Enumération de motifs intéressants dans les bases de données : application à la santé et à l'environnement

Jean-Marc Petit[1] Maguelonne Teisseire[2]

[1]**Université de Lyon, CNRS**
INSA Lyon, LIRIS UMR5205
[2]**CEMAGREF, Montpellier**

19 mai 2010
Ecole d'été MASSES DE DONNÉES DISTRIBUÉES,
Les Houches, 17-21 mai 2010

Avant propos

Focus sur la découverte de motifs intéressants :

- motifs fréquents
  ⇨ Application au BI (e.g. problème du "panier de la ménagère"),
- motifs séquentiels fréquents
  ⇨ Application en bilogie (e.g. séquences génomiques)
- dépendances fonctionnelles ou d'inclusion
  ⇨ Application à la compréhension des BD existantes (e.g. self-tuning logique des BD)
- sous-arbres fréquents dans des forests
  ⇨ Application au Web (e.g. fragments de documents XML), etc.

Première partie :

- Focus sur les problèmes dits "représentables par des ensembles"

Seconde partie :

- Autres types de problèmes d'énumération
- Applications à la santé et à l'environnement

Première partie

Intervenants : Jean-Marc Petit
Responsable de l'équipe `Base de données` du LIRIS
Prof. à l'INSA de Lyon

Travail joint avec : Marie Agier, Fabien De Marchi, Frédéric Flouvat

Pattern mining problems

Examples :

- frequent itemsets (and variants), sequences, trees, graphs
  ...
- functional, inclusion, multivalued dependencies
- learning monotone function
- minimal transversals of hypergraph

⇨ A wide class of problems, some being studied for years in combinatorics, artificial intelligence and databases

Main issue : scalability wrt the size of the data

Motivations

Data mining research in this (sub-)area ?
⇨ most of the time, ad-hoc solutions (with customized data structures)

- Can be seen as a competition to devise (low-level) code (to beat previous implementations)
- I/O routines sometimes as important as algorithmic strategies !

For one problem common to many applications, one solution per application !

- efficient low level code **very** difficult to reuse
- a slight change in the problem statement (data, pattern or predicate) often means to re-start development from scratch

Objectives

1. Allowing a wider dissemination of data mining techniques
   - ⇨ Development time of algorithms should be reduced
   - ⇨ Low-level details should be hidden to developers
   - ⇨ Efficient and scalable implementations

⇨ Identifying some subclasses of pattern mining problems

⇨ Pushing forward **declarative approaches** in this context

Main trends for declarative approaches in data mining

- Inductive logic programming [Wro00, NR06]
- Constraint programming [RGN08]
- Databases [IM96, Cha98, CW01]

Many attempts, not very successful yet

Compromise to be found between many opposite goals : genericity, efficiency, easy of use, seamless integration with SQL . . .

In this course

A first step toward a declarative approach for a well-defined class of pattern mining problems

- Which class of problems ?
  ⇨ pattern mining problems said to be “representable as sets” [MT97]
- For what ?
  ⇨ Being able to re-use both known “set-oriented” algorithms and set-oriented data structures (to handle large collection of sets)
  ⇨ should easier the development of new instance of such problems
  ⇨ should be efficient enough and scalable
- iZi Library [FDP09]

Main related contributions : DMTL of M. Zaki Team [CHSZ08]

About the content of this course

- not an introduction to some declarative language for pattern mining problems : remains an open problem
- iZi is a first step ... but still remains programmer-dependent
  - lack of a declarative language
  - no optimization possible by some query processor

Outline

Outline

Notations

Mainly from (Mannila and Toivonen, DMKD, 1997) [MT97]

Consider the following framework :

1. Let $\mathcal{D}$ be a database
2. Let $\mathcal{L}$ be a set of patterns (or a finite language)
3. Let $\mathcal{P}$ be a predicate to qualify interesting patterns X in $\mathcal{D}$, noted $\mathcal{P}(X, \mathcal{D})$

Definition (Problem statement P)

Given $\mathcal{D}$, $\mathcal{L}$ and $\mathcal{P}$, enumerate all interesting patterns of $\mathcal{L}$ in $\mathcal{D}$

In other words, enumerate the set
$Th(\mathcal{D}, \mathcal{L}\mathcal{P}) = \{X \in \mathcal{L} \mid \mathcal{P}(X, \mathcal{D}) true\}$

`Exercises` : Instances of such a framework ?

Sometimes, $\mathcal{D}$ is made up of patterns of $\mathcal{L}$
Without any other knowledge, how to solve P ?

Notations

Mainly from (Mannila and Toivonen, DMKD, 1997) [MT97]

Consider the following framework :

1. Let $\mathcal{D}$ be a database
2. Let $\mathcal{L}$ be a set of patterns (or a finite language)
3. Let $\mathcal{P}$ be a predicate to qualify interesting patterns X in $\mathcal{D}$, noted $\mathcal{P}(X, \mathcal{D})$

Definition (Problem statement P)

Given $\mathcal{D}$, $\mathcal{L}$ and $\mathcal{P}$, enumerate all interesting patterns of $\mathcal{L}$ in $\mathcal{D}$

In other words, enumerate the set
$Th(\mathcal{D}, \mathcal{L}\,\mathcal{P}) = \{X \in \mathcal{L} \mid \mathcal{P}(X, \mathcal{D}) true\}$

`Exercises` : Instances of such a framework ?

Sometimes, $\mathcal{D}$ is made up of patterns of $\mathcal{L}$
Without any other knowledge, how to solve P ?

Outline

Structuring the search space (1/2)

Sometimes some order may exist among patterns, i.e. a specialization/generalization relation

4 Let $\preceq$ be a partial order on $\mathcal{L}$

$X \preceq Y$: X generalizes Y and Y specializes X

`Exercises` : What are the possible partial orders for sets ? for sequences ? for trees ? for inclusion dependencies ?

Structuring the search space (2/2)

Influence of the partial order on the predicate ?
The most studied property in data mining : monotonic property

Definition

$\mathcal{P}$ is said to be monotone with respect to $\preceq$ if for all $X, Y \in \mathcal{L}$ such that $X \preceq Y$, we have : $\mathcal{P}(Y, \mathcal{D})true \Rightarrow \mathcal{P}(X, \mathcal{D})true$

`Exercises` : Let A be a set of articles, ϵ a user-defined threshold, $\mathcal{D}$ a transactional database, $\mathcal{L} = 2^A$ and $\mathcal{P}(X, \mathcal{D})$ defined as : $\mathcal{P}(X, \mathcal{D})$ true wrt ϵ iff $card(\{t \in \mathcal{D} | X \subseteq t\}) \geq \epsilon$
$\mathcal{P}(X, \mathcal{D})$ is monotone wrt $\subseteq$?

Nevertheless, a predicate is not restricted to frequency : for example, the satisfaction of a functional dependency in a relation is a predicate

Equivalent problem statements

Two (complementary) notions emerges : the positive and negative borders, i.e. the most specialized interesting patterns and the most generalized non interesting patterns

Definition (New problem statement P')

Given $\mathcal{D}$, $\mathcal{L}$ and $\mathcal{P}$, enumerate positive (or negative) border of interesting patterns of $\mathcal{L}$ in $\mathcal{D}$

In other words, enumerate the sets :

$$bd^{+}(\mathcal{D},\mathcal{L},\mathcal{P},\preceq) = \{X \in Th \mid \nexists Y \in \mathcal{L}\,(X \preceq Y \Rightarrow Y \in Th)\}$$
$$bd^{-}(\mathcal{D},\mathcal{L},\mathcal{P},\preceq) = \{X \in \mathcal{L} \mid X \notin Th, \forall Y \in \mathcal{L}(Y \preceq X \Rightarrow Y \in Th)\}$$

Now, how to solve P' ?

Outline

Example of frequent itemset mining (FIM)

Continuing the previous example :
$\mathcal{D}$ a transactional database (a bag of patterns)
$\mathcal{L} = 2^A$
$\preceq = \subseteq$
$\mathcal{P}(X, \mathcal{D})$ monotone wrt $\subseteq$

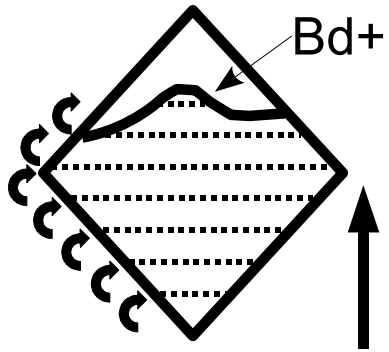
So many contributions for FIM ... cf FIMI, OSDM workshops

Main algorithmic contributions :

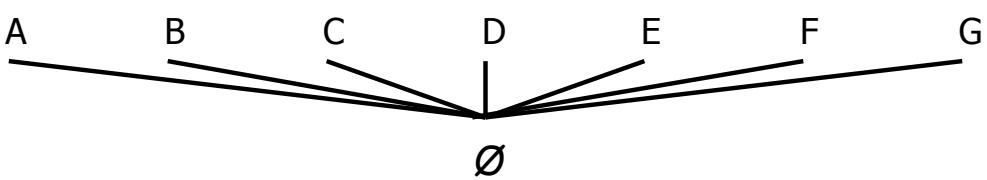
- Apriori levelwise search : **candidate generation**
- **Relationship between borders**
- Specialized data structures to optimize the counting operation, to compress the database ...

Levelwise strategy

Levelwise search

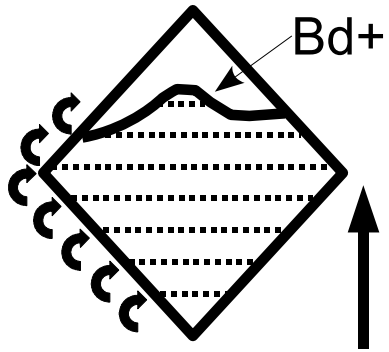


Pruning strategy : based on the monotonicity property

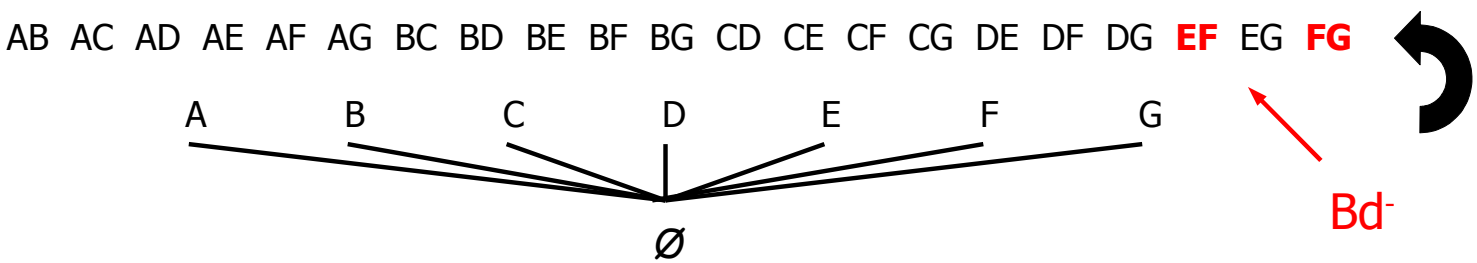


Levelwise strategy

Levelwise search

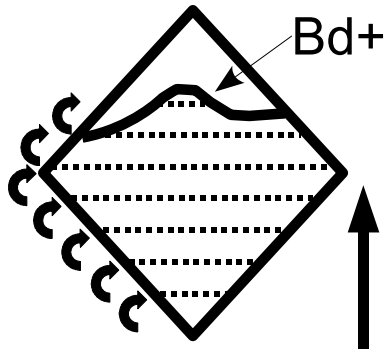


Pruning strategy : based on the monotonicity property

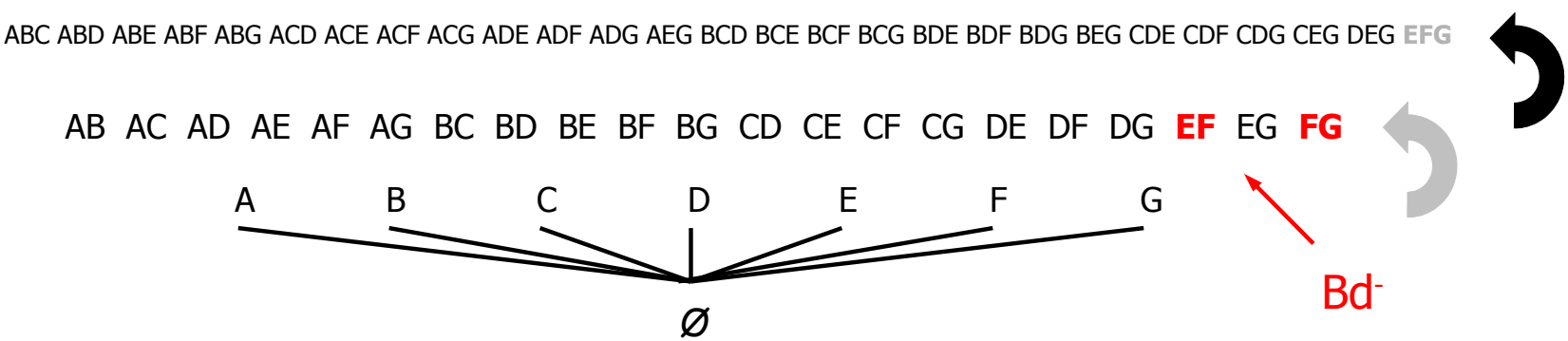


Levelwise strategy

Levelwise search

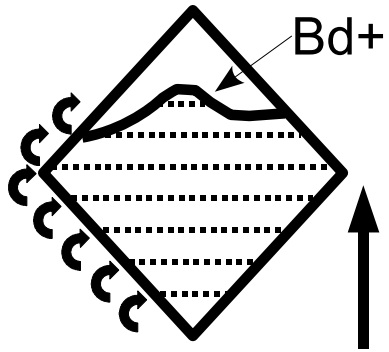


Pruning strategy : based on the monotonicity property

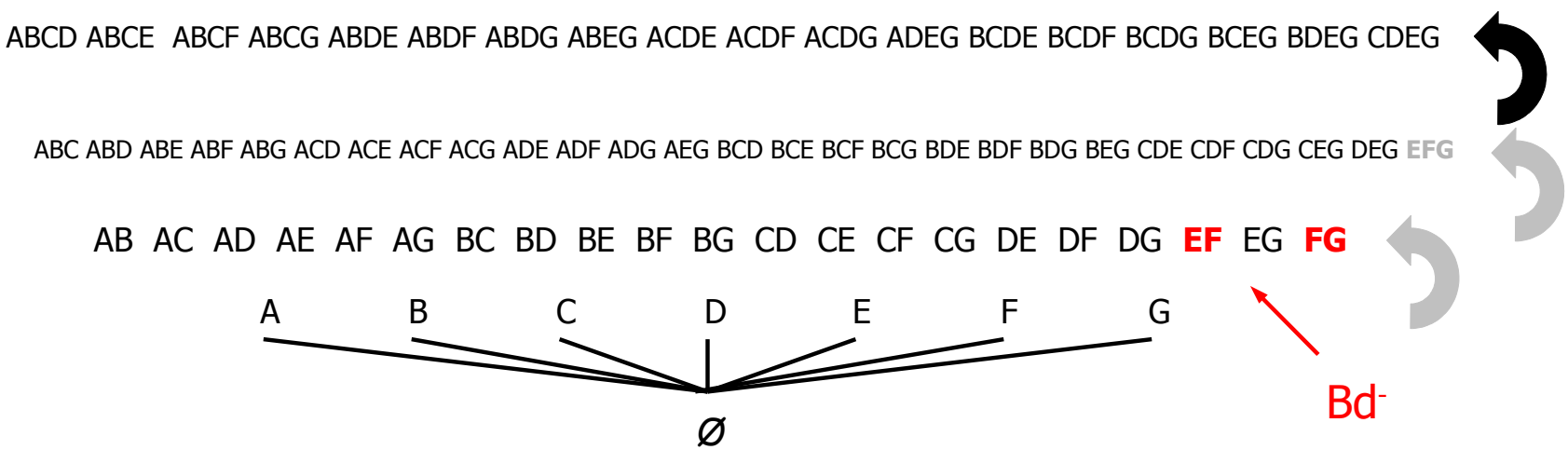


Levelwise strategy

Levelwise search

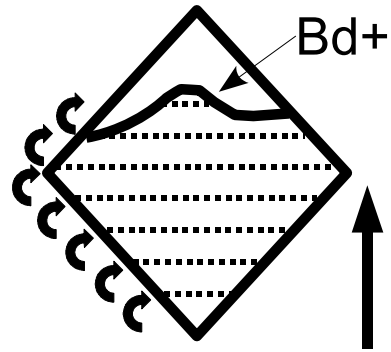


Pruning strategy : based on the monotonicity property

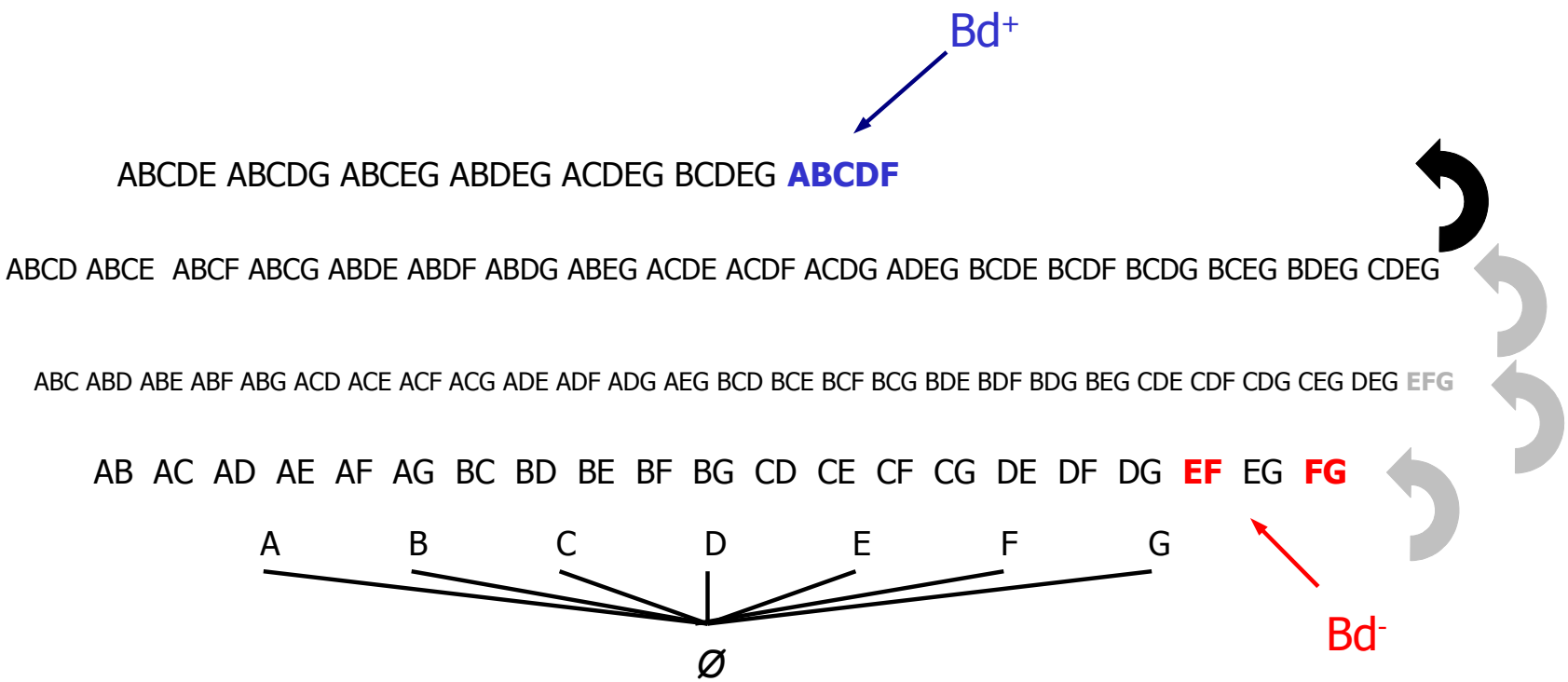


Levelwise strategy

Levelwise search

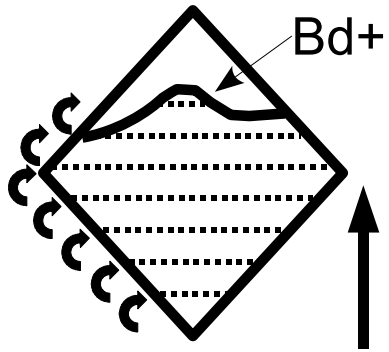


Pruning strategy : based on the monotonicity property

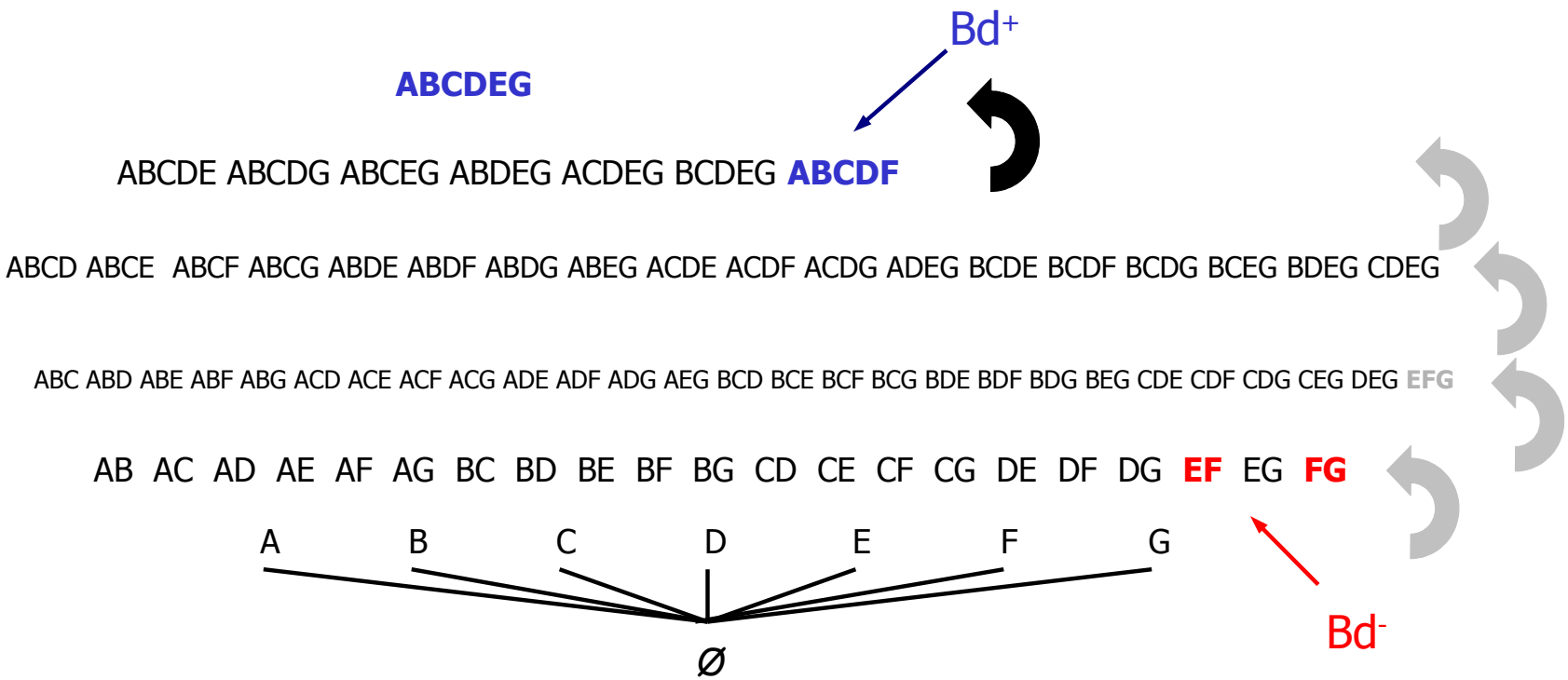


Levelwise strategy

Levelwise search



Pruning strategy : based on the monotonicity property



Outline

Last restriction : Isomorphism with a boolean lattice

Basic idea : patterns that can be “translated” as elements of the powerset of some set and inversely.

5 For some finite set E, a function f from $\mathcal{L}$ to 2^E has to exist such that :
- f^{-1} is computable
- f bijective
- f preserves the partial order, i.e. $X \preceq Y \Leftrightarrow f(X) \subseteq f(Y)$

⇨ Quite restrictive assumption

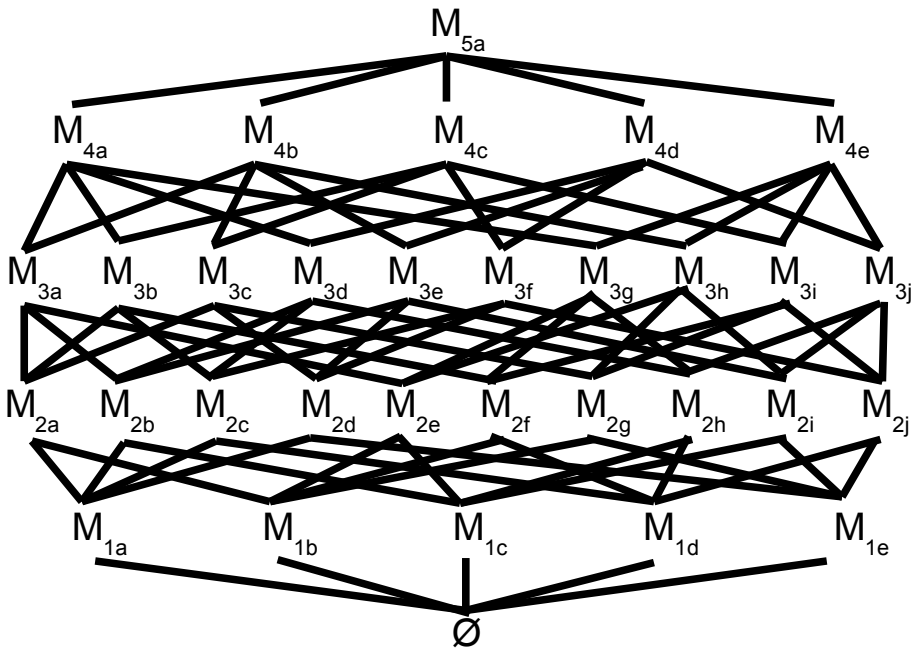
Main interests

For any problem complying with the 5 components of the framework :

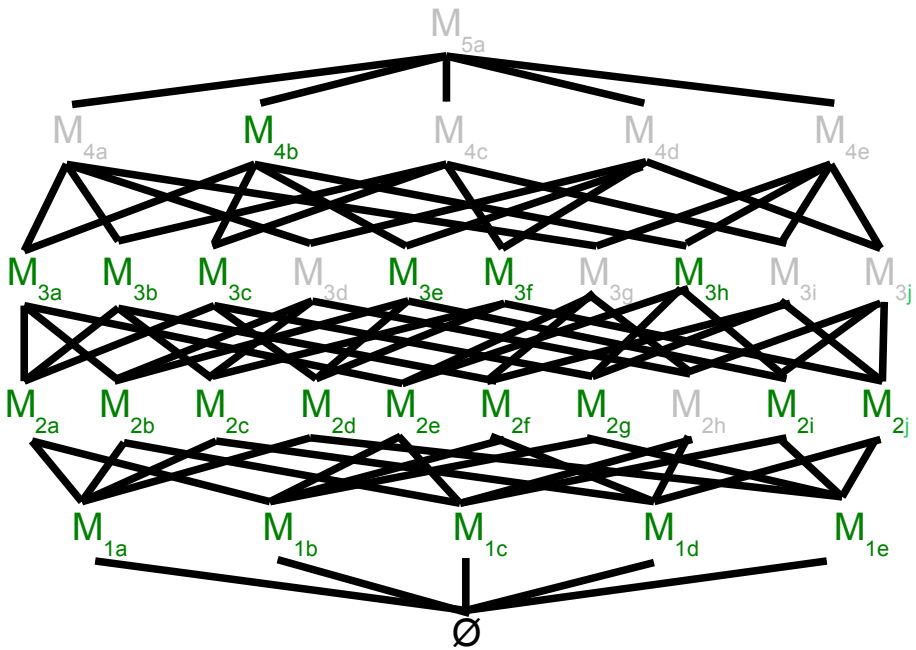
1. Clear separation between DB accesses for predicate evaluation and candidate enumerations on patterns
2. Set oriented algorithms can be used everywhere
   - 2.1 candidate generation in levelwise algorithms
   - 2.2 relationship between borders : notion of dualization (minimal transversal enumeration in an hypergraph)
3. Same algorithm principles can be applied to every problem

Outline

Synthesis


M5a
M4a M4b M4c M4d M4e
M3a M3b M3c M3d M3e M3f M3g M3h M3i M3j
M2a M2b M2c M2d M2e M2f M2g M2h M2i M2j
M1a M1b M1c M1d M1e
Ø

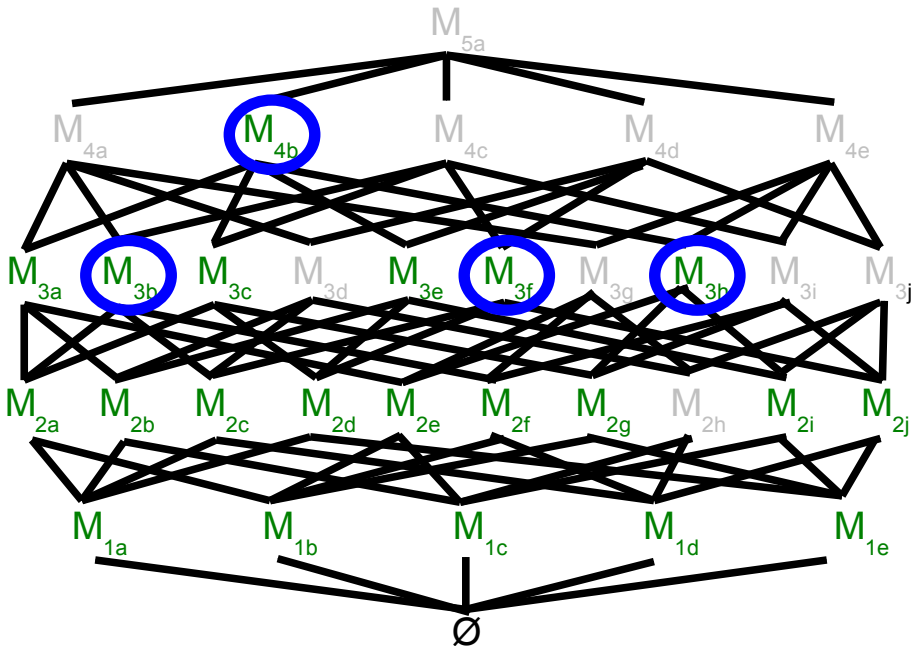
Synthesis



Solutions

- all **patterns**, i.e. every interesting pattern

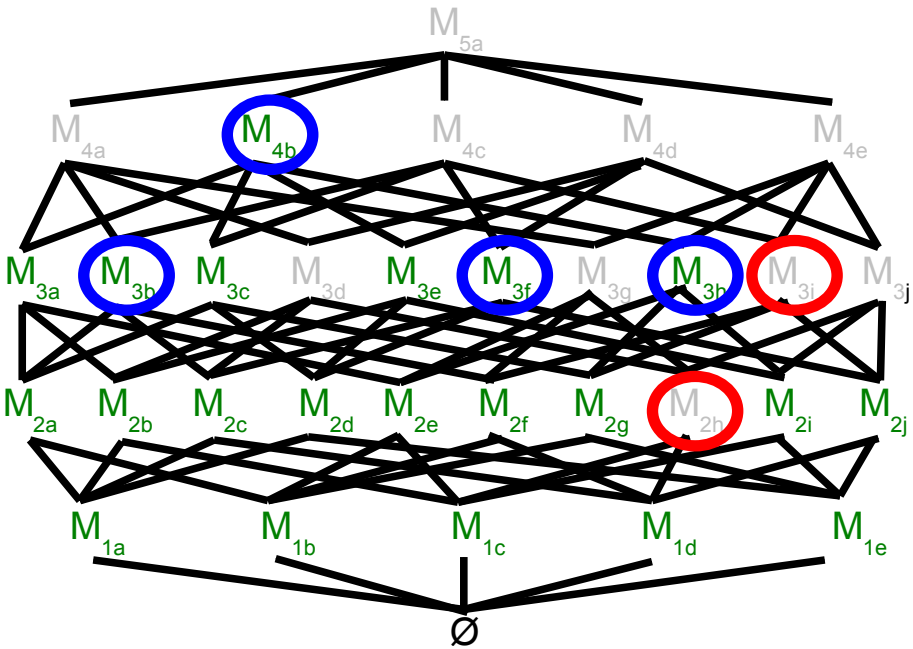
Synthesis



Solutions

- all **patterns**, i.e. every interesting pattern
- **positive border** Bd^+ : most specific interesting patterns

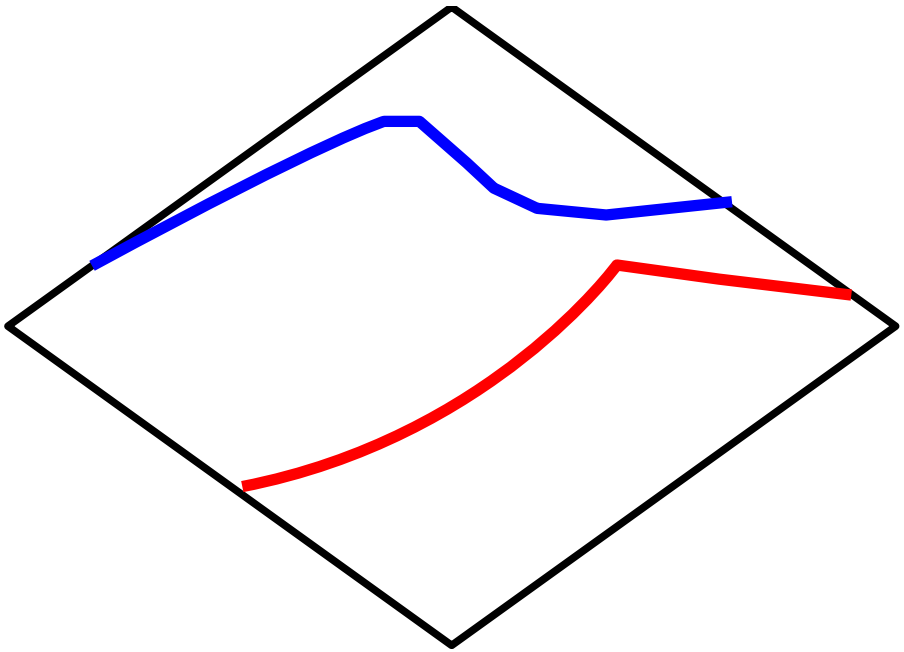
Synthesis



Solutions

- all **patterns**, i.e. every interesting pattern
- **positive border** Bd^+ : most specific interesting patterns
- **negative bordure** Bd^- : most general non interesting patterns

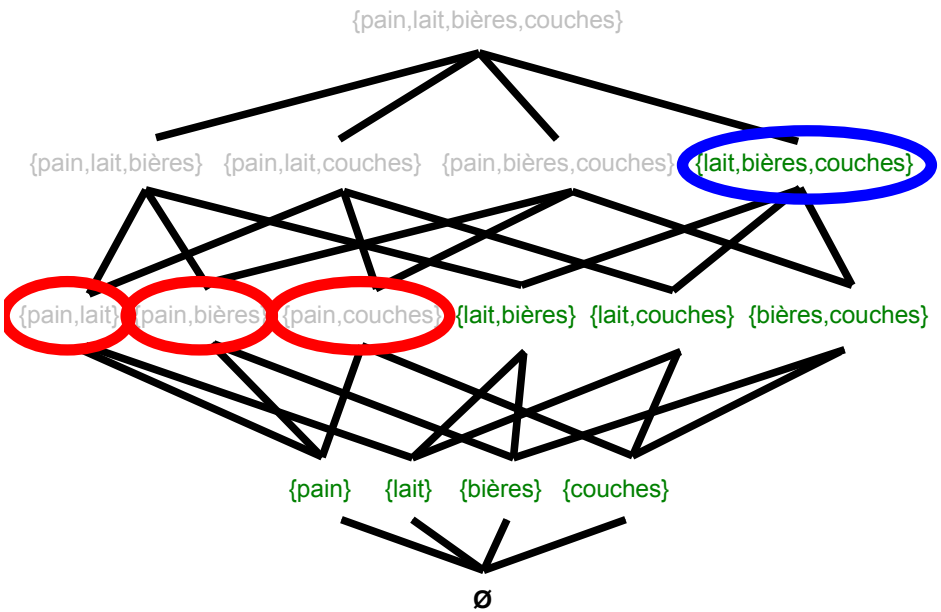
Synthesis



Solutions

- all **patterns**, i.e. every interesting pattern
- **positive border** Bd^+ : most specific interesting patterns
- **negative bordure** Bd^- : most general non interesting patterns

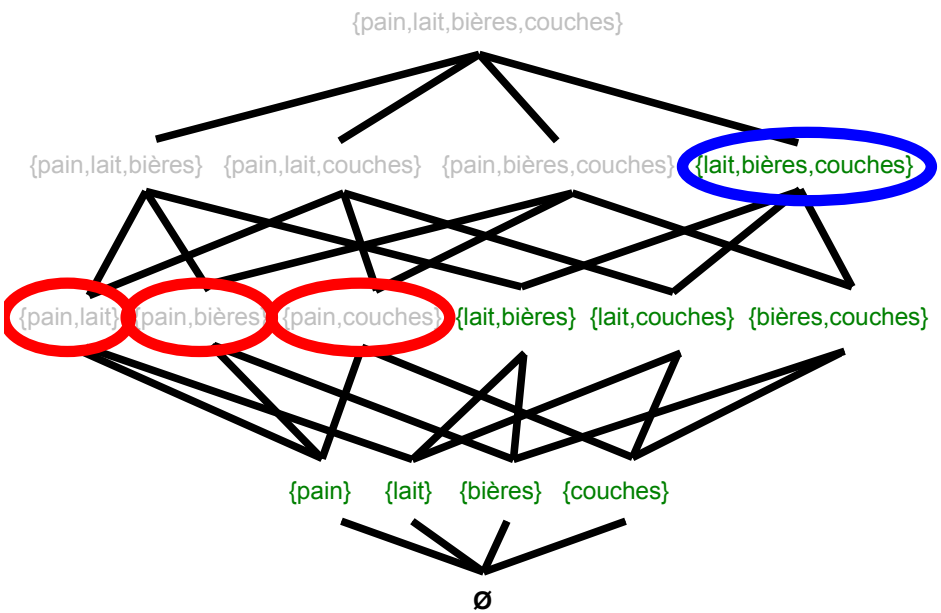
Synthesis



Solutions

- all **patterns**, i.e. every interesting pattern
- **positive border** Bd^+ : most specific interesting patterns
- **negative bordure** Bd^- : most general non interesting patterns

Synthesis



Solutions

- all **patterns**, i.e. every interesting pattern
- **positive border** Bd^+ : most specific interesting patterns
- **negative bordure** Bd^- : most general non interesting patterns

⇨ Exponential search space

⇨ Pruning strategies and traversals very important

Outline

Outline

Key

Functional dependencies generalize key in DB
First, let us define the key patterns

Definition

Let R be a relation schema over some universe U
An key over **R** is a subset of U

Second, what does “interesting” means for key ?

Definition

Let r be a relation over R
An key K is *satisfied* in r, noted $r \models K$ ssi $r \models K \rightarrow R$

Let `Pred_Key` be the predicate defined as follows :

Definition

Let r be a relation and K a key.
`Pred_Key`(K, r) true if $r \models K$

Problem statement

Definition (Problem statement)

Enumerate satisfied (minimal) keys in a relation

Is there a “natural” order among keys ?
Yes ...
Proposition `Pred_Key` is monotone with respect to $\subseteq$

The transformation function is the **identity function**

New problem statement ?

Outline

Inclusion dependencies (IND) : definitions

generalize foreign key in databases (as functional dependencies generalize keys)
First, let us define the kind of IND patterns

Definition

Let **R** be database schema and $R_1, R_2 \in \mathbf{R}$
An IND over **R** is a *statement* of the form $R_1[X] \subseteq R_2[Y]$, with X and Y are distinct attribute sequences of $schema(R_1)$ and $schema(R_2)$ respectively.

IND satisfaction

Second, what does “interesting” means for IND ?

Definition

Let d a database over $\mathbf{R}$ and $r_1, r_2 \in d$ defined over $R_1, R_2 \in \mathbf{R}$ respectively.
An IND $R_1[X] \subseteq R_2[Y]$ is *satisfied* in d, noted
$d \models R_1[X] \subseteq R_2[Y]$, ssi $\forall t_1 \in r_1$, $\exists t_2 \in r_2$ tq $t_1[X] = t_2[Y]$

Let `Pred_Ind` be the predicate defined as follows :

Definition

Let d be a database and i an IND.
`Pred_Ind`(i, d) true if $d \models i$.

Problem statement

Definition (Problem statement IND)

Enumerate satisfied INDs in a database

contrived decision problem associated to it already NP-hard (Kantola and al., IJIS, 92) [KMRS92]

Is there a “natural” order among INDs ?

Partial order among INDs

Casanova et al. axioms (JCCS, 1984) [CFP84] :

- Reflexivity
- Projection and permutation : if $I \vdash R_1[X] \subseteq R_2[Y]$ where $X = \{A_1, \ldots, A_m\} \subseteq schema(R_1)$, $Y = \{B_1, \ldots, B_m\} \subseteq schema(R_2)$ and $i_1, \ldots, i_k$ is a sequence of natural numbers $\{1, \ldots, m\}$, then $I \vdash R_1[A_{i_1}, \ldots, A_{i_k}] \subseteq R_2[B_{i_1}, \ldots, B_{i_k}]$
- Transitivity

Definition

Let i_1 and i_2 two INDs. $i_1 \preceq i_2$ if i_1 can be obtained from i_2 using the 2nd axioms of Casanova and al.

Monotone predicate for IND

Proposition `Pred_Ind` is monotone with respect to $\preceq$
Influence on the problem statement :

Definition (Problem statement IND')

Enumerate the **positive border** of satisfied INDs in a database

From INDs to a boolean lattice

How to identify the set ?

Definition

Let E be the set of satisfied unary INDs in d, i.e.
$E = \{i \mid |i| = 1 \text{ and } d \models i\}$.
We define the function f from I to 2^E as follows :
Let $i = R_1[A_1, \ldots, A_m] \subseteq R_2[B_1, \ldots, B_m]$

$$f(i) = \{e_1, \ldots, e_n\}$$

where $e_i = R_1[A_i] \subseteq R_2[B_i]$ for all $i = 1, n$

From INDs to a boolean lattice

How to identify the set ?

Definition

Let E be the set of satisfied unary INDs in d, i.e.
$E = \{i \mid |i| = 1 \text{ and } d \models i\}$.
We define the function f from I to 2^E as follows :
Let $i = R_1[A_1, \ldots, A_m] \subseteq R_2[B_1, \ldots, B_m]$

$$f(i) = \{e_1, \ldots, e_n\}$$

where $e_i = R_1[A_i] \subseteq R_2[B_i]$ for all $i = 1, n$

From INDs to a boolean lattice

Proposition Under some restrictions on I, the set of allowed IND, f is a bijection and preserve the partial relation $\preceq$

- only two relations in a given order (e.g. $R[A] \subseteq S[C]$ and $R[B] \subseteq T[D]$)
- need an order on LHS attributes, (e.g. $\{R[A] \subseteq S[C], R[B] \subseteq S[D]\}$ gives $R[A, B] \subseteq S[C, D]$? or $R[B, A] \subseteq S[D, C]$
- need also duplicated attributes on the LHS (e.g. $f^{-1}(\{R[A] \subseteq S[C], R[A] \subseteq S[D]\}) =?$

Easy to do, the lesson is that the pattern language needs to be restricted (biais)

Definition (Problem statement IND")

Enumerate the **positive border** of satisfied INDs from r to s

Outline

iZi library

iZi stands for "`Easy` prototyping of interesting pattern mining algorithms"

Website : `http://liris.cnrs.fr/~iZi`

- Integrate different algorithmic strategies : levelwise (bottom-up or top-down) and dualization-based techniques (cf ABS)
- Prefix tree data structure to manage huge collection of sets

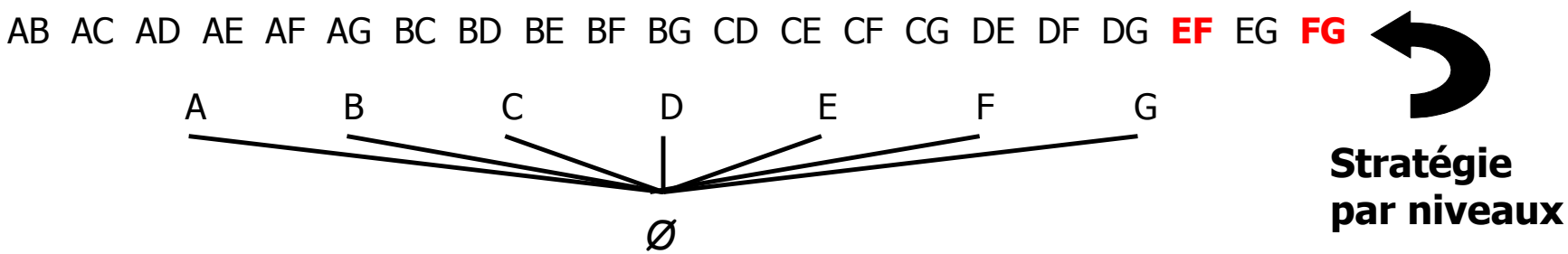
ABS algorithm (Adaptive Borders Search)

An hybrid strategy between levelwise and “dualize and advance” strategies [MT97]
Based on [DP03, FMP04]

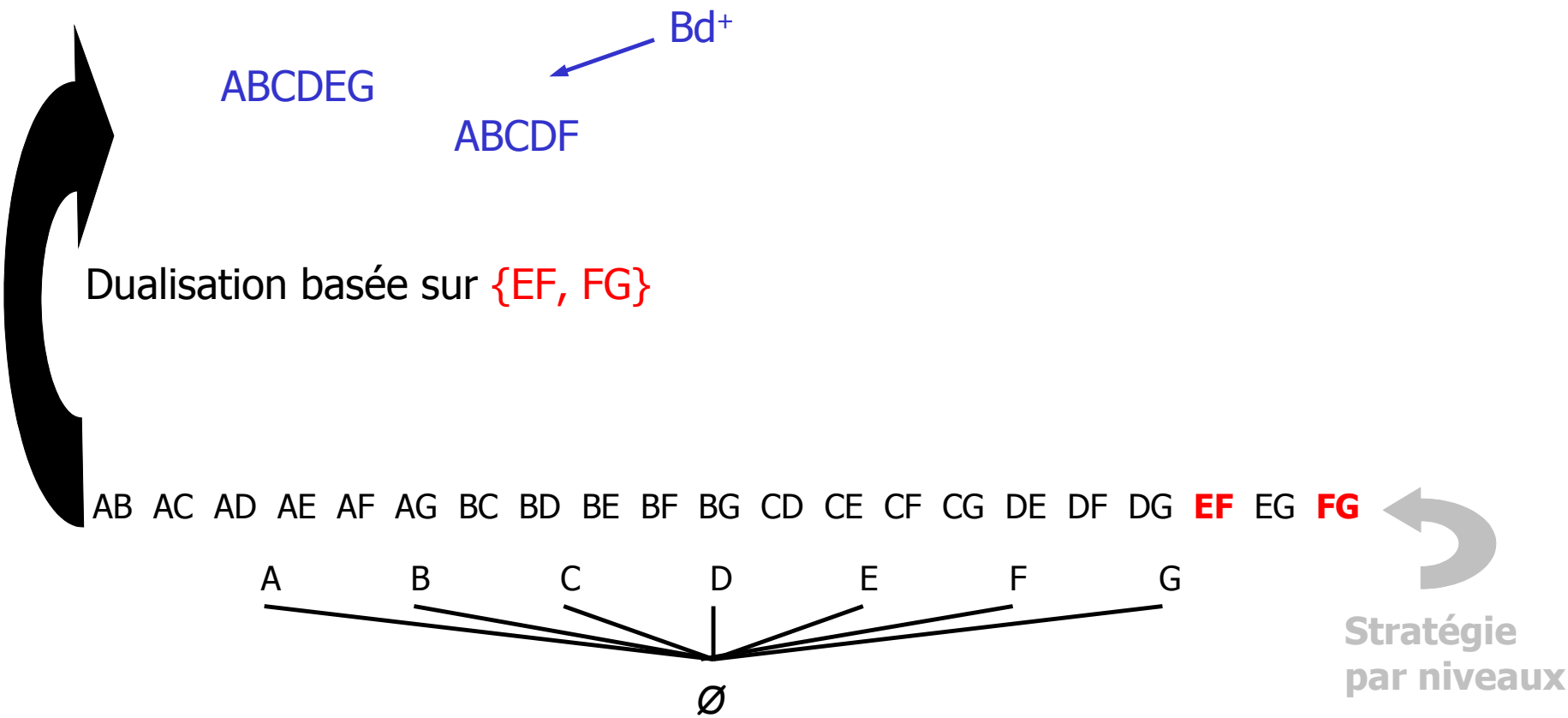
Principle :

- initialization phase : levelwise until a given level
- dualization until convergence

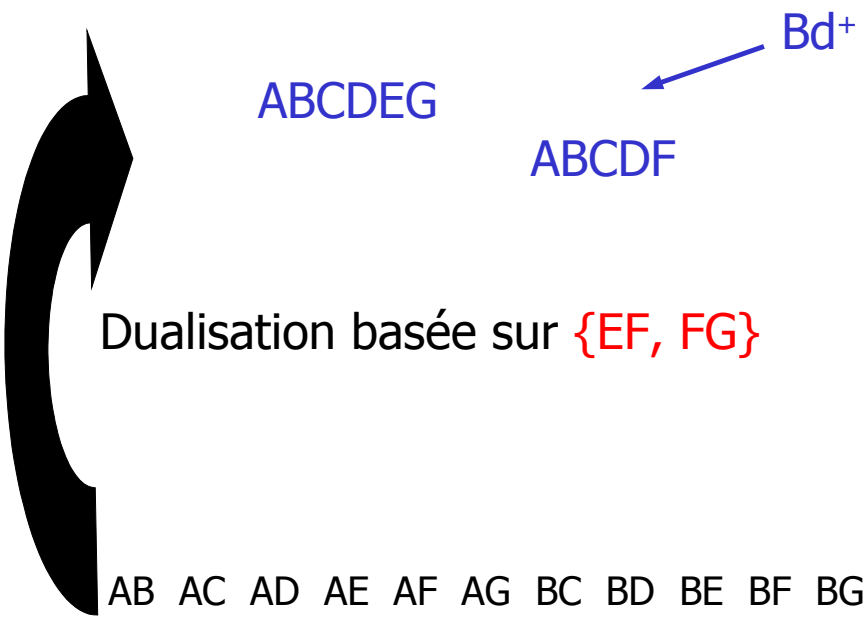
Intuition on ABS strategy



Intuition on ABS strategy

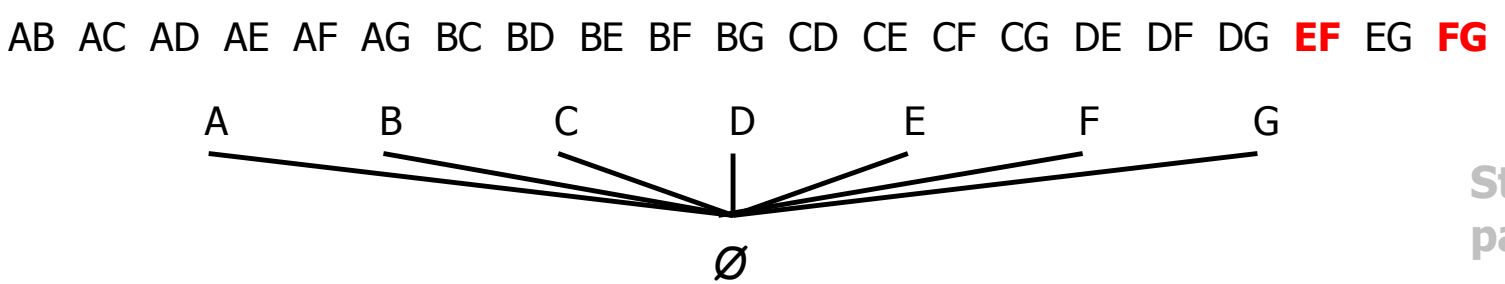


Intuition on ABS strategy



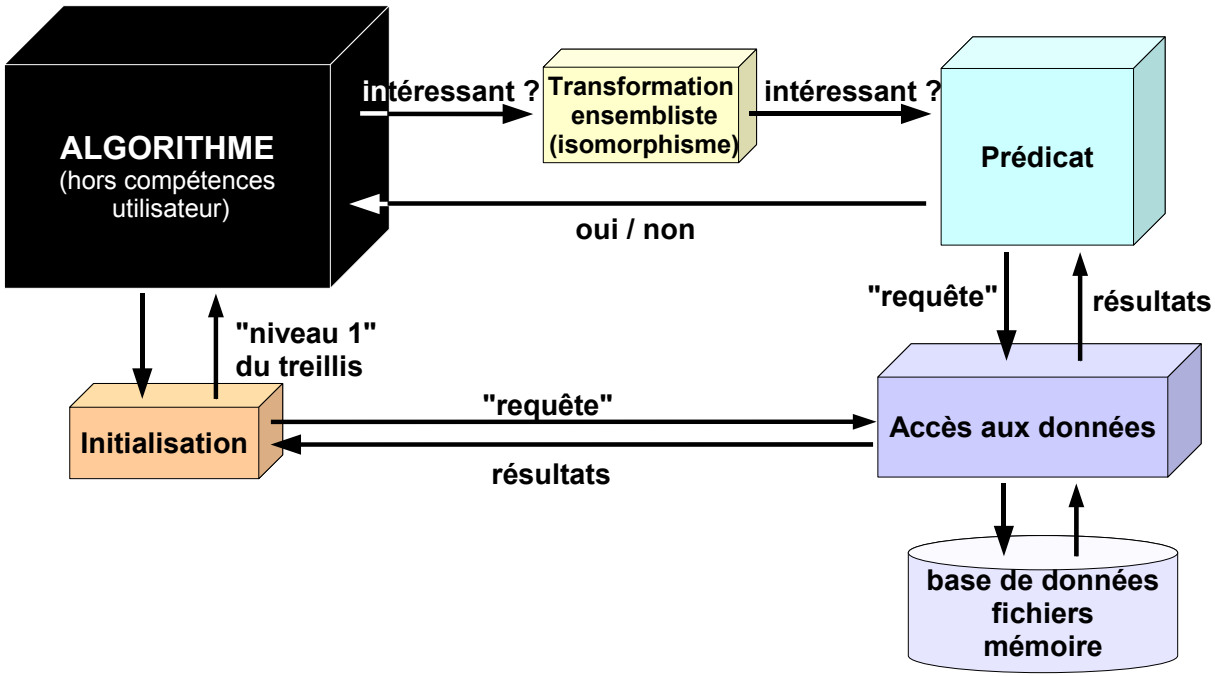
=> caractérisation exacte

=> 1 dualisation à la place de 4 itérations avec la stratégie par niveaux



Stratégie par niveaux

Architecture



Total independence between :

- → Algorithms and memory usage
- → Patterns and predicates
- → Data accesses

Currently proposed components

Problem	File (format)	DBMS
inclusion dependencies	CVS FDEP	MySQL
keys	CVS FDEP	MySQL
frequent itemsets	FIMI	
frequent essential itemsets	FIMI	
query rewriting	specific	

Algorithms and data structures

→ Levelwise search (bottom-up and top-down)
→ Dualization computation
→ prefix trees

Getting started with iZi (1/2)

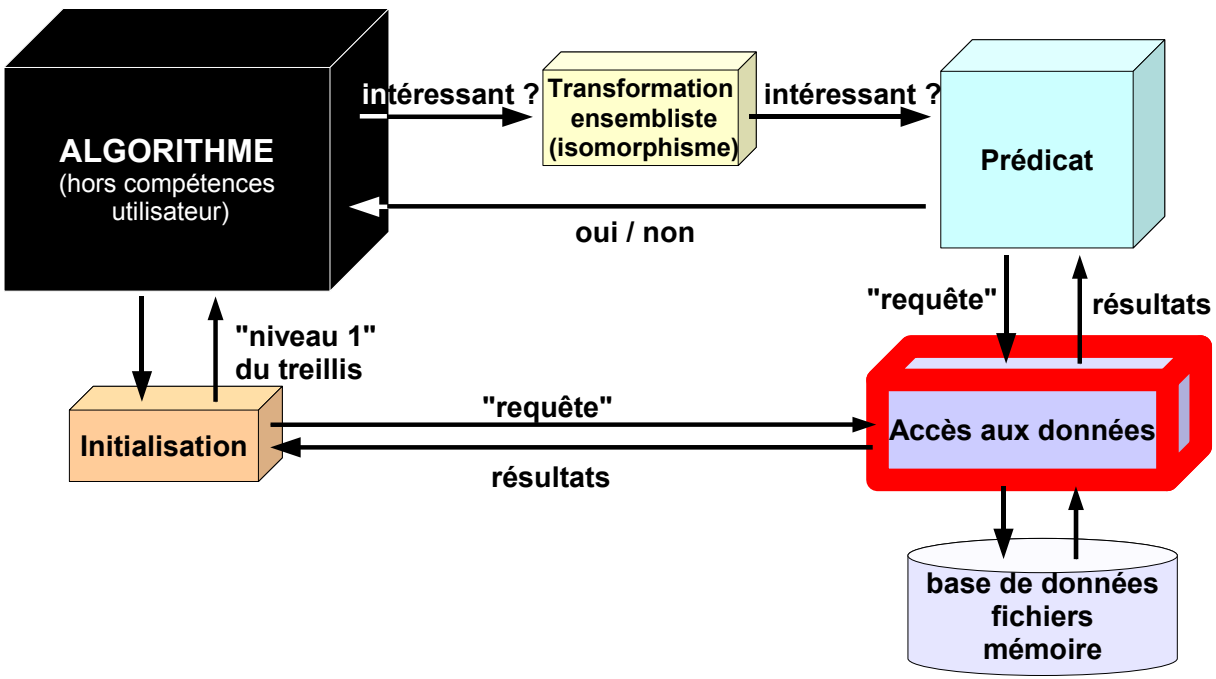
Let us consider a new problem *P*

- What should be done ?
  - what kind of databases ?
  - pattern definition ?
  - interesting pattern ?
  - Partial order among patterns ?
  - monotonicity (proof to be done) ?
  - What would be the solution ?
  - isomorphism to a boolean lattice (proof to be done)

- Algorithmic choice left to the analyst wrt data and solution characteristics

Getting started with iZi (2/2)

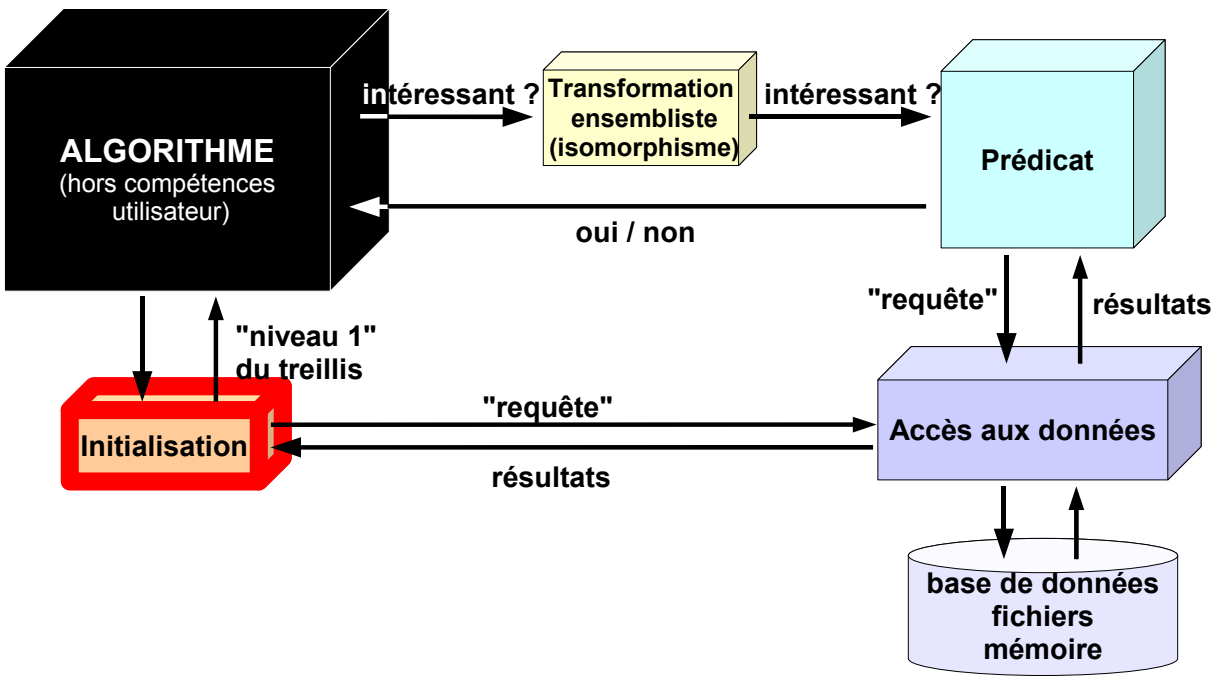
- Data access
- Initialization phase, patterns of size 1
- Implementing the function *f*
- Implementing the predicate



Getting started with iZi (2/2)

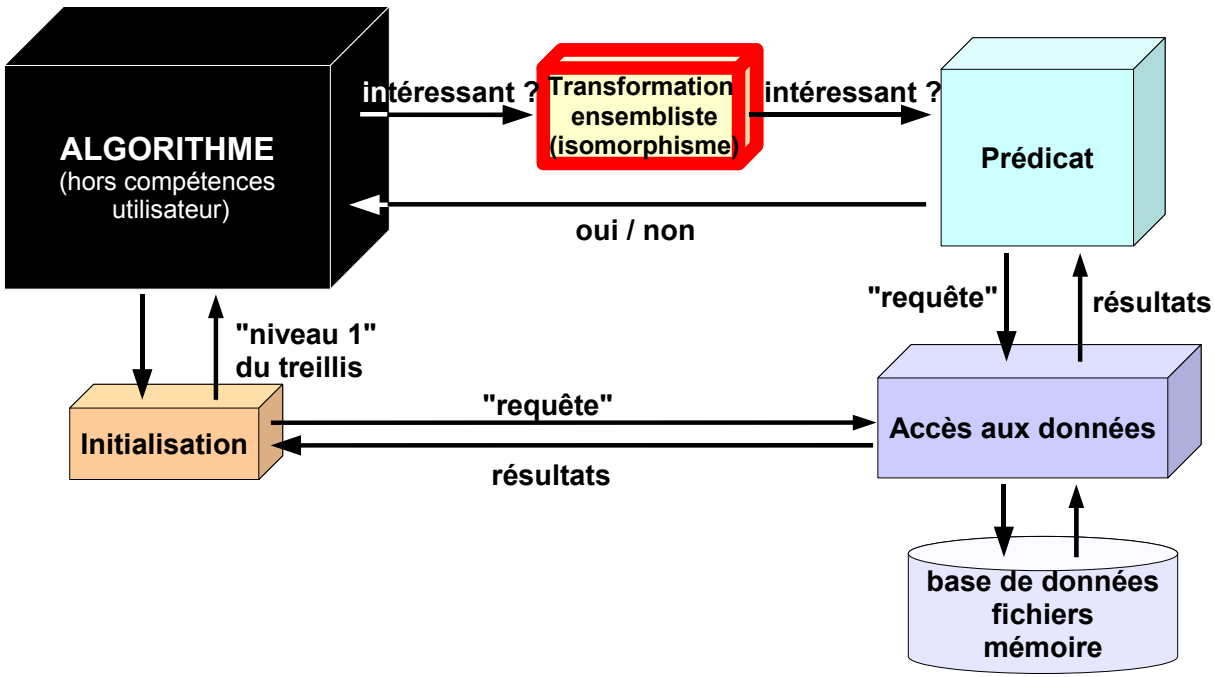
- Data access
- Initialization phase, patterns of size 1

- Implementing the function *f*
- Implementing the predicate



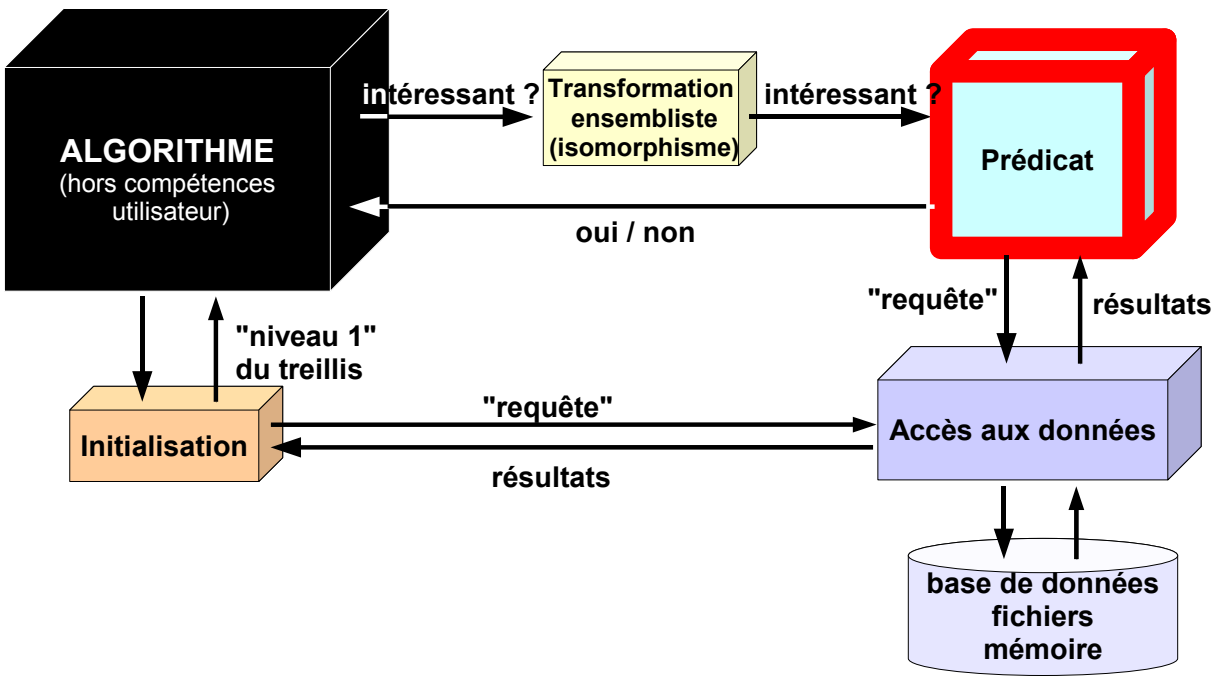
Getting started with iZi (2/2)

- Data access
- Initialization phase, patterns of size 1
- Implementing the function *f*
- Implementing the predicate



Getting started with iZi (2/2)

- Data access
- Initialization phase, patterns of size 1
- Implementing the function *f*
- Implementing the predicate



Experiments

Our feedbacks :

- A few hours to implement a new problem
- Many problems already implemented and available on the website
- Scalable and efficient implementations
  For frequent itemset mining
  - comparison with efficient and specialized implementations of *Apriori*

  ⇨ comparable performances

Outline

Conclusion

- iZi allows to deal with many pattern mining problems, often hidden behind practical applications
  1. Query rewriting in data integration (H. Jaudoin et al, DL'05) [JFPT09]
  2. Discovering complex matchings across web query interfaces : a correlation mining approach (B. He et al, KDD'04) [HCH04]
- Allows to cope with data-centric step of many applications
- Open-source, extensible, easy of use for "C++ aware" developers

Want to try it ?
just click at `http://liris.cnrs.fr/ iZi` !

Perspectives

Upgrading the library

1. Depth first algorithms, exploiting bitmaps
2. Patterns, predicates and data accesses

Declarative approach

1. Define a logical/algebraic language for some classes of problems, optimization, cost model ...

Extending the theoretical framework

1. Closure systems
2. New classes of problems

ANR DEFI 2009 : ongoing DAG project

M.A. Casanova, R. Fagin, and C. Papadimitriou.

Inclusion Dependencies and their Interaction with Functional Dependencies.
Journal of Computer and System Sciences, 28(1) :29–59, February 1984.

Surajit Chaudhuri.

Data mining and database systems : Where is the intersection ?
Data Engineering Bulletin, 21(1) :4–8, 1998.

Vineet Chaoji, Mohammad Al Hasan, Saeed Salem, and Mohammed Javeed Zaki.

An integrated, generic approach to pattern mining : data mining template library.
Data Min. Knowl. Discov., 17(3) :457–495, 2008.

Toon Calders and Jef Wijsen.

On monotone data mining languages.
In *DBPL*, pages 119–132, 2001.

Fabien De Marchi and Jean-Marc Petit.

Zigzag : a new algorithm for mining large inclusion dependencies in databases.
In *IEEE International Conference on Data Mining (ICDM'03), Floride, USA*, pages 27–34. IEEE Computer Society, November 2003.

Frédéric Flouvat, Fabien De Marchi, and Jean-Marc Petit.

Advanced Techniques for Data Mining and Knowledge Discovery, chapter The iZi project : easy prototyping of interesting pattern mining algorithms, pages 1–15.
LNCS. Springer-Verlag, September 2009.

F. Flouvat, F. De Marchi, and J-M. Petit.

Abs : Adaptive borders search of frequent itemsets.
In *FIMI'04*, 2004.

Bin He, Kevin Chen-Chuan Chang, and Jiawei Han.

Discovering complex matchings across web query interfaces : a correlation mining approach.
In *KDD*, pages 148–157, 2004.

Tomasz Imielinski and Heikki Mannila.

A database perspective on knowledge discovery.

Commun. ACM, 39(11) :58–64, 1996.

Hélène Jaudoin, Frédéric Flouvat, Jean-Marc Petit, and Farouk Toumani.

Towards a scalable query rewriting algorithm in presence of value constraints.
J. Data Semantics, 12 :37–65, 2009.

M. Kantola, H. Mannila, K-J. Räihä, and H. Siirtola.

Discovering functional and inclusion dependencies in relational databases.
International Journal of Intelligent Systems, 7 :591–607, 1992.

H. Mannila and Hannu Toivonen.

Levelwise search and borders of theories in knowledge discovery.
Data Mining and Knowledge Discovery, 1(3) :241–258, 1997.

Siegfried Nijssen and Luc De Raedt.

Iql : A proposal for an inductive query language.
In *KDID*, pages 189–207, 2006.

Luc De Raedt, Tias Guns, and Siegfried Nijssen.

Constraint programming for itemset mining.
In *KDD*, pages 204–212, 2008.

Stefan Wrobel.

Inductive logic programming for knowedge discovery in databases.
pages 74–99, 2000.

Seconde partie