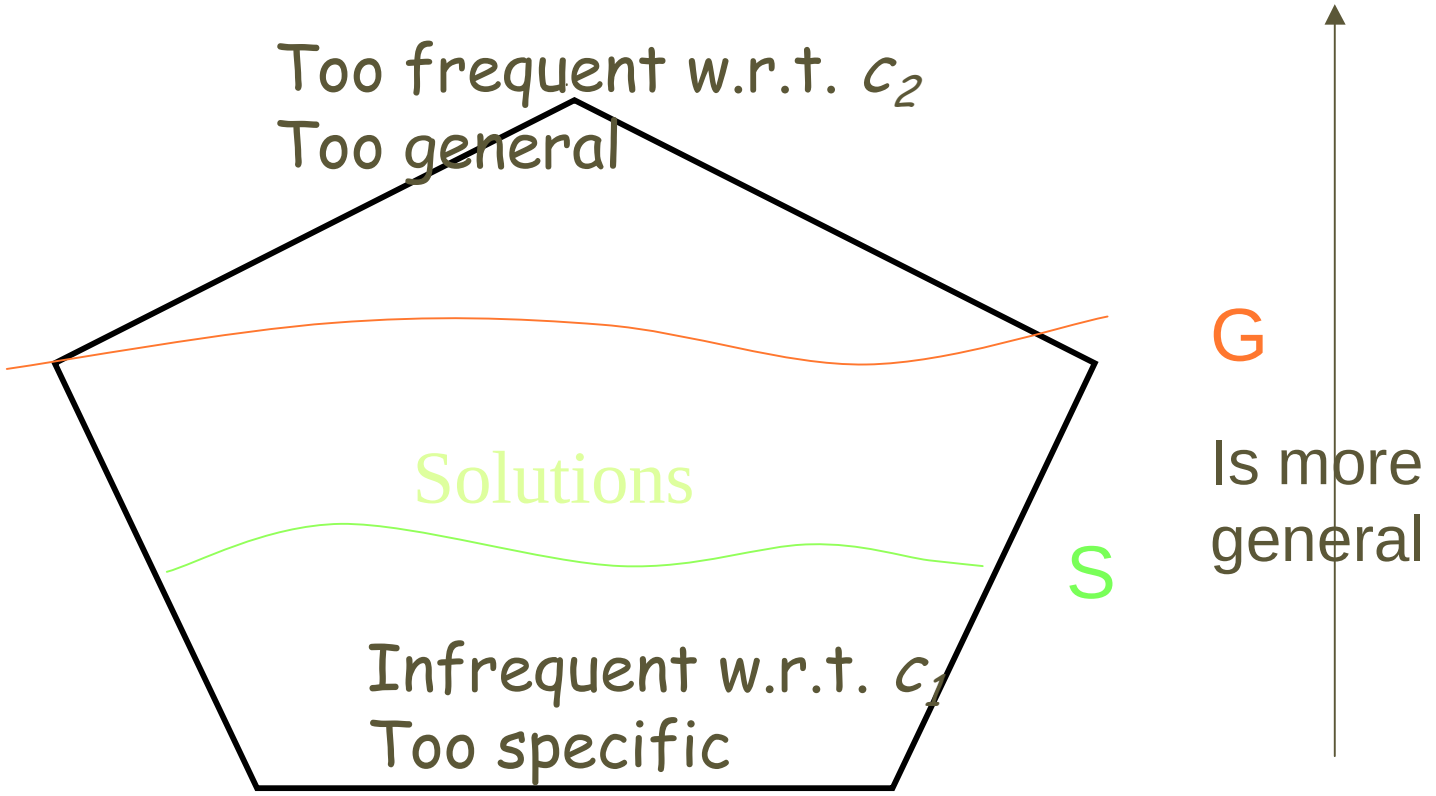
$$c_1 = freq(f, D) \geq x$$
$$c_2 = freq(f, E) \leq y$$

- We want to compute

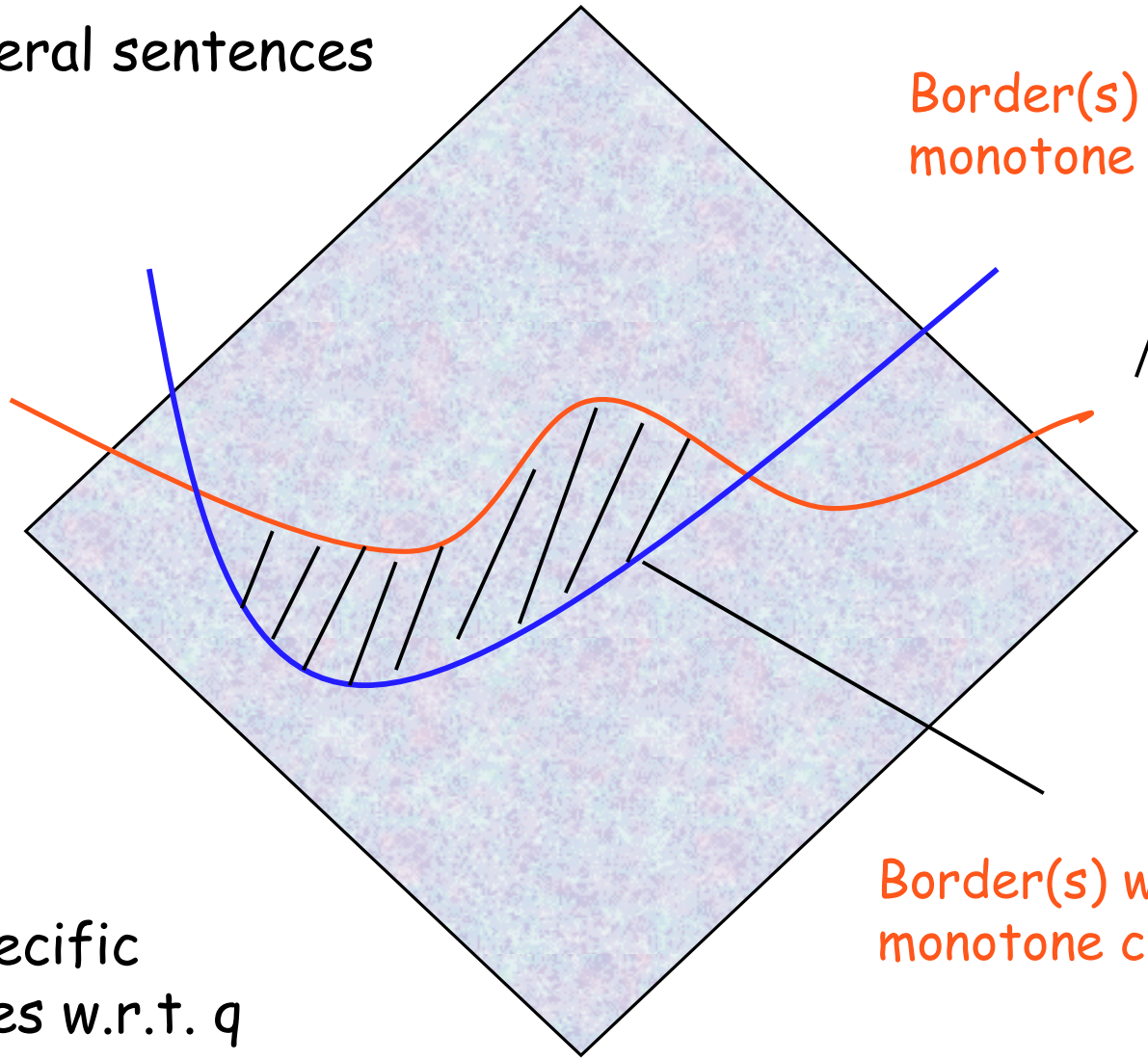
$$sol(c_1 \wedge c_2) = \{ f \mid \exists s \in S, g \in G : g \leq f \leq s \}$$

where S and G are defined w.r.t. $c_1 \wedge c_2$

Mitchell's Version Spaces



Most general sentences w.r.t. q



Border(s) w.r.t. monotone constraints

Most specific sentences w.r.t. q

Border(s) w.r.t. anti-monotone constraints

Constraints

Anti-monotonic

$$freq(f, D) \geq x$$

$$f \leq P$$

$$not(P \leq f)$$

In ML

$$f \leq P$$

$$\sim$$

P is a positive example

Monotonic

$$freq(f, D) \leq x$$

$$f \geq P$$

$$not(P \geq f)$$

In ML

$$not(f \leq P)$$

$$\sim$$

P is a negative example

Computing borders

- Borders completely characterize the set of solutions
  - Pro and cons
- Combination of well-known algorithms
  - Levelwise algorithm
  - Mitchell's and Mellish's version space algorithms

Generic algorithms for solving conjunctive constraints

- Condensed representation
  - Level wise version space algorithm (De Raedt 01)
- Theory level
  - Dual Miner (Gehrke et al. Kdd 02)
  - A generic levelwise algorithm for pushing conjunctions of anti-monotone and monotone constraints Boulicaut & Jeudy 01,02 ideas-ida
    - Using anti-monotone constraints for pruning
    - Using monotone constraints for candidate generation

Mellish's Description Identification Algorithm

$f \leq ABCD$

$G = G' = \{A, B, C\}$

$S = \{BCDE, FABC\}$

$S' = \{BCD, ABC\}$

Incrementally process constraints c

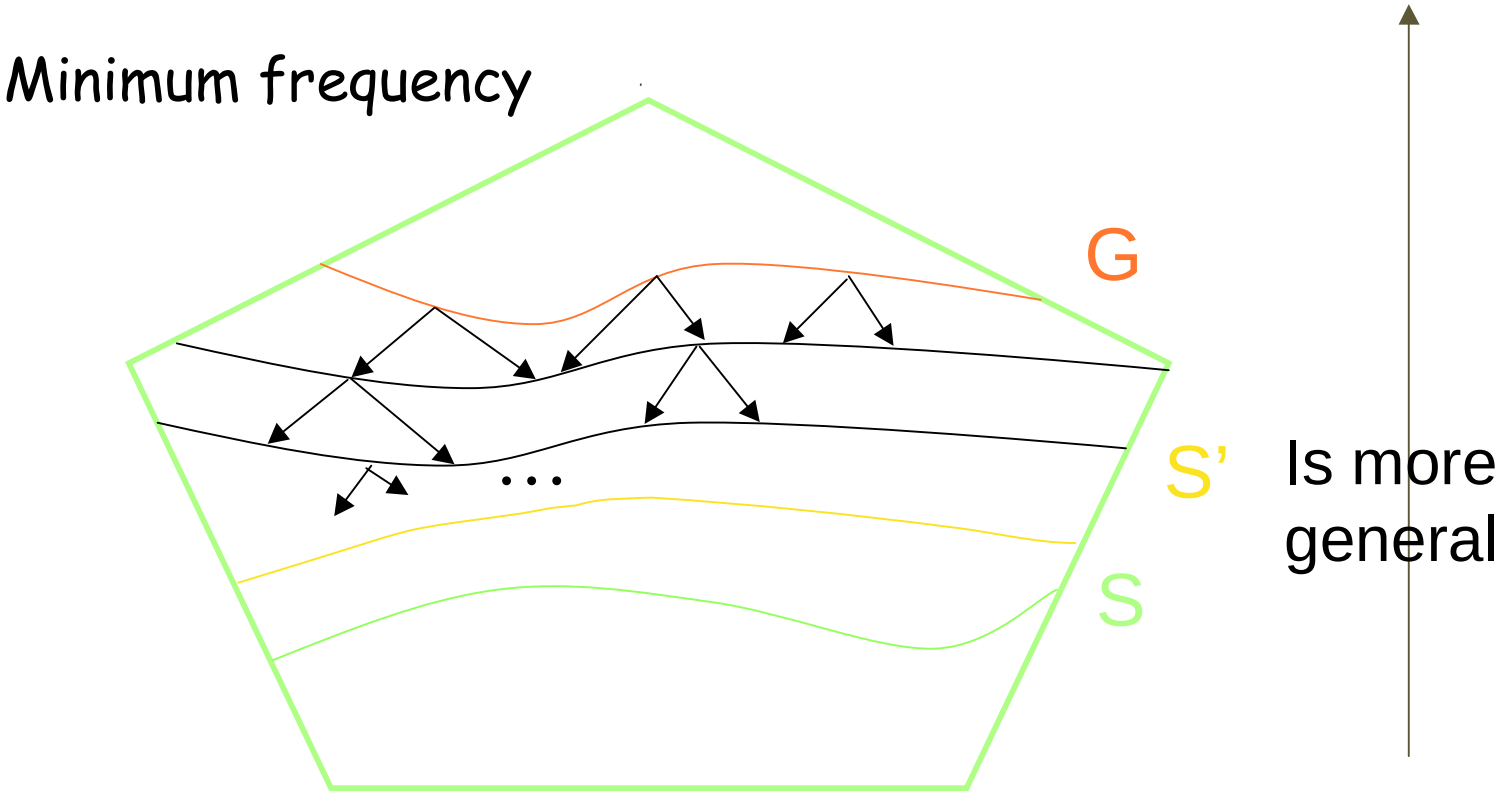
Case c of $f \leq P$ (P is a positive example)

$G = \{g \in G \mid g \leq P\}$

$S = \min\{l \mid l \in \text{lub}(s, P) \text{ and } s \in S \text{ and } \exists g \in G : g \leq l\}$

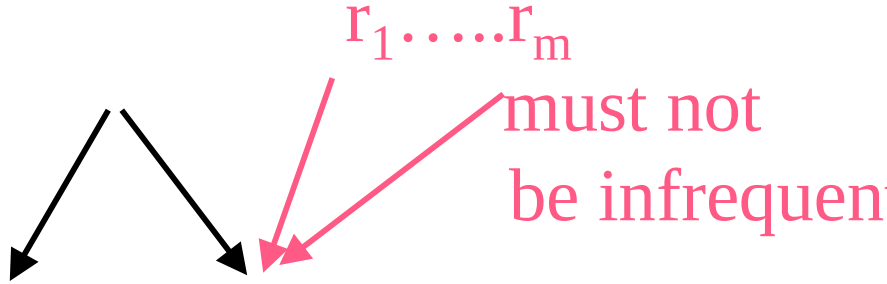
Here $\text{lub}(s, P) = \min\{l \mid l \leq s \text{ and } l \leq P\}$

Levelwise Version Spaces

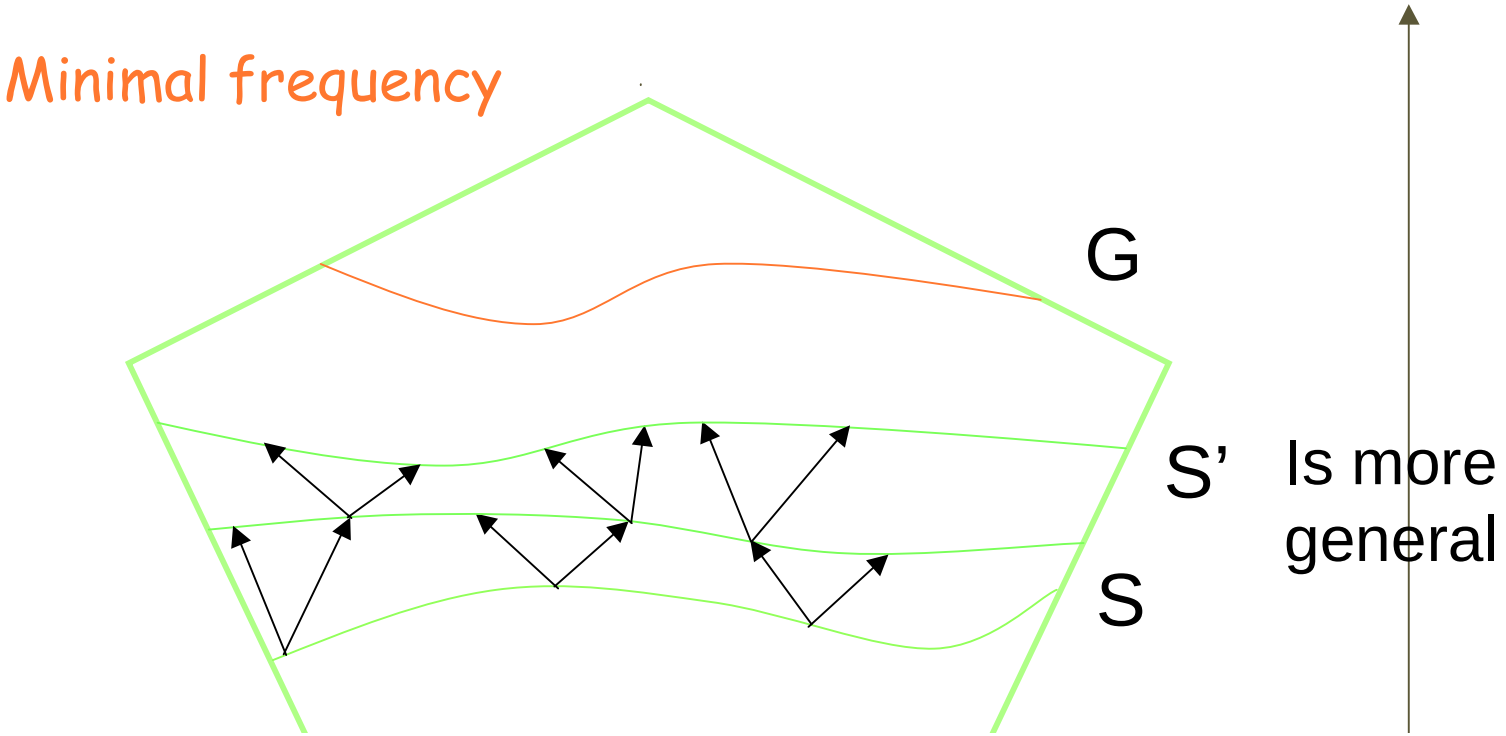


Minimum frequency constraints

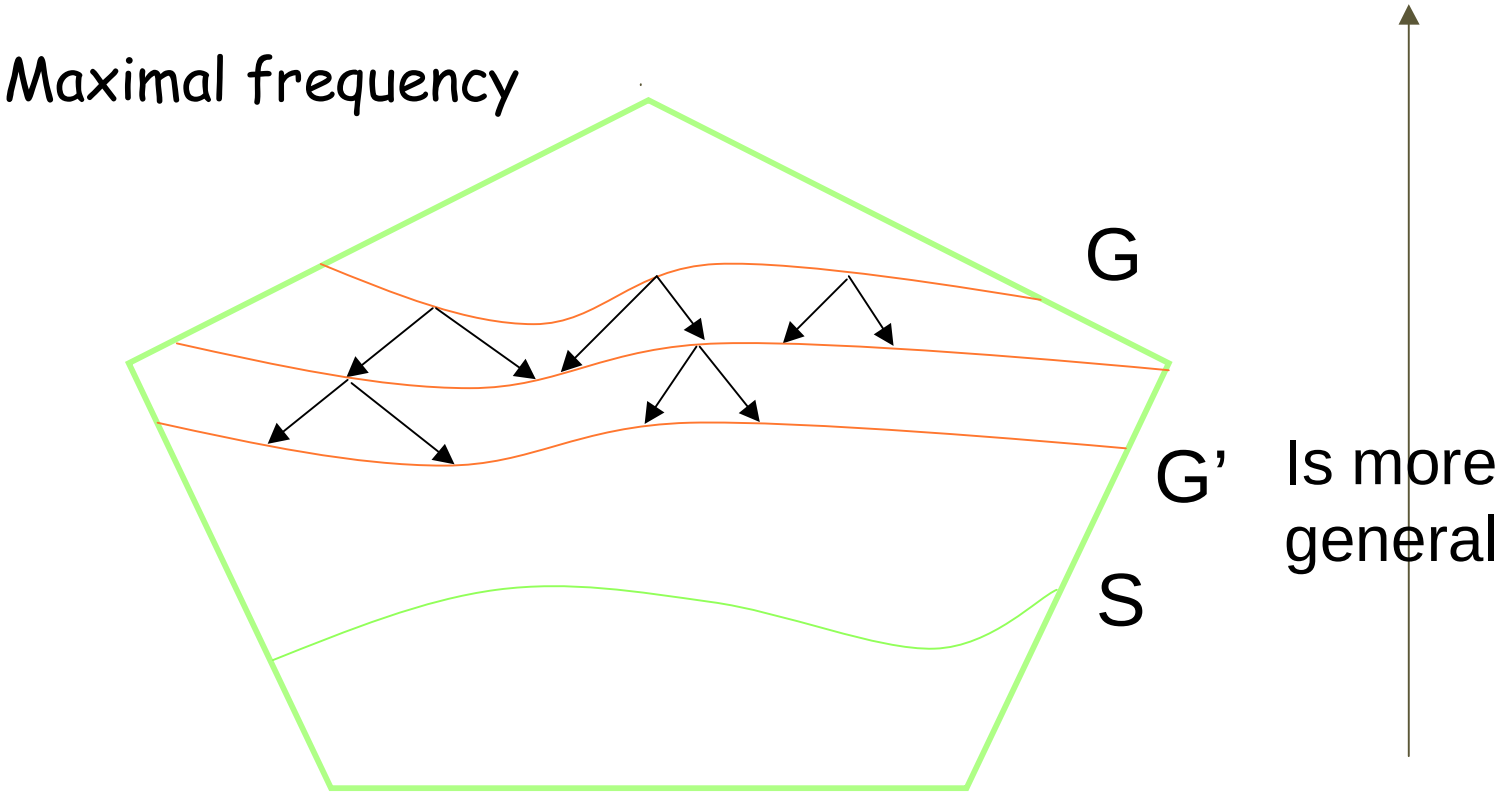
Let c be a constraint of type $freq(f, D) \geq m$
$L_0 := G$; $i := 0$
while $L_i \neq \emptyset$ **do**
    $F_i := \{p \mid p \in L_i$ *and* p *satisfies constraint* $c\}$
    $I_i := L_i - F_i$ the set of infrequent fragments considered
    $L_{i+1} := \{p \mid \exists q \in F_i : p \in \rho_s(q)$
        and $\exists s \in S : p \leq s$ *and* $\rho_g(p) \cap (\cup_j I_j) = \emptyset\ \}$
    $i := i + 1$
endwhile
$G := F_0$
$S := min(\cup_j F_j)$



Dual computation



Levelwise Version Space algorithm



Levelwise Version Space algorithm

- Dualities
  - General to specific versus Specific to general
  - Minimum / Maximum frequency
- Use refinement operators on single fragments (and check) instead of self joining two fragments
- Hashing is important
- Generalizes both description identification and levelwise algorithm

Consended representations: application to frequency queries

- $Th(L \otimes E, r, q) = \{(\phi, e) \in L \otimes E \mid q(r, \varphi) \text{ is true}\}$
- Other types of condensed representatoins
  - Requires e (e.g. frequency) to be known or approximated !
  - In version spaces E/e is not used
  - Based on closedness concept

Constraints on itemsets

- C_{freq} (S) — See e.g., Ng & al. 98 (sigmod)
- $A \notin S$ — $A \in S$
- $\{A,B,C,D\} \supset S$ — $\{A,B,C,D\} \subseteq S$
- $S \cap \{A,B,C\} = \emptyset$ — $S \cap \{A,B,C\} \neq \emptyset$
- sum(S.Price) $\leq$ v — sum(S.Price) > v
- $\nexists A \in 2^n$, Interest(A) > Interest (S)

Primitive constraints based on closures

The "closure" evaluation function

- The closure of X is the maximal superset of X that has the same frequency

closure (X,r) = Items (Object (X,r),r)

A	B	C	D
1	0	1	0
1	1	1	0
0	1	1	1
0	1	0	1
1	1	1	0

closure{*A*} ={*A*,*C*}

Closed sets

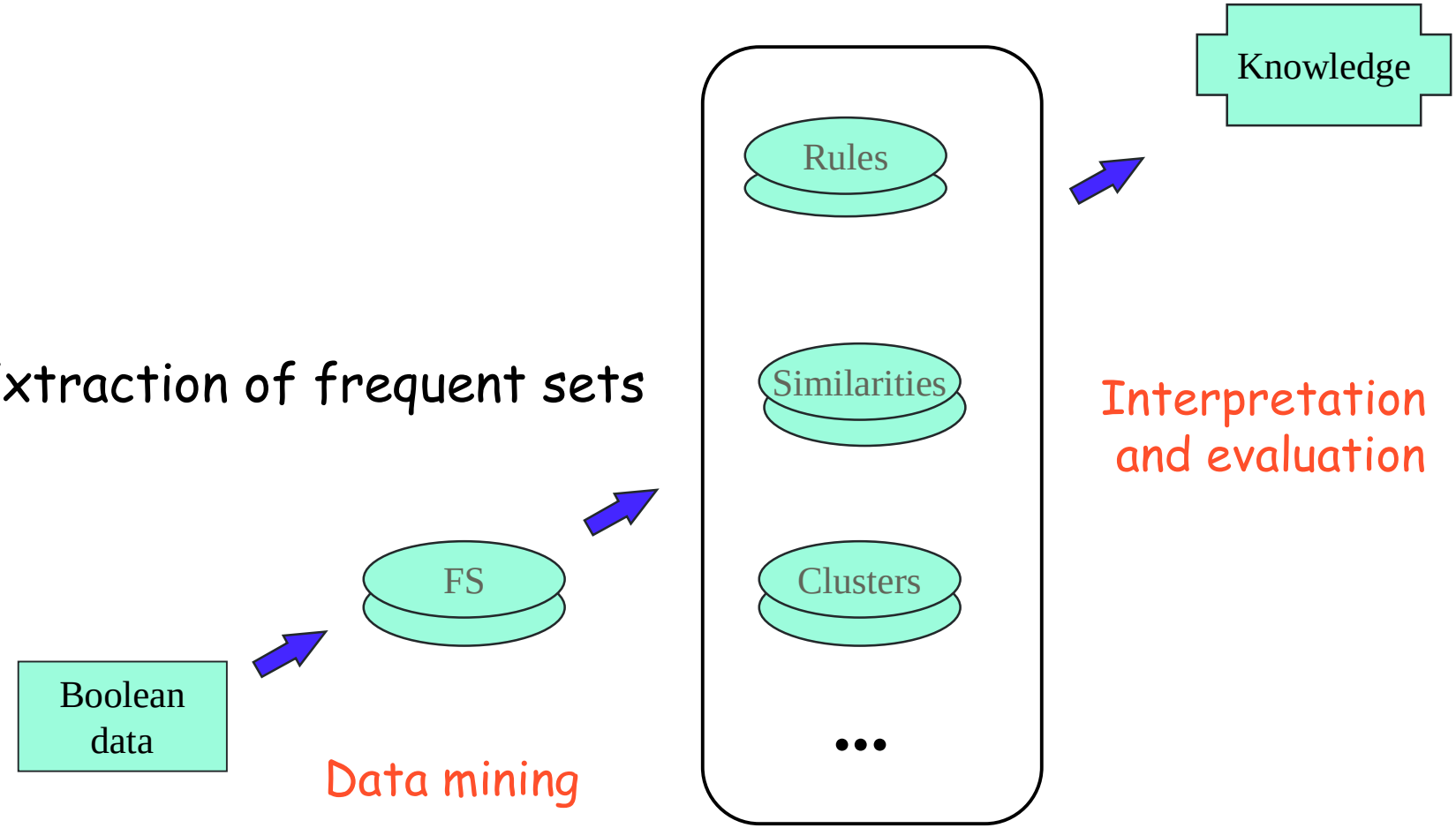
- A closed set is equal to its closure. It is a maximal set of items that support the same transactions

A	B	C	D
1	0	1	0
1	1	1	0
0	1	1	1
0	1	0	1
1	1	1	0

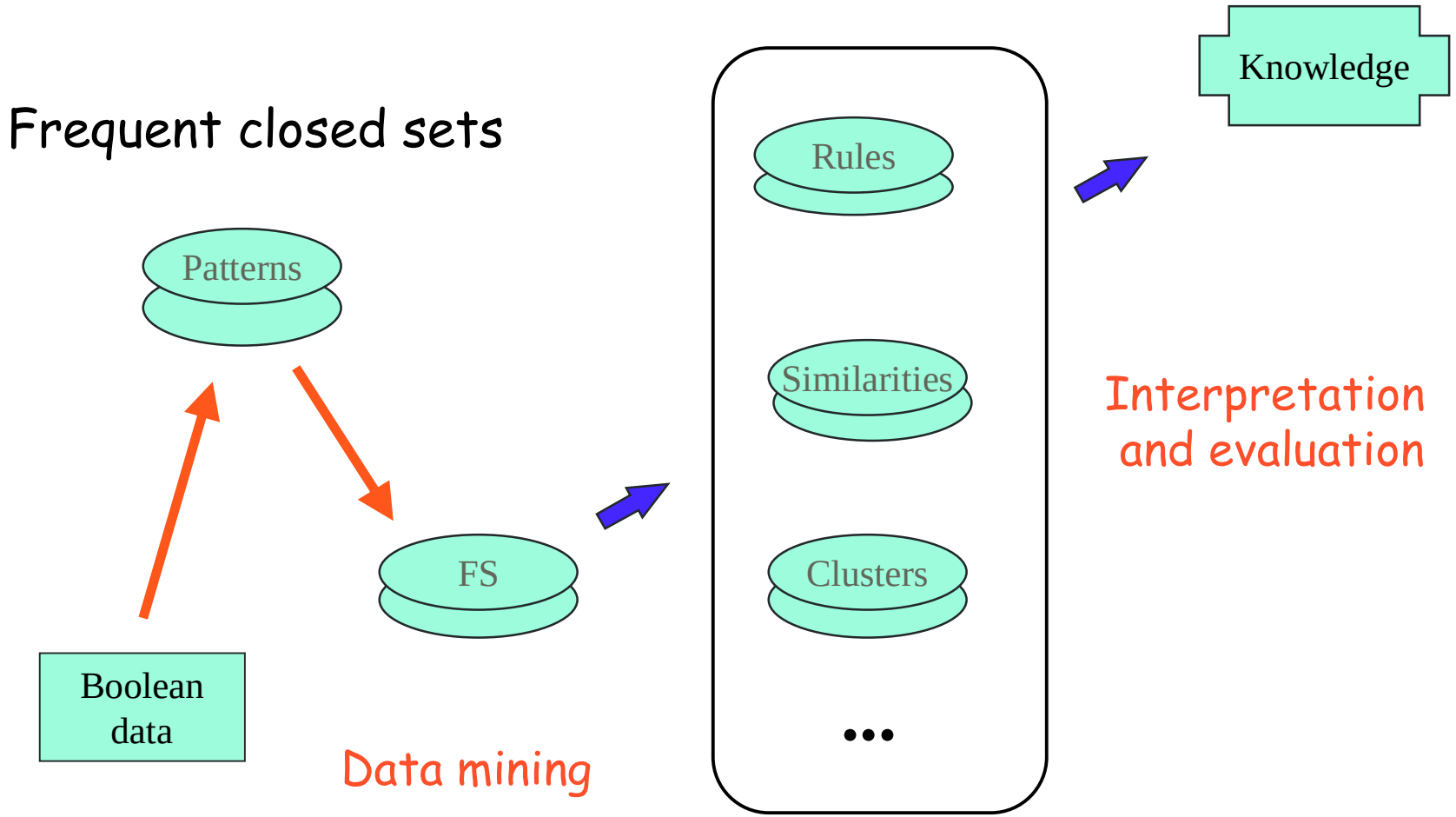
{A,C} is closed {A,B} is not closed

$c_{Close}(S)$

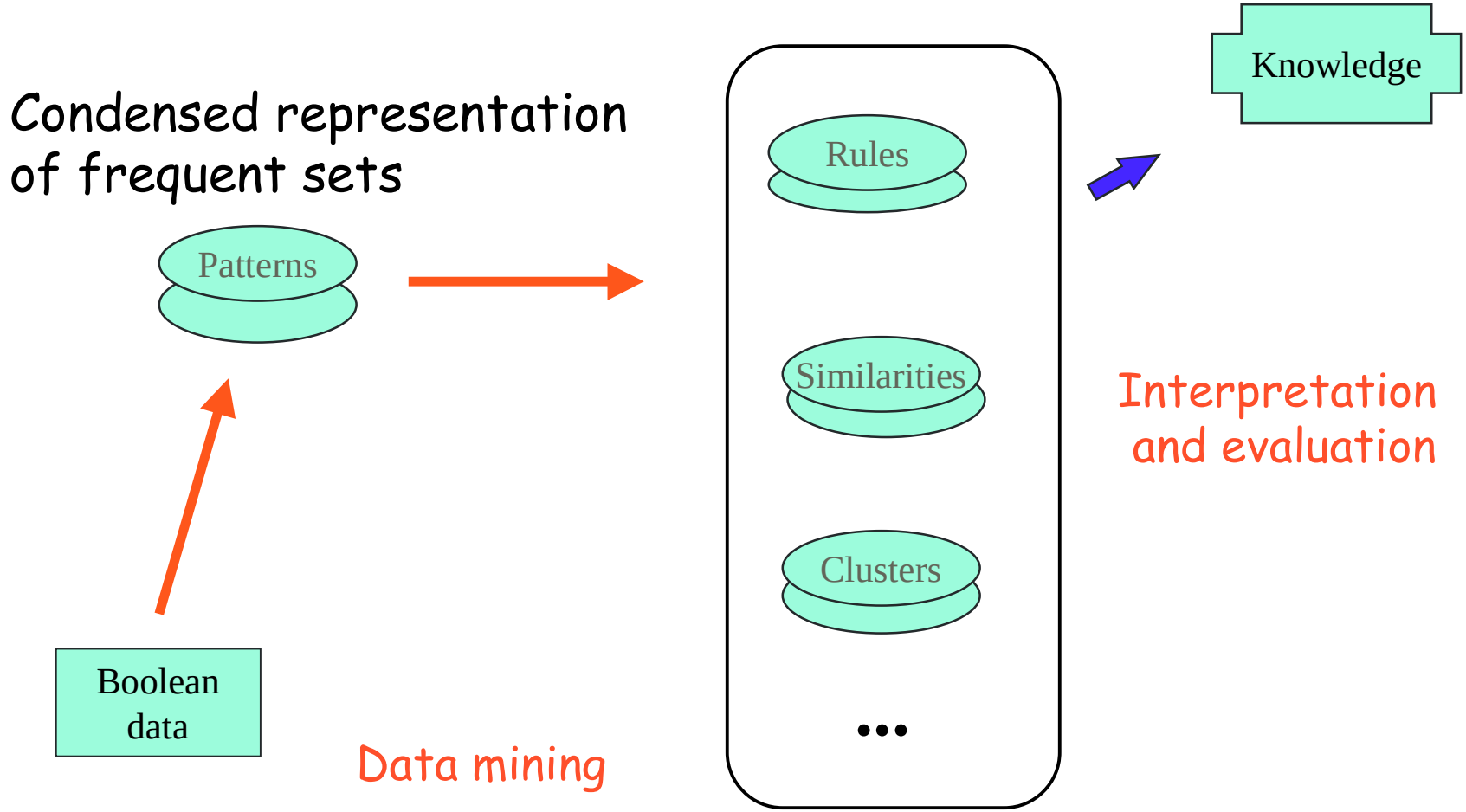
... introducing condensed representations



Frequent set mining in difficult cases



An up-to-date view



ε-adequate representations

- Assume the class of queries that returns the frequency of an itemset, look for alternative representations of data on which we can provide its frequency with a precision of at most ε
  - e.g., the collection of γ-frequent sets is $\gamma/2$-adequate

    Is it possible to find smaller representations, i.e., condensed representations
- This concept is quite general Mannila & Toivonen 96 (kdd)

Condensed representations of frequent itemsets

- Maximal itemsets e.g., Bayardo 97 (sigmod) Max-Miner
- Version spaces e.g. De Raedt 01 (ijcai)
- Closed sets — Pasquier & al. 99 (icdt) - Boulicaut & Bykowski 00 (pakdd) - Han & Pei 00 (wdmkd) - Zaki 00 (sigkdd) Close - Closet - Charm
- Free sets — Boulicaut & al. 00 (pkdd) - Bastide & al. 00 (sigkdd explorations) Min-Ex - Pascal
- δ-free sets
- $\vee$-free sets — Bykowski & Rigotti 01 (pods) - Kryskiewicz 01 (icdm)
- Extraction complexity vs. compacity vs. accuracy

Apriori vs. Close

Dataset/ Frequency threshold	Time in sec.	$\|\|FS_\sigma\|\|$	Scans	Time in sec. (1^{st}/2^{nd} step)	$\|\|FC_\sigma\|\|$	Scans
ANPE/σ=0.05	1 463.9	25 781	11	69.2 / 6.2	11 125	9
Census/σ=0.05	7 377.6	90 755	13	61.7 / 25.8	10 513	9
ANPE/σ=0.1	254.5	6 370	10	25.5 / 1.1	2 798	8
Census/σ=0.1	2 316.9	26 307	12	34.6 / 6.0	4 041	9
ANPE/σ=0.2	108.4	1 516	9	11.8 / 0.2	638	7
Census/σ=0.2	565.5	5 771	11	18.0 / 1.1	1 064	9

Freeness

- A free-set is such that there is no logical rules that holds between its subsets

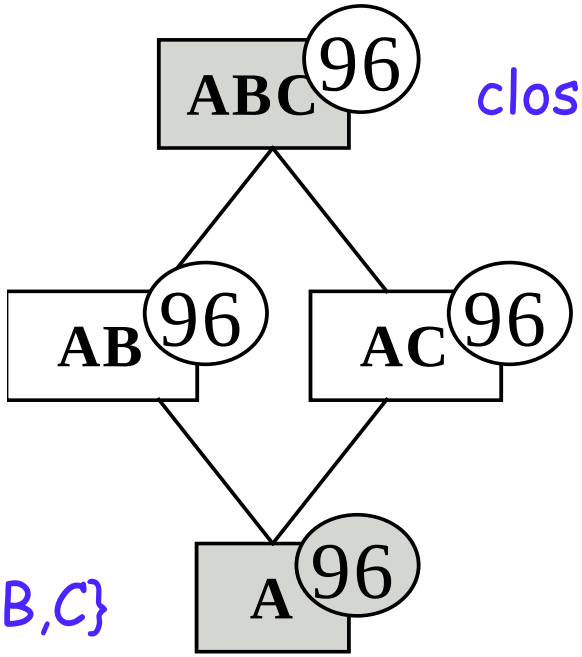
A	B	C	D
1	0	1	0
1	1	1	0
0	1	1	1
0	1	0	1
1	1	1	0

{A,B} is free {A,C} is not free

$C_{Free}(S)$ checking freeness ?

- Closed sets are the closures of free sets

Free and closed sets



closure({ABC})={ABC}

closure({A})={A,B,C}

δ-freeness

- A δ-free-set is such that there is no δ-strong rules that holds between its subsets

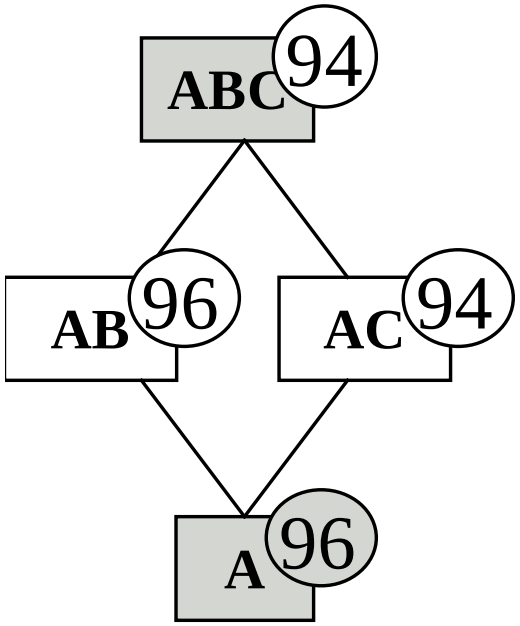
  $X \Rightarrow_{\delta} Y$ is δ-strong if it has at most δ exceptions

A	B	C	D
1	0	1	0
1	1	1	0
0	1	1	1
0	1	0	1
1	1	1	0

{A,B} was free but is not 1-free

$C_{\delta\text{-Free}}(S)$ checking δ-freeness ?

An example of a 2-free sets



$B, C \in \text{closure}_2(\{A\})$

Examples of condensed representations

1	ABCD
2	AC
3	AC
4	ABCD
5	BC
6	ABC

Threshold 2

16 frequent sets

1 maximal frequent set

Frequent closed sets

C, AC, BC, ABC, ABCD

Frequent free sets

∅, A, B, D, AB

Frequent 1-free sets

∅, B, D

« Approximation » from closed sets

- ε-adequate representation

  If S is not included in a γ-frequent closed set

  Then S is not frequent (return Freq(S,r) =0)

  Else S is frequent

  Let choose the frequent closed set X s.t. $S \subseteq X$ that has the maximal support and return Freq(S,r) = Freq(X,r)

Approximation from δ-free sets

- ε-adequate representation

  If S is a superset of an element from $FreeBd^-$

  Then S is not frequent (return $Freq(S,r) = 0$)

  Else S is frequent

  Let choose the frequent δ-free set $Y \subseteq X$ that has the minimal support and
  $Freq(Y,r) - Freq(X,r) \leq |X \setminus Y|\delta$

Computing frequent δ-free-sets

- Min-Ex is an effective levelwise algorithm that computes every frequent δ-free set in r
  - thanks to freeness anti-monotonicity and an effective freeness test …
- Forthcoming Ph. D thesis by A. Bykowski
- Promising experimental validation on dense datasets
  - High condensation and pruning even for low δ
  - Low error in practice even for « large » δ values

Experimental validation

- Experiment
  - PUMSB* data set (size=49046 rows), $\gamma = 0.3$
  - 432699 γ-frequent sets, the largest has N = 16 items
  - Condensed representation for $\delta = 20$, 11079 frequent δ-free-sets
  - Theoretical error bound: maximal absolute (resp. relative) support error $\delta * N = 20 * 16 = 320$ rows (resp. $\delta * N$ / size$*\gamma$ = 2.18 %
  - Practical observed error: maximal absolute (resp. relative) support error 45 rows (resp. 0.29 %), average absolute (resp. relative) support error 6.01 rows (resp. 0.037 %)

4. Where to go from here ?

- Other forms of primitives ?
  - E.g. accuracy of rule / hypotheses is larger than x
  - E.g. average cost of transaction is larger than x
  - Neither monotone nor anti-monotone
- Optimization primitives ?
  - Find item sets with maximum frequency
  - Find rule with maximum accuracy

- Other forms of tasks ?
  - Clustering (some initial works exist)
    - Formulate constraints on no. of desired clusters, and cluster membership
  - Prediction
    - Some approaches to decision tree learning exist
- Other forms of algorithms ?
  - Instead of "all solutions" find "best" or "plausible" solutions
  - Approximation/heuristic algorithms
  - Cf. constraint programming

- Other form of queries
  - Boolean inductive queries
- Query optimisation
  - E.g. Baralis and Psaila Dawak 98
- Operations on solution sets
  - E.g. version spaces
  - E.g. version space trees

Query Optimisation and Reasoning

Claim (subsumption)

Let q_1 and q_2 be two queries such that $q_1 \models q_2$.

Then $sol(q_1) \subseteq sol(q_2)$

Background knowledge can also be used in this process.

E.g. $freq(f, D) > x$ and $x \geq y \rightarrow freq(f, D) > y$

E.g. $freq(f, D1) > x$ and $D1 \subseteq D2 \rightarrow freq(f, D2) > x$

E.g. $freq(f2, D) > x$ and $f1 \leq f2 \rightarrow freq(f1, D) > x$

Useful :

axioms about sets, generality, number theory

Subsumption is useful in the light of interactive querying

and reuse of the results of previous queries

Memory organisation

- Consider
  - q1 : freq(f,D) > m
  - q2: freq(f,D U M) > m (q1 |= q2)
  - q3:freq(f,D) > m OR freq(f,M) > m (q3 |= q2)
- Scenario's
  - q1 answered and stored; q2 asked
  - q2 answered and stored; q1 asked
- Keep track of subset relations among pattern sets / data sets
- Keep track of relations among patterns (generality – lattice structure) within given pattern set
- Operations on solution sets ? On border sets ?

Boolean Inductive Queries

Any monotonic or anti-monotonic constraint c,

and any membership function (e.g. $f \in P$)

is an atom.

An inductive query is a boolean formula over atoms.

E.g. $(f \in P)$ and $[freq(f, D1) > x$ or $freq(f, D2) < y]$ and $f < abbbcccc$

The query evaluation problem

Given

an inductive database

an inductive query q

Find a characterisation of $sol(q)$

So far : solutions for conjunction of anti-monotonic and monotonic

Query Evaluation

Theorem

Let q be and inductive query.

Then $sol(q)$ can be represented using a set of versionspaces

(a set of versionspaces represents the union of the versionspaces)

Proof

Write q in Disjunctive Normal Form, i.e.

in the form of disjunction of conjunctions of the form $a_1 \wedge ... a_k \wedge m_1 \wedge ... m_n$

Each conjunction corresponds to a versionspace

$sol(q)$ can be represented using disjunctive versionspace (Cf. Gunther Sablon)

Query Evaluation

Theorem

Let q be and inductive query.

Then $sol(q)$ can be represented using a set of versionspaces

(a set of versionspaces represents the union of the versionspaces)

Proof

Write q in Disjunctive Normal Form, i.e.

in the form of disjunction of conjunctions of the form $a_1 \wedge ... a_k \wedge m_1 \wedge ... m_n$

Each conjunction corresponds to a versionspace

$sol(q)$ can be represented using disjunctive versionspace (Cf. Gunther Sablon)

Divide and conquer approach

To evaluate/solve a query

rewrite in DNF

for each conjunct in DNF

call level wise version space algo.

Query Optimisation

Claim

Let q_1 and $q_2$be two queries that are logically equivalent.

Then $sol(q_1) = sol(q_2)$

Using logical rewrites to optimize the mining process.

E.g. $(a_1 \vee a_2) \wedge (m_1 \vee m_2)$ is logically equivalent to

$$(a_1 \wedge m_1) \vee (a_2 \vee m_1) \vee (a_1 \wedge m_2) \vee (a_2 \vee m_2)$$

One versionspace versus the disjunction of four

What is best ?

Operations on solution spaces

- Logical operations on primitives have a set oriented counter part ?
- E.g. q1 or q2 corresponds to sol(q1) U sol(q2)
- What can we say about the corresponding operations on solution sets ?
  - Analogy with relational database
  - We assume solution sets are version spaces
  - Version spaces closed under intersection but not for union ! Difference ?

Version space union

Let $sol(q_1)$ and $sol(q_2)$ be boundary set representable, i.e. representable using a versionspace.

Then in general $G(q_1 \vee q_2) \neq G(q_1) \vee G(q_2)$ and

$$S(q_1 \vee q_2) \neq S(q_1) \vee S(q_2)$$

Counter Example

A C

AB ACAC CD

ABC ACD

Acknowledgments

cInQ consortium

- INSA Lyon
- University of Torino
- Politecnico di Milano
- Albert-Ludwigs University Freiburg
- Nokia Research Center Helsinki
- Institute Jozef Stefan