A story of rules

ICFCA - 29 Jun-2 Jul 2021 - Strasbourg

A multiple voices talk

Karell Bertet

Christophe Demko

University of La Rochelle, France

Kira Adaricheva

Hofstra University, New York, USA

Angel Mora

Manuel Enciso

Malaga's Team, University of Malaga, Spain

One voice is missing ....

Vincent Duquenne

"Where is Guigues ?"

This talk to pay tribute ....

... to the memory of Vincent Duquenne, who left us so prematurely.
Vincent is "one of the two men" of the canonical basis.

Preliminaries: Implication

- An *implication* is a rule $A \to B$ on a set S with $A \subseteq S$ and $B \subseteq S$
  - A is the premise, B the conclusion,
  - meaning A *implies* B
  - refering to strong association rules (confidence=1)
- An *implicational system* (IS) Σ is a set of implications
- A subset $X \subseteq S$ *respects* an implication $A \to B$ when

$$A \subseteq X \text{ implies } B \subseteq X$$

Preliminaries: Closure system

- A *closure operator* φ on a set S is a map $\varphi : P(S) \to P(S)$ that satisfies, for all $X, Y \subseteq S$:
  - increasing : $X \subseteq \varphi(X)$
  - isotone : $X \subseteq Y$ implies $\varphi(X) \subseteq \varphi(Y)$
  - idempotent : $\varphi(\varphi(X)) = \varphi(X)$
- A *closure system* refers to the pair (φ, S) of a set S and a closure operator φ on S

Preliminaries: Closure lattice

- A subset $X \subseteq S$ is called *closed* or a *closure* when $X = \varphi(X)$
- The set of all closed sets equipped with the inclusion relation forms a lattice called the *closure lattice*
- Closure systems are *equivalent* when they have the same closure lattice

Preliminaries: Links with FCA

FCA

Let (G, M, α, β) be a context with :

- $\alpha : P(G) \to P(M)$ associates to objects their common attributes
- $\beta : P(M) \to P(G)$ associates to attributes their common objects

Then $(\alpha \circ \beta, M)$ and $(\beta \circ \alpha, G)$ are two closure systems

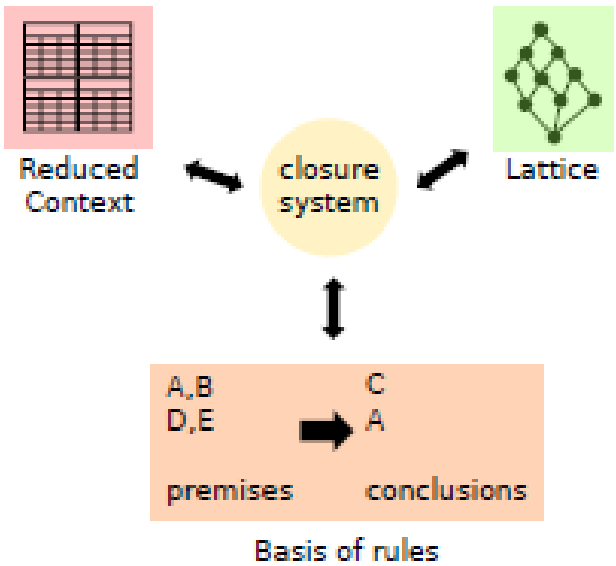
Implications

Let Σ be an implicational system on S
For $X \subseteq S$, let $\varphi_\Sigma(X)$ be the closure of X, i.e. the smallest set

- containing X and
- respecting any implication $A \to B \in \Sigma$

Then (φ_Σ, S) is a closure system

Preliminaries: Links with FCA



Concept lattice

The *concept lattice* of a context is the mix of the two closure lattices on G and M

Basis

IS with some minimality properties

Reduced context

Context reduced to the *irreducible* elements of the lattice

Bijections where closure system is central[1]

[1] K.Bertet, C. Demko, J-F. Viaud, C. Guérin. Lattices, closures systems and implication bases: A survey of structural aspects and algorithms. TCS(2018)

Table of contents

Table of contents

What bases of rules ?

Kira Adaricheva

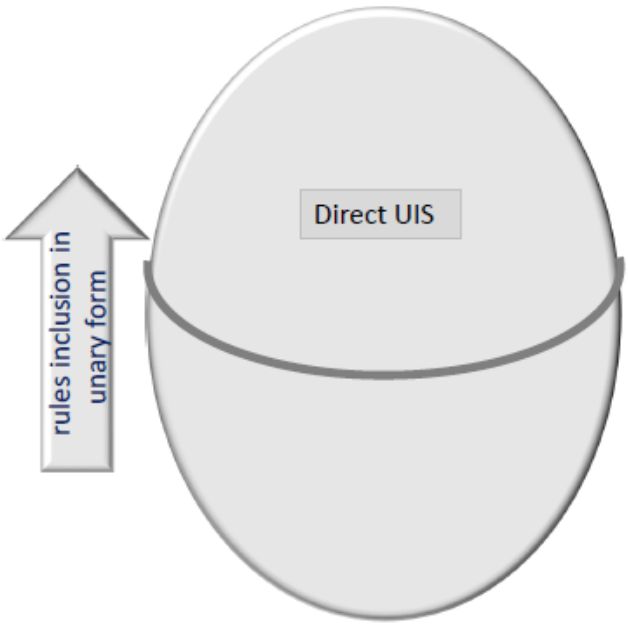
Hofstra University, New York, USA

Karell Bertet

University of La Rochelle, France

Table of contents

The main bases



Equivalent IS ordered by inclusion of their unit form

Equivalent IS

Their associated closure systems are equivalent (the same closure lattice)

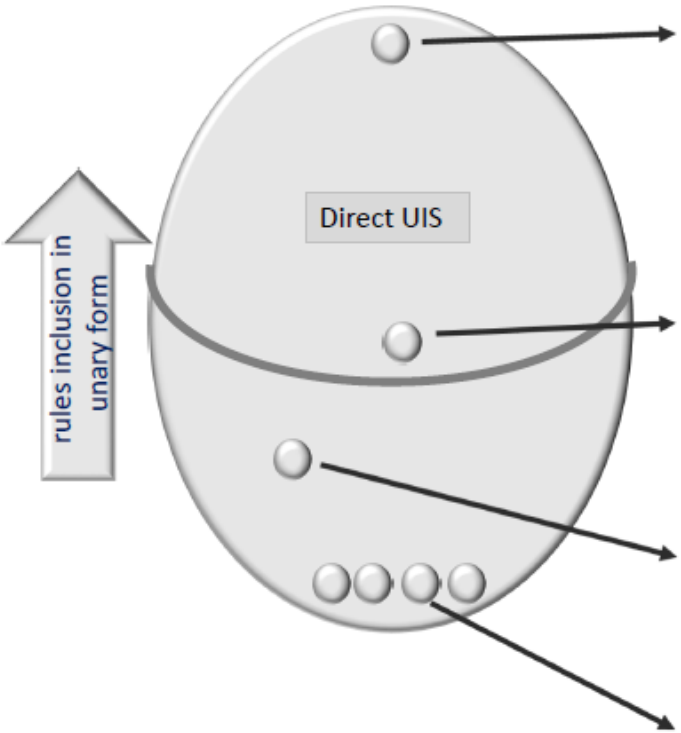
Unit form

Each implication $A \rightarrow B$ is replaced by the set of unit implications $\{A \rightarrow b, \forall b \in B\}$

Directness

An IS is direct when each closure $\varphi(X)$ can be obtained by only one-pass on its implications

The main bases



Equivalent IS ordered by inclusion of their unit form

The full UIS (maximal)

$\{X \to \varphi(X) \setminus X : X \subseteq S\}$

The canonical direct basis (minimal among the direct ones)

$\{X \to \varphi(X) \setminus X : X \text{ is a min. gen.}\}$

The D-basis

Direct according to a special ordering

The canonical basis (one minimal)

$\{X \to \varphi(X) \setminus X : X \text{ is a pseudo-cl.}\}$

The canonical direct basis

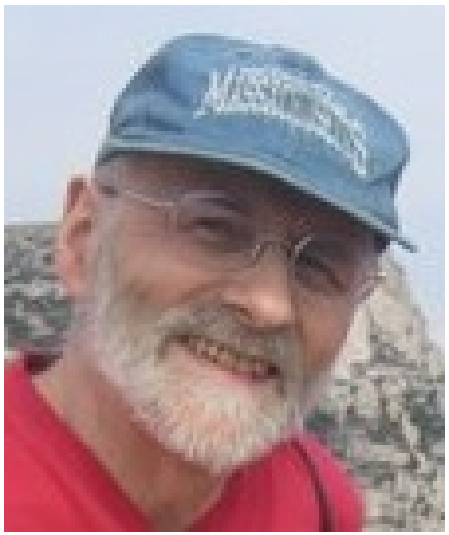
The *canonical direct basis* is a basis unifying various definition of direct basis[2]:

- The canonical and iteration-free basis (Wild, 1994)
- The weak implication basis (Rush, Wille, 1995)
- The minimal functional dependencies (Maier, 1983)
- The proper implications basis (Taouil, Bastide, 2002)
- The direct-optimal basis (Bertet, Nebut, 2004)
- The dependance relation basis (Bertet, 1998)

[2] K.Bertet, B. Monjardet. The multiple factes of the canonical direct basis, TCS, 411 (2010)

Dependence relation

Binary relation defined on the set of join-irreducible elements of a lattice (including the sub-order)



Karell (oct 2004, CAMS seminary, EHESS):

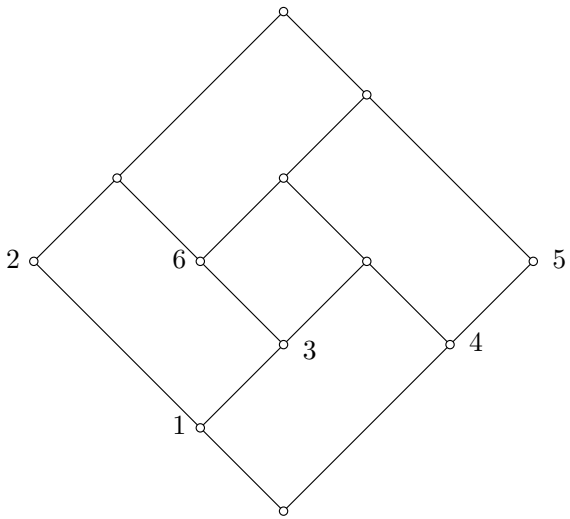
The dependance relation basis is the unique direct-optimal basis

Bernard Monjardet (oct 2004, CAMS seminary, EHESS):

It looks like other bases I worked on a few years ago. Is it also left-minimal ?

Table of contents

The journey to the D-basis



Finite lattice with six join-irreducibles

A lattice

Partially ordered set where each pair of elements has a least upper bound $\vee$ and a greatest lower bound $\wedge$.

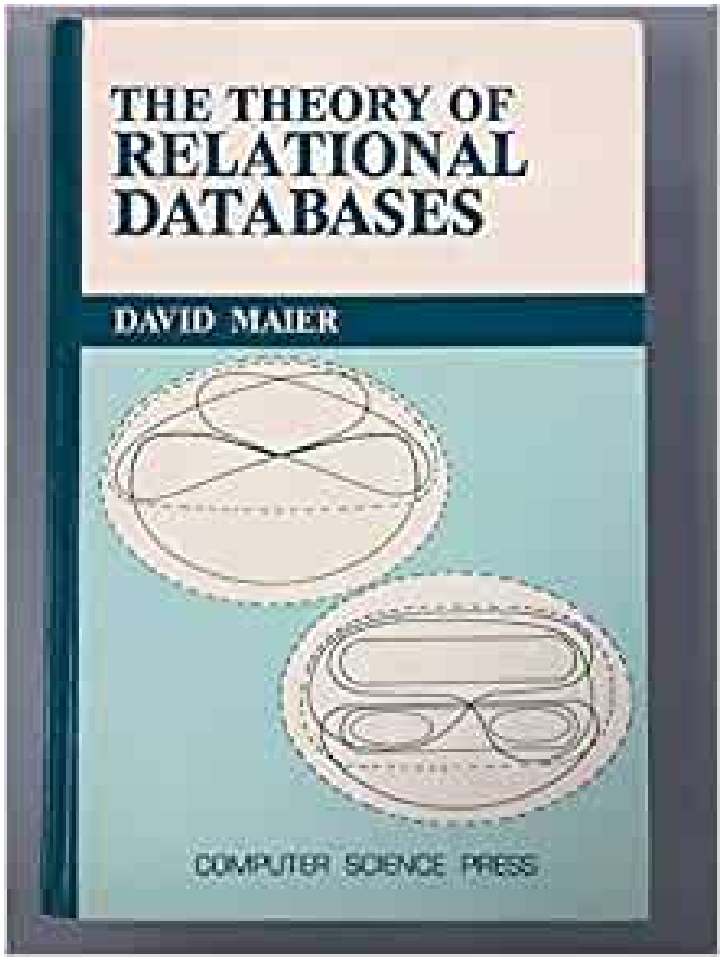
Join irreducible

An element j of lattice such that $j = a \vee b$ implies $j = a$ or $j = b$.

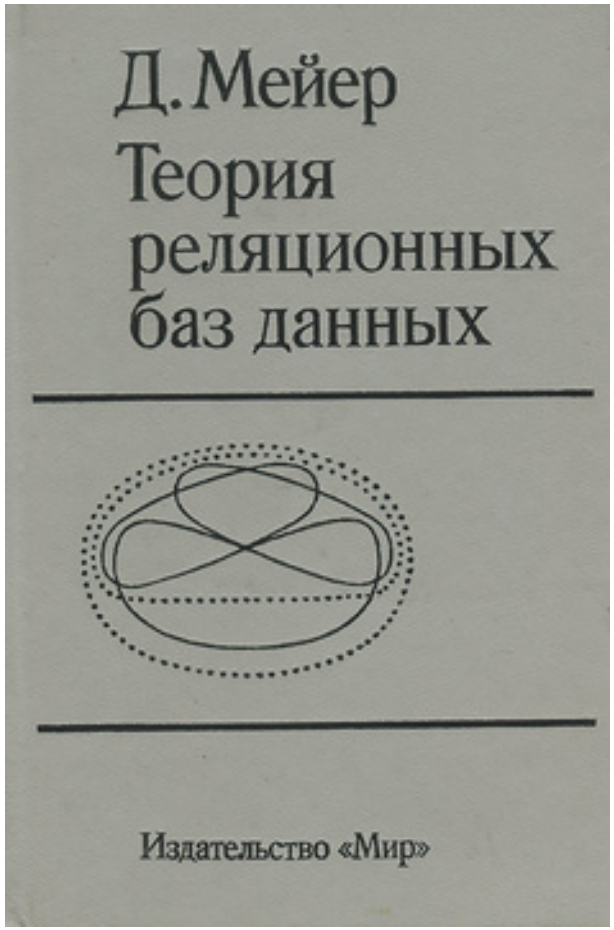
Example of a finite lattice

Relations on join irreducibles: $3 \leq 6$, or $3 \leq 2 \vee 5$

Portland-Novosibirsk, 1983-87



1983 issue



1987 issue

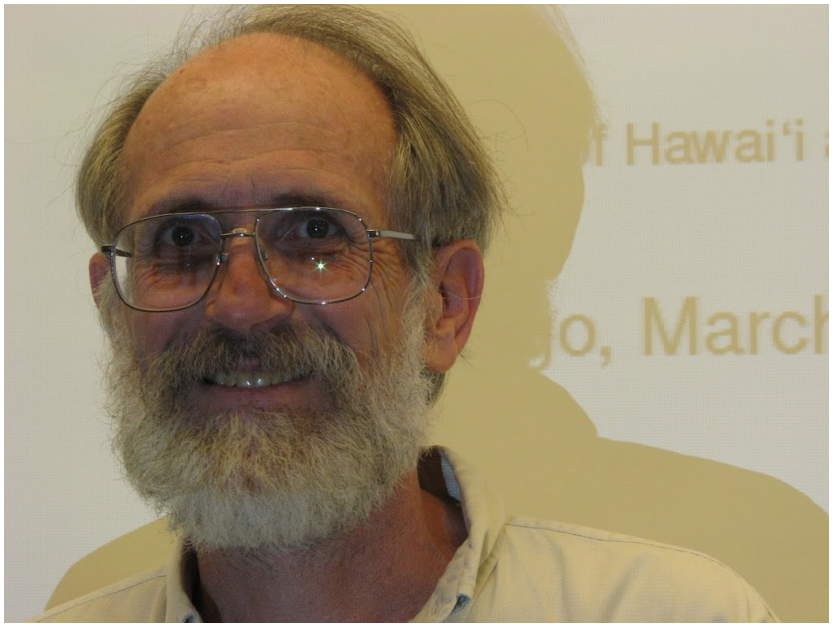
Alan Day

A. Day, *The lattice theory of functional dependencies and normal decompositions*, IJAC 1992

Day's conference: McMaster University, 1992

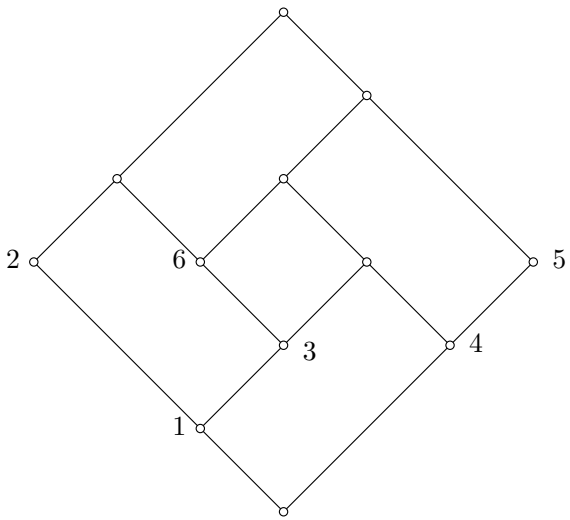
M. Wild, *A theory of finite closure spaces based on implications*, Adv. Math. 1996

Day's conference: McMaster University, 1992



J.B.Nation, *An approach to lattice varieties of finite height*, Alg. Universalis 1990

OD-graph of a finite lattice



Finite lattice with six join-irreducibles

OD-graph of a finite lattice

(I) Partially ordered set of join irreducibles.
(II) Minimal *join-covers* of join irreducibles.

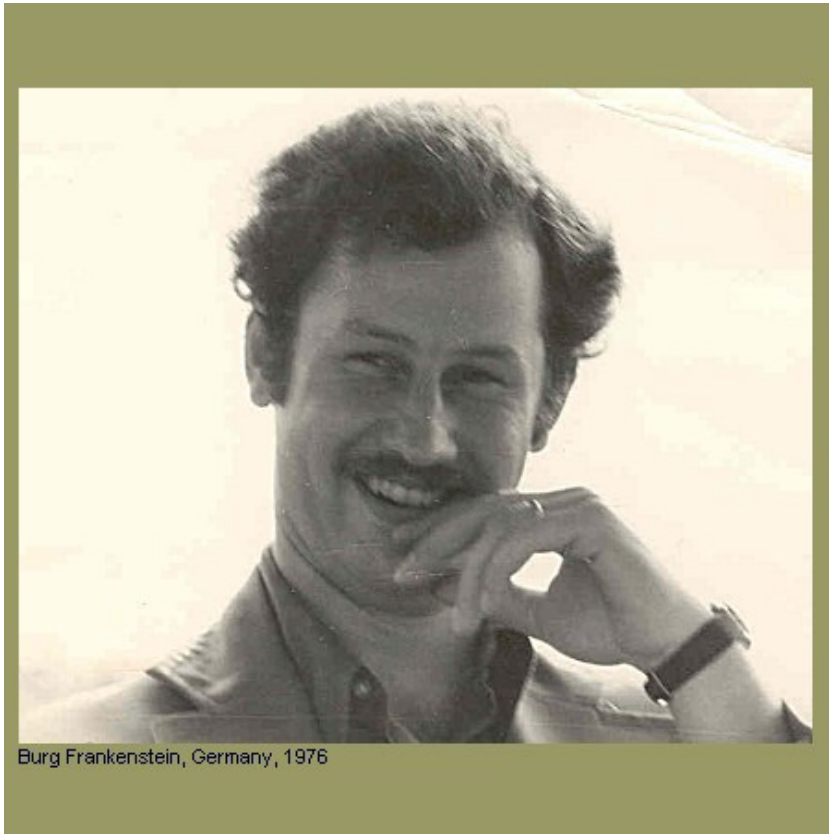
Example: part (I)

$1 \leq 2$, $1 \leq 3 \leq 6$, $4 \leq 5$

Example: part (II)

$3 \leq 2 \vee 5$ is a join-cover of 3, but it is not *minimal* cover, since there is $3 \leq 1 \vee 4$, where $1 \leq 2$ and $4 \leq 5$, from (I)

ORDAL'96 : Ottawa



D. Duffus, I. Rival, *A structure theory for ordered sets*, Disc. Math. 1981

ORDAL'96 : Ottawa

V. Duquenne, *GLAD: General Lattice Analysis and Design*, Invited talk in two parts

Fast forward to 2011 ...

Kira:

Is that right? So you were there at Ivan's conference? Somehow, I am not good at remembering things. In 2008, on ROGICS conference in Tunisia, I met Guy Jourdan, and he said he remembers me in Ivan Rival's home, when there were just a few people there. Only when he told me I vaguely remembered.

Vincent

Hence you don't need a forgetful functor...

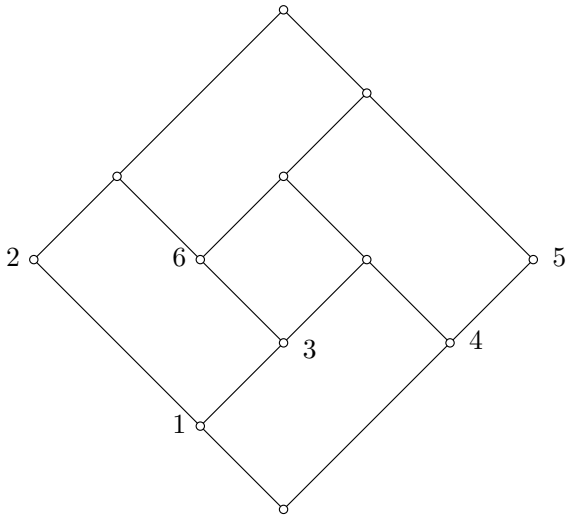
AMS meeting: University of Illinois, 2009

M. Langlois, R. H. Sloan, Gy. Turán: *Horn upper bounds and renaming*, Journal on Satisfiability, Boolean Modeling and Computation 2010.

University of Illinois, Chicago 2010

K. Bertet and B. Monjardet, *The multiple facets of the canonical direct unit implicational basis*, Theoretical Computer Science 2010.

Connecting the dots



Closure lattice of
six-element set
(= set of join-irreducibles)

Lattice perspective

(I) Partially ordered set of join irreducibles.
(II) Minimal *join-covers* of join irreducibles.

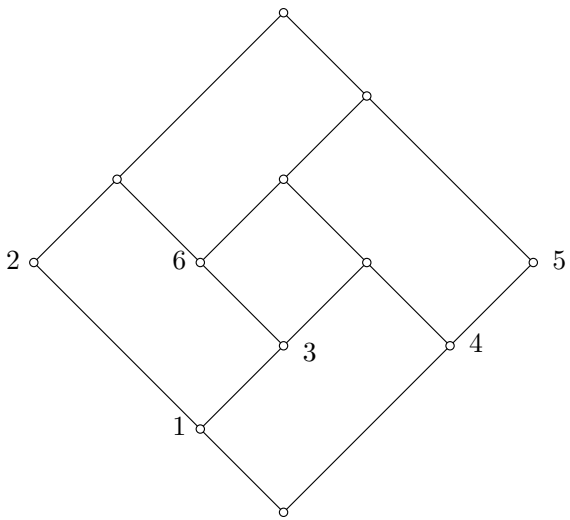
Closure system perspective

Part (I) becomes: $2 \to 1$,
$6 \to 3 \to 1$, $5 \to 4$

Example: part (II)

$2\&5 \to 3$ is an implication from the join-cover of 3, and $1\&4 \to 3$ from the *minimal* cover of 3.

The D-basis



Closure lattice of
six-element set
(= set of join-irreducibles)

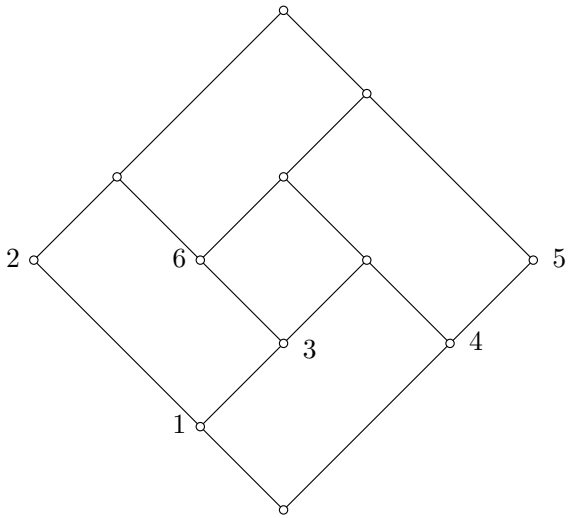
The D-basis has implications coming from OD-graph

(I) it is a subset of the canonical direct unit basis.
(II) it is *ordered-direct*: apply implications from part (I) before part (II).

Closure system perspective

D-basis is easy to compute from any direct basis.

The D-basis



Closure lattice of
six-element set
(= set of join-irreducibles)

Canonical direct unit basis vs. D-basis

D-basis has 9 implications,
canonical direct unit basis has 13.
Extra: $2\&5 \to 3$, $2\&4 \to 3$,
$1\&5 \to 3$ and $2\&5 \to 6$.

The D-basis

```
1 0 0 0 0 1 0 0 0 1 1 0 1 1 0 1 1 1 0 1 1 1
1 1 1 0 0 1 0 0 1 1 1 0 0 0 1 1 0 1 0 1 0 1
1 1 1 1 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1
1 1 0 1 0 1 1 1 0 0 0 0 1 1 1 0 0 0 1 1 1 1
0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0
0 0 0 1 1 1 1 0 0 1 1 0 0 1 1 0 0 0 1 1 0 1
1 1 1 1 1 1 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
1 1 0 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1
1 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 1 1 1 0 0 0
0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1
```

Benchmark 10x22 binary matrix for testing bases on its concept lattice

Canonical basis

has 97 implications

Canonical direct unit basis

has 1534 implications.

The D-basis

has 719 implications.

Yeshiva University, New York 2011

K.Adaricheva, J.B.Nation,R. Rand, *Ordered direct implicational basis of a finite closure system*, Discr. Appl. Math. 2013.

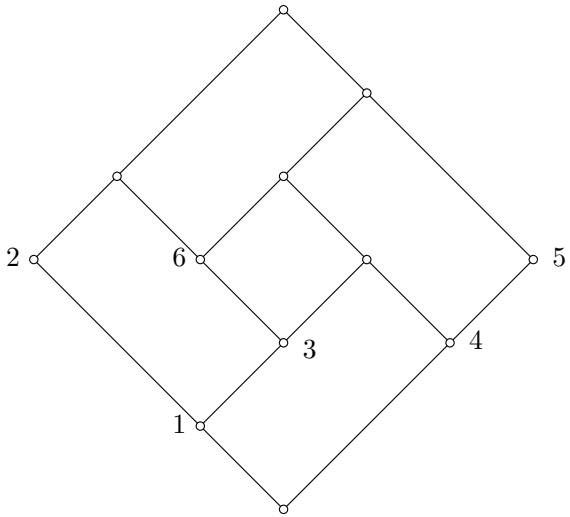
Vincent in communication 2011-12

Mushrooms in Sennely, which he collected

From letter in July 2012

Here is a small idea (I take only 10% if it worked!!). I never took seriously the optimal business, because the right way to calculate with implications doesn't gain anything from it. OK. But there are potentially situations where optimality and your order directness should be crucial: biology, chemistry etc. For classifying an object, you have its partial description, you want to calculate its (quasi-)closure, and for this you have reactives / products / procedures but you can use them only one time, and even with some order constraint.

Dependence relation vs. D-relation



Closure lattice of six-element set (= set of join-irreducibles)

Dependence relation

Connects an element in the conclusion and elements in the premise of a rule in canonical direct unit basis. Since $2\&5 \to 3$, $2\&4 \to 3$, $1\&5 \to 3$ and $1\&4 \to 3$ are in the canonical direct unit basis, 3 *depends* on $1, 2, 4, 5$.

D-relation

Similar relation for the D-basis.Since only $1\&4 \to 3$ is in the D-basis, are in the canonical direct unit basis, 3 *D-depends* on $1, 4$.

D-relation in the context

1 0 0 0 0 1 0 0 0 1 1 0 1 1 0 1 1 1 0 1 1 1
1 1 1 0 0 1 0 0 1 1 1 0 0 0 1 1 0 1 0 1 0 1
1 1 1 1 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1
1 1 0 1 0 1 1 1 0 0 0 0 1 1 1 0 0 0 1 1 1 1
0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0
0 0 0 1 1 1 1 0 0 1 1 0 0 1 1 0 0 0 1 1 0
1 1 1 1 1 1 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
1 1 0 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1
1 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0
0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1

The D-relation

can be found in the context using the standard arrow relations.

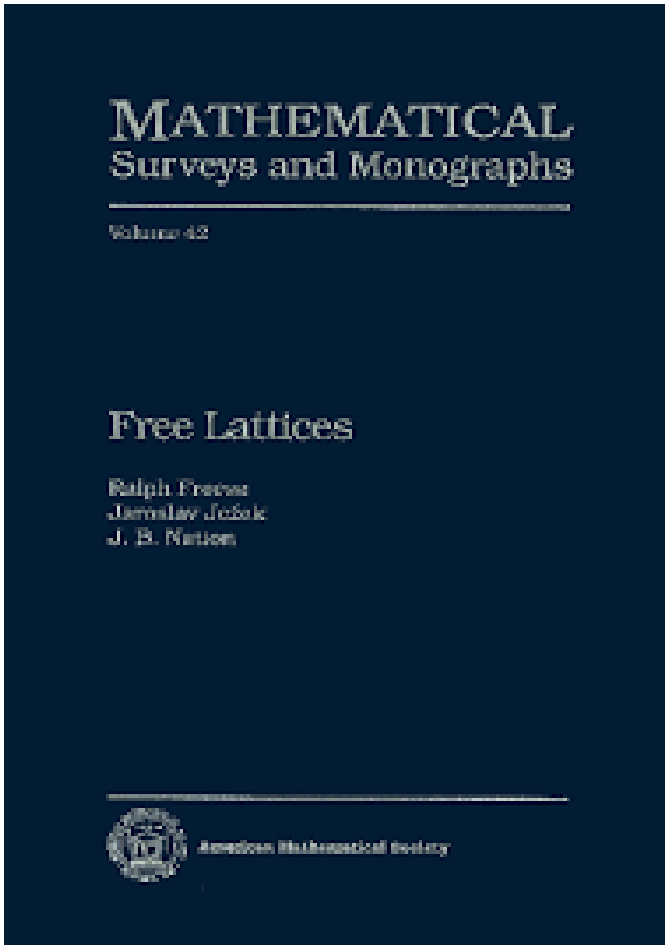
The D-basis

can be extracted from the context using hypergraph dualization.

The D-relation

plays an important role in the study of structure of free lattices.

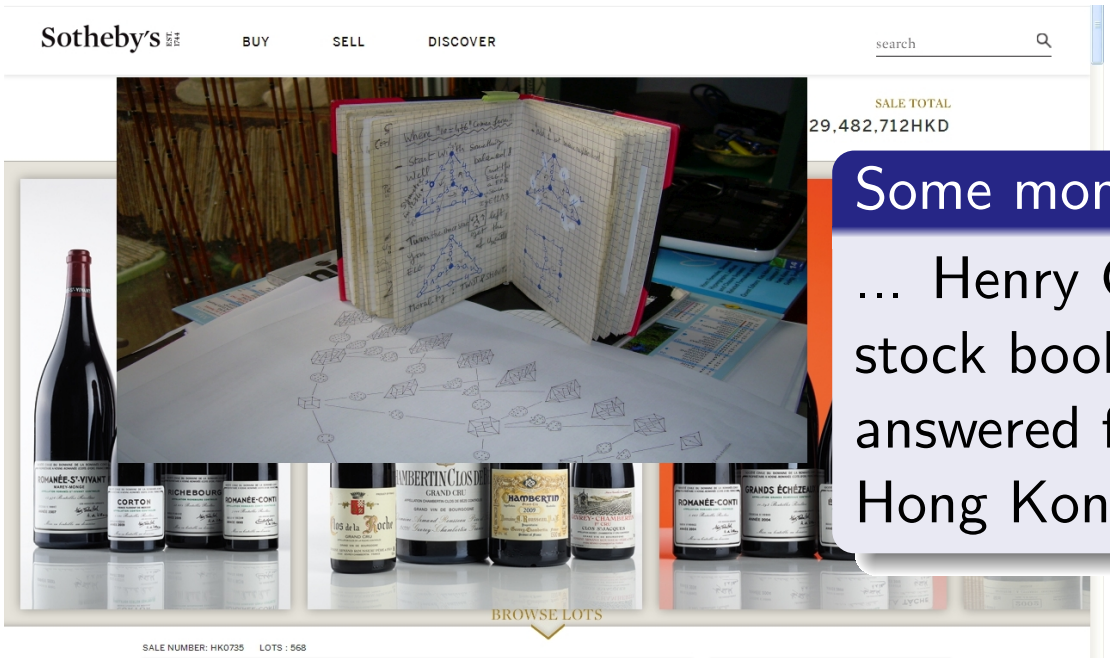
Free lattices - 1992



Vincent in Dagstuhl - 2014

Vincent is the third on the left!

Vincent on his stock booklet



Some months ago ...

... Henry Crapo asked me my stock booklet, to which I answered for fun that I sold it in a Hong Kong auction.

Fast backward to 2011 ...

Kira:

During my stay in Hawai'i I was able to show that aggregated E-basis is an optimization of your canonical basis.

Vincent

Hard to see that, one day, you are no more optimal, but it's life! (I'm kidding, which you understood is my favorite hobby...)

Table of contents

What about non binary data ?

Christophe Demko

Karell Bertet

University of La Rochelle, France

Table of contents

The first approaches aiming to adapt data to FCA are not really effective

- Multivalued context
- Discretization/scaling of numerical data into intervals
- Embedding of more complex data into numerical vector
- ....
  => Huge number of binary attributes
  => Loss of value
  => Loss of structure

The question of an extension of FCA framework to the non-binary case has been adressed

- For what kind of data ?
- How to define a concept/closure lattice ?
  - What is the closure operator ?
  - Is there a Galois connection ?
- How to extract/deduce (bases of rules) from the lattice ?
  - What are the minimal generators ?
  - What are the pseudo-closed sets ?

Some approaches are dedicated to particular data

Triadic concept analysis (Lehmann, Wille, 1995)

- Dedicated to triadic contexts: 3 sets (Obj, Att, Cond), with a 3-ary relation
- Triadic concepts and triadic implications are defined
- $N \underset{ad}{\longrightarrow} P$ ($N \rightarrow P$ under ad)
- Extensions to n-adic (Voutsadakis, 2002)
- Triadic association rules (Missaoui et al, 2013)

Relational concept analysis(Huchard et al 2007)

- Defined for relational data
- An iterative process generates a concept lattice

$\mathbb{K}$	P	N	R	K	S
1	abd	abd	ac	ab	a
2	ad	bcd	abd	ad	d
3	abd	d	ab	ab	a
4	abd	bd	ab	ab	d
5	ad	ad	abd	abc	a

Others works extend FCA to non-binary data

Logical Concept Analysis

Logical Concept Analysis[a] is a generalization of FCA in which sets of attributes are replaced by logical expressions. The power set of attributes is replaced by an arbitrary set of formulas to which are associated a subsumption relation, and conjunctive and disjunctive operations, and therefore forms a lattice.

[a] S. Ferré, O. Ridoux. A Logical Generalization of Formal Concept Analysis. LNCS. 2000

LCA is the fundamental base of Abstract Conceptual Navigation (Ferré 2014) whose principle is to navigate in a conceptual space where places are logical concepts connected by navigation links.

Other works extends FCA to non-binary data

Pattern structures

Pattern structures[a] allows to extend FCA algorithms to non-binary data when a Galois connection exists between obects and their description, i.e. the space description provides a subsumption relation.

[a] B. Ganter, S. Kuznetsov. Pattern structures and their projections. International Conference on Conceptual Structures. 2001.

- Pattern structures for numerical data (Kaytoue, 2020)
- Pattern structure for sequential data (Buzmakov et al, 2017)
- ...

But pattern lattices are huge, often intractable.

From the deluge of data to the deluge of patterns and rules!!

Table of contents

NextPriorityConcept

Founding ideas

Inspired by discussions with colleagues, and in particular with Sergei K.during his stay at La Rochelle

NextPriorityConcept

Founding ideas

NextPriorityConcept[a] generates concepts from heterogeneous and complex data :

- A pattern structure approach
- Inspired by the Bordat algorithm, it maintains the lattice structure using a priority queue and a constraint propagation mechanism
- Generic strategies allows to limit the generation of patterns
- Patterns are described in a generic way by first order monadic predicates

[a] Demko, Bertet, Faucher, Viaud, Kuznetsov. NextPriorityConcept: A new and generic algorithm computing concepts from complex and heterogeneous data. TCS. 2020

Bordat algorithm as basis

A dual version of Bordat theorem

There is a bijection between the immediate predecessors of a concept (A, B) and the inclusion maximal subsets of the family:

$$\left\{ \beta(b) \cap A \ : \ b \in M \setminus B \right\}$$

Bordat algorithm as basis

A dual version of Bordat theorem

There is a bijection between the immediate predecessors of a concept (A, B) and the inclusion maximal subsets of the family:

$$\left\{ \beta(b) \cap A \ : \ b \in M \setminus B \right\}$$

Concepts($\langle G, M, (\alpha, \beta) \rangle$)

```
begin
    top ← (G, α(G));
    Add top to a queue Q;
    while Q not empty do
        (A, B) ← Q.pop();
        produce (A, B);
        LP ← Immediate-Predecessors((A, B));
        forall (A′, B′) ∈ LP do
            Add (A′, B′) to Q
        end
    end
end
```

Bordat algorithm as basis

A dual version of Bordat theorem

There is a bijection between the immediate predecessors of a concept (A, B) and the inclusion maximal subsets of the family:

$$\left\{ \beta(b) \cap A \ : \ b \in M \setminus B \right\}$$

Concepts($\langle G, M, (\alpha, \beta)\rangle$)

```
begin
    top ← (G, α(G));
    Add top to a queue Q;
    while Q not empty do
        (A, B) ← Q.pop();
        produce (A, B);
        LP ← Immediate-Predecessors((A, B));
        forall (A′, B′) ∈ LP do
            Add (A′, B′) to Q
        end
    end
end
```

Immediate-Predecessors((A, B))

```
begin
    L ← ∅ ;
end
```

Bordat algorithm as basis

A dual version of Bordat theorem

There is a bijection between the immediate predecessors of a concept (A, B) and the inclusion maximal subsets of the family:

$$\left\{ \beta(b) \cap A \ : \ b \in M \setminus B \right\}$$

Concepts($\langle G, M, (\alpha, \beta)\rangle$)

```
begin
    top ← (G, α(G));
    Add top to a queue Q;
    while Q not empty do
        (A, B) ← Q.pop();
        produce (A, B);
        LP ← Immediate-Predecessors((A, B));
        forall (A′, B′) ∈ LP do
            Add (A′, B′) to Q
        end
    end
end
```

Immediate-Predecessors((A, B))

```
begin
    L ← ∅ ;
    forall b ∈ M \ B do
    end
end
```

Bordat algorithm as basis

A dual version of Bordat theorem

There is a bijection between the immediate predecessors of a concept (A, B) and the inclusion maximal subsets of the family:

$$\left\{ \beta(b) \cap A \ : b \in M \setminus B \right\}$$

Concepts($\langle G, M, (\alpha, \beta) \rangle$)

```
begin
    top ← (G, α(G));
    Add top to a queue Q;
    while Q not empty do
        (A, B) ← Q.pop();
        produce (A, B);
        LP ← Immediate-Predecessors((A, B));
        forall (A′, B′) ∈ LP do
            Add (A′, B′) to Q
        end
    end
end
```

Immediate-Predecessors((A, B))

```
begin
    L ← ∅ ;
    forall b ∈ M \ B do
    end
end
```

Bordat algorithm as basis

A dual version of Bordat theorem

There is a bijection between the immediate predecessors of a concept (A, B) and the inclusion maximal subsets of the family:

$$\left\{ \beta(b) \cap A \ : b \in M \setminus B \right\}$$

Concepts($\langle G, M, (\alpha, \beta) \rangle$)

```
begin
    top ← (G, α(G));
    Add top to a queue Q;
    while Q not empty do
        (A, B) ← Q.pop();
        produce (A, B);
        LP ← Immediate-Predecessors((A, B));
        forall (A′, B′) ∈ LP do
            Add (A′, B′) to Q
        end
    end
end
```

Immediate-Predecessors((A, B))

```
begin
    L ← ∅ ;
    forall b ∈ M \ B do
        A′ ← β(b) ∩ A ;
    end
end
```

Bordat algorithm as basis

A dual version of Bordat theorem

There is a bijection between the immediate predecessors of a concept (A, B) and the inclusion maximal subsets of the family:

$$\left\{ \beta(b) \cap A \ : \ b \in M \setminus B \right\}$$

Concepts($\langle G, M, (\alpha, \beta) \rangle$)

```
begin
    top ← (G, α(G));
    Add top to a queue Q;
    while Q not empty do
        (A, B) ← Q.pop();
        produce (A, B);
        LP ← Immediate-Predecessors((A, B));
        forall (A′, B′) ∈ LP do
            Add (A′, B′) to Q
        end
    end
end
```

Immediate-Predecessors((A, B))

```
begin
    L ← ∅ ;
    forall b ∈ M \ B do
        A′ ↩ β(b) ∩ A ;
    end
end
```

Bordat algorithm as basis

A dual version of Bordat theorem

There is a bijection between the immediate predecessors of a concept (A, B) and the inclusion maximal subsets of the family:

$$\{\beta(b) \cap A : b \in M \setminus B\}$$

Concepts($\langle G, M, (\alpha, \beta)\rangle$)

```
begin
    top ← (G, α(G));
    Add top to a queue Q;
    while Q not empty do
        (A, B) ← Q.pop();
        produce (A, B);
        LP ← Immediate-Predecessors((A, B));
        forall (A′, B′) ∈ LP do
            Add (A′, B′) to Q
        end
    end
end
```

Immediate-Predecessors((A, B))

```
begin
    L ← ∅ ;
    forall b ∈ M \ B do
        A′ ← β(b) ∩ A ;
        if A′ maximal in L then Add A′ to L;
    end
end
```

Bordat algorithm as basis

A dual version of Bordat theorem

There is a bijection between the immediate predecessors of a concept (A, B) and the inclusion maximal subsets of the family:

$$\left\{ \beta(b) \cap A \ : \ b \in M \setminus B \right\}$$

Concepts($\langle G, M, (\alpha, \beta) \rangle$)

```
begin
    top ← (G, α(G));
    Add top to a queue Q;
    while Q not empty do
        (A, B) ← Q.pop();
        produce (A, B);
        LP ← Immediate-Predecessors((A, B));
        forall (A′, B′) ∈ LP do
            Add (A′, B′) to Q
        end
    end
end
```

Immediate-Predecessors((A, B))

```
begin
    L ← ∅ ;
    forall b ∈ M \ B do
        A′ ← β(b) ∩ A ;
        if A′ maximal in L then Add A′ to L;
    end
end
```

Bordat algorithm as basis

A dual version of Bordat theorem

There is a bijection between the immediate predecessors of a concept (A, B) and the inclusion maximal subsets of the family:

$$\left\{ \beta(b) \cap A \ : \ b \in M \setminus B \right\}$$

Concepts($\langle G, M, (\alpha, \beta)\rangle$)

```
begin
    top ← (G, α(G));
    Add top to a queue Q;
    while Q not empty do
        (A, B) ← Q.pop();
        produce (A, B);
        LP ← Immediate-Predecessors((A, B));
        forall (A′, B′) ∈ LP do
            Add (A′, B′) to Q
        end
    end
end
```

Immediate-Predecessors((A, B))

```
begin
    L ← ∅ ;
    forall b ∈ M \ B do
        A′ ↞ β(b) ∩ A ;
        if A′ maximal in L then Add A′ to L;
    end
    LP ← ∅ ;
    forall A′ ∈ L do
        Add (A′, α(A′)) to LP
    end
    return LP
end
```

Selection of attributes: a strategy σ

Definition

Instead of all the possibles attributes in $M \setminus B$, we only consider some attributes, given by a strategy. A strategy σ is an application from 2^G to 2^M which associates a subset of selected attributes $\sigma(A) \subseteq M$ to every $A \subseteq G$.

Selection of attributes: a strategy σ

Definition

Instead of all the possibles attributes in $M \setminus B$, we only consider some attributes, given by a strategy. A strategy σ is an application from 2^G to 2^M which associates a subset of selected attributes $\sigma(A) \subseteq M$ to every $A \subseteq G$.

Immediate-Predecessors((A, D))

begin
  $L \leftarrow \emptyset$;
  forall $b \in M \setminus B$ **do**
    $A' \leftarrow \beta(b) \cap A$;
    if A' *maximal in* L **then** Add A' to L;
  end
  $LP \leftarrow \emptyset$;
  forall $A' \in L$ **do**
    Add $(A', \alpha(A'))$ to LP;
  end
  return LP
end

Selected attributes P

The set of selected attributes is denoted P.
We denote (A, D) a concept of $\langle G, P, I_P \rangle$.

Selection of attributes: a strategy σ

Definition

Instead of all the possibles attributes in $M \setminus B$, we only consider some attributes, given by a strategy. A strategy σ is an application from 2^G to 2^M which associates a subset of selected attributes $\sigma(A) \subseteq M$ to every $A \subseteq G$.

Immediate-Predecessors((A, D) ,σ)

begin
  $L \leftarrow \emptyset$;
  forall $b \in M \setminus B$ **do**
    $A' \leftarrow \beta(b) \cap A$;
    if A' *maximal in* L **then** Add A' to L;
  end
  $LP \leftarrow \emptyset$;
  forall $A' \in L$ **do**
    Add $(A', \alpha(A'))$ to LP;
  end
  return LP
end

Selected attributes P

The set of selected attributes is denoted P.
We denote (A, D) a concept of $\langle G, P, I_P \rangle$.

Selection of attributes: a strategy σ

Definition

Instead of all the possibles attributes in $M \setminus B$, we only consider some attributes, given by a strategy. A strategy σ is an application from 2^G to 2^M which associates a subset of selected attributes $\sigma(A) \subseteq M$ to every $A \subseteq G$.

Immediate-Predecessors((A, D) ,σ)

begin
  $L \leftarrow \emptyset$;
  forall $b \in \sigma(A)$ **do**
    $A' \leftarrow \beta(b) \cap A$;
    if A' *maximal in* $L \wedge A' \subset A$ **then** Add A' to L;
  end
  $LP \leftarrow \emptyset$;
  forall $A' \in L$ **do**
    Add $(A', \alpha(A'))$ to LP;
  end
  return LP
end

Selected attributes P

The set of selected attributes is denoted P.
We denote (A, D) a concept of $\langle G, P, I_P \rangle$.

Selection of attributes: a strategy σ

Definition

Instead of all the possibles attributes in $M \setminus B$, we only consider some attributes, given by a strategy. A strategy σ is an application from 2^G to 2^M which associates a subset of selected attributes $\sigma(A) \subseteq M$ to every $A \subseteq G$.

Immediate-Predecessors((A, D) ,σ)

begin
  $L \leftarrow \emptyset$;
  forall $b \in \sigma(A)$ **do**
    $A' \leftarrow \beta(b) \cap A$;
    if A' *maximal in* $L \wedge A' \subset A$ **then** Add A' to L;
  end
  $LP \leftarrow \emptyset$;
  forall $A' \in L$ **do**
    Add $(A', \alpha(A'))$ to LP;
  end
  return LP
end

Selected attributes P

The set of selected attributes is denoted P.
We denote (A, D) a concept of $\langle G, P, I_P \rangle$.

Selection of attributes: a strategy σ

Definition

Instead of all the possibles attributes in $M \setminus B$, we only consider some attributes, given by a strategy. A strategy σ is an application from 2^G to 2^M which associates a subset of selected attributes $\sigma(A) \subseteq M$ to every $A \subseteq G$.

Immediate-Predecessors((A, D) ,σ)

begin
  $L \leftarrow \emptyset$;
  forall $b \in \sigma(A)$ **do**
    $A' \leftarrow \beta(b) \cap A$;
    if A' *maximal in* $L \wedge A' \subset A$ **then** Add A' to L;
  end
  $LP \leftarrow \emptyset$;
  forall $A' \in L$ **do**
    Add $(A', \alpha(A'))$ to LP;
  end
  return LP
end

Selected attributes P

The set of selected attributes is denoted P.
We denote (A, D) a concept of $\langle G, P, I_P \rangle$.

Constraints

Constraints are needed to ensure that meet are correctly computed.
Constraints associate attributes $\mathcal{C}[A]$ to each subset $A \subseteq G$.

Selection of attributes: a strategy σ

Definition

Instead of all the possibles attributes in $M \setminus B$, we only consider some attributes, given by a strategy. A strategy σ is an application from 2^G to 2^M which associates a subset of selected attributes $\sigma(A) \subseteq M$ to every $A \subseteq G$.

Immediate-Predecessors((A, D) ,σ)

begin
  $L \leftarrow \emptyset$;
  forall $b \in \sigma(A) \cup \mathcal{C}[A]$ **do**
    $A' \leftarrow \beta(b) \cap A$;
    if A' *maximal in* $L \wedge A' \subset A$ **then** Add A' to L;
  end
  $LP \leftarrow \emptyset$;
  forall $A' \in L$ **do**
    Add $(A', \alpha(A'))$ to LP;
  end
  return LP
end

Selected attributes P

The set of selected attributes is denoted P.
We denote (A, D) a concept of $\langle G, P, I_P \rangle$.

Constraints

Constraints are needed to ensure that meet are correctly computed.
Constraints associate attributes $\mathcal{C}[A]$ to each subset $A \subseteq G$.

Selection of attributes: a strategy σ

Definition

Instead of all the possibles attributes in $M \setminus B$, we only consider some attributes, given by a strategy. A strategy σ is an application from 2^G to 2^M which associates a subset of selected attributes $\sigma(A) \subseteq M$ to every $A \subseteq G$.

Immediate-Predecessors((A, D) ,σ)

begin
  $L \leftarrow \emptyset$;
  forall $b \in \sigma(A) \cup \mathcal{C}[A]$ **do**
    $A' \leftarrow \beta(b) \cap A$;
    if A' *maximal in* $L \wedge A' \subset A$ **then** Add A' to L;
  end
  $LP \leftarrow \emptyset$;
  forall $A' \in L$ **do**
    Add $(A', \alpha(A'))$ to LP;
    Compute the cross and residual constraints $\mathcal{C}[A']$
  end
  return LP
end

Selected attributes P

The set of selected attributes is denoted P.
We denote (A, D) a concept of $\langle G, P, I_P \rangle$.

Constraints

Constraints are needed to ensure that meet are correctly computed.
Constraints associate attributes $\mathcal{C}[A]$ to each subset $A \subseteq G$.

Selection of attributes: a strategy σ

Definition

Instead of all the possibles attributes in $M \setminus B$, we only consider some attributes, given by a strategy. A strategy σ is an application from 2^G to 2^M which associates a subset of selected attributes $\sigma(A) \subseteq M$ to every $A \subseteq G$.

Immediate-Predecessors((A, D) ,σ)

begin
  $L \leftarrow \emptyset$;
  forall $b \in \sigma(A) \cup \mathcal{C}[A]$ **do**
    $A' \leftarrow \beta(b) \cap A$;
    if A' *maximal in* $L \wedge A' \subset A$ **then** Add A' to L;
  end
  $LP \leftarrow \emptyset$;
  forall $A' \in L$ **do**
    Add $(A', \alpha(A'))$ to LP;
    Compute the cross and residual constraints $\mathcal{C}[A']$
  end
  return LP
end

Selected attributes P

The set of selected attributes is denoted P.
We denote (A, D) a concept of $\langle G, P, I_P \rangle$.

Constraints

Constraints are needed to ensure that meet are correctly computed.
Constraints associate attributes $\mathcal{C}[A]$ to each subset $A \subseteq G$.

Selection of concepts: a priority queue

Concepts($\langle G, M, (\alpha, \beta)\rangle$)

begin
  top $\leftarrow (G, \alpha(G))$;
  Add top to a queue Q;
  while Q **not empty do**
    $(A,\ B\ ) \leftarrow Q$.pop();
    produce $(A,\ B\ )$;
    $LP \leftarrow$ Immediate-Predecessors$((A,\ B\ ))$;
    forall $(A',\ B'\ ) \in LP$ **do**
      Add $(A',\ B'\ )$ to Q;
    end
  end
end

Selection of concepts: a priority queue

Concepts($\langle G, M, (\alpha, \beta)\rangle$)

begin
  top $\leftarrow (G, \alpha(G))$;
  Add top to a queue Q;
  while Q **not empty do**
    $(A,\ B\ ) \leftarrow Q$.pop();
    produce $(A,\ B\ )$;
    $LP \leftarrow$ Immediate-Predecessors$((A,\ B\ ))$;
    forall $(A',\ B'\ ) \in LP$ **do**
      Add $(A',\ B'\ )$ to Q;
    end
  end
end

Strategy

The strategy σ is given as input of the main algorithm.

Selection of concepts: a priority queue

Concepts($\langle G, M, (\alpha, \beta)\rangle$,σ)

begin
  top $\leftarrow (G, \alpha(G))$;
  Add top to a queue Q;
  while Q **not empty do**
    $(A, D) \leftarrow Q$.pop();
    produce (A, D);
    $LP \leftarrow$ Immediate-Predecessors((A, D) ,σ);
    forall $(A', D') \in LP$ **do**
      Add (A', D') to Q;
    end
  end
end

Strategy

The strategy σ is given as input of the main algorithm.

Selection of concepts: a priority queue

Concepts($\langle G, M, (\alpha, \beta)\rangle$,σ)

begin
  top $\leftarrow (G, \alpha(G))$;
  Add top to a queue Q;
  while Q **not empty do**
    $(A,\ D\ ) \leftarrow Q$.pop();
    produce $(A,\ D\ )$;
    $LP \leftarrow$ Immediate-Predecessors($(A,\ D\ )$,σ);
    forall $(A',\ D'\ ) \in LP$ **do**
      Add $(A',\ D'\ )$ to Q;
    end
  end
end

Strategy

The strategy σ is given as input of the main algorithm.

The priority queue Q

We use a priority queue according to the support of concepts to ensure that concepts are generated level by level, i.e. each concept is generated before its predecessors.

Selection of concepts: a priority queue

Concepts($\langle G, M, (\alpha, \beta)\rangle$,σ)

begin
  top $\leftarrow (G, \alpha(G))$;
  Add ($|G|$, top) to a priority queue Q;
  while Q **not empty do**
    $(A,\ D\ ) \leftarrow Q$.pop();
    produce $(A,\ D\ )$;
    $LP \leftarrow$ Immediate-Predecessors($(A,\ D\ )$,σ);
    forall $(A',\ D'\ ) \in LP$ **do**
      Add ($|A'|$,$(A',\ D'\ )$) to Q;
    end
  end
end

Strategy

The strategy σ is given as input of the main algorithm.

The priority queue Q

We use a priority queue according to the support of concepts to ensure that concepts are generated level by level, i.e. each concept is generated before its predecessors.

Selection of concepts: a priority queue

Concepts($\langle G, M, (\alpha, \beta)\rangle$,σ)

begin
  top $\leftarrow (G, \alpha(G))$;
  Add ($|G|$, top) to a priority queue Q;
  while Q **not empty do**
    $(A,\ D\ ) \leftarrow Q$.pop();
    produce $(A,\ D\ )$;
    $LP \leftarrow$ Immediate-Predecessors($(A,\ D\ )$,σ);
    forall $(A',\ D'\ ) \in LP$ **do**
      Add ($|A'|$, $(A',\ D'\ )$) to Q;
    end
  end
end

Strategy

The strategy σ is given as input of the main algorithm.

The priority queue Q

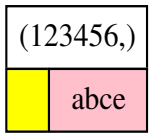
We use a priority queue according to the support of concepts to ensure that concepts are generated level by level, i.e. each concept is generated before its predecessors.

Example

Sample data

(α, β)	a	b	c	d	e
1	✓	✓	✓	✓	
2	✓	✓	✓		✓
3	✓	✓			✓
4			✓	✓	✓
5		✓	✓		
6	✓		✓		

- $\sigma(A) = \{b \in M | \text{Conf}(\alpha(A) + b) \geq 0.5\}$
- constraints
- current concept

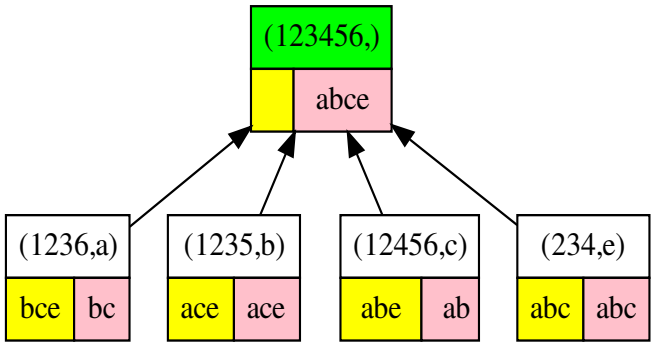


Example

Sample data

(α, β)	a	b	c	d	e
1	✓	✓	✓	✓	
2	✓	✓	✓		✓
3	✓	✓			✓
4			✓	✓	✓
5		✓	✓		
6	✓		✓		

- $\sigma(A) = \{b \in M | \text{Conf}(\alpha(A) + b) \geq 0.5\}$
- constraints
- current concept

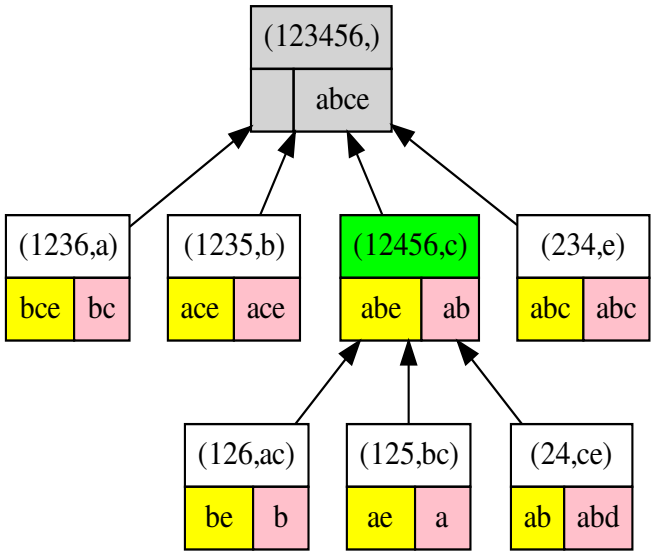


Example

Sample data

(α, β)	a	b	c	d	e
1	✓	✓	✓	✓	
2	✓	✓	✓		✓
3	✓	✓			✓
4			✓	✓	✓
5		✓	✓		
6	✓		✓		

- $\sigma(A) = \{b \in M | \text{Conf}(\alpha(A) + b) \geq 0.5\}$
- constraints
- current concept

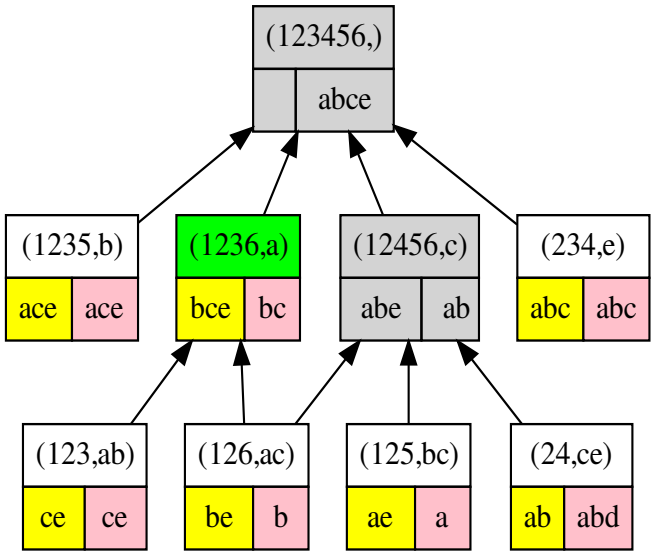


Example

Sample data

(α, β)	a	b	c	d	e
1	✓	✓	✓	✓	
2	✓	✓	✓		✓
3	✓	✓			✓
4			✓	✓	✓
5		✓	✓		
6	✓		✓		

- $\sigma(A) = \{b \in M | \text{Conf}(\alpha(A) + b) \geq 0.5\}$
- constraints
- current concept

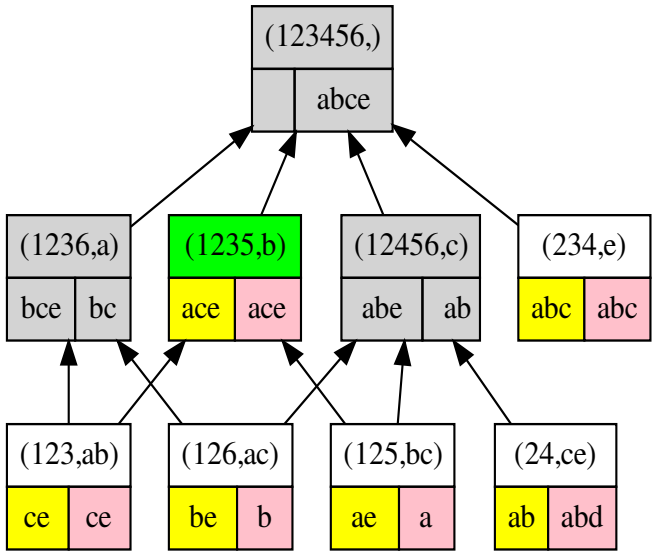


Example

Sample data

(α, β)	a	b	c	d	e
1	✓	✓	✓	✓	
2	✓	✓	✓		✓
3	✓	✓			✓
4			✓	✓	✓
5		✓	✓		
6	✓		✓		

- $\sigma(A) = \{b \in M | \text{Conf}(\alpha(A) + b) \geq 0.5\}$
- constraints
- current concept

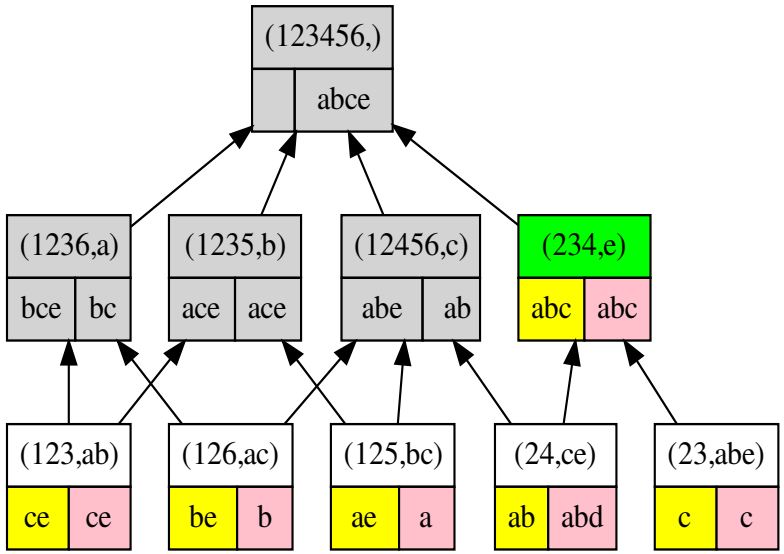


Example

Sample data

(α, β)	a	b	c	d	e
1	✓	✓	✓	✓	
2	✓	✓	✓		✓
3	✓	✓			✓
4			✓	✓	✓
5		✓	✓		
6	✓		✓		

- $\sigma(A) = \{b \in M | \text{Conf}(\alpha(A) + b) \geq 0.5\}$
- constraints
- current concept

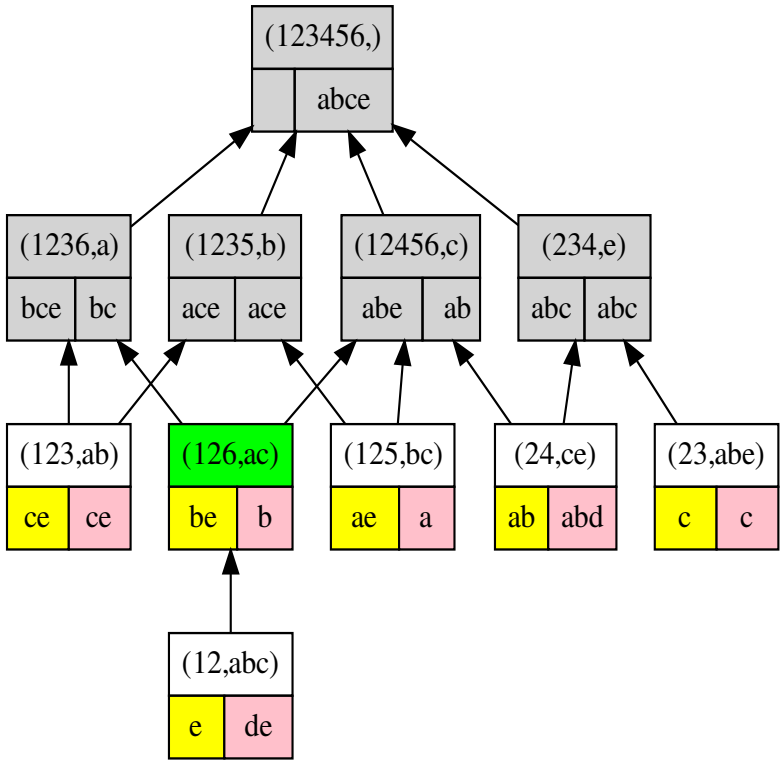


Example

Sample data

(α, β)	a	b	c	d	e
1	✓	✓	✓	✓	
2	✓	✓	✓		✓
3	✓	✓			✓
4			✓	✓	✓
5		✓	✓		
6	✓		✓		

- $\sigma(A) = \{b \in M | \text{Conf}(\alpha(A) + b) \geq 0.5\}$
- constraints
- current concept

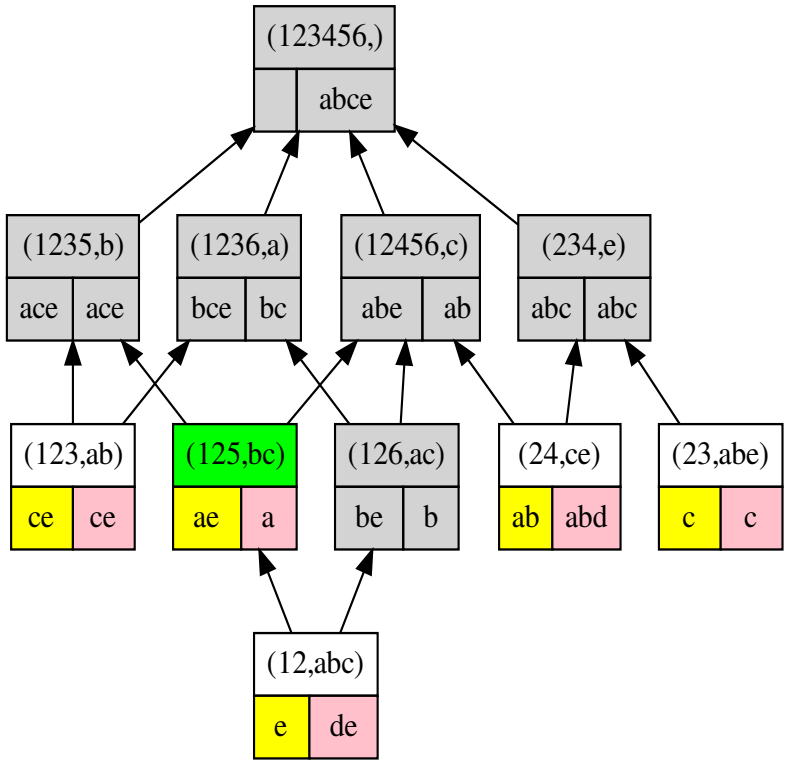


Example

Sample data

(α, β)	a	b	c	d	e
1	✓	✓	✓	✓	
2	✓	✓	✓		✓
3	✓	✓			✓
4			✓	✓	✓
5		✓	✓		
6	✓		✓		

- $\sigma(A) = \{b \in M | \text{Conf}(\alpha(A) + b) \geq 0.5\}$
- constraints
- current concept

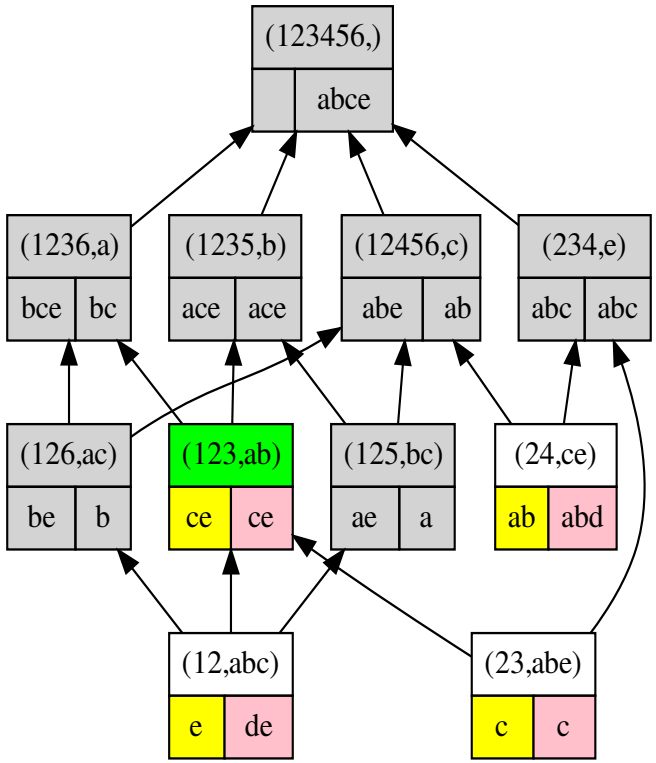


Example

Sample data

(α, β)	a	b	c	d	e
1	✓	✓	✓	✓	
2	✓	✓	✓		✓
3	✓	✓			✓
4			✓	✓	✓
5		✓	✓		
6	✓		✓		

- $\sigma(A) = \{b \in M | \text{Conf}(\alpha(A) + b) \geq 0.5\}$
- constraints
- current concept

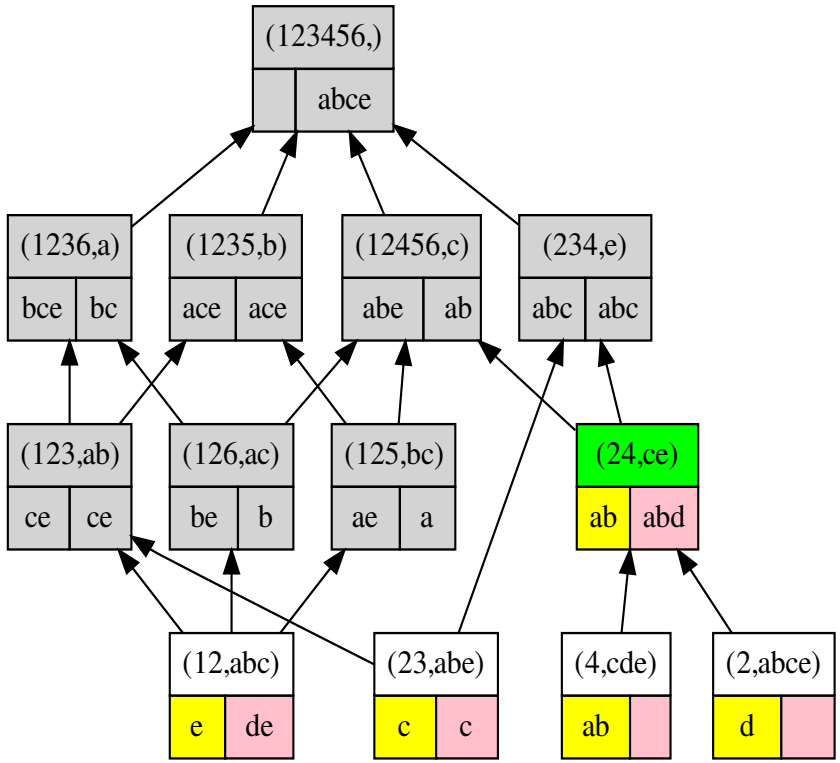


Example

Sample data

(α, β)	a	b	c	d	e
1	✓	✓	✓	✓	
2	✓	✓	✓		✓
3	✓	✓			✓
4			✓	✓	✓
5		✓	✓		
6	✓		✓		

- $\sigma(A) = \{b \in M | \text{Conf}(\alpha(A) + b) \geq 0.5\}$
- constraints
- current concept

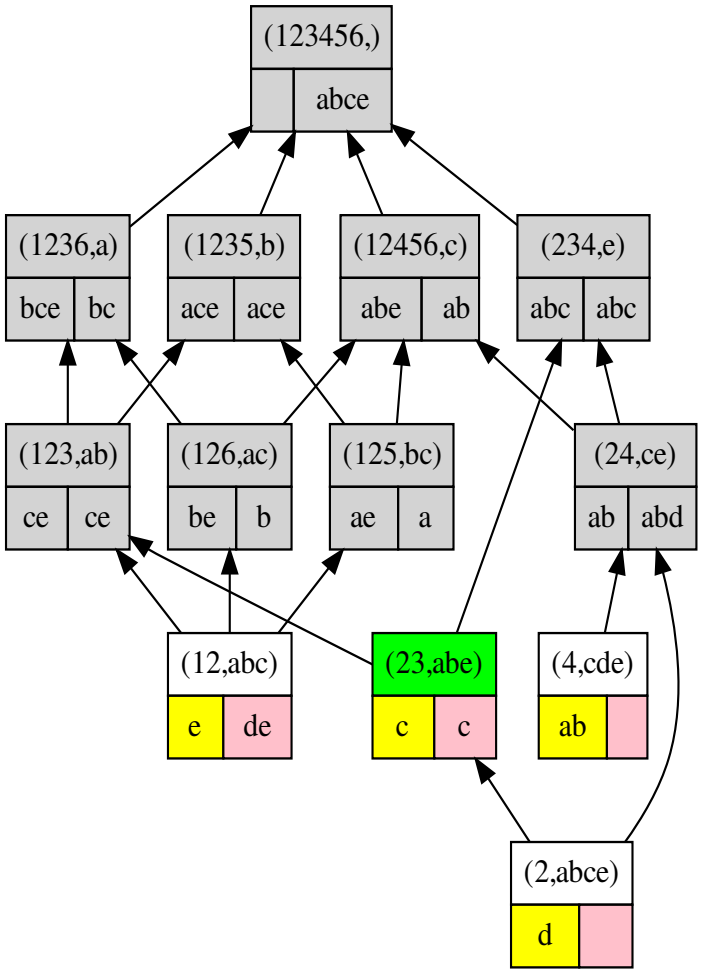


Example

Sample data

(α, β)	a	b	c	d	e
1	✓	✓	✓	✓	
2	✓	✓	✓		✓
3	✓	✓			✓
4			✓	✓	✓
5		✓	✓		
6	✓		✓		

- $\sigma(A) = \{b \in M | \text{Conf}(\alpha(A) + b) \geq 0.5\}$
- constraints
- current concept

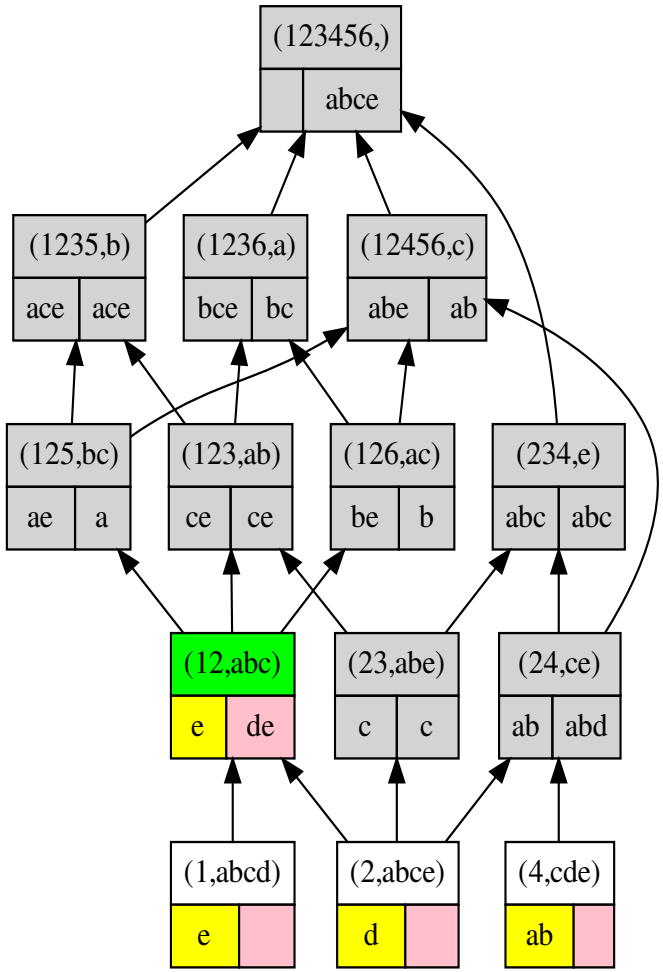


Example

Sample data

(α, β)	a	b	c	d	e
1	✓	✓	✓	✓	
2	✓	✓	✓		✓
3	✓	✓			✓
4			✓	✓	✓
5		✓	✓		
6	✓		✓		

- $\sigma(A) = \{b \in M | \text{Conf}(\alpha(A) + b) \geq 0.5\}$
- constraints
- current concept

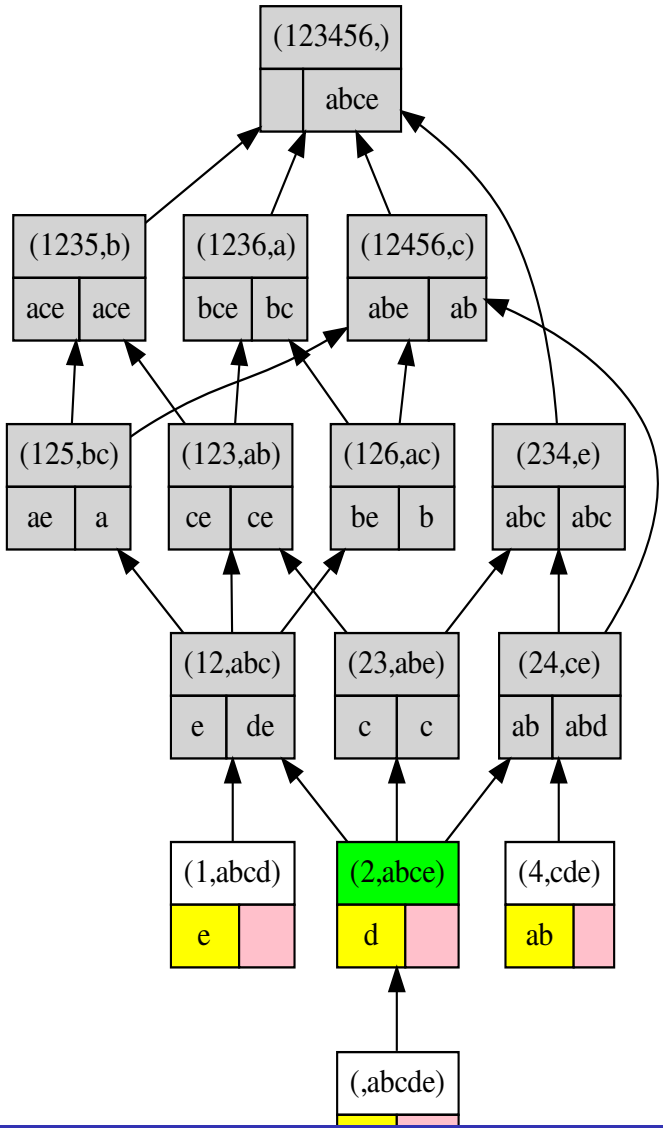


Example

Sample data

(α, β)	a	b	c	d	e
1	✓	✓	✓	✓	
2	✓	✓	✓		✓
3	✓	✓			✓
4			✓	✓	✓
5		✓	✓		
6	✓		✓		

- $\sigma(A) = \{b \in M | \text{Conf}(\alpha(A) + b) \geq 0.5\}$
- constraints
- current concept

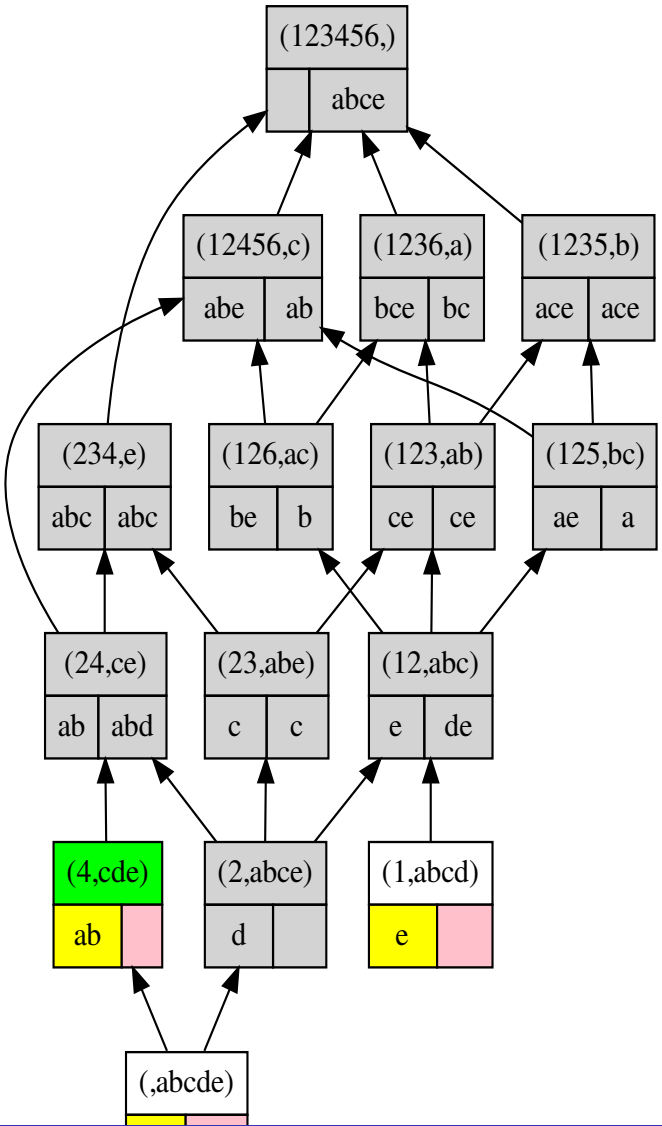


Example

Sample data

(α, β)	a	b	c	d	e
1	✓	✓	✓	✓	
2	✓	✓	✓		✓
3	✓	✓			✓
4			✓	✓	✓
5		✓	✓		
6	✓		✓		

- $\sigma(A) = \{b \in M | \text{Conf}(\alpha(A) + b) \geq 0.5\}$
- constraints
- current concept

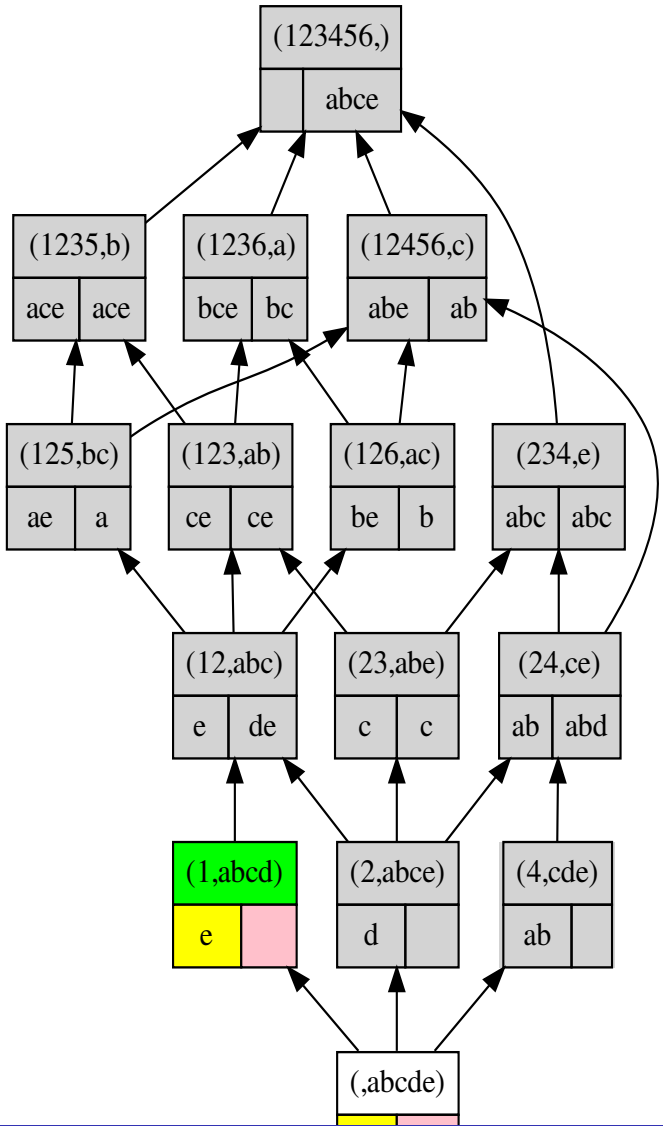


Example

Sample data

(α, β)	a	b	c	d	e
1	✓	✓	✓	✓	
2	✓	✓	✓		✓
3	✓	✓			✓
4			✓	✓	✓
5		✓	✓		
6	✓		✓		

- $\sigma(A) = \{b \in M | \text{Conf}(\alpha(A) + b) \geq 0.5\}$
- constraints
- current concept

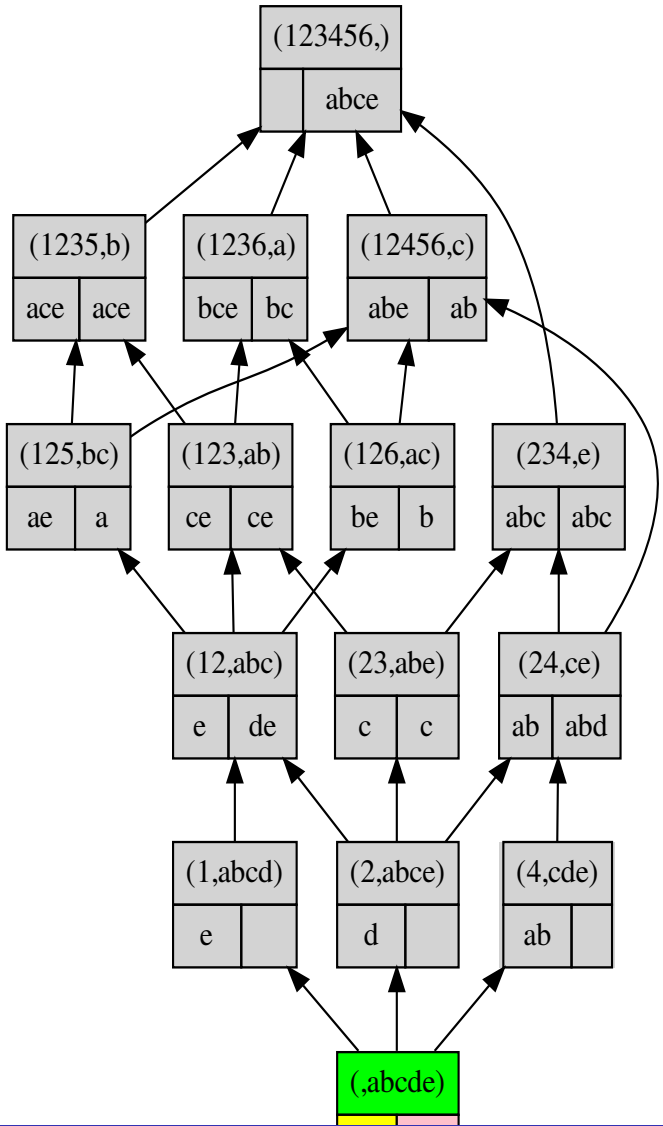


Example

Sample data

(α, β)	a	b	c	d	e
1	✓	✓	✓	✓	
2	✓	✓	✓		✓
3	✓	✓			✓
4			✓	✓	✓
5		✓	✓		
6	✓		✓		

- $\sigma(A) = \{b \in M | \text{Conf}(\alpha(A) + b) \geq 0.5\}$
- constraints
- current concept

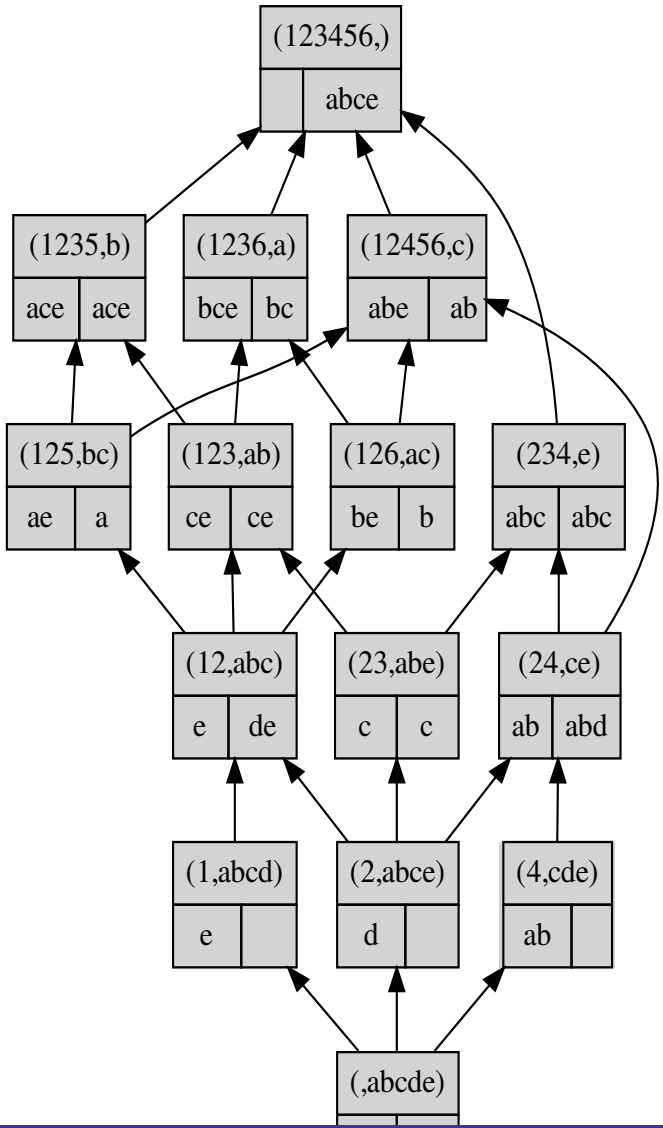


Example

Sample data

(α, β)	a	b	c	d	e
1	✓	✓	✓	✓	
2	✓	✓	✓		✓
3	✓	✓			✓
4			✓	✓	✓
5		✓	✓		
6	✓		✓		

- $\sigma(A) = \{b \in M | \text{Conf}(\alpha(A) + b) \geq 0.5\}$
- constraints
- current concept

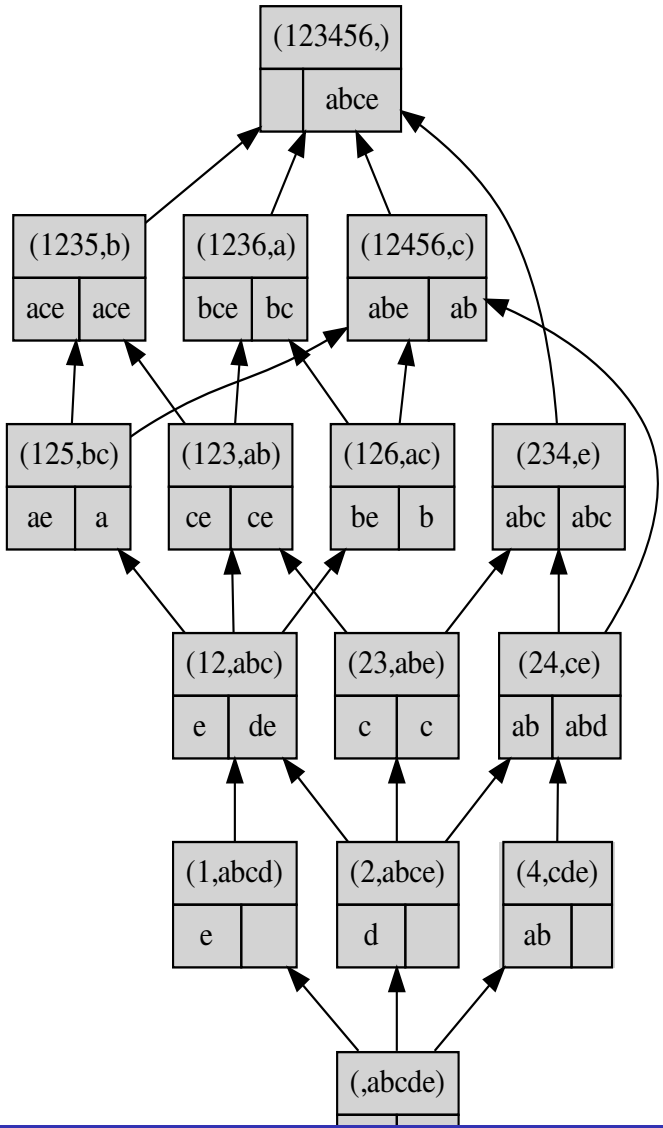


Example

Sample data

(α_P, β_P)	a	b	c	d|abc	d|ce	e
1	✓	✓	✓	✓		
2	✓	✓	✓			✓
3	✓	✓				✓
4			✓		✓	✓
5		✓	✓			
6	✓		✓			

- $\sigma(A) = \{b \in M | \text{Conf}(\alpha(A) + b) \geq 0.5\}$
- constraints
- current concept



Generic description by predicates for heterogeneous data

Concepts$(\langle G, S, (S^i, \sigma^i, \delta^i))\rangle)$

begin
  top $\leftarrow (G, \delta(G))$;
  Add $(|G|, \text{top})$ to a priority queue Q;
  while Q **not empty do**
    $(A, D) \leftarrow Q$.pop();
    produce (A, D);
    $LP \leftarrow$ Immediate-Predecessors$((A, D), \sigma, \delta)$;
    forall $(A', D') \in LP$ **do**
      Add $(|A'|, (A', D'))$ to Q;
    end
  end
end

Generic description by predicates for heterogeneous data

Concepts($\langle G, S, (S^i, \sigma^i, \delta^i)\rangle$)

begin
  top $\leftarrow (G, \delta(G))$;
  Add $(|G|, \text{top})$ to a priority queue Q;
  while Q **not empty do**
    $(A, D) \leftarrow Q$.pop();
    produce (A, D);
    $LP \leftarrow$ Immediate-Predecessors$((A, D), \sigma, \delta)$;
    forall $(A', D') \in LP$ **do**
      Add $(|A'|, (A', D'))$ to Q;
    end
  end
end

Groups of charateristics

Charateristics are given by a **family** (S^i) where each S^i contains charateristics of the same domain.

Generic description by predicates for heterogeneous data

Concepts$(\langle G, S, (S^i, \sigma^i, \delta^i)\rangle)$

begin
  top $\leftarrow (G, \delta(G))$;
  Add $(|G|, \text{top})$ to a priority queue Q;
  while Q **not empty do**
    $(A, D) \leftarrow Q.\text{pop}()$;
    produce (A, D);
    $LP \leftarrow$ Immediate-Predecessors$((A, D), \sigma, \delta)$;
    forall $(A', D') \in LP$ **do**
      Add $(|A'|, (A', D'))$ to Q;
    end
  end
end

Groups of charateristics

Charateristics are given by a **family** (S^i) where each S^i contains charateristics of the same domain.

Descriptions and predicates

Each group of charateristics S^i is provided with a **description** δ^i of objects by predicates. A description is an application which associates a subset of predicates $\delta(A)$ describing a subset of objects $A \subseteq G$.

Generic description by predicates for heterogeneous data

Concepts($\langle G, S, (S^i, \sigma^i, \delta^i)\rangle$)

begin
  top $\leftarrow (G, \delta(G))$;
  Add $(|G|, \text{top})$ to a priority queue Q;
  while Q **not empty do**
    $(A, D) \leftarrow Q$.pop();
    produce (A, D);
    $LP \leftarrow$ Immediate-Predecessors$((A, D), \sigma, \delta)$;
    forall $(A', D') \in LP$ **do**
      Add $(|A'|, (A', D'))$ to Q;
    end
  end
end

Groups of charateristics

Charateristics are given by a **family** (S^i) where each S^i contains charateristics of the same domain.

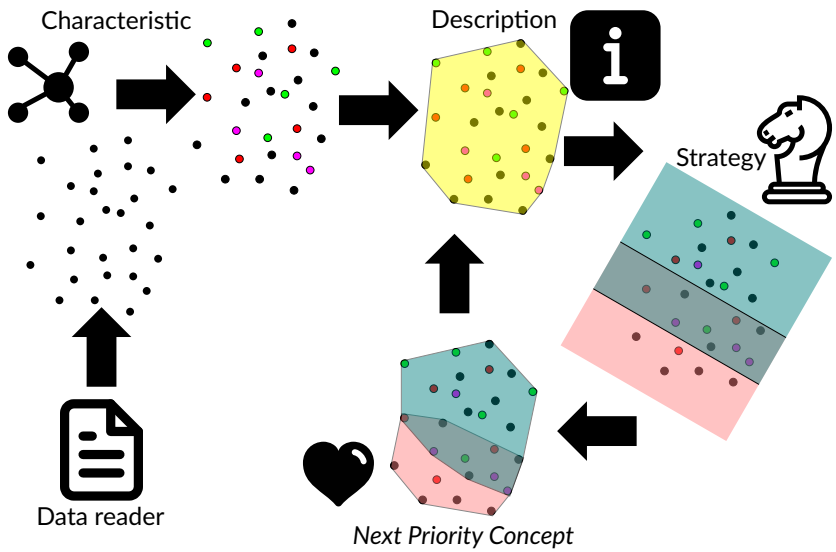
Descriptions and predicates

Each group of charateristics S^i is provided with a **description** δ^i of objects by predicates. A description is an application which associates a subset of predicates $\delta(A)$ describing a subset of objects $A \subseteq G$.

Strategies

Each group of charateristics S^i is provided with a **strategy** σ^i which defines a set of predicates from which the predecessors are generated.

Descriptions and strategies for heterogenous data



Description

The description $\delta(A)$ is composed of predicates describing the borders of the **generalized** convex hull of A

Strategy

The strategy $\sigma(A)$ is composed of predicates describing a "cut" of the **generalized** convex hull of A from which the predecessors are generated.

NextPriorityConcept

Remark

The NextPriorityConcept algorithm is a **pattern discovery** approach where each $(S^i, \sigma^i, \delta^i)$ corresponds to a pattern structure:

- the description δ^i corresponds to the patterns given by predicates
  $=>$ **heterogeneous data are possible**
- the strategy σ^i allows a predecessor generation "on the fly" for each subsets A of objects
  $=>$ **discovered patterns are more suited to the data**

NextPriorityConcept

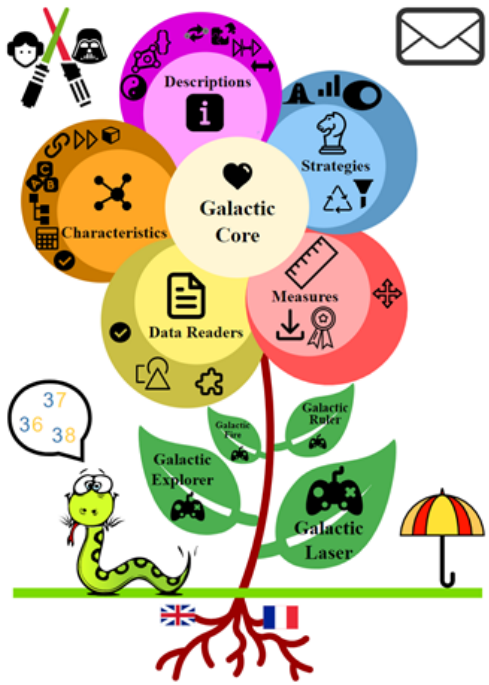
Theorem (Demko et al. 2020)

If each description δ^i verifies $\delta^i(A) \sqsubseteq \delta^i(A')$ for $A' \subseteq A$ then:
The NextPriorityConcept algorithm computes the concept lattice of (G, P, δ) where P is the set of predicates issued from the descriptions.

GALACTIC

GALACTIC: Platform written in python, fully extensible

- A **Core** library, with an implementation of NextPriorityConcept
- An eco-systems of plugins:
  - data readers plugins
  - characteristics plugins
  - description plugins
  - strategies plugins
  - measures plugins
- Applications
  - Galactic-laser
  - Galactic-ruler



(demonstration on thursday afternoon)

And what about rules ?

Canonical direct basis of rules

- Rules are composed of set of predicates in premise and conclusion
- The minimal generators of a concepts corresponds to "the borders" of the convex hull
- Only the irreducible predicates of P must be considered

For example, the predicate *"is multiple of 12"* is not irreducible when the two predicates *"is multiple of 3"* and *"is multiple of 4"* have been generated

"is multiple of 3" and *"is multiple of 4"* $\Rightarrow$ *"is multiple of 12"* ?

Example dataset

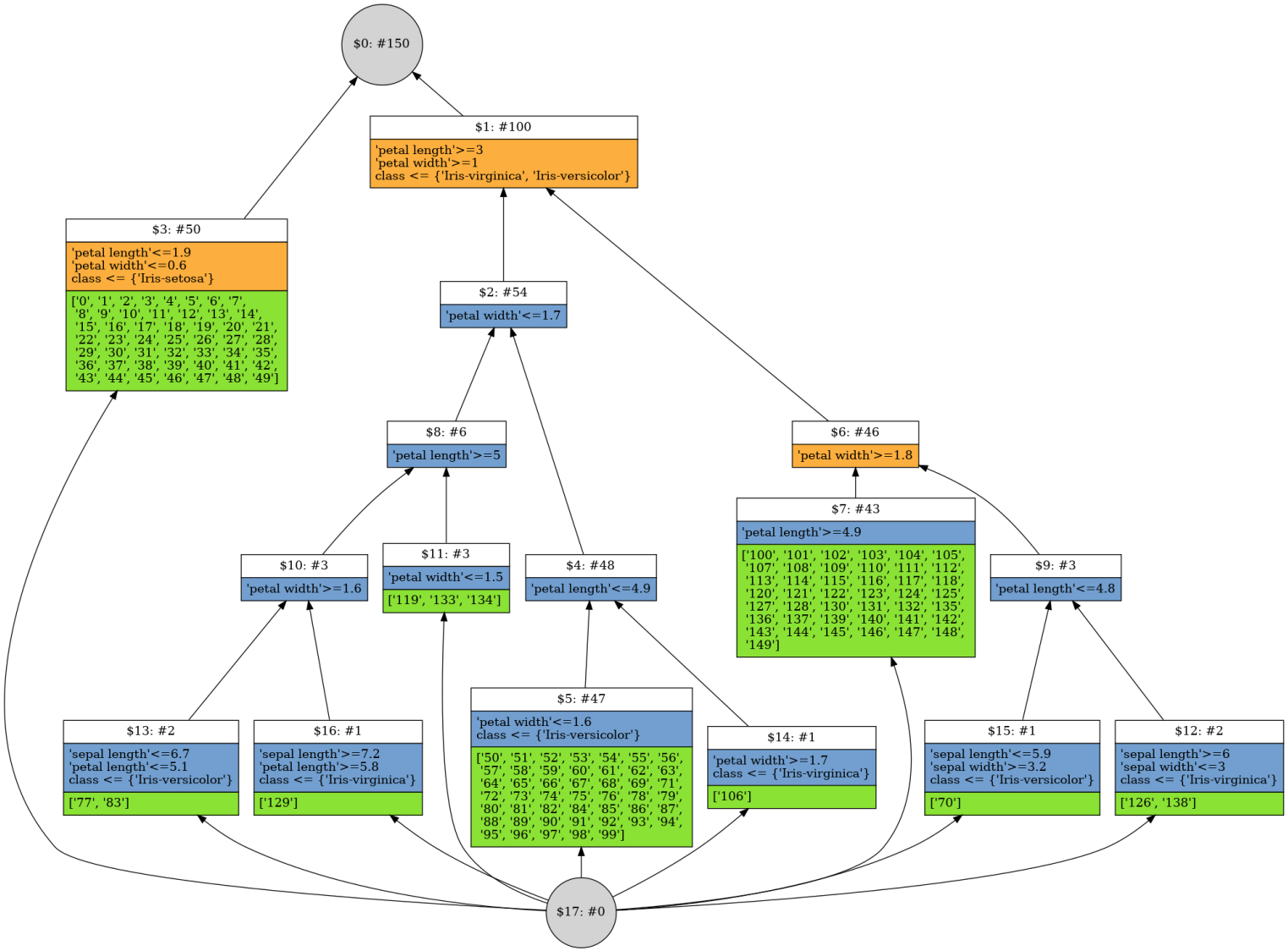
Iris dataset (Fisher, 1930)

- 150 individuals
- 4 numerical parameters and one modal parameter
- the whole lattice contains over 6 millions concepts

Strategies

Guided by the user providing the strategies, the NextPriorityConcept algorithm produces a lattice of 18 concepts

Example lattice



Example rules

Top rule

```
if (Support 1, Confidence 1)
then
    'sepal length'>=4.3
    'sepal length'<=7.9
    'sepal width'>=2
    'sepal width'<=4.4
    'petal length'>=1
    'petal length'<=6.9
    'petal width'>=0.1
    'petal width'<=2.5
    class <= {'virginica', 'setosa', 'versicolor'}
```

Example rules

Second rule

```
if (Support 0.666667, Confidence 1)
    'petal length'>=3
then
    'sepal length'>=4.9
    'sepal length'<=7.9
    'sepal width'>=2
    'sepal width'<=3.8
    'petal length'<=6.9
    'petal width'>=1
    'petal width'<=2.5
    class <= {'virginica', 'versicolor'}
```

Example rules

Specific rule

```
if (Support 0.333333, Confidence 1)
    'petal length'<=1.9
then
    'sepal length'>=4.3
    'sepal length'<=5.8
    'sepal width'>=2.3
    'sepal width'<=4.4
    'petal length'>=1
    'petal width'>=0.1
    'petal width'<=0.6
    class <= {'setosa'}
```

Table of contents

How to deduce new knowledge?

Manuel Enciso

Angel Mora

Malaga's Team, University of Malaga, Spain

If-Then rules are popular in Computer Science

- Modus Ponens
- Horn Clauses
- Functional Dependencies
- Implications


ENCYCLOPÉDIE,
OU
DICTIONNAIRE RAISONNÉ
DES SCIENCES,
DES ARTS ET DES MÉTIERS,
RECUEILLI
DES MEILLEURS AUTEURS
ET PARTICULIEREMENT

If-Then rules are popular in Computer Science

- Modus Ponens
- Horn Clauses
- Functional Dependencies
- Implications

$$\texttt{[MP]}\ \frac{p \rightarrow q, p}{\therefore q}$$

If-Then rules are popular in Computer Science

- Modus Ponens
- Horn Clauses
- Functional Dependencies
- Implications

If-Then rules are popular in Computer Science

- Modus Ponens
- Horn Clauses
- Functional Dependencies
- Implications

$$u \leftarrow q, p, \ldots t$$

If-Then rules are popular in Computer Science

- Modus Ponens
- Horn Clauses
- Functional Dependencies
- Implications

UTD

Example

SSN	LastName	FirstName	City
111111111	Smith	John	Richardson
222222222	Li	Peng	Richardson
333333333	Kant	John	Plano
444444444	Smith	Mark	Plano

- Does {ssn} → {LastName} hold?
- Does {ssn} → {LastName,FirstName} hold ?
- Does {LastName, FirstName}→ {City} hold?
- Does {City} → {FirstName} hold?

If-Then rules are popular in Computer Science

- Modus Ponens
- Horn Clauses
- Functional Dependencies
- Implications

If-Then rules are popular in Computer Science

- Modus Ponens
- Horn Clauses
- Functional Dependencies
- Implications

If-Then rules are popular in Computer Science

- Modus Ponens
- Horn Clauses
- Functional Dependencies
- Implications

If-Then rules are popular in Computer Science

- Modus Ponens
- Horn Clauses
- Functional Dependencies
- Implications

If-Then rules are popular in Computer Science

- Modus Ponens
- Horn Clauses
- Functional Dependencies
- Implications

If-Then rules interpretations

- Classical Logic: validity.
- Funcional Dependencies: value matching.
- Association rules: probabilistic measure.
- FCA Implications: *epistemic interpretation*.

If-Then rules interpretations

- Classical Logic: validity.
- Funcional Dependencies: value matching.
- Association rules: probabilistic measure.
- FCA Implications: *epistemic interpretation*.

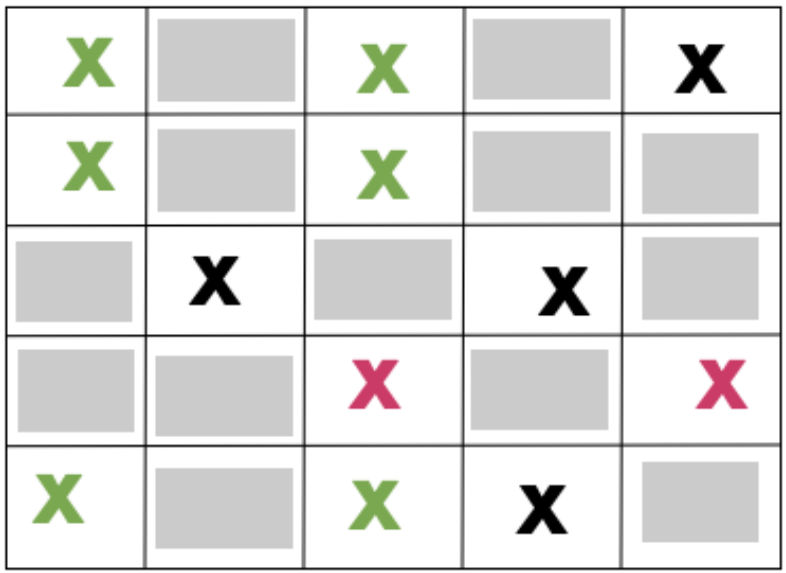
X		X		X
X		X		
	X		X	
		X		X
X		X	X	

If-Then rules interpretations

- Classical Logic: validity.
- Funcional Dependencies: value matching.
- Association rules: probabilistic measure.
- FCA Implications: *epistemic interpretation*.

If-Then rules interpretations

- Classical Logic: validity.
- Funcional Dependencies: value matching.
- Association rules: probabilistic measure.
- FCA Implications: *epistemic interpretation*.



If-Then rules interpretations

- Classical Logic: validity.
- Funcional Dependencies: value matching.
- Association rules: probabilistic measure.
- FCA Implications: *epistemic interpretation*.

Axiomatic Systems

Armstrong Axioms

Reflexivity	[Ref]		$\vdash_{\mathcal{A}} AB \to A$
Augmentation	[Aug]	$A \to B$	$\vdash_{\mathcal{A}} AC \to BC$
Transitivity	[Tran]	$A \to B, B \to C$	$\vdash_{\mathcal{A}} A \to C$

Armstrong Axioms

The system shows how rules works (semantics).

Decomposition	[Dec]	$A \to BC$	$\vdash_{\mathcal{A}} A \to B$
Composition	[Com]	$\{A \to B, C \to D\}$	$\vdash_{\mathcal{A}} AC \to BD$
Union	[Uni]	$A \to B, A \to C$	$\vdash_{\mathcal{A}} A \to BC$
Pseudo Transitivity	[Pse]	$A \to B, BC \to D$	$\vdash_{\mathcal{A}} AC \to D$
Sel	[Tran]		$\vdash_{\mathcal{A}} A \to A$

Axiomatic System for implications

Theorem

Armstrong axioms is a sound and complete axiomatic system for FCA implications.

Logic for implications. Simplification logic.

One axiom scheme

[Ref] $\vdash_{\mathcal{S}} A \to A$;

And the following inference rules:

[Frag] $A \to BC \vdash_{\mathcal{S}} A \to B$;

[Comp] $A \to B,\ C \to D \vdash_{\mathcal{S}} AC \to BD$;

[Simp] If $A \subseteq C$ and $A \cap B = \varnothing$, then
$A \to B,\ C \to D \vdash_{\mathcal{S}} C - B \to D - B$.

Other extensions

Negative attributes

Negative attributes

[Ref] If $A \supseteq B$ then $\vdash A \to B$

[Augm] $A \to B \vdash AC \to BC$

[Trans] $A \to B,\ B \to C \vdash A \to C$

Plus the following two rules:

[Cont] $\vdash a\overline{a} \to Y\overline{Y}$.

[Rft] $Aa \to b \vdash A\overline{b} \to \overline{a}$.

The closure of A wrt Γ, denoted as A^+, is the biggest set such that $\Gamma \vdash A \to A^+$.

Therefore,

$$\Gamma \vdash A \to B \text{ if and only } B \subseteq A^+$$

Fuzzy implications

[Belohlavek R. and Vychodil V.]

Fuzzy implications

Grades

$\{^{0.2}/y_1, y_2\} \Rightarrow \{^{0.8}/y_3\}$

Object-attribute data with grades

Every object that has attribute y_1 to degree at least 0.2 and attribute y_2 to degree 1, has also attribute y_3 to degree at least 0.8

Ranked tables over domains with similarities

Every two tuples that are similar on attribute y_1 to degree at least 0.2 and are equal on attribute y_2 are similar on attribute y_3 to degree at least 0.8

Fuzzy implications

FFDs over Ranked data tables with grades

- Axioms: $\vdash AB \Rightarrow A.$
- Cut rule: $\{A \Rightarrow B, BC \Rightarrow D\} \vdash AC \Rightarrow D.$
- Multiplication rule: $\{A \Rightarrow B\} \vdash c^* \otimes A \Rightarrow c^* \otimes B.$

Fuzzy implications

Fuzzy Attribute Simplification Logic

- Axioms: $\vdash AB \Rightarrow A.$
- Simplification rule: $\{A \Rightarrow B, C \Rightarrow D\} \vdash A(C \smallsetminus B) \Rightarrow D.$
- Multiplication rule: $\{A \Rightarrow B\} \vdash c^* \otimes A \Rightarrow c^* \otimes B.$

Derived rules

- Decomposition rule: $\{A \Rightarrow BC\} \vdash A \Rightarrow B.$
- Composition rule: $\{A \Rightarrow B, C \Rightarrow D\} \vdash AC \Rightarrow BD.$

Triadic implications

Triadic implications

Conditional Attribute Implication

The CAIL axiomatic system consists of the following rules:

[Non-constraint] $\vdash_{\mathscr{C}} \emptyset \xrightarrow{\emptyset} \Omega$.

[Inclusion] $\vdash_{\mathscr{C}} XY \xrightarrow{\Gamma} X$.

[Augmentation] $X \xrightarrow{\mathcal{C}} Y \vdash_{\mathscr{C}} XZ \xrightarrow{\mathcal{C}} YZ$.

[Transitivity] $\{X \xrightarrow{\mathcal{C}_1} Y, Y \xrightarrow{\mathcal{C}_2} Z\} \vdash_{\mathscr{C}} X \xrightarrow{\mathcal{C}_1 \cap \mathcal{C}_2} Z$.

[Conditional Decomposition] $X \xrightarrow{\mathcal{C}_1\mathcal{C}_2} Y \vdash_{\mathscr{C}} X \xrightarrow{\mathcal{C}_1} Y$.

[Conditional Composition]

$$\{X \xrightarrow{\mathcal{C}_1} Y, Z \xrightarrow{\mathcal{C}_2} W\} \vdash_{\mathscr{C}} XZ \xrightarrow{\mathcal{C}_1\mathcal{C}_2} Y \cap W.$$

Simplification Logic for *CAIs*

The CAISL axiomatic system has two axiom schemes:

[Non-constraint] $\vdash_{\mathscr{S}} \emptyset \xrightarrow{\emptyset} \Omega$.

[Reflexivity] $\vdash_{\mathscr{S}} X \xrightarrow{\Gamma} X$.

and four inference rules:

[Decomposition] $\{X \xrightarrow{\mathcal{C}_1\mathcal{C}_2} YZ\} \vdash_{\mathscr{S}} X \xrightarrow{\mathcal{C}_1} Y$.

[Composition] $\{X \xrightarrow{\mathcal{C}_1} Y, Z \xrightarrow{\mathcal{C}_2} W\} \vdash_{\mathscr{S}} XZ \xrightarrow{\mathcal{C}_1 \cap \mathcal{C}_2} YW$.

[Conditional Composition]

$$\{X \xrightarrow{\mathcal{C}_1} Y, Z \xrightarrow{\mathcal{C}_2} W\} \vdash_{\mathscr{S}} XZ \xrightarrow{\mathcal{C}_1\mathcal{C}_2} Y \cap W.$$

[Simplification] If $X \cap Y = \varnothing$,

$$\{X \xrightarrow{\mathcal{C}_1} Y, XZ \xrightarrow{\mathcal{C}_2} W\} \vdash_{\mathscr{S}} XZ \setminus Y \xrightarrow{\mathcal{C}_1 \cap \mathcal{C}_2} W \setminus Y.$$

Automated Reasoning

Automated Reasoning

Automated reasoning. Attribute closure operator.

Classical attribute closure

Input: Γ, A
Output: A_Γ^+
begin
$A_\Gamma^+ := A$
repeat
$A' := A_\Gamma^+$
for each $X \to Y \in \Gamma$ **if** $X \subseteq A_\Gamma^+$ **and** $Y \not\subseteq A_\Gamma^+$ **then**
$A_\Gamma^+ := A_\Gamma^+ \cup \{Y\}$
until $A_\Gamma^+ = A'$
return A_Γ^+

Automated reasoning. Implication theorem.

Implication Theorem

Let M be a set of attributes, $\Sigma \subseteq \mathcal{L}_M$ and $A, B \subseteq M$. The following conditions are equivalent:

$$\Sigma \vdash A \to B \quad \text{iff} \quad B \subseteq A^+_\Sigma \quad \text{iff} \quad \{\varnothing \to A\} \cup \Sigma \vdash \varnothing \to B.$$

Automated reasoning. Equivalences

Theorem

In Simplification logic, the following equivalences hold:

1. (`Fr-Eq`): $\{A \to B\} \equiv \{A \to B - A\}$
2. (`Co-Eq`): $\{A \to B, A \to C\} \equiv \{A \to BC\}$
3. (`Si-Eq`): *If* $A \cap B = \varnothing$ *and* $A \subseteq C$ *then*

$$\{A \to B, C \to D\} \equiv \{A \to B, C - B \to D - B\}$$

4. (`rSi-Eq`): *If* $A \cap B = \varnothing$ *and* $A \subseteq C \cup D$ *then*

$$\{A \to B, C \to D\} \equiv \{A \to B, C \to D - B\}$$

Automated reasoning. Closure method.

1. $\varnothing \to X$ is added to Σ. It is called the *guide*.
2. The procedure compares the guide with the rest of the implications and applies the following SLFD equivalences:

   Eq. I: If $B \subseteq A$ then $\{\varnothing \to A, B \to C\} \equiv \{\varnothing \to AC\}$
   Eq. II: If $C \subseteq A$ then $\{\varnothing \to A, B \to C\} \equiv \{\varnothing \to A\}$
   Eq. III: Otherwise,
   $\{\varnothing \to A, B \to C\} \equiv \{\varnothing \to A, B - A \to C - A\}$
3. When a fix point is achieved, if the guide is $\varnothing \to A$, then we have that $X^+_\Sigma = A$.

This Closure returns:

- the closure of X
- a set of simplified implications, enclosing the knowledge not used (yet)

Attribute closure. Example.

$\{ad \to c, b \to e, be \to cg, bc \to g, c \to a, cd \to b, cf \to bh, cg \to af, lm \to n, \varnothing \to bd\}$

$\{a \to c, b \to e, be \to cg, bc \to g, c \to a, cd \to b, cf \to bh, cg \to af, lm \to n, \varnothing \to bd\}$

$\{a \to c, be \to cg, bc \to g, c \to a, cd \to b, cf \to bh, cg \to af, lm \to n, \varnothing \to bde\}$

$\{a \to c, bc \to g, c \to a, cd \to b, cf \to bh, cg \to af, lm \to n, \varnothing \to bcdeg\}$

$\{a \to c, c \to a, cd \to b, cf \to bh, cg \to af, lm \to n, \varnothing \to bcdeg\}$

$\{a \to c, cd \to b, cf \to bh, cg \to af, lm \to n, \varnothing \to abcdeg\}$

$\{a \to c, cf \to bh, cg \to af, lm \to n, \varnothing \to abcdeg\}$

$\{a \to c, cf \to bh, lm \to n, \varnothing \to abcdefg\}$

$\{cf \to bh, lm \to n, \varnothing \to abcdefg\}$

$\{lm \to$

Other extensions

Fuzzy implications :: Equivalences

Implication Theorem

Let $A, B \in \mathrm{L}^Y$, $c \in L$ and $T \in \mathrm{L}^{\mathcal{L}}$.

$$T \vdash A \overset{c}{\Rightarrow} B \text{ if and only if } \{\top \Rightarrow A\} \cup c(T) \vdash \top \Rightarrow c \otimes B$$

Equivalences

For all $A, B, C \in \mathrm{L}^Y$, if $A' = A(S(B, A)^* \otimes C)$ then

1. $\{\top \Rightarrow A, B \Rightarrow C\} \equiv \{\top \Rightarrow A', B - A' \Rightarrow C - A'\}$ [gSiEq]
2. If $B \setminus A' = \varnothing$ then $\{\top \Rightarrow A, B \Rightarrow C\} \equiv \{\top \Rightarrow A'C\}$ [gSiUnEq]
3. If $C \setminus A' = \varnothing$ then $\{\top \Rightarrow A, B \Rightarrow C\} \equiv \{\top \Rightarrow A'\}$ [gSiAxEq]

Fuzzy implications :: Closure method

Input: T (theory), A (L-set of attributes)
Output: A^+ (closure of A with respect to $c(T)$)
BEGIN
Let $A_0 := \emptyset$, $T_0 := \emptyset$, and $i := 0$
Let $A_1 := A$;
Apply (**DeEq**) to every formula in $c(T)$ obtaining T_1
WHILE ($A_i \neq A_{i+1}$ or $T_i \neq T_{i+1}$) DO
$i := i + 1$; $A_{i+1} := A_i$; $T_{i+1} := T_i$
Modify T_{i+1} and A_{i+1} applying
(**gSiEq**), (**gSiUnEq**), and (**gSiAxEq**) to
$\top \Rightarrow A_i$ and each $U \Rightarrow V \in T_i$
END WHILE
RETURN A_i
END

Triadic implications :: equivalences

Equivalences

The following equivalences hold:

Axiom Eq.: $\{X \xrightarrow{\varnothing} Y\} \equiv \{X \xrightarrow{\mathcal{C}} \varnothing\} \equiv \varnothing$

Decomposition Eq.: $\{X \xrightarrow{\mathcal{C}} Y\} \equiv \{X \xrightarrow{\mathcal{C}} Y \setminus X\}$

Composition Eq.: $\{X \xrightarrow{\mathcal{C}} Y, X \xrightarrow{\mathcal{C}} W\} \equiv \{X \xrightarrow{\mathcal{C}} YW\}$

Conditional Composition Eq.:

$$\{X \xrightarrow{\mathcal{C}_1} Y, X \xrightarrow{\mathcal{C}_2} Y\} \equiv \{X \xrightarrow{\mathcal{C}_1\mathcal{C}_2} Y\}$$

Simplification Eq.: If $X \cap Y = \varnothing$, then $\{X \xrightarrow{\mathcal{C}_1\mathcal{C}_2} Y, XV \xrightarrow{\mathcal{C}_2} W\} \equiv \{X \xrightarrow{\mathcal{C}_1\mathcal{C}_2} Y, XV \setminus Y \xrightarrow{\mathcal{C}_2} W \setminus Y\}$

Triadic implications :: Implication theorem

Theorem

For any $\Sigma \subseteq \mathcal{L}_{\Omega,\Gamma}$ *and* $X \xrightarrow{\mathcal{C}} Y \in \mathcal{L}_{\Omega,\Gamma}$, *one has*

$$\Sigma \vdash X \xrightarrow{\mathcal{C}} Y \text{ if and only if } \Sigma \cup \{\varnothing \xrightarrow{\mathcal{C}} X\} \vdash \varnothing \xrightarrow{\mathcal{C}} Y$$

Proposition

The following equivalences hold:

$$\{\varnothing \xrightarrow{\mathcal{C}_1} X, U \xrightarrow{\mathcal{C}_2} V\} \equiv \{\varnothing \xrightarrow{\mathcal{C}_1} X, U \setminus X \xrightarrow{\mathcal{C}_1 \cap \mathcal{C}_2} V \setminus X, U \xrightarrow{\mathcal{C}_2 \setminus \mathcal{C}_1} V\} \quad (1)$$

$$\{\varnothing \xrightarrow{\mathcal{C}_1} X, U \xrightarrow{\mathcal{C}_2} V\} \equiv \{\varnothing \xrightarrow{\mathcal{C}_1 \cap \mathcal{C}_2} XV, \varnothing \xrightarrow{\mathcal{C}_1 \setminus \mathcal{C}_2} X, U \xrightarrow{\mathcal{C}_2 \setminus \mathcal{C}_1} V\}, \text{ when } U \subseteq X \quad (2)$$

$$\{\varnothing \xrightarrow{\mathcal{C}_1} X, U \xrightarrow{\mathcal{C}_2} V\} \equiv \{\varnothing \xrightarrow{\mathcal{C}_1} X, U \xrightarrow{\mathcal{C}_2 \setminus \mathcal{C}_1} V\}, \text{ when } V \subseteq X \quad (3)$$

Triadic :: Closure method

input : A set of implications Σ, and a CAI $X \xrightarrow{\mathcal{C}} Y$

output: A boolean answer

begin

$\Delta_X := X \times \mathcal{C}$, $\Delta_Y := (Y \times \mathcal{C}) \setminus (X \times \mathcal{C})$

repeat

flag:=false

foreach $U \xrightarrow{\mathcal{C}_1} V \in \Sigma$ *with* $\mathcal{C}_1 \cap \mathcal{C} \neq \varnothing$ **do**

$\Delta_C := \{c \in \mathcal{C}_1 \cap \mathcal{C} \mid U \times \{c\} \subseteq \Delta_X\}$

if $\Delta_C \neq \varnothing$ **then** ..... Equivalence (2)

$\Delta_X := \Delta_X \cup (V \times \Delta_C)$

$\Delta_Y := \Delta_Y \setminus (V \times \Delta_C)$

$\Sigma := \Sigma \setminus \{U \xrightarrow{\mathcal{C}_1} V\}$

$\mathcal{C}_1 := \mathcal{C}_1 \setminus \Delta_C$

if $\mathcal{C}_1 \neq \varnothing$ **then** $\Sigma := \Sigma \cup \{U \xrightarrow{\mathcal{C}_1} V\}$

flag:=true

end

$\Delta_C := \{c \in \mathcal{C}_1 \cap \mathcal{C} \mid V \times \{c\} \subseteq \Delta_X\}$

if $\Delta_C \neq \varnothing$ **then** ..... Equivalence (3)

if $\Delta_C = \mathcal{C}_1$ **then** $\Sigma := \Sigma \setminus \{U \xrightarrow{\mathcal{C}_1} V\}$

else $\Sigma := (\Sigma \setminus \{U \xrightarrow{\mathcal{C}_1} V\}) \cup \{U \xrightarrow{\mathcal{C}_1 \setminus \Delta_C} V\}$

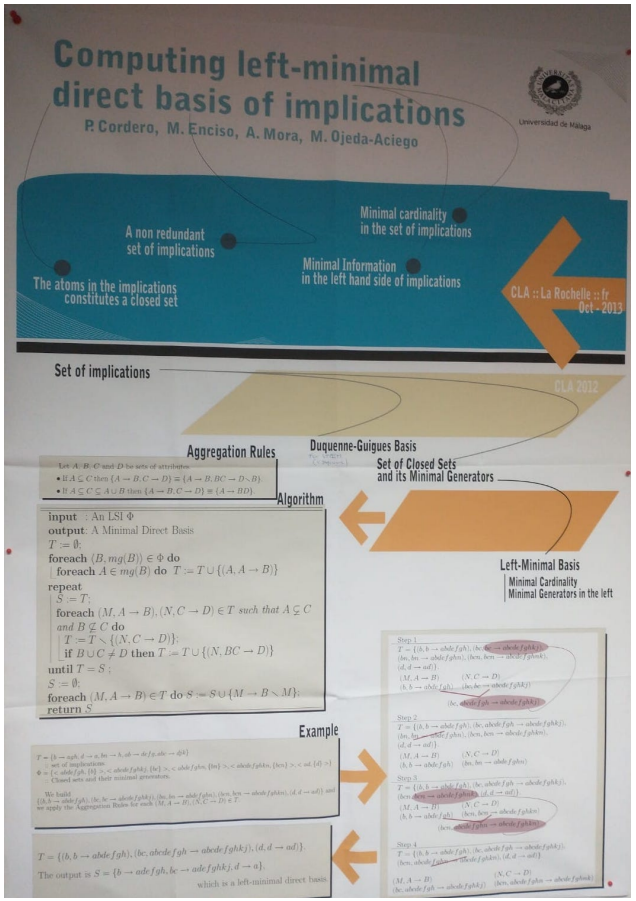
end

end

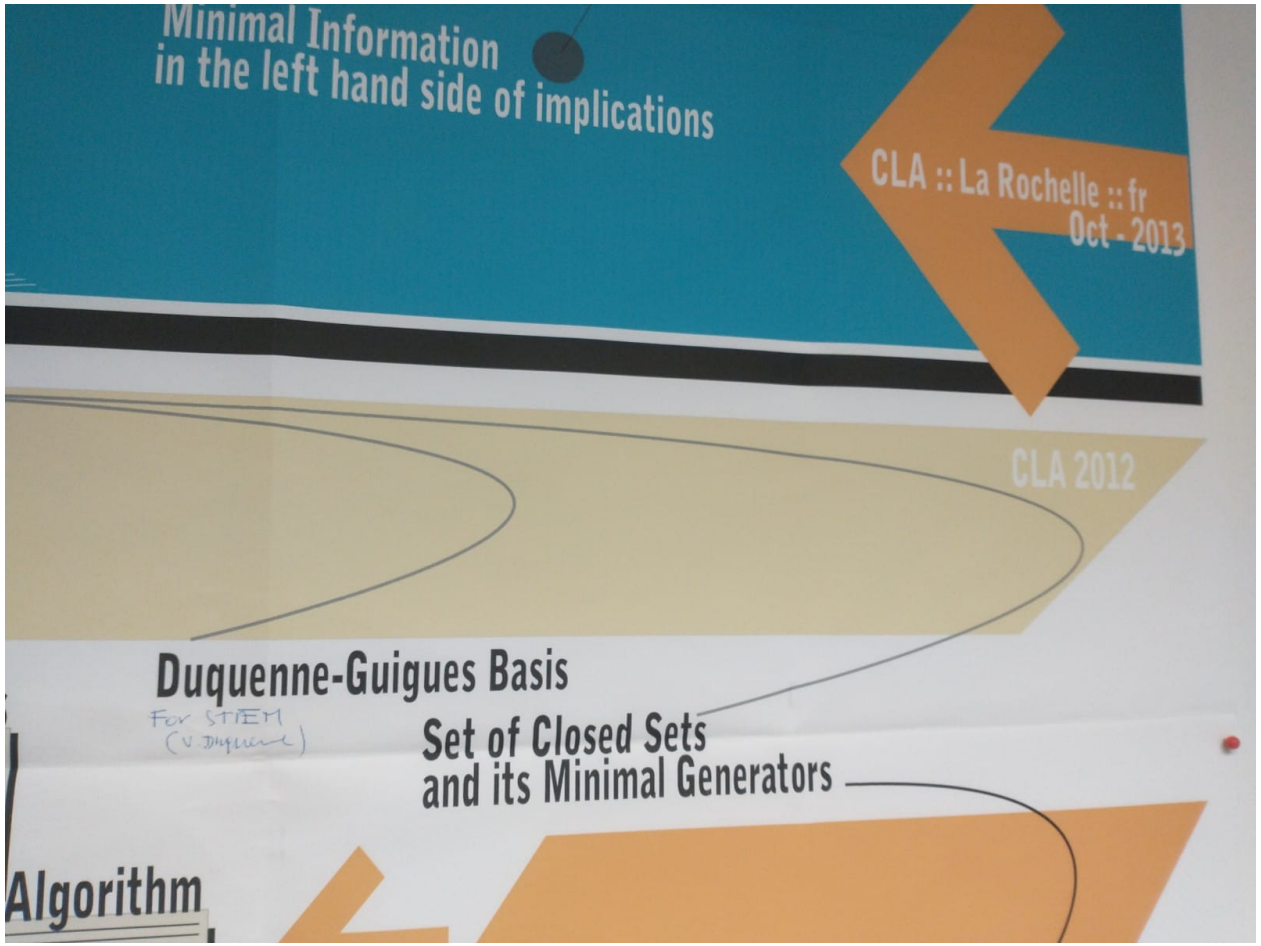
until $(\Delta_Y = \varnothing)$ *or* (flag=false)

Asking Vincent for an autograph

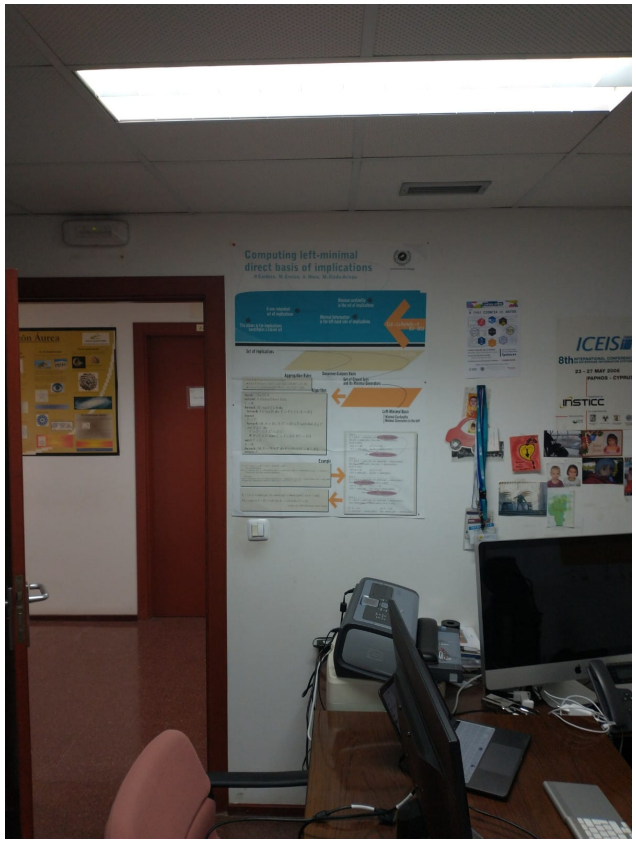
Asking Vincent for an autograph



Asking Vincent for an autograph



Asking Vincent for an autograph



Moving towards applications

How to guide an user (users) in a recommendation system?

- Closures of preferences of the users - blindly
- Closures of preferences of the users - guided by attributes on the left-hand side of the implications
- Preferences of the users - guided by attributes on minimal generators
- Some of the above options enhanced by parallelism and/or cooperative user guidance

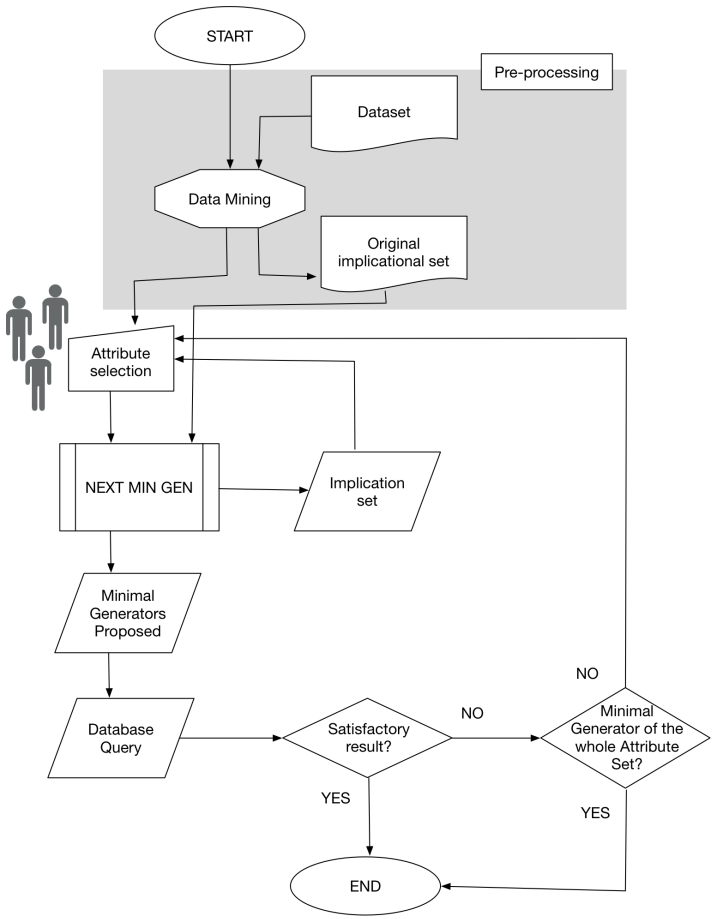
A Formal Concept Analysis approach to cooperative conversational recommendation

P. Cordero, M. Enciso, A. Moram, M. Ojeda, C. Rossi - International Journal of Computational Intelligence Systems (2020) 13(1) 1243–1252

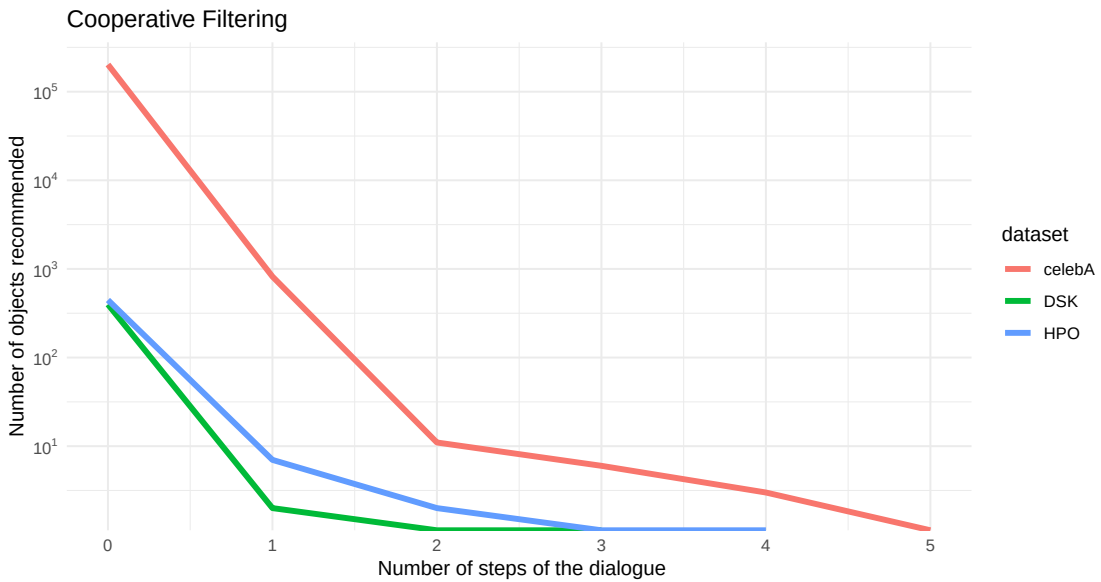
The knowledge inside the implications addresses us towards the development of some recommender systems.

- Based on logic which allows to manage the users' preferences by following a conversational paradigm.
- Lazzy computation of minimal generators - on-demand computation of minimal generators by means of an algorithm with polynomial delay.

A Formal Concept Analysis approach to cooperative conversational recommendation



A Formal Concept Analysis approach to cooperative conversational recommendation



step	Objects celebA	Objects DSK	Objects HPO
0	202599	398	446
1	820	2	7
2	11	0	2
3	6	0	0
4	3	0	0
5	0	0	0

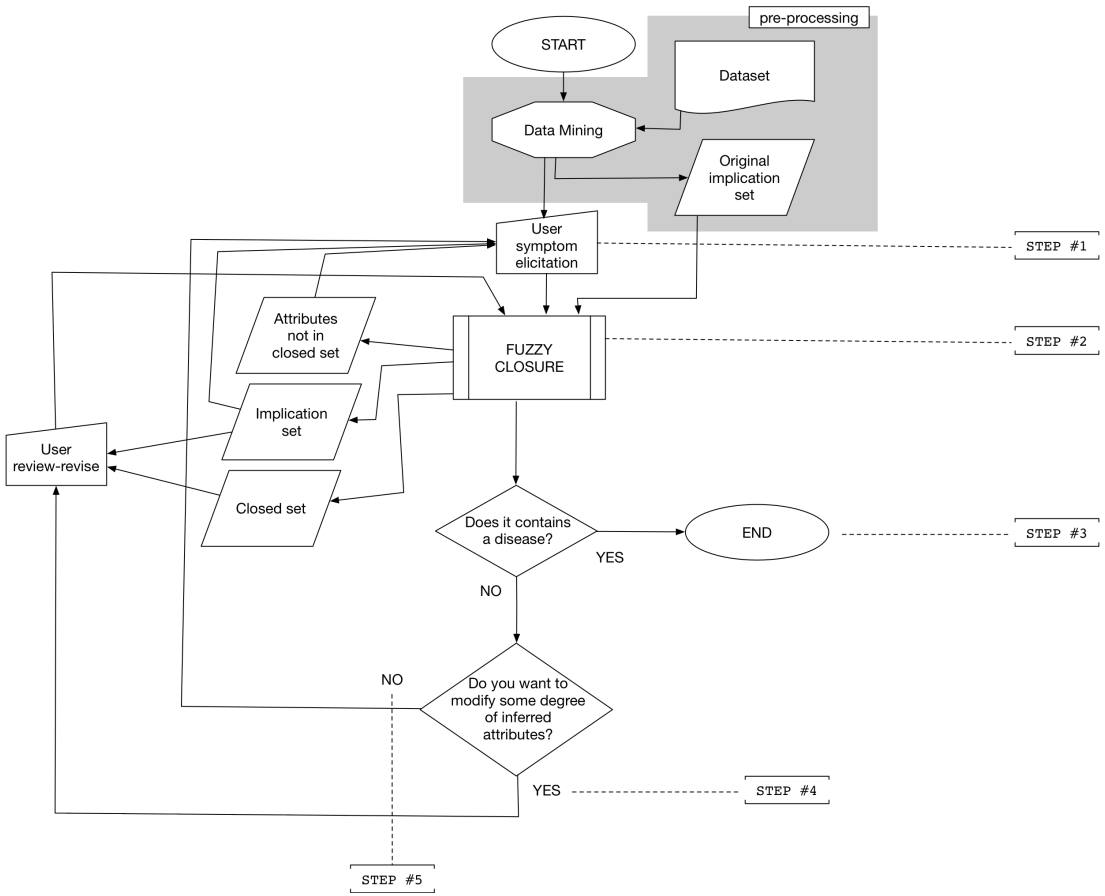
A conversational recommender system for diagnosis using fuzzy rules

P. Cordero, M. Enciso, D. López, A. Mora - Expert Systems With Applications 154 (2020) 113449

Graded implications in the framework of Fuzzy Formal Concept Analysis are used as the knowledge guiding the recommendations.

- Conversational recommender systems have proven to be a good approach in telemedicine, building a dialogue between the user and the recommender based on user preferences provided at each step of the conversation.
- We propose a conversational recommender system for medical diagnosis using fuzzy logic.
- The recommender system has been used to provide differential diagnosis between schizophrenia and schizoaffective and bipolar disorders.

A conversational recommender system for diagnosis using fuzzy rules



A conversational recommender system for diagnosis using fuzzy rules

	Accuracy	Sensitivity	Specificity	Precision
ALS	0.360	0.333	0.380	0.290
IBCF (Cosine)	0.555	0.475	0.615	0.483
IBCF (Pearson)	0.770	0.466	1.000	1.000
LIBMF	0.491	0.901	0.181	0.455
SVD	0.376	0.515	0.271	0.349
SVDF	0.431	1.000	0.000	0.431
UBCF (Cosine)	0.608	0.967	0.335	0.524
UBCF (Pearson)	0.525	0.783	0.330	0.470
C5.0	0.674	0.636	1.000	1.000
PART	0.883	0.847	0.950	0.970
JRip	0.752	0.814	0.688	0.731
Random Forest	0.953	0.924	1.000	1.000
xgboost	0.818	0.963	0.713	0.706
k-nn	0.589	0.603	0.544	0.815
Proposal	0.982	0.996	0.948	0.955

Table: Comparison of the current proposal to other recommender systems and machine learning methods.

Demo - fcaR

How to use Simplification Logic using fcaR to compute closures:
`https://github.com/Malaga-FCA-group/fcaR_demos/blob/main/SLrecommender_DSK_C.html`

Conclusion

1. What bases of rules ?
   - The main bases
   - The D-basis
2. What about non binary data ?
   - Some approaches for non-binary data
   - NextPriorityConcept algorithm
3. How to reason on rules ?
   - Implication logics
   - Automated reasoning
   - Applications

Tools demonstration

Tools demonstration: Thursday afternoon

- **fcaR**: R package for FCA
- **Galactic**: Concept generation for heterogenous and complex data

A story of rules ...

A story of encounters and people ....

... behind the story of results

On the importance ...

... of encounters
... of exchanges
... of (non virtual) conferences

To the memory of Vincent Duquenne

Vincent's house at Sennelly

Rest in peace