The NextPriorityConcept Algorithm

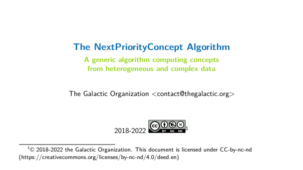
A generic algorithm computing concepts from heterogeneous and complex data

The Galactic Organization <contact@thegalactic.org>

2018-2022 BY NC ND[1]

2022-02-03

The NextPriorityConcept Algorithm



The NextPriorityConcept Algorithm
A generic algorithm computing concepts
from heterogeneous and complex data

The Galactic Organization <contact@thegalactic.org>

2018-2022 BY NC ND[1]

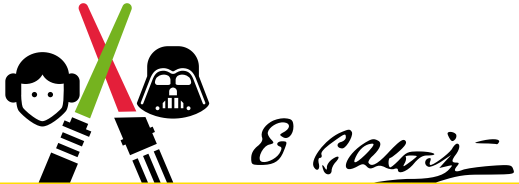
[1] © 2018-2022 the Galactic Organization. This document is licensed under CC-by-nc-nd (https://creativecommons.org/licenses/by-nc-nd/4.0/deed.en)

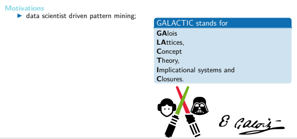
Motivations

- data scientist driven pattern mining;

GALACTIC stands for

GAlois
LAttices,
Concept
Theory,
Implicational systems and
Closures.




Motivations
- data scientist driven pattern mining;

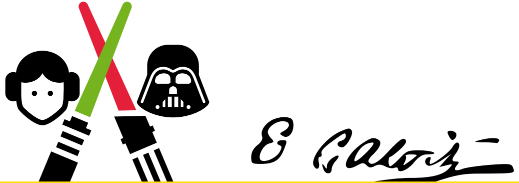
GALACTIC stands for
GAlois
LAttices,
Concept
Theory,
Implicational systems and
Closures.

Motivations

- data scientist driven pattern mining;
- consideration of heterogeneous and complex data;

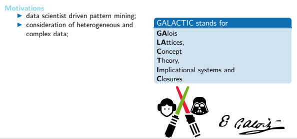
GALACTIC stands for

GAlois
LAttices,
Concept
Theory,
Implicational systems and
Closures.



2022-02-03

The NextPriorityConcept Algorithm
└Introduction
└Motivations
└Motivations



Motivations
- data scientist driven pattern mining;
- consideration of heterogeneous and complex data;

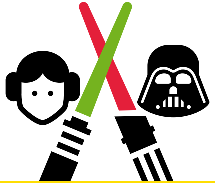
GALACTIC stands for
GAlois
LAttices,
Concept
Theory,
Implicational systems and
Closures.

Motivations

- data scientist driven pattern mining;
- consideration of heterogeneous and complex data;
- generation of implication rules;

GALACTIC stands for

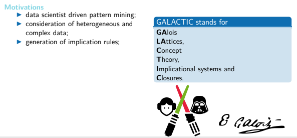
GAlois
LAttices,
Concept
Theory,
Implicational systems and
Closures.



E Galois

2022-02-03

The NextPriorityConcept Algorithm
└Introduction
└Motivations
└Motivations



Motivations
- data scientist driven pattern mining;
- consideration of heterogeneous and complex data;
- generation of implication rules;

GALACTIC stands for
GAlois
LAttices,
Concept
Theory,
Implicational systems and
Closures.

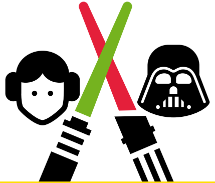
La Rochelle Université

Motivations

- data scientist driven pattern mining;
- consideration of heterogeneous and complex data;
- generation of implication rules;
- extracted information size adapted to the goals.

GALACTIC stands for

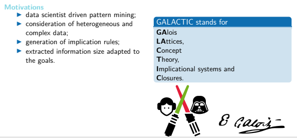
GAlois
LAttices,
Concept
Theory,
Implicational systems and
Closures.



E Galois

2022-02-03

The NextPriorityConcept Algorithm
└Introduction
└Motivations
└Motivations



Motivations
- data scientist driven pattern mining;
- consideration of heterogeneous and complex data;
- generation of implication rules;
- extracted information size adapted to the goals.

GALACTIC stands for
GAlois
LAttices,
Concept
Theory,
Implicational systems and
Closures.

E Galois

Founding ideas

Founding ideas

- inspired by the Bordat algorithm;

Founding ideas
Founding ideas
- inspired by the Bordat algorithm;

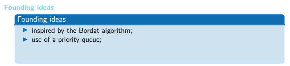
Founding ideas

Founding ideas

- inspired by the Bordat algorithm;
- use of a priority queue;

2022-02-03

The NextPriorityConcept Algorithm
└Introduction
  └Founding ideas
    └Founding ideas



Founding ideas
Founding ideas
- inspired by the Bordat algorithm;
- use of a priority queue;

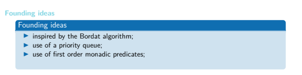
Founding ideas

Founding ideas

- inspired by the Bordat algorithm;
- use of a priority queue;
- use of first order monadic predicates;

2022-02-03

The NextPriorityConcept Algorithm
└Introduction
  └Founding ideas
    └Founding ideas



Founding ideas

Founding ideas

- inspired by the Bordat algorithm;
- use of a priority queue;
- use of first order monadic predicates;

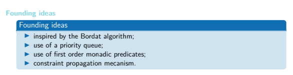
Founding ideas

Founding ideas

- inspired by the Bordat algorithm;
- use of a priority queue;
- use of first order monadic predicates;
- constraint propagation mecanism.

2022-02-03

The NextPriorityConcept Algorithm
└Introduction
  └Founding ideas
    └Founding ideas


Founding ideas

Founding ideas

- inspired by the Bordat algorithm;
- use of a priority queue;
- use of first order monadic predicates;
- constraint propagation mecanism.

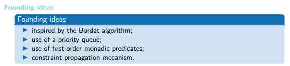
Founding ideas

Founding ideas

- inspired by the Bordat algorithm;
- use of a priority queue;
- use of first order monadic predicates;
- constraint propagation mecanism.

2022-02-03

The NextPriorityConcept Algorithm
└Introduction
  └Founding ideas
    └Founding ideas



Founding ideas

Founding ideas

- inspired by the Bordat algorithm;
- use of a priority queue;
- use of first order monadic predicates;
- constraint propagation mecanism.

Founding ideas

Founding ideas

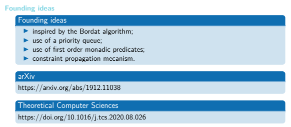
- inspired by the Bordat algorithm;
- use of a priority queue;
- use of first order monadic predicates;
- constraint propagation mecanism.

arXiv

https://arxiv.org/abs/1912.11038

Theoretical Computer Sciences

https://doi.org/10.1016/j.tcs.2020.08.026



Founding ideas
Founding ideas
- inspired by the Bordat algorithm;
- use of a priority queue;
- use of first order monadic predicates;
- constraint propagation mecanism.
arXiv
https://arxiv.org/abs/1912.11038
Theoretical Computer Sciences
https://doi.org/10.1016/j.tcs.2020.08.026

Bordat algorithm as basis

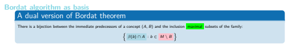
A dual version of Bordat theorem

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

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

2022-02-03

The NextPriorityConcept Algorithm
└Algorithm
└Bordat algorithm
└Bordat algorithm as basis



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:

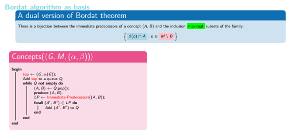
$$\left\{ \beta(b) \cap A \ : b \in \ M \setminus B \right\}$$

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

```
begin
    top ← $(G, \alpha(G))$;
    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
```

2022-02-03

The NextPriorityConcept Algorithm
└Algorithm
└Bordat algorithm
└Bordat algorithm as basis



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

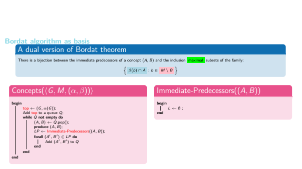
```
begin
    top ← $(G, \alpha(G))$;
    Add top to a queue $Q$;
    while $Q$ not empty do
        $(A, B) \leftarrow Q$.pop();
        produce $(A, B)$;
        $LP \leftarrow$ Immediate-Predecessors($(A, B)$);
        forall $(A', B') \in LP$ do
            Add $(A', B')$ to $Q$
        end
    end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

Immediate-Predecessors((A, B))

```
begin
    $L \leftarrow \emptyset$ ;
end
```

2022-02-03

The NextPriorityConcept Algorithm
└Algorithm
└Bordat algorithm
└Bordat algorithm as basis



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:

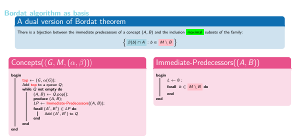
$$\left\{ \beta(b) \cap A \ : b \in \ M \setminus B \right\}$$

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

```
begin
    top ← $(G, \alpha(G))$;
    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
```

Immediate-Predecessors((A, B))

```
begin
    $L \leftarrow \emptyset$ ;
    forall $b \in M \setminus B$ 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($\langle G, M, (\alpha, \beta) \rangle$)

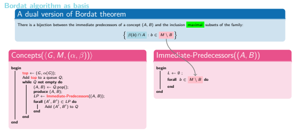
```
begin
    top ← $(G, \alpha(G))$;
    Add top to a queue $Q$;
    while $Q$ not empty do
        $(A, B) \leftarrow Q$.pop();
        produce $(A, B)$;
        $LP \leftarrow$ Immediate-Predecessors($(A, B)$);
        forall $(A', B') \in LP$ do
            Add $(A', B')$ to $Q$
        end
    end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

Immediate-Predecessors((A, B))

```
begin
    $L \leftarrow \emptyset$ ;
    forall $b \in M \setminus B$ do
    end
end
```

2022-02-03

The NextPriorityConcept Algorithm
└Algorithm
  └Bordat algorithm
    └Bordat algorithm as basis


Bordat algorithm as basis
A dual version of Bordat theorem
Concepts($\langle G, M, (\alpha, \beta) \rangle$)
Immediate-Predecessors((A, B))

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:

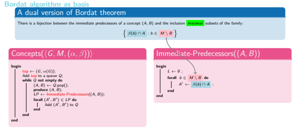
$$\left\{ \beta(b) \cap A \ : b \in \ M \setminus B \right\}$$

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

```
begin
    top ← $(G, \alpha(G))$;
    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
```

Immediate-Predecessors((A, B))

```
begin
    $L \leftarrow \emptyset$ ;
    forall $b \in M \setminus B$ do
        $A' \leftarrow \beta(b) \cap A$ ;
    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:
Concepts($\langle G, M, (\alpha, \beta) \rangle$)
Immediate-Predecessors((A, B))

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

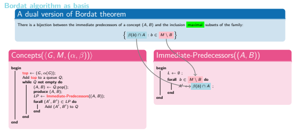
```
begin
    top ← $(G, \alpha(G))$;
    Add top to a queue $Q$;
    while $Q$ not empty do
        $(A, B) \leftarrow Q$.pop();
        produce $(A, B)$;
        $LP \leftarrow$ Immediate-Predecessors($(A, B)$);
        forall $(A', B') \in LP$ do
            Add $(A', B')$ to $Q$
        end
    end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

Immediate-Predecessors((A, B))

```
begin
    $L \leftarrow \emptyset$ ;
    forall $b \in M \setminus B$ do
        $A' \leftarrow \beta(b) \cap A$ ;
    end
end
```

2022-02-03

The NextPriorityConcept Algorithm
└Algorithm
└Bordat algorithm
└Bordat algorithm as basis



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

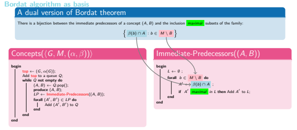
```
begin
    top ← $(G, \alpha(G))$;
    Add top to a queue $Q$;
    while $Q$ not empty do
        $(A, B) \leftarrow Q$.pop();
        produce $(A, B)$;
        $LP \leftarrow$ Immediate-Predecessors($(A, B)$);
        forall $(A', B') \in LP$ do
            Add $(A', B')$ to $Q$
        end
    end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

Immediate-Predecessors((A, B))

```
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
end
```

The NextPriorityConcept Algorithm
└Algorithm
└Bordat algorithm
└Bordat algorithm as basis

2022-02-03



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

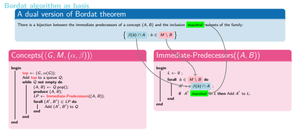
```
begin
    top ← $(G, \alpha(G))$;
    Add top to a queue $Q$;
    while $Q$ not empty do
        $(A, B) \leftarrow Q$.pop();
        produce $(A, B)$;
        $LP \leftarrow$ Immediate-Predecessors($(A, B)$);
        forall $(A', B') \in LP$ do
            Add $(A', B')$ to $Q$
        end
    end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

Immediate-Predecessors((A, B))

```
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
end
```

2022-02-03

The NextPriorityConcept Algorithm
└Algorithm
  └Bordat algorithm
    └Bordat algorithm as basis



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:

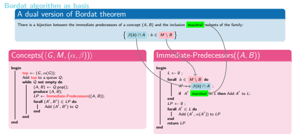
$$\left\{ \beta(b) \cap A \ : b \in \ M \setminus B \right\}$$

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

```
begin
    top ← $(G, \alpha(G))$;
    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
```

Immediate-Predecessors((A, B))

```
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
```



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:
{ β(b) ∩ A : b ∈ M \ B }
Concepts(⟨G, M, (α, β)⟩)
Immediate-Predecessors((A, B))

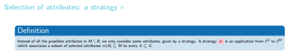
Selection of attributes: a strategy σ

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 $A \subseteq G$.

2022-02-03

The NextPriorityConcept Algorithm
└Algorithm
└Strategies
└Selection of attributes: a strategy σ



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))

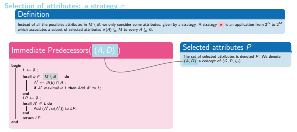
```
begin
    $L \leftarrow \emptyset$ ;
    forall $b \in$ $M \setminus B$ do
        $A' \leftarrow \beta(b) \cap A$ ;
        if $A'$ maximal in $L$ then Add $A'$ to $L$;
    end
    $LP \leftarrow \emptyset$ ;
    forall $A' \in L$ do
        Add $(A', \alpha(A'))$ to $LP$;
    end
    return $LP$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

Selected attributes P

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

2022-02-03

The NextPriorityConcept Algorithm
└Algorithm
└Strategies
└Selection of attributes: a strategy σ



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) $,\sigma$)

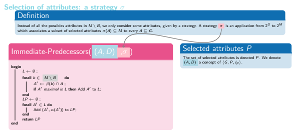
```
begin
    $L \leftarrow \emptyset$ ;
    forall $b \in$ $M \setminus B$ do
        $A' \leftarrow \beta(b) \cap A$ ;
        if $A'$ maximal in $L$ then Add $A'$ to $L$;
    end
    $LP \leftarrow \emptyset$ ;
    forall $A' \in L$ do
        Add $(A', \alpha(A'))$ to $LP$;
    end
    return $LP$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

Selected attributes P

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

2022-02-03

The NextPriorityConcept Algorithm
└Algorithm
  └Strategies
    └Selection of attributes: a strategy σ



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) $,\sigma$)

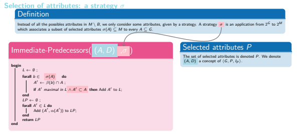
```
begin
    $L \leftarrow \emptyset$ ;
    forall $b \in \sigma(A)$ do
        $A' \leftarrow \beta(b) \cap A$ ;
        if $A'$ maximal in $L \wedge A' \subset A$ then Add $A'$ to $L$;
    end
    $LP \leftarrow \emptyset$ ;
    forall $A' \in L$ do
        Add $(A', \alpha(A'))$ to $LP$;
    end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

Selected attributes P

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

2022-02-03

The NextPriorityConcept Algorithm
└Algorithm
  └Strategies
    └Selection of attributes: a strategy σ



Selection of attributes: a strategy σ

Definition

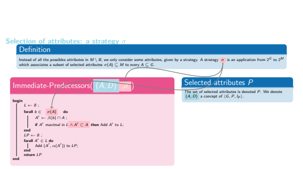
Instead of all the possibles attributes in $M \setminus B$, we only consider some attributes, given by a strategy. A strategy σ is an application from 2^G to 2^M which associates a subset of selected attributes $\sigma(A) \subseteq M$ to every $A \subseteq G$.

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) $,\sigma$)

```
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
```

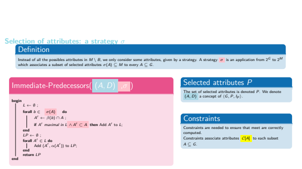
Selected attributes P

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 $\langle G, P, I_P \rangle$.

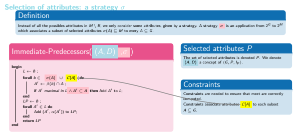
Constraints

Constraints are needed to ensure that meet are correctly computed.

Constraints associate attributes $\mathcal{C}[A]$ to each subset $A \subseteq G$.

2022-02-03

The NextPriorityConcept Algorithm
└Algorithm
  └Strategies
    └Selection of attributes: a strategy σ



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 $LP$
```

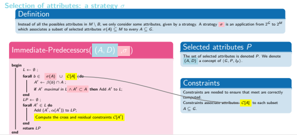
Selected attributes P

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 $LP$
```

Selected attributes P

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

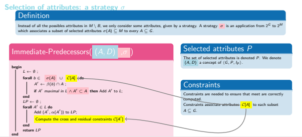
Constraints

Constraints are needed to ensure that meet are correctly computed.

Constraints associate attributes $\mathcal{C}[A]$ to each subset $A \subseteq G$.

2022-02-03

The NextPriorityConcept Algorithm
└Algorithm
└Strategies
└Selection of attributes: a strategy σ



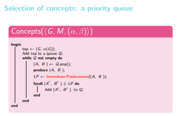
Selection of concepts: a priority queue

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

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

2022-02-03

The NextPriorityConcept Algorithm
└Algorithm
  └Strategies
    └Selection of concepts: a priority queue



Selection of concepts: a priority queue

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

```
begin
    top ← $(G, \alpha(G))$;
    Add top to a queue $Q$;
    while $Q$ not empty do
        $(A, B) \leftarrow Q$.pop();
        produce $(A, B)$;
        $LP$ ← 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($\langle G, M, (\alpha, \beta)\rangle$)

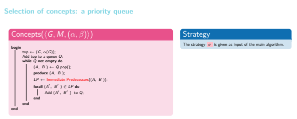
```
begin
    top ← $(G, \alpha(G))$;
    Add top to a queue $Q$;
    while $Q$ not empty do
        $(A, B) \leftarrow Q$.pop();
        produce $(A, B)$;
        $LP \leftarrow$ Immediate-Predecessors($(A, B)$);
        forall $(A', B') \in LP$ do
            Add $(A', B')$ to $Q$;
        end
   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

Strategy

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

2022-02-03

The NextPriorityConcept Algorithm
└Algorithm
  └Strategies
    └Selection of concepts: a priority queue


Selection of concepts: a priority queue
Concepts($\langle G, M, (\alpha, \beta)\rangle$)
Strategy
The strategy σ is given as input of the main algorithm.

Selection of concepts: a priority queue

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

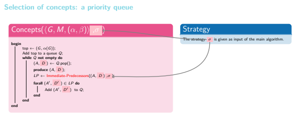
```
begin
    top ← $(G, \alpha(G))$;
    Add top to a queue $Q$;
    while $Q$ not empty do
        $(A, D) \leftarrow Q$.pop();
        produce $(A, D)$;
        $LP \leftarrow$ Immediate-Predecessors($(A, D)$ $,\sigma$ );
        forall $(A', D') \in LP$ do
            Add $(A', D')$ to $Q$;
    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

Strategy

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

2022-02-03

The NextPriorityConcept Algorithm
└Algorithm
└Strategies
└Selection of concepts: a priority queue



Selection of concepts: a priority queue

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

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

Strategy

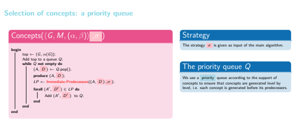
The strategy σ is given as input of the main algorithm.

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.

2022-02-03

The NextPriorityConcept Algorithm
└Algorithm
  └Strategies
    └Selection of concepts: a priority queue



Selection of concepts: a priority queue

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

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

```
begin
    top ← $(G, \alpha(G))$;
    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($(A,$ $D$ $)$ $,\sigma$ );
        forall $(A'$, $D'$ $) \in LP$ do
            Add ( $|A'|$ , $(A'$, $D'$ )) to $Q$;
        end
    end
end
```

Strategy

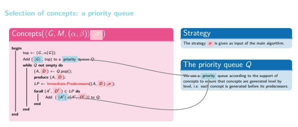
The strategy σ is given as input of the main algorithm.

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.

2022-02-03

The NextPriorityConcept Algorithm
└Algorithm
  └Strategies
    └Selection of concepts: a priority queue



Selection of concepts: a priority queue

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

```
begin
    top ← $(G, \alpha(G))$;
    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($(A,$ $D$ $)$ $,\sigma$ );
        forall $(A',$ $D'$ $) \in LP$ do
            Add ( $|A'|$ , $(A',$ $D'$ $)$) to $Q$;
        end
    end
end
```

Strategy

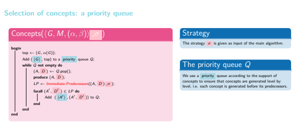
The strategy σ is given as input of the main algorithm.

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.

2022-02-03

The NextPriorityConcept Algorithm
└Algorithm
└Strategies
└Selection of concepts: a priority queue

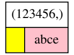


Example

Sample data

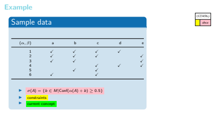
(α, β)	a	b	c	d	e
1	✓	✓	✓	✓	
2	✓	✓	✓		✓
3	✓	✓			✓
4			✓	✓	✓
5		✓	✓		
6	✓		✓		

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


(123456,)
abce

2022-02-03

The NextPriorityConcept Algorithm
└Algorithm
└Strategies
└Example

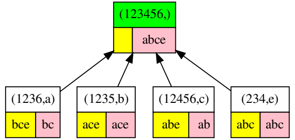


Example

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



(123456,)
abce

(1236,a)
bce | bc

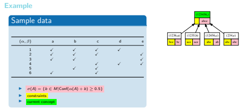
(1235,b)
ace | ace

(12456,c)
abe | ab

(234,e)
abc | abc

2022-02-03

The NextPriorityConcept Algorithm
└Algorithm
 └Strategies
  └Example

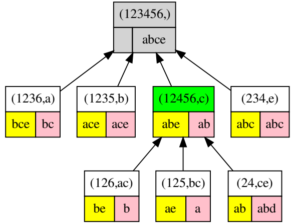


Example

Sample data

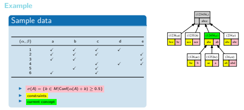
(α, β)	a	b	c	d	e
1	✓	✓	✓	✓	
2	✓	✓	✓		✓
3	✓	✓			✓
4			✓	✓	✓
5		✓	✓		
6	✓		✓		

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



The NextPriorityConcept Algorithm
└Algorithm
└Strategies
└Example

2022-02-03

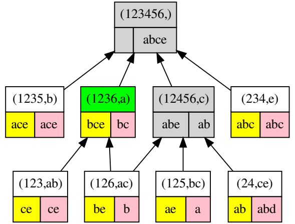


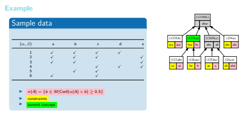
Example

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



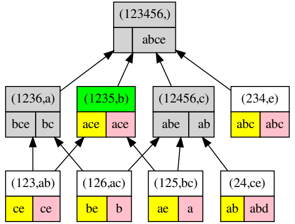


Example

Sample data

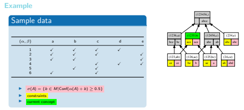
(α, β)	a	b	c	d	e
1	✓	✓	✓	✓	
2	✓	✓	✓		✓
3	✓	✓			✓
4			✓	✓	✓
5		✓	✓		
6	✓		✓		

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



2022-02-03

The NextPriorityConcept Algorithm
└Algorithm
└Strategies
└Example

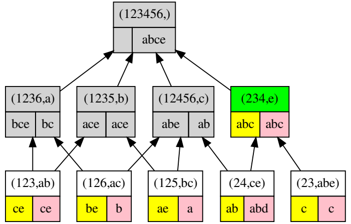


Example

Sample data

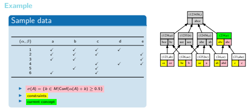
(α, β)	a	b	c	d	e
1	✓	✓	✓	✓	
2	✓	✓	✓		✓
3	✓	✓			✓
4			✓	✓	✓
5		✓	✓		
6	✓		✓		

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



2022-02-03

The NextPriorityConcept Algorithm
└Algorithm
└Strategies
└Example

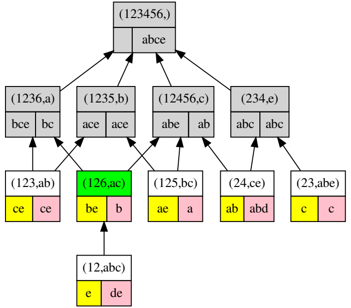


Example

Sample data

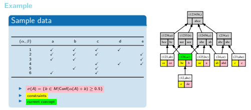
(α, β)	a	b	c	d	e
1	✓	✓	✓	✓	
2	✓	✓	✓		✓
3	✓	✓			✓
4			✓	✓	✓
5		✓	✓		
6	✓		✓		

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



2022-02-03

The NextPriorityConcept Algorithm
└Algorithm
└Strategies
└Example

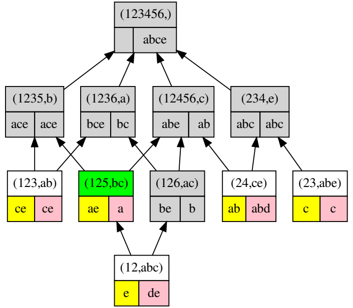


Example

Sample data

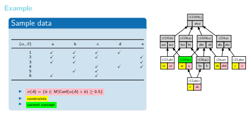
(α, β)	a	b	c	d	e
1	✓	✓	✓	✓	
2	✓	✓	✓		✓
3	✓	✓			✓
4			✓	✓	✓
5		✓	✓		
6	✓		✓		

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



2022-02-03

The NextPriorityConcept Algorithm
└Algorithm
 └Strategies
  └Example

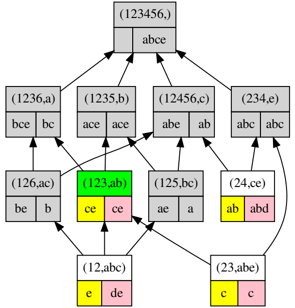


Example

Sample data

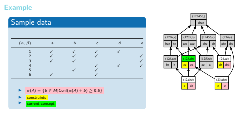
(α,β)	a	b	c	d	e
1	✓	✓	✓	✓	
2	✓	✓	✓		✓
3	✓	✓			✓
4			✓	✓	✓
5		✓	✓		
6	✓		✓		

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



2022-02-03

The NextPriorityConcept Algorithm
└Algorithm
└Strategies
└Example

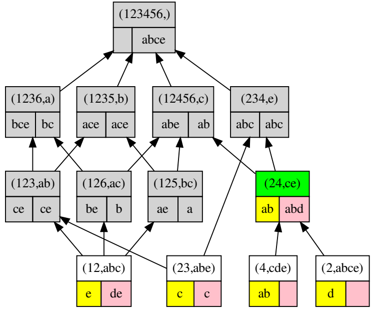


Example

Sample data

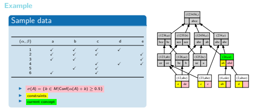
(α, β)	a	b	c	d	e
1	✓	✓	✓	✓	
2	✓	✓	✓		✓
3	✓	✓			✓
4			✓	✓	✓
5		✓	✓		
6	✓		✓		

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



2022-02-03

The NextPriorityConcept Algorithm
└Algorithm
└Strategies
└Example

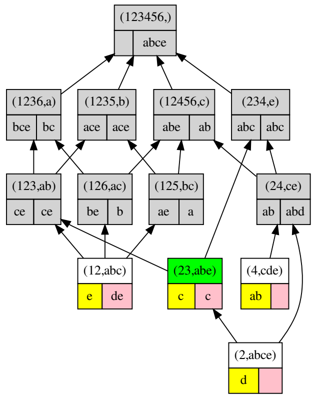


Example

Sample data

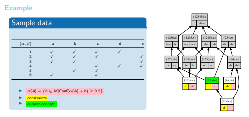
(α, β)	a	b	c	d	e
1	✓	✓	✓	✓	
2	✓	✓	✓		✓
3	✓	✓			✓
4			✓	✓	✓
5		✓	✓		
6	✓		✓		

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



2022-02-03

The NextPriorityConcept Algorithm
└Algorithm
└Strategies
└Example

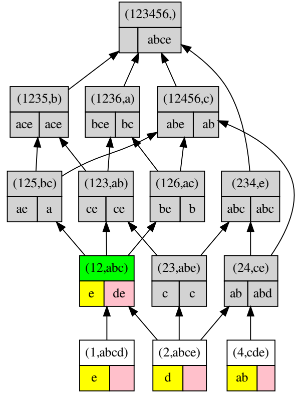


Example

Sample data

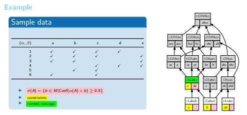
(α,β)	a	b	c	d	e
1	✓	✓	✓	✓	
2	✓	✓	✓		✓
3	✓	✓			✓
4			✓	✓	✓
5		✓	✓		
6	✓		✓		

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



2022-02-03

The NextPriorityConcept Algorithm
└Algorithm
└Strategies
└Example

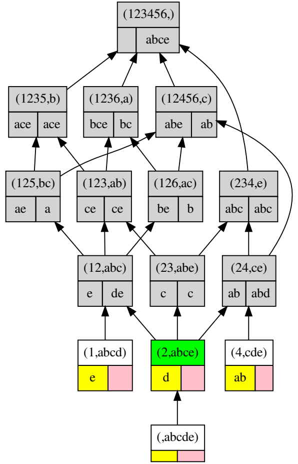

Example
Sample data

Example

Sample data

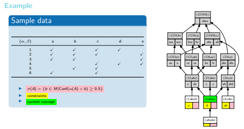
(α, β)	a	b	c	d	e
1	✓	✓	✓	✓	
2	✓	✓	✓		✓
3	✓	✓			✓
4			✓	✓	✓
5		✓	✓		
6	✓		✓		

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



2022-02-03

The NextPriorityConcept Algorithm
└Algorithm
└Strategies
└Example

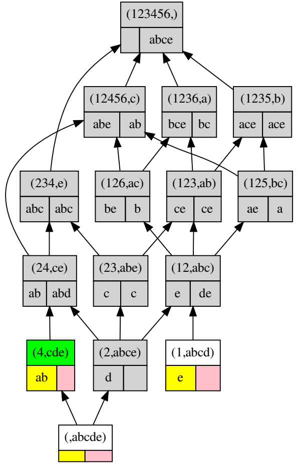


Example

Sample data

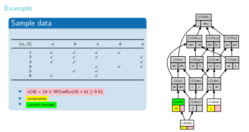
(α, β)	a	b	c	d	e
1	✓	✓	✓	✓	
2	✓	✓	✓		✓
3	✓	✓			✓
4			✓	✓	✓
5		✓	✓		
6	✓		✓		

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



2022-02-03

The NextPriorityConcept Algorithm
└Algorithm
└Strategies
└Example

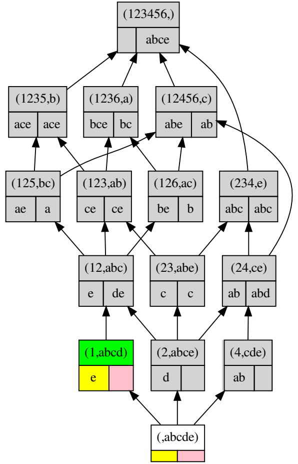


Example

Sample data

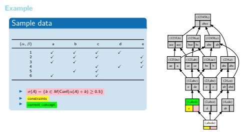
(α, β)	a	b	c	d	e
1	✓	✓	✓	✓	
2	✓	✓	✓		✓
3	✓	✓			✓
4			✓	✓	✓
5		✓	✓		
6	✓		✓		

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


(123456,) | abce
(1235,b) | ace | ace
(1236,a) | bce | bc
(12456,c) | abe | ab
(125,bc) | ae | a
(123,ab) | ce | ce
(126,ac) | be | b
(234,e) | abc | abc
(12,abc) | e | de
(23,abe) | c | c
(24,ce) | ab | abd
(1,abcd) | e
(2,abce) | d
(4,cde) | ab
(,abcde)

2022-02-03

The NextPriorityConcept Algorithm
└Algorithm
└Strategies
└Example

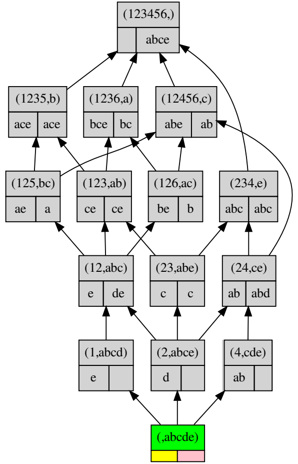


Example

Sample data

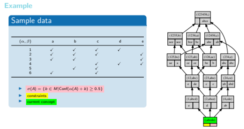
(α, β)	a	b	c	d	e
1	✓	✓	✓	✓	
2	✓	✓	✓		✓
3	✓	✓			✓
4			✓	✓	✓
5		✓	✓		
6	✓		✓		

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



2022-02-03

The NextPriorityConcept Algorithm
└Algorithm
  └Strategies
    └Example

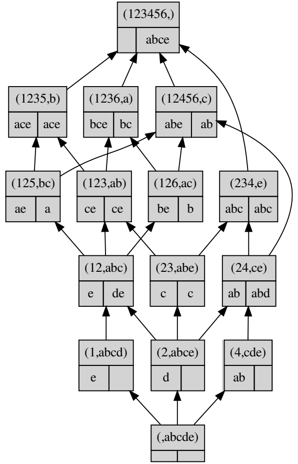


Example

Sample data

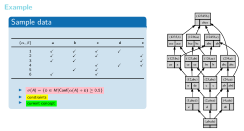
(α, β)	a	b	c	d	e
1	✓	✓	✓	✓	
2	✓	✓	✓		✓
3	✓	✓			✓
4			✓	✓	✓
5		✓	✓		
6	✓		✓		

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



2022-02-03

The NextPriorityConcept Algorithm
└Algorithm
└Strategies
└Example

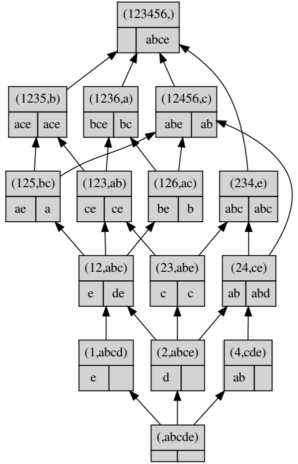


Example

Sample data

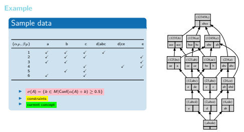
(α_P, β_P)	a	b	c	d|abc	d|ce	e
1	✓	✓	✓	✓		
2	✓	✓	✓			✓
3	✓	✓				✓
4			✓		✓	✓
5		✓	✓			
6	✓		✓			

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



2022-02-03

The NextPriorityConcept Algorithm
└Algorithm
└Strategies
└Example



NextPriorityConcept: the main theorem

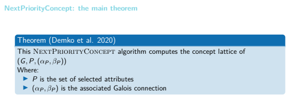
Theorem (Demko et al. 2020)

This NextPriorityConcept algorithm computes the concept lattice of $(G, P, (\alpha_P, \beta_P))$
Where:

- P is the set of selected attributes
- (α_P, β_P) is the associated Galois connection

2022-02-03

The NextPriorityConcept Algorithm
└Algorithm
└Strategies
└NextPriorityConcept: the main theorem



NextPriorityConcept: the main theorem

Theorem (Demko et al. 2020)
This NextPriorityConcept algorithm computes the concept lattice of $(G, P, (\alpha_P, \beta_P))$
Where:
- P is the set of selected attributes
- (α_P, β_P) is the associated Galois connection

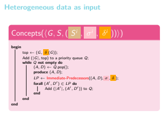
Heterogeneous data as input

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

```
begin
    top ← $(G, \delta(G))$;
    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
```

2022-02-03

The NextPriorityConcept Algorithm
└Algorithm
  └Heterogeneous data
    └Heterogeneous data as input



Heterogeneous data as input

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

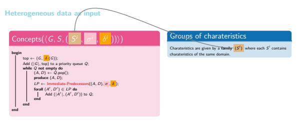
```
begin
    top ← $(G, \boxed{\delta}(G))$;
    Add $(|G|, \text{top})$ to a priority queue $Q$;
    while $Q$ not empty do
        $(A, D) \leftarrow Q.\text{pop}()$;
        produce $(A, D)$;
        $LP \leftarrow$ Immediate-Predecessors($(A, D), \boxed{\sigma}, \boxed{\delta}$);
        forall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

Groups of charateristics

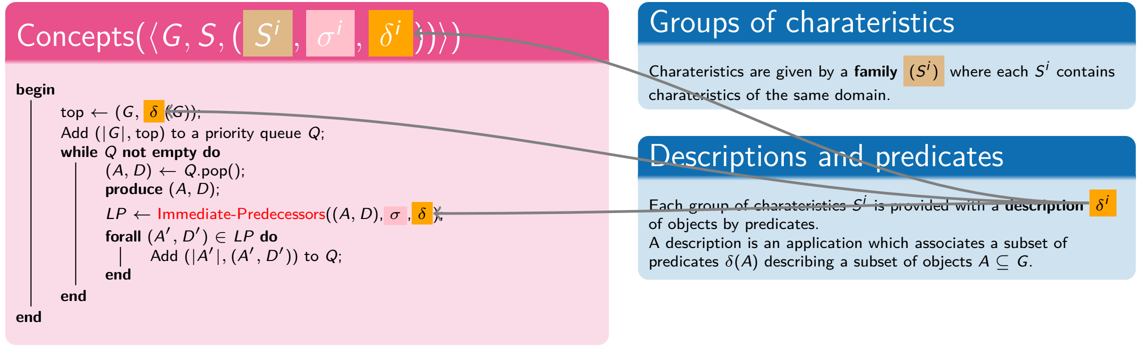
Charateristics are given by a **family** (S^i) where each S^i contains charateristics of the same domain.

2022-02-03

The NextPriorityConcept Algorithm
└Algorithm
 └Heterogeneous data
  └Heterogeneous data as input



Heterogeneous data as input



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

```
begin
    top ← $(G, \delta(G))$;
    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

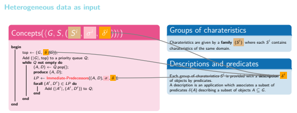
Charateristics are given by a **family** (S^i) where each S^i contains characteristics of the same domain.

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 $A \subseteq G$.

2022-02-03

The NextPriorityConcept Algorithm
└Algorithm
 └Heterogeneous data
  └Heterogeneous data as input


Heterogeneous data as input

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

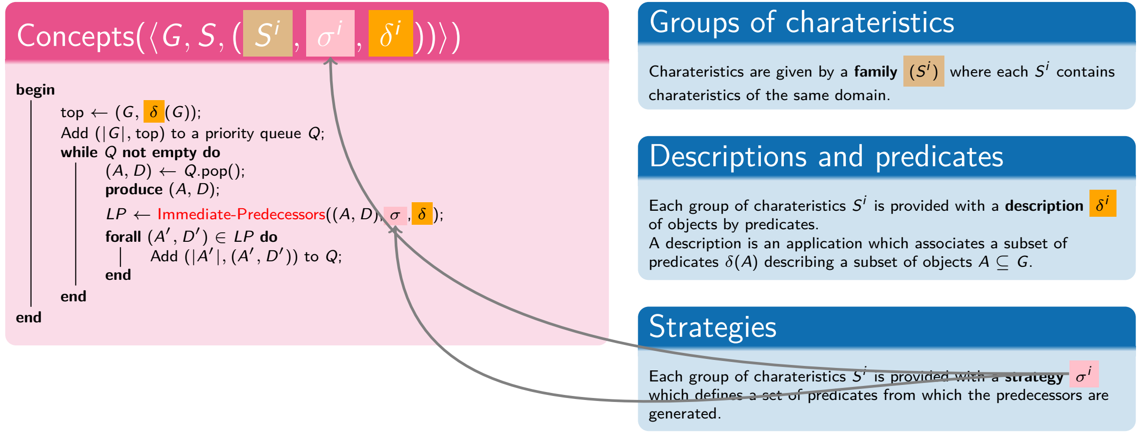
Groups of charateristics

Charateristics are given by a **family** (S^i) where each S^i contains characteristics 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 $A \subseteq G$.

Heterogeneous data as input



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

```
begin
    top ← $(G, \delta(G))$;
    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

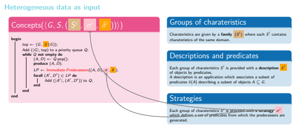
Charateristics are given by a **family** (S^i) where each S^i contains charateristics of the same domain.

Descriptions and predicates

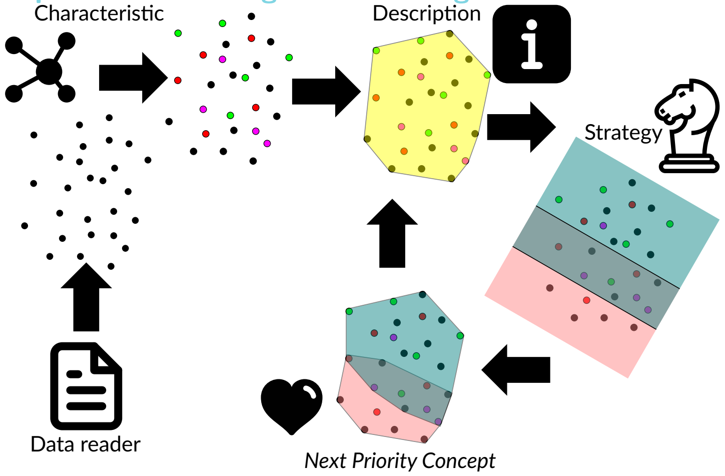
Each group of charateristics S^i is provided with a **description** δ^i of objects by predicates.
A description is an application which associates a subset of predicates $\delta(A)$ describing a subset of objects $A \subseteq G$.

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.

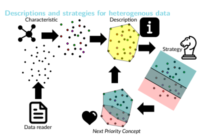


Descriptions and strategies for heterogenous data



2022-02-03

The NextPriorityConcept Algorithm
└Algorithm
└Heterogeneous data
└Descriptions and strategies for heterogenous data



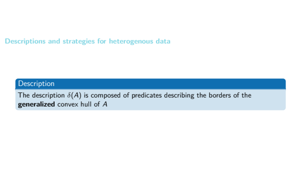
Descriptions and strategies for heterogenous data

Description

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

2022-02-03

The NextPriorityConcept Algorithm
└Algorithm
  └Heterogeneous data
    └Descriptions and strategies for heterogenous data



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

Descriptions and strategies for heterogenous data

Description

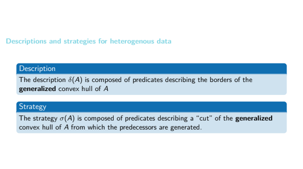
The description $\delta(A)$ is composed of predicates describing the borders of the **generalized** convex hull of A

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.

2022-02-03

The NextPriorityConcept Algorithm
└Algorithm
└Heterogeneous data
└Descriptions and strategies for heterogenous data



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

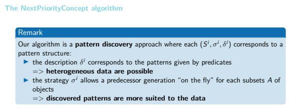
The strategy $\sigma(A)$ is composed of predicates describing a “cut” of the **generalized** convex hull of A from which the predecessors are generated.

The NextPriorityConcept algorithm

Remark

Our 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**



The NextPriorityConcept algorithm

Remark

Our 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**

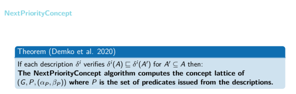
NextPriorityConcept

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 $(G, P, (\alpha_P, \beta_P))$ where P is the set of predicates issued from the descriptions.

2022-02-03

The NextPriorityConcept Algorithm
└Conclusion
└Theorem
└NextPriorityConcept



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 $(G, P, (\alpha_P, \beta_P))$ where P is the set of predicates issued from the descriptions.