Finger Exercises in Formal Concept Analysis

Bernhard Ganter

Institut für Algebra
TU Dresden
D-01062 Dresden
ganter@math.tu-dresden.de

Dresden ICCL Summer School, June/July 2006

Outline

Outline

The approach of FCA

The basic procedure of Formal Concept Analysis is the following:

- Data is represented in a very basic data type, called **formal context**.
- Each formal context is transformed into a mathematical structure called **concept lattice**.
  The information contained in the formal context is preserved.
- The concept lattice is the basis for further data analysis. It may be represented graphically to support communication, or it may be investigated with with algebraic methods to unravel its structure.

The approach of FCA

The basic procedure of Formal Concept Analysis is the following:

- Data is represented in a very basic data type, called **formal context**.
- Each formal context is transformed into a mathematical structure called **concept lattice**.
  The information contained in the formal context is preserved.
- The concept lattice is the basis for further data analysis. It may be represented graphically to support communication, or it may be investigated with with algebraic methods to unravel its structure.

The approach of FCA

The basic procedure of Formal Concept Analysis is the following:

- Data is represented in a very basic data type, called **formal context**.
- Each formal context is transformed into a mathematical structure called **concept lattice**.
  The information contained in the formal context is preserved.
- The concept lattice is the basis for further data analysis. It may be represented graphically to support communication, or it may be investigated with with algebraic methods to unravel its structure.

The approach of FCA

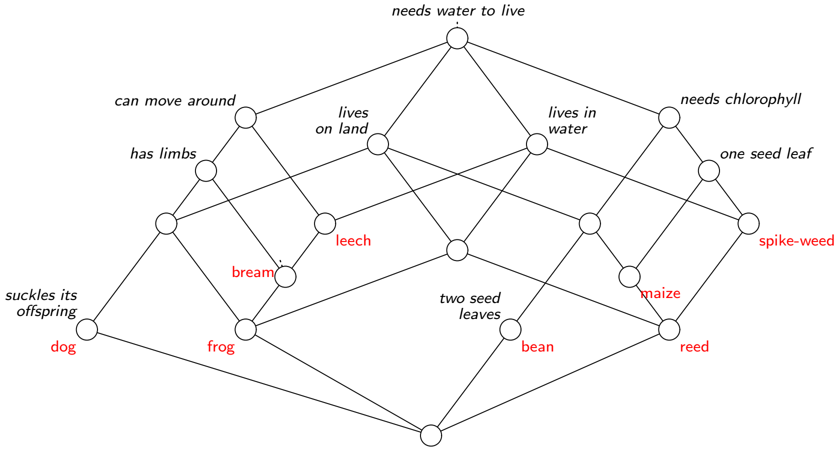
The basic procedure of Formal Concept Analysis is the following:

- Data is represented in a very basic data type, called **formal context**.
- Each formal context is transformed into a mathematical structure called **concept lattice**.
  The information contained in the formal context is preserved.
- The concept lattice is the basis for further data analysis. It may be represented graphically to support communication, or it may be investigated with with algebraic methods to unravel its structure.

A formal context

	needs water to live	lives in water	lives on land	needs chlorophyll	two seed leaves	one seed leaf	can move around	has limbs	suckles its offspring
Leech	×	×					×		
Bream	×	×					×	×	
Frog	×	×	×				×	×	
Dog	×		×				×	×	×
Spike-weed	×	×		×		×			
Reed	×	×	×	×		×			
Bean	×		×	×	×				
Maize	×		×	×		×			

and its concept lattice



Basic definitions

A **formal context** (G, M, I) consists of sets G, M and a binary relation $I \subseteq G \times M$.

For $A \subseteq G$ and $B \subseteq M$, define

$$\begin{aligned} A' &:= \{m \in M \mid g\, I\, m \text{ for all } g \in A\} \\ B' &:= \{g \in G \mid g\, I\, m \text{ for all } m \in B\}. \end{aligned}$$

The mappings

$$X \to X''$$

are closure operators.

Formal concepts

(A, B) is a **formal concept** of (G, M, I) iff

$$A \subseteq G, \quad B \subseteq M, \quad A' = B, \quad A = B'.$$

A is the **extent** and B is the **intent** of (A, B).

Concept lattice

Formal concepts can be ordered by

$$(A_1, B_1) \leq (A_2, B_2) :\Longleftrightarrow A_1 \subseteq A_2.$$

The set $\underline{\mathfrak{B}}(G, M, I)$ of all formal concepts of (G, M, I), with this order, is a complete lattice, called the **concept lattice** of (G, M, I).

The basic theorem

Theorem (The basic theorem of Formal Concept Analysis.) *The concept lattice of any formal context (G, M, I) is a complete lattice. For an arbitrary set $\{(A_i, B_i) \mid i \in I\} \subseteq \mathfrak{B}(G, M, I)$ of formal concepts, the supremum is given by*

$$\bigvee_{i\in I}(A_i, B_i) = \left((\bigcup_{i\in I} A_i)'', \bigcap_{i\in I} B_i\right)$$

and the infimum is given by

$$\bigwedge_{i\in I}(A_i, B_i) = \left(\bigcap_{i\in I} A_i, (\bigcup_{i\in I} B_i)''\right).$$

A complete lattice L is isomorphic to $\underline{\mathfrak{B}}(G, M, I)$ iff there are mappings $\tilde{\gamma} : G \to L$ and $\tilde{\mu} : M \to L$ such that $\tilde{\gamma}(G)$ is supremum-dense and $\tilde{\mu}(M)$ is infimum-dense in L, and

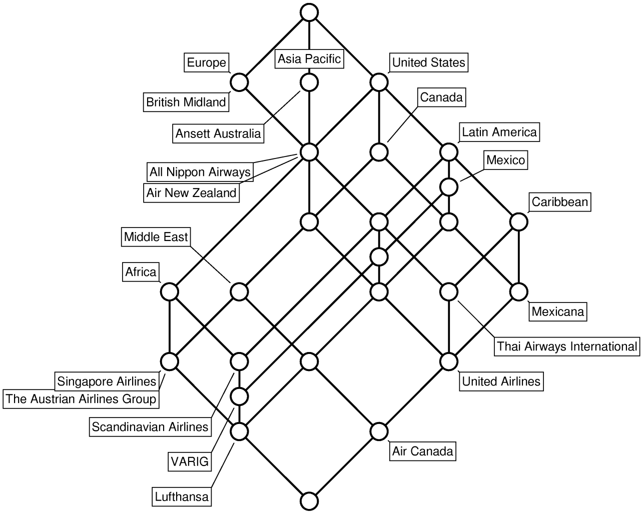
$$g\ I\ m \iff \tilde{\gamma}(g) \leq \tilde{\mu}(m).$$

In particular, $L \cong \underline{\mathfrak{B}}(L, L, \leq)$.

An example about airlines . . .

	Latin America	Europe	Canada	Asia Pacific	Middle East	Africa	Mexico	Caribbean	United States
Air Canada	×	×	×	×	×		×	×	×
Air New Zealand		×		×					×
All Nippon Airways		×		×					×
Ansett Australia				×					
The Austrian Airlines Group		×	×	×	×	×			×
British Midland		×							
Lufthansa	×	×	×	×	×	×	×		×
Mexicana	×		×				×	×	×
Scandinavian Airlines	×	×		×		×			×
Singapore Airlines		×	×	×	×	×			×
Thai Airways International	×	×		×				×	×
United Airlines	×	×	×	×			×	×	×
VARIG	×	×		×		×	×		×

. . . and its concept lattice



What if the data set is large?

With growing size of the data set, it quickly becomes unreasonable to display the full data in a single lattice diagram.

The theory of Formal Concept Analysis offers methods to

- split large diagrams into smaller ones, so that the information content is preserved,
- browse through lattices and thereby build *conceptual landscapes* of information,
- coarsen the view, showing rough images that can (locally) be refined to the original data.

This utilizes the mathematical theory of congruences, homomorphisms, embeddings, tensor products, dimension, etc. . . .

What if the data set is large?

With growing size of the data set, it quickly becomes unreasonable to display the full data in a single lattice diagram.

The theory of Formal Concept Analysis offers methods to

- split large diagrams into smaller ones, so that the information content is preserved,
- browse through lattices and thereby build *conceptual landscapes* of information,
- coarsen the view, showing rough images that can (locally) be refined to the original data.

This utilizes the mathematical theory of congruences, homomorphisms, embeddings, tensor products, dimension, etc. . . .

What if the data set is large?

With growing size of the data set, it quickly becomes unreasonable to display the full data in a single lattice diagram.

The theory of Formal Concept Analysis offers methods to

- split large diagrams into smaller ones, so that the information content is preserved,
- browse through lattices and thereby build *conceptual landscapes* of information,
- coarsen the view, showing rough images that can (locally) be refined to the original data.

This utilizes the mathematical theory of congruences, homomorphisms, embeddings, tensor products, dimension, etc. . . .

What if the data set is large?

With growing size of the data set, it quickly becomes unreasonable to display the full data in a single lattice diagram.

The theory of Formal Concept Analysis offers methods to

- split large diagrams into smaller ones, so that the information content is preserved,
- browse through lattices and thereby build *conceptual landscapes* of information,
- coarsen the view, showing rough images that can (locally) be refined to the original data.

This utilizes the mathematical theory of congruences, homomorphisms, embeddings, tensor products, dimension, etc. . .

Outline

Formal contexts that frequently occur

There are many series of formal contexts that have an suggestive interpretation. Such formal contexts will be called **scales**.

So formally, a scale is the same as a formal context. But it is meant to have a special interpretation.

Examples of scales are *nominal*, *ordinal*, *multiordinal*, *contranominal*, and *dichotomic* scales.

Conceptual Scaling

The basic data type of Formal Concept Analysis is that of a formal context.
But data is often given in form of a **many-valued context**.

Many–valued contexts are translated to one-valued context via **conceptual scaling**. This is not automatic; it is an act of *interpretation.*

A many-valued context: driving test

All possible outcomes of a driving test:

	theory	practice	total
1	pass	pass	pass
2	pass	fail	fail
3	fail	pass	fail
4	fail	fail	fail

Is there a concept lattice?

Scaling the driving test context

A natural choice here is the **dichotomic scale**, that stands for a pair of attribute values being negations of each other:

	theory	practice	total
1	pass	pass	pass
2	pass	fail	fail
3	fail	pass	fail
4	fail	fail	fail

	pass	fail
pass	×	
fail		×

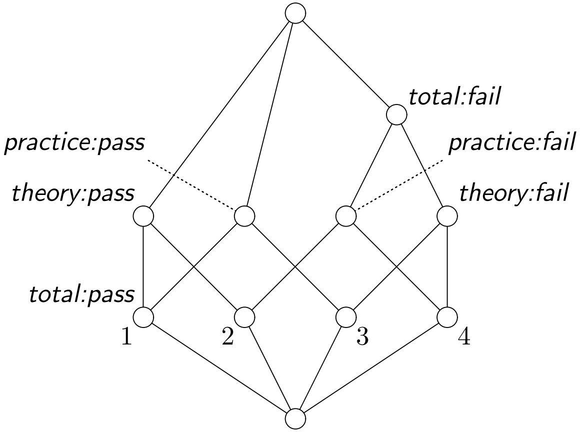
To obtain the **derived context**, replace each many valued attribute by the scale attributes and each attribute value by the corresponding row of the scale.

The derived context for the scaled driving test

	theory	practice	total
1	pass	pass	pass
2	pass	fail	fail
3	fail	pass	fail
4	fail	fail	fail

	theory: pass	theory: fail	practice: pass	practice: fail	total: pass	total: fail
1	×		×		×	
2	×			×		×
3		×	×			×
4		×		×		×

A concept lattice for the scaled driving test



Choosing a scale for each many-valued attribute

Here is a tiny many-valued context with only one many-valued attribute.

	Size
trickle	very small
brook	small
river	large
stream	very large

How can this be transformed into a one-valued context?

A naive scaling

	Size
trickle	very small
brook	small
river	large
stream	very large

	Size:			
	very small	small	large	very large
trickle	×			
brook		×		
river			×	
stream				×

A more intuitive scaling

	Size
trickle	very small
brook	small
river	large
stream	very large

	Size:			
	very small	small	large	very large
trickle	×	×		
brook		×		
river			×	
stream			×	×

Elephant turtles on the Galapagos

Galapagos island	Island size	Opuntion		Turtle type		
		bushy	treelike	dome	intermediate	saddle
Albemarle	4278 km^2	+	-	+	-	-
Indefatigable	1025 km^2	+	-	+	-	-
Narborough	650 km^2	+	-	+	-	-
James	574 km^2	+	-	-	+	-
Chatham	ca. 500 km^2	+	-	-	+	-
Charles	ca. 200 km^2	+	-	-	+	+
Hood	$<$100 km^2	+	-	-	-	+
Bindloe	$<$100 km^2	-	+	-	-	-
Abingdon	$<$100 km^2	+	-	-	-	+
Barringdon	$<$100 km^2	+	-	-	+	+
Tower	$<$100 km^2	-	+	-	-	-
Wenman	$<$100 km^2	-	+	-	-	-
Culpepper	$<$100 km^2	-	+	-	-	-
Jervis	$<$100 km^2	+	-	-	+	+

Joachim Jaenicke (Ed.). *Materialienhandbuch Kursunterricht Biologie, Band 6: Evolution.* Aulis Verlag Köln 1997, ISBN 3-7614-1966-X.

Scales for Island Size and for Opuntia

Island size	small	not large	not small	large
< 100 km^2	×	×		
100–1000 km^2		×	×	
> 1000 km^2			×	×

Opuntia bushy | treelike		bushy	treelike
$+$	$-$	×	
$-$	$+$		×

Each many-valued attribute is replaced by the scale attributes.
Each value is replaced by the corresponding row of the scale.

Scales for Island Size and for Opuntia

Island size	small	not large	not small	large
< 100 km^2	×	×		
100–1000 km^2		×	×	
> 1000 km^2			×	×

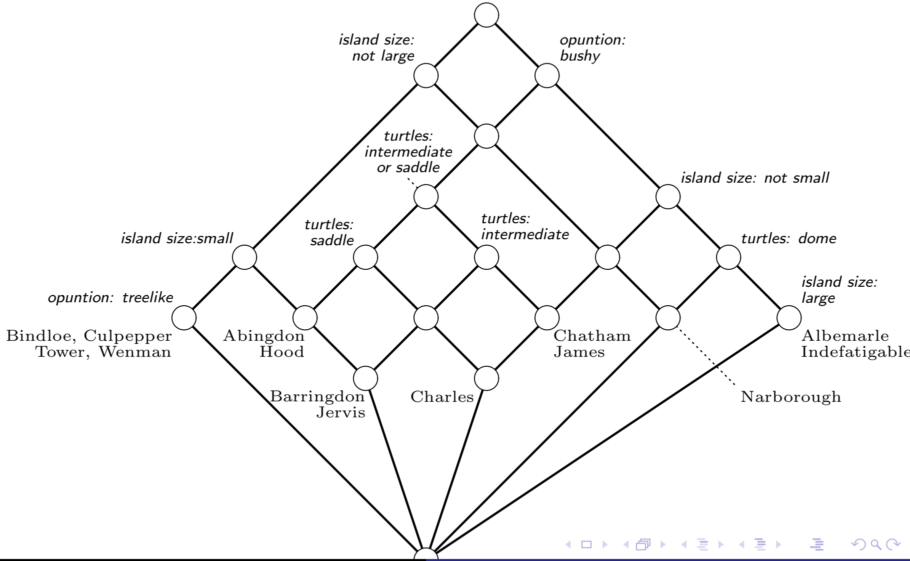
Opuntia bushy | treelike	bushy	treelike
$+$ $-$	×	
$-$ $+$		×

Each many-valued attribute is replaced by the scale attributes.
Each value is replaced by the corresponding row of the scale.

The derived context

	island size: large	island size: not small	island size: not large	island size: small	opuntion: bushy	opuntion: treelike	turtles: dome	turtles: intermediate or saddle	turtles: saddle	turtles: intermediate
Albemarle	×	×			×		×			
Indefatigable	×	×			×		×			
Narborough		×	×		×		×			
James		×	×		×			×		×
Chatham		×	×		×			×		×
Charles		×	×		×			×	×	×
Hood			×	×	×			×	×	
Bindloe			×	×		×				
Abingdon			×	×	×			×	×	
Barringdon			×	×	×			×	×	×
Tower			×	×		×				
Wenman			×	×		×				
Culpepper			×	×		×				
Jervis			×	×	×			×	×	×

Turtles concept lattice



Outline

Recall: The derivation operators

Recall a definition from the beginning:

For a formal context (G, M, I) and given $A \subseteq G$, we define

$$A' := \{m \in M \mid g\ I\ m \text{ for all } g \in A\}.$$

Dually, for $B \subseteq M$, we have by

$$B' := \{g \in G \mid g\ I\ m \text{ for all } m \in B\}.$$

These two operators are the **derivation operators** for (G, M, I).

Iterating derivation

The derivation operators can be combined:

Starting with a set $A \subseteq G$, we obtain that A' is a subset of M.
Applying the second operator on this set, we get $(A')'$, or A'' for short.
This is a set of objects.
Continuing, we obtain A''', A'''', and so on.
But these sets are not all different, as the next proposition shows:

Properties of derivation

Proposition

For subsets $A, A_1, A_2 \subseteq G$ *we have*

1. $A_1 \subseteq A_2 \implies A_2' \subseteq A_1'$,
2. $A \subseteq A''$,
3. $A' = A'''$.

Dually, for subsets $B, B_1, B_2 \subseteq M$ *we have*

1. $B_1 \subseteq B_2 \implies B_2' \subseteq B_1'$,
2. $B \subseteq B''$,
3. $B' = B'''$.

Extent closure, intent closure

Combining the derivation operators, we get two operators of the form

$$X \mapsto X'',$$

one on G, the other on M. For $A \subseteq G$ we have that $A'' \subseteq G$. The set A'' is called the **extent closure** of A. Dually, when $B \subseteq M$, then also $B'' \subseteq M$, and B'' is called the **intent closure** of B. Note that A'' and B'' have a clear meaning for the data:

- Whenever all objects from a set $A \subseteq G$ have a common attribute m, then also all objects from A'' have that attribute.
- Whenever an object $g \in G$ has all attributes from $B \subseteq M$, then this object also has all attributes from B''.

Properties of closure

Some simple properties of these operators are listed in the next proposition. They are easy consequences of Proposition 1.

Proposition

For subsets $A, A_1, A_2 \subseteq G$ we have

1. $A_1 \subseteq A_2$ *implies* $A_1'' \subseteq A_2''$,
2. $A \subseteq A''$,
3. $(A'')'' = A''$.

Dually, for subsets $B, B_1, B_2 \subseteq M$ we have

1. $B_1 \subseteq B_2$ *implies* $B_1'' \subseteq B_2''$,
2. $B \subseteq B''$,
3. $(B'')'' = B''$.

Closure operators

There is a technical term for operators satisfying the properties given in Proposition 2:
They are called **closure operators**.

The sets which are images of a closure operator are the **closed sets**.

Thus, in the case of a closure operator $X \mapsto X''$, the closed sets are the sets of the form X''.

Closed sets are intents and extents

Proposition

If (G, M, I) is a formal context and $A \subseteq G$, then A'' is an extent. Conversely, if A is an extent of (G, M, I), then $A = A''$. Dually if $B \subseteq M$, then B'' is an intent, and every intent B satisfies $B'' = B$.

Proof.

This follows from the fact that for each subset $A \subseteq G$ the pair (A'', A') is a formal concept, and that similarly (B', B'') is a concept whenever $B \subseteq M$. □

Therefore, the closed sets of the closure operator $A \mapsto A''$, $A \subseteq G$ are precisely the extents of (G, M, I), and the closed sets of the operator $B \mapsto B''$, $B \subseteq M$, are precisely the intents.

Representing sets by bit vectors

We start by giving our base set M an arbitrary linear order,

$$M = \{m_1 < m_2 < \cdots < m_n\}.$$

Every subset $S \subseteq M$ can conveniently be described by a row of crosses and blanks.

For example, if $M = \{a < b < c < d < e < f < g\}$, the subset $S := \{a, c, d, f\}$ will be written as

×	.	×	×	.	×	.

.

In this notation it is easy to see if a given set is a subset of another given set, etc.

The lectic order

We introduce a *linear* order of the subsets of M, called the **lectic order** $\leq$, as follows:

Let $A, B \subseteq M$ be two distinct subsets. We say that A is *lectically smaller* than B, if the smallest element in which A and B differ belongs to B.

Formally,

$$A < B \quad :\Longleftrightarrow \quad \exists_i \, (i \in B,\ i \notin A,\ \forall_{j<i} \, (j \in A \iff j \in B)).$$

For example $\{a, c, e, f\} < \{a, c, d, f\}$, because the smallest element in which the two sets differ is D, and this element belongs to the larger set.

Understanding the lectic order

The lectic order is just the lexicographic order of the incidence vectors:

×	.	×	.	×	×	.

$$\updownarrow$$

×	.	×	×	.	×	.

.

Note that the lectic order extends the subset-order, i.e.,

$$A \subseteq B \implies A \leq B.$$

A proposition

The following notation is helpful:

$$A <_i B \quad :\Longleftrightarrow \quad (i \in B,\ i \notin A,\ \forall_{j<i}\,(j \in A \iff j \in B)).$$

In words: $A <_i B$ iff i is the smallest element in which A and B differ, and $i \in B$.

Proposition

1. *$A < B$ if and only if $A <_i B$ for some $i \in M$.*
2. *If $A <_i B$ and $A <_j C$ with $i < j$, then $C <_i B$.*

Finding all closed sets

We consider a closure operator

$$A \mapsto A''$$

on the base set M. To each subset $A \subseteq M$ it gives its closure $A'' \subseteq M$.

Our task is to find a list of all these closures.

The naive algorithm “for each $A \subseteq M$ and check if the result is already listed” requires an exponential number of lookups in a list that may have exponential size.

Closures in lectic order

A better idea is to generate the closures in lectic order.

We will show how to compute, given a closed set, the *lectically next* one.

Then no lookups in the list of found solutions are necessary. Actually, it will not even be necessary to store the list. For many applications it will suffice to generate the list elements on demand. Therefore we do not have to store exponentially many closed sets. Instead, we shall store just *one*!.

The $\oplus$-construction

To find the next closure we define for $A \subseteq M$ and $m_i \in M$

$$A \oplus m_i := ((A \cap \{m_1, \ldots, m_{i-1}\}) \cup \{m_i\})''.$$

For example, let $A := \{a, c, d, f\}$ and $m_i := e$.

$$\begin{array}{|c|c|c|c|c|c|c|} \multicolumn{4}{c}{} & \multicolumn{1}{c}{\downarrow} & \multicolumn{2}{c}{} \\ \hline \times & . & \times & \times & . & \times & . \\ \hline \end{array}.$$

We first remove all elements that are greater or equal m_i from A:

$$\begin{array}{|c|c|c|c|c|c|c|} \multicolumn{4}{c}{} & \multicolumn{1}{c}{\downarrow} & \multicolumn{2}{c}{} \\ \hline \times & . & \times & \times & . & . & . \\ \hline \end{array}.$$

Then we insert m_i

$$\begin{array}{|c|c|c|c|c|c|c|} \multicolumn{4}{c}{} & \multicolumn{1}{c}{\downarrow} & \multicolumn{2}{c}{} \\ \hline \times & . & \times & \times & \times & . & . \\ \hline \end{array}$$

and form the closure.

The next closed set

Proposition

1. *If $i \notin A$ then $A < A \oplus i$.*
2. *If B is closed and $A <_i B$ then $A \oplus i \subseteq B$, in particular $A \oplus i \leq B$.*
3. *If B is closed and $A <_i B$ then $A <_i A \oplus i$.*

Using this, it is easy to characterize the "next" closed set:

Theorem

The smallest closed set larger than a given set $A \subset M$ with respect to the lectic order is

$$A \oplus i,$$

i being the largest element of M with $A <_i A \oplus i$.

Turning the theorem into an algorithm

```
Algorithm ALL CLOSURES
Input:  A closure operator $X \mapsto X''$ on a finite set $M$.
Output:  All closed sets in lectic order.
begin
  First_Closure;
  repeat
    Output $A$;
    Next_Closure;
  until not *success*;
end.
```

Finding the first closed set . . .

```
Algorithm FIRST CLOSURE
Input:  A closure operator $X \mapsto X''$ on a finite set $M$.
Output:  The closure $A$ of the empty set.
begin
   $A := \emptyset''$;
end.
```

. . . and all others

```
Algorithm Next Closure
Input:    A closure operator $X \mapsto X''$ on a finite set $M$,
              and a subset $A \subseteq M$.
Output:   $A$ is replaced by the lectically next closed set.
begin
  $i :=$ largest element of $M$;
  $i :=$ succ($i$);
  success := false;
  repeat
    $i :=$ pred($i$);
    if $i \notin A$ then
    begin
      $A := A \cup \{i\}$;
      $B := A''$;
      if $B \setminus A$ contains no element $< i$ then
      begin
        A:= B;
        success := true;
      end;
    end else $A := A \setminus \{i\}$;
  until success or $i =$ smallest element of $M$.
end.
```

Finding formal concepts

Recall that for any formal context (G, M, I), the set of extents is a closure system on G and the set of intents is a closure system on M. We can therefore apply the NEXT CLOSURE algorithm to compute all extents or all intents, and thereby all formal concepts, of a formal context.

When we use the NEXT CLOSURE algorithm for extents, we call it NEXT EXTENT, and if we use it for concept intents, we speak of the NEXT INTENT algorithm. These algorithms therefore are not new, but merely a specialization of the general algorithm to specific closure systems.

We demonstrate NEXT INTENT on a small example:

		a	b	c	d	e	f	g	h
		brilliant white	fine white	white	for high performance copiers	for copiers	for liquid toner copiers	for type writers	for for double sided copying
1	Copy-Lux	×			×	×		×	×
2	Copy-X		×		×	×		×	×
3	Copy			×	×	×		×	×
4	Liquid-Copy			×		×	×		×
5	Office			×		×		×	
6	Offset			×				×	

The formal context (which was copied from an advertisement) shows which papers of the respective company serve which copying purposes.

A run of Next intent

A	i	$(A \cap \{1,\ldots,i\}) \cup \{i\}$	$A \oplus i$	smallest new element	success?
$\emptyset$	$= \emptyset''$ (first intent).				yes
$\emptyset$	h	$\{h\}$	$\{e,h\}$	e	no
$\emptyset$	g	$\{g\}$	$\{g\}$	g	yes
$\{g\}$	h	$\{g,h\}$	$\{d,e,g,h\}$	d	no
$\{g\}$	f	$\{f\}$	$\{c,e,f,h\}$	c	no
$\{g\}$	e	$\{e\}$	$\{e\}$	e	yes
$\{e\}$	h	$\{e,h\}$	$\{e,h\}$	h	yes
$\{e,h\}$	g	$\{e,g\}$	$\{e,g\}$	g	yes
$\{e,g\}$	h	$\{e,g,h\}$	$\{d,e,g,h\}$	d	no
$\{e,g\}$	f	$\{e,f\}$	$\{c,e,f,h\}$	c	no
$\{e,g\}$	d	$\{d\}$	$\{d,e,g,h\}$	d	yes
$\{d,e,g,h\}$	f	$\{d,e,f\}$	$\{a,b,c,d,e,f,g,h\}$	a	no
$\{d,e,g,h\}$	c	$\{c\}$	$\{c\}$	c	yes
$\{c\}$	h	$\{c,h\}$	$\{c,e,h\}$	e	no
$\{c\}$	g	$\{c,g\}$	$\{c,g\}$	g	yes
$\{c,g\}$	h	$\{c,g,h\}$	$\{c,d,e,g,h\}$	d	no
$\{c,g\}$	f	$\{c,f\}$	$\{c,e,f,h\}$	e	no
$\{c,g\}$	e	$\{c,e\}$	$\{c,e\}$	e	yes
$\{c,e\}$	h	$\{c,e,h\}$	$\{c,e,h\}$	h	yes

A run . . . (contd.)

A	i	$(A \cap \{1,\ldots,i\}) \cup \{i\}$	$A \oplus i$	smallest new element	success?
$\{c,e\}$	h	$\{c,e,h\}$	$\{c,e,h\}$	h	yes
$\{c,e,h\}$	g	$\{c,e,g\}$	$\{c,e,g\}$	g	yes
$\{c,e,g\}$	h	$\{c,e,g,h\}$	$\{c,d,e,g,h\}$	d	no
$\{c,e,g\}$	f	$\{c,e,f\}$	$\{c,e,f,h\}$	f	yes
$\{c,e,f,h\}$	g	$\{c,e,f,g\}$	$\{a,b,c,d,e,f,g,h\}$	a	no
$\{c,e,f,h\}$	d	$\{c,d\}$	$\{c,d,e,g,h\}$	d	yes
$\{c,d,e,g,h\}$	f	$\{c,d,e,f\}$	$\{a,b,c,d,e,f,g,h\}$	a	no
$\{c,d,e,g,h\}$	b	$\{b\}$	$\{b,d,e,g,h\}$	b	yes
$\{b,d,e,g,h\}$	f	$\{b,d,e,f\}$	$\{a,b,c,d,e,f,g,h\}$	a	no
$\{b,d,e,g,h\}$	c	$\{b,c\}$	$\{a,b,c,d,e,f,g,h\}$	a	no
$\{b,d,e,g,h\}$	a	$\{a\}$	$\{a,d,e,g,h\}$	a	yes
$\{a,d,e,g,h\}$	f	$\{a,d,e,f\}$	$\{a,b,c,d,e,f,g,h\}$	b	no
$\{a,d,e,g,h\}$	c	$\{a,c\}$	$\{a,b,c,d,e,f,g,h\}$	b	no
$\{a,d,e,g,h\}$	b	$\{a,b\}$	$\{a,b,c,d,e,f,g,h\}$	b	yes
$\{a,b,c,d,e,f,g,h\}$	$= M$ (last intent).				

Frequent closed itemsets

The data mining community often uses the word “item” for what we call “attribute” here. “Itemsets” are thus just the subsets of M.

An itemset $B \subseteq M$ is *frequent* if it is common to many objects, in other words, if B' is large (i.e., above some given threshold). Of course, all subsets of a frequent set are frequent.

We know from Proposition 1 that $B' = (B'')'$, which says that if B is frequent then also its intent closure B'' is frequent. Such sets which are both frequent and closed are called *frequent closed itemsets*.

Translated to Formal Concept Analysis, frequent closed itemsets are just the intents of formal concepts with large extents.

Icebergs

Definition

Let (G, M, I) be a nontrivial formal context and let $\textsc{minsupp}$ be a real number with $0 \leq \textsc{minsupp} \leq 1$. A subset $B \subseteq M$ is called **frequent** if

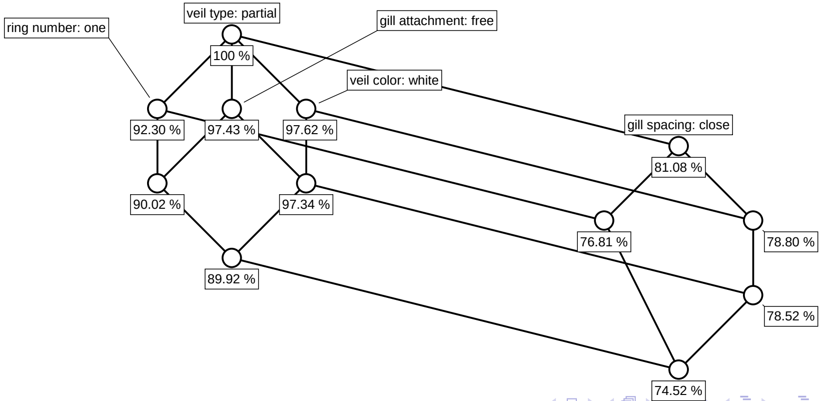
$$\frac{|B'|}{|G|} \geq \textsc{minsupp}.$$

The formal concepts with frequent intent form an "upper end" in the concept lattice (more precisely, an *order filter*, also called an *upset*).

They are called the **iceberg** of the concept lattice (for minimum support $\textsc{minsupp}$). Adding a smallest element makes the iceberg a lattice again, called the **iceberg lattice**.

An Iceberg of Mushrooms

As an example, we show an iceberg concept lattice for the popular MUSHROOM data set, for a minimum support of 70 %. The full concept lattice of this data set has 32.086 elements.



Next frequent intent

The Next closure algorithm can also be used to compute all frequent intents and thereby the iceberg lattice.

Theorem

The smallest frequent intent larger than a given set $A \subset M$ with respect to the lectic order, if it exists, is

$$A \oplus i,$$

i being the largest element of M satisfying

- $A <_i A \oplus i$ *and*
- $A \oplus i$ *frequent.*

It is easy to include the second condition added by this Theorem in the Next closure–algorithm.

Outline

Implications in a formal context

Let (G, M, I) be a formal context and let $A, B \subseteq M$.

We say that the **implication**

$$A \rightarrow B$$

holds in (G, M, I) iff every object that has all the attributes from A also has all the attributes from B.

There are simple rules for implication inference in formal contexts (Armstrong rules).

The theory of contextual implications can be applied to *functional dependencies*, *association rules*, and generalizations thereof.

Implications in a formal context

Let (G, M, I) be a formal context and let $A, B \subseteq M$.

We say that the **implication**

$$A \to B$$

holds in (G, M, I) iff every object that has all the attributes from A also has all the attributes from B.

There are simple rules for implication inference in formal contexts (Armstrong rules).

The theory of contextual implications can be applied to *functional dependencies*, *association rules*, and generalizations thereof.

Why implications

Implications are not very expressive. They represent only a *fragment of Propositional Logic*. To obtain full propositional expressivity, one had to use *clauses*.

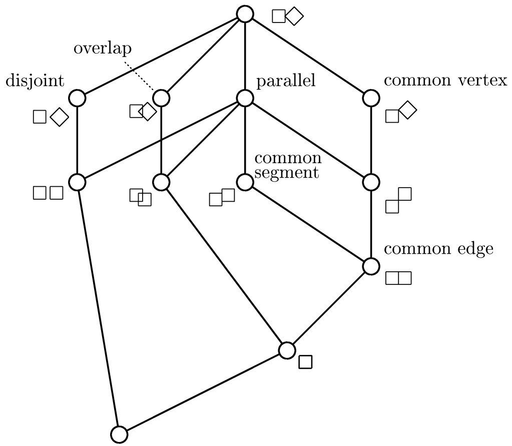
But

- implications are intuitive and easy to handle, while
- clause inference is $\mathcal{NP}$-complete.

In other words:

- Implications are nice,
- clauses are not.

An example: Pairs of (unit-) squares



Implications can be read from the diagram using infima.

The stem base

There is a canonical minimal generating set for the implicational theory of every formal context, called the **stem base**.

The stem base is sound, complete, and of minimal cardinality.

We have convenient algorithms for computing the stem base, but better ones may be possible.

The stem base

There is a canonical minimal generating set for the implicational theory of every formal context, called the **stem base**.

The stem base is sound, complete, and of minimal cardinality.

We have convenient algorithms for computing the stem base, but better ones may be possible.

Pseudo–intents

Let (G, M, I) be a formal context, and let M be finite.

A set $P \subseteq M$ is a **pseudo–intent** of (G, M, I) iff the following two conditions are satisfied:

1. $P \neq P''$, and
2. if $Q \subset P$, $Q \neq P$ is a pseudo–intent of (G, M, I), then $Q'' \subseteq P$.

The stem base of a formal context is the set

$$\{P \to P'' \mid P \text{ pseudo–intent }\}.$$

Proof

To show that the stem base is complete, it suffices to show that the subsets of M which respect all stem-base implications are precisely the concept intents.

So let $B \subseteq M$ be such a set, which respects the stem base, and let $Q \subseteq B$ be a pseudo-intent.

Then $Q \to Q''$ is in the stem base and therefore $Q'' \subseteq B$.

Thus B satisfies the second condition in the definition of pseudo-intent. B cannot be a pseudo-intent and therefore must violate the first condition. Thus $B = B''$.

A closure operator for closed or pseudo

Form the proof we can guess a generating principle for pseudo-intents:

A set $B \subseteq M$ is an intent or a pseudo-intent iff B respects all implications of the stem base except possibly $B \rightarrow B''$.

This can be used to define a closure operator, the closed sets of which are all intents and all pseudo-intents.

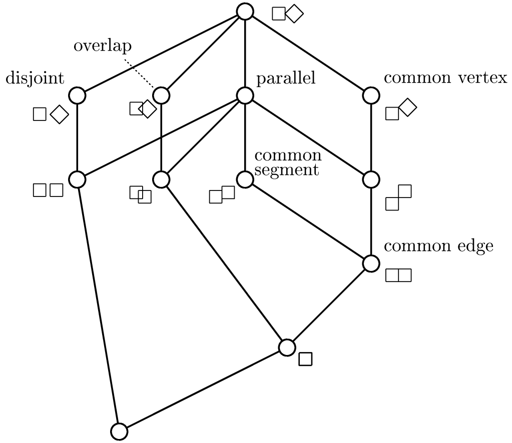
Computing the stem base

Pseudo-intents can be computed with a modified NEXT-INTENT algorithm.

However, the complexity status has not yet fully been clarified.

There are implementations that compute the stem base for examples of moderate size in reasonable time.

Is “Pairs of squares” complete?



Are there other examples of pairs of unit squares?

Did we consider all possible cases?

How can we decide if our selection of examples is complete?

A possible strategy is to prove that every implication, that holds for these examples, holds in general.

- Compute the stem base of the context of examples, and
- prove that these implications hold in general,
- or find counterexamples and extend the example set.

This can nicely be organized in an algorithm, called **attribute exploration**.

Did we consider all possible cases?

How can we decide if our selection of examples is complete?

A possible strategy is to prove that every implication, that holds for these examples, holds in general.

- Compute the stem base of the context of examples, and
- prove that these implications hold in general,
- or find counterexamples and extend the example set.

This can nicely be organized in an algorithm, called **attribute exploration**.

Did we consider all possible cases?

How can we decide if our selection of examples is complete?

A possible strategy is to prove that every implication, that holds for these examples, holds in general.

- Compute the stem base of the context of examples, and
- prove that these implications hold in general,
- or find counterexamples and extend the example set.

This can nicely be organized in an algorithm, called **attribute exploration**.

Stem base of the example set

- common edge $\longrightarrow$ parallel, common vertex, common segment
- common segment $\longrightarrow$ parallel
- parallel, common vertex, common segment $\longrightarrow$ common edge
- overlap, common vertex $\longrightarrow$ parallel, common segment, common edge
- overlap, parallel, common segment $\longrightarrow$ common edge, common vertex
- overlap, parallel, common vertex $\longrightarrow$ common segment, common edge
- disjoint, common vertex $\longrightarrow \perp$
- disjoint, parallel, common segment $\longrightarrow \perp$
- disjoint, overlap $\longrightarrow \perp$

Two of the implications do not hold in general

- common edge $\longrightarrow$ parallel, common vertex, common segment
- common segment $\longrightarrow$ parallel
- parallel, common vertex, common segment $\longrightarrow$ common edge
- overlap, common vertex $\longrightarrow$ parallel, common segment, common edge
- overlap, parallel, common segment $\longrightarrow$ common edge, common vertex
- overlap, parallel, common vertex $\longrightarrow$ common segment, common edge
- disjoint, common vertex $\longrightarrow \bot$
- disjoint, parallel, common segment $\longrightarrow \bot$
- disjoint, overlap $\longrightarrow \bot$

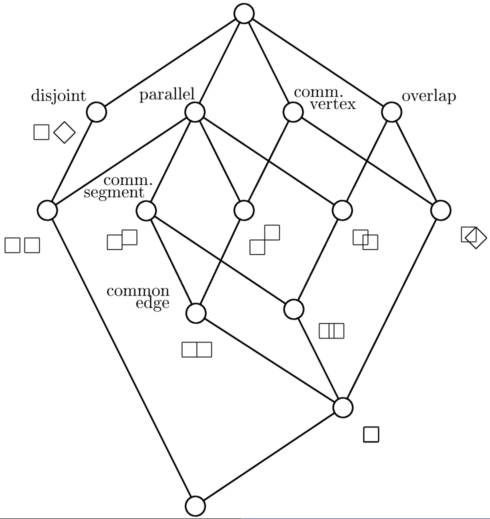
Conterexamples for the two implications

- overlap, common vertex $\longrightarrow$ parallel, common segment, common edge
- Counterexample:
- overlap, parallel, common segment $\longrightarrow$ common edge, common vertex
- Counterexample:

Conterexamples for the two implications

- overlap, common vertex → parallel, common segment, common edge
- Counterexample:
- overlap, parallel, common segment → common edge, common vertex
- Counterexample:

A better choice of examples



An exploration of lattice properties

We investigate finite lattices and properties they may have:

- $g \nearrow m, g \swarrow n \Rightarrow g \neswarrow m$
- semi-convex
- $g \swarrow m, h \nearrow m \Rightarrow g \neswarrow m$
- semimodular
- $SD_\wedge$
- $SD_\vee$
- dually semimodular
- meet-distributive
- join-distributive
- modular
- B
- distributive

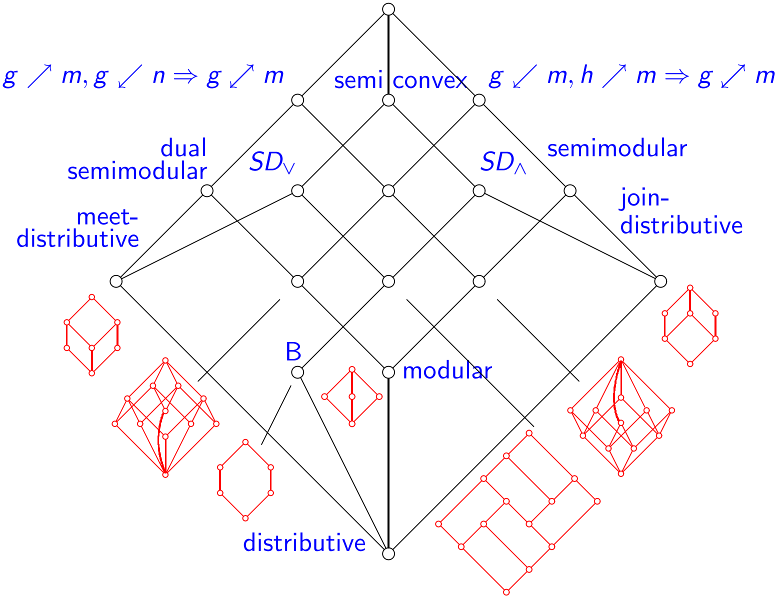
Interactive exploration query

A typical question that would be asked during the exploration process would be:

Does every lattice with the attributes
$SD_\vee$ and dually semimodular
also have the attribute join-distributive?

The user must either confirm or provide a counter example.

Exploring attribute combinations


$g \nearrow m, g \swarrow n \Rightarrow g \swarrow\!\!\!\nearrow m$
semi convex
$g \swarrow m, h \nearrow m \Rightarrow g \swarrow\!\!\!\nearrow m$
dual semimodular
$SD_\vee$
$SD_\wedge$
semimodular
meet-distributive
join-distributive
B
modular
distributive

An unpleasant example: Driving test

	theory		driving		license	
	pass	fail	pass	fail	pass	fail
1	×		×		×	
2	×			×		×
3		×	×			×
4		×		×		×

We expect that the stem base, which is irredundant, sound and complete, expresses that

$$\text{license} = \text{pass} \quad \leftrightarrow \quad \text{theory} = \text{pass } \textit{and} \text{ practice} = \text{pass}.$$

. . . and its stem base

$$
\begin{array}{lcl}
\text{driving} = \text{fail} & \to & \text{license} = \text{fail} \\
\text{theory} = \text{fail} & \to & \text{license} = \text{fail} \\
\text{license} = \text{fail}, \text{driving} = \text{pass} & \to & \text{theory} = \text{fail} \\
\text{license} = \text{fail}, \text{theory} = \text{pass} & \to & \text{driving} = \text{fail} \\
\text{driving} = \text{pass}, \text{theory} = \text{pass} & \to & \text{license} = \text{pass} \\
\text{license} = \text{pass} & \to & \text{driving} = \text{pass}, \text{theory} = \text{pass} \\
\left.\begin{array}{l}\text{license} = \text{fail}, \text{theory} = \text{fail}, \\ \text{driving} = \text{pass}, \text{driving} = \text{fail}\end{array}\right\} & \to & \perp \\
\left.\begin{array}{l}\text{license} = \text{fail}, \text{theory} = \text{fail}, \\ \text{theory} = \text{pass}, \text{driving} = \text{fail}\end{array}\right\} & \to & \perp
\end{array}
$$

Something must be wrong?

Background knowledge

The reason why the driving test stem base is so complicated is that the stem base algorithm does not "know" that PASS and FAIL are negations of each other.

This shows the necessity to include **background knowledge**, which may be non-implicational.

One frequent instance is that the background knowledge encodes the scaling information of a many-valued context.

Modified inference

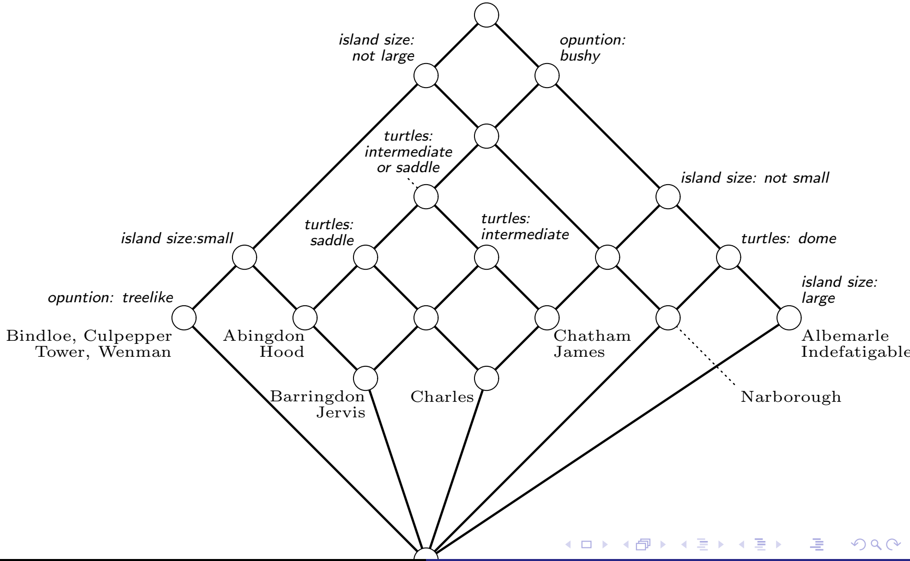
Including background knowledge requires a modified inference mechanism for implications.

In general, this would make the inference problem intractable.

But it can be shown that for a *fixed amount* of non-implicational background knowledge, implication inference remains tractable.

This typically applies to scaling-induced background knowledge.

Logic of the Galapagos turtles



The stem base for the turtles context

1 <5 >turtles type: intermediate ==>size: not large opuntion: treelike turtles type: intermediate or saddle;
2 <5 >turtles type: saddle ==>size: not large opuntion: treelike turtles type: intermediate or saddle;
3 <7 >turtles type: intermediate or saddle ==>size: not large opuntion: treelike;
4 <3 >turtles type: dome ==>size: not small opuntion: treelike;
5 <4 >opuntion: bushy ==>size: not large size: small;
6 <8 >size: small ==>size: not large;
7 <4 >size: not large size: small opuntion: treelike ==>turtles type: intermediate or saddle turtles type: saddle;
8 <0 >size: not large size: small opuntion: treelike opuntion: bushy turtles type: intermediate or saddle turtles type: saddle ==>size: large size: not small turtles type: dome turtles type: intermediate;
9 <6 >size: not small ==>opuntion: treelike;
10 <3 >size: not small size: not large opuntion: treelike turtles type: intermediate or saddle ==>turtles type: intermediate;
11 <0 >size: not small size: not large opuntion: treelike turtles type: dome turtles type: intermediate or saddle turtles type: intermediate ==>size: large size: small opuntion: bushy turtles type: saddle;
12 <0 >size: not small size: not large size: small opuntion: treelike turtles type: intermediate or saddle turtles type: saddle turtles type: intermediate ==>size: large opuntion: bushy turtles type: dome;
13 <2 >size: large ==>size: not small opuntion: treelike turtles type: dome;
14 <0 >size: large size: not small size: not large opuntion: treelike turtles type: dome ==>

size: small opuntion: bushy turtles type: intermediate or saddle turtles type: saddle turtles type: intermediate;

The scaling–induced logic

Island size:

small, not small	$\multimap$	$\emptyset$
$\emptyset$	$\multimap$	small, not small
large, not large	$\multimap$	$\emptyset$
$\emptyset$	$\multimap$	large, not large
small	$\multimap$	not large

Opuntia:

bushy, treelike	$\multimap$	$\emptyset$
$\emptyset$	$\multimap$	bushy, treelike

Turtles:

dome, intermediate_or_saddle	$\multimap$	$\emptyset$
intermediate_or_saddle	$\multimap$	intermediate, saddle
saddle	$\multimap$	intermediate_or_saddle
intermediate	$\multimap$	intermediate_or_saddle

The remaining implicational logic

$$
\begin{aligned}
\text{island size: large} &\rightarrow \text{turtles: dome}\\
\text{island size: not small} &\rightarrow \text{opuntia: bushy}\\
\text{turtles: dome} &\rightarrow \text{island size: not small}\\
\left.\begin{array}{r}\text{island size: not large,}\\ \text{turtles: intermediate\_or\_saddle}\end{array}\right\} &\rightarrow \text{opuntia: bushy}\\
\left.\begin{array}{r}\text{island size: small,}\\ \text{opuntia: bushy}\end{array}\right\} &\rightarrow \text{turtles: saddle}\\
\left.\begin{array}{r}\text{island size: not small, not large,}\\ \text{turtles: intermediate\_or\_saddle}\end{array}\right\} &\rightarrow \text{turtles: intermediate.}
\end{aligned}
$$

NEXT INTENT modulo symmetry

Another possible generalization is based on the fact that the NEXT INTENT algorithm can respect *context automorphisms*.

This version does not compute all intents, but only one from each symmetry class.

This can be used to generalize attribute exploration to FOL **rule exploration**, even with background knowledge.

We show the method by means two examples.

A context of implications . . .

	$\emptyset \to a$	$\emptyset \to b$	$\emptyset \to c$	$a \to b$	$a \to c$	$b \to a$	$b \to c$	$c \to a$	$c \to b$	$a, b \to c$	$a, c \to b$	$b, c \to a$
$\emptyset$				×	×	×	×	×	×	×	×	×
$\{a\}$	×					×	×	×	×	×	×	×
$\{b\}$		×		×	×			×	×	×	×	×
$\{c\}$			×	×	×	×	×			×	×	×
$\{a, b\}$	×	×		×		×		×	×		×	×
$\{a, c\}$	×		×		×	×	×	×		×		×
$\{b, c\}$		×	×	×	×		×		×	×	×	
$\{a, b, c\}$	×	×	×	×	×	×	×	×	×	×	×	×

... and its stem base

1. $c \to b$ implies $a, c \to b$;
2. $c \to b;\ a, c \to b;\ b, c \to a$ implies $c \to a$;
3. $c \to a$ implies $b, c \to a$;
4. $c \to a;\ a, c \to b;\ b, c \to a$ implies $c \to b$;
5. $b \to c$ implies $a, b \to c$;
6. $b \to c;\ a, b \to c;\ b, c \to a$ implies $b \to a$;
7. $b \to a$ implies $b, c \to a$;
8. $b \to a;\ a, b \to c;\ b, c \to a$ implies $b \to c$;
9. $a \to c$ implies $a, b \to c$;
10. $a \to c;\ a, b \to c;\ a, c \to b$ implies $a \to b$;
11. $a \to b$ implies $a, c \to b$;
12. $a \to b;\ a, b \to c;\ a, c \to b$ implies $a \to c$;
13. $\emptyset \to c$ implies $a \to c;\ b \to c;\ a, b \to c$;
14. $\emptyset \to c;\ a \to c;\ b \to a;\ b \to c;\ c \to a;\ a, b \to c;\ b, c \to a$ implies $\emptyset \to a$;
15. $\emptyset \to c;\ a \to b;\ a \to c;\ b \to c;\ c \to b;\ a, b \to c;\ a, c \to b$ implies $\emptyset \to b$;
16. $\emptyset \to b$ implies $a \to b;\ c \to b;\ a, c \to b$;
17. $\emptyset \to b;\ a \to b;\ b \to a;\ c \to a;\ c \to b;\ a, c \to b;\ b, c \to a$ implies $\emptyset \to a$;
18. $\emptyset \to b;\ a \to b;\ a \to c;\ b \to c;\ c \to b;\ a, b \to c;\ a, c \to b$ implies $\emptyset \to c$;
19. $\emptyset \to a$ implies $b \to a;\ c \to a;\ b, c \to a$;
20. $\emptyset \to a;\ a \to c;\ b \to a;\ b \to c;\ c \to a;\ a, b \to c;\ b, c \to a$ implies $\emptyset \to c$;
21. $\emptyset \to a;\ a \to b;\ b \to a;\ c \to a;\ c \to b;\ a, c \to b;\ b, c \to a$ implies $\emptyset \to b$;

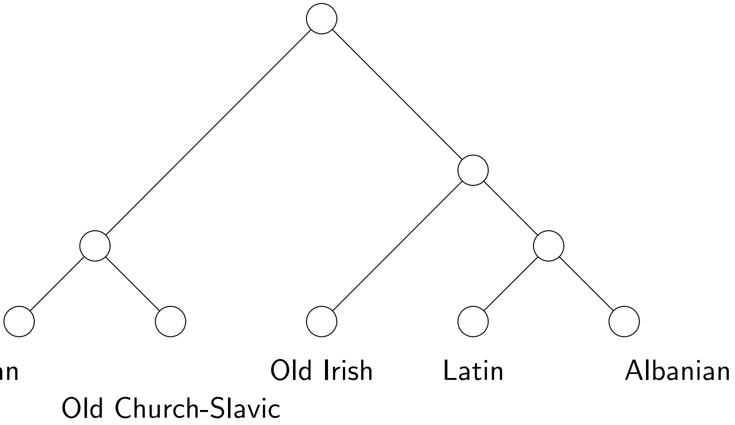
Stem base “modulo automorphisms”

1. $$\frac{x \longrightarrow y}{x, z \longrightarrow y}$$
2. $$\frac{x \longrightarrow y;\ x, y \longrightarrow z;\ \color{red}{x, z \longrightarrow y}}{y \longrightarrow x}$$
3. $$\frac{\emptyset \longrightarrow x}{y \longrightarrow x}$$
4. $$\frac{\emptyset \longrightarrow x;\ x \longrightarrow y;\ \color{red}{y \longrightarrow x;\ z \longrightarrow y;\ z \longrightarrow x;\ y, z \longrightarrow x;\ x, z \longrightarrow y}}{\emptyset \longrightarrow y}$$

Stem base “modulo automorphisms”, simplified

1. $$\frac{x \longrightarrow y}{x, z \longrightarrow y}$$
2. $$\frac{x \longrightarrow y;\ x, y \longrightarrow z;}{y \longrightarrow x}$$
3. $$\frac{\emptyset \longrightarrow x}{y \longrightarrow x}$$
4. $$\frac{\emptyset \longrightarrow x;\ x \longrightarrow y;}{\emptyset \longrightarrow y}$$

An evolutionary tree of languages



The versus relation of a tree

Let T be a (finite) (rooted) tree. Most naturally, we can interprete T as a $\vee$-semilattice, in which the root is the largest element and the leaves are minimal elements.

On the set L of leaves we can define a ternay relation

$$xy \mid z :\Longleftrightarrow z \not\leq x \vee y.$$

(The **versus**-relation, read: "x, y versus z").

It is easy to see that the tree structure can be reconstructed from its versus relation.

The problem

We shall discuss the following problem:

Characterize treelike versus-relations!

That means: Find axioms that describe versus-relations derived from trees similarly as

orders	are reflexive, transitive, antisymmetric relations,
equivalences	are reflexive, transitive, symmetric relations.

Can this perhaps be done automatically?

The problem

We shall discuss the following problem:

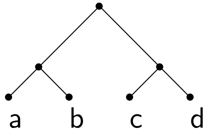
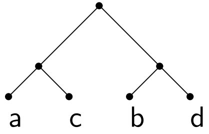
Characterize treelike versus-relations!

That means: Find axioms that describe versus-relations derived from trees similarly as

orders	are reflexive, transitive, antisymmetric relations,
equivalences	are reflexive, transitive, symmetric relations.

Can this perhaps be done automatically?

A context of trees

	$\{a, b, c\}$	$\{a, b, d\}$	$\{a, c, d\}$	$\{b, c, d\}$
	$ab \mid c$	$ab \mid d$	$cd \mid a$	$cd \mid b$
	$ac \mid b$	$bd \mid a$	$ac \mid d$	$bd \mid c$
13 more trees with 4 leaves	$\cdots$	$\cdots$	$\cdots$	$\cdots$

A scale $\mathbb{S}_{\{x,y,z\}}$ for the context of trees

$\mathbb{S}_{\{x,y,z\}}$	$yz \mid x$	$xz \mid y$	$xy \mid z$	$\neg yz \mid x$	$\neg xz \mid y$	$\neg xy \mid z$
$yz \mid x$	×				×	×
$xz \mid y$		×		×		×
$xy \mid z$			×	×	×	

Implications of the derived context

The derived context has 15 objects and 24 attributes.

Its stem base consists of 88 implications.

Removing the scale–induced information leaves 24 implications.

It remains to factor modulo automorphisms, thereby transforming implications into Horn rules (with variables).

Implications of the derived context

The derived context has 15 objects and 24 attributes.

Its stem base consists of 88 implications.

Removing the scale–induced information leaves 24 implications.

It remains to factor modulo automorphisms, thereby transforming implications into Horn rules (with variables).

Implications of the derived context

The derived context has 15 objects and 24 attributes.

Its stem base consists of 88 implications.

Removing the scale–induced information leaves 24 implications.

It remains to factor modulo automorphisms, thereby transforming implications into Horn rules (with variables).

Implications of the derived context

The derived context has 15 objects and 24 attributes.

Its stem base consists of 88 implications.

Removing the scale–induced information leaves 24 implications.

It remains to factor modulo automorphisms, thereby transforming implications into Horn rules (with variables).

A single rule for treelike relations

Treelike versus-relations can be characterized by a single rule:

$$wx \mid y, \neg wz \mid y \quad \longrightarrow \quad wx \mid z.$$

Outline

So what?

- The method of Formal Concept Analysis offers an algebraic approach to data analysis and knowledge processing.
- Its strengths are
  - a solid mathematical and philosophical foundation,
  - $\approx$ 1000 research publications,
  - experience of several hundred application projects,
  - an expressive and intuitive graphical representation,
  - and a good algorithmic basis.
- Due to its elementary yet powerful formal theory, FCA can express other methods, and therefore has the potential to unify the methodology of data analysis.

So what?

- The method of Formal Concept Analysis offers an algebraic approach to data analysis and knowledge processing.
- Its strengths are
  - a solid mathematical and philosophical foundation,
  - $\approx$ 1000 research publications,
  - experience of several hundred application projects,
  - an expressive and intuitive graphical representation,
  - and a good algorithmic basis.
- Due to its elementary yet powerful formal theory, FCA can express other methods, and therefore has the potential to unify the methodology of data analysis.

So what?

- The method of Formal Concept Analysis offers an algebraic approach to data analysis and knowledge processing.
- Its strengths are
  - a solid mathematical and philosophical foundation,
  - $\approx$ 1000 research publications,
  - experience of several hundred application projects,
  - an expressive and intuitive graphical representation,
  - and a good algorithmic basis.
- Due to its elementary yet powerful formal theory, FCA can express other methods, and therefore has the potential to unify the methodology of data analysis.

Thank you for your attention