Learning it quickly (or not): Algorithmic complexity of learning with FCA

S.O. Kuznetsov

skuznetsov@hse.ru

Higher School of Economics,
All-Russia Institute for Scientific and Technical Information (VINITI),
Moscow, Russia
Outline

1. Generating all concepts
2. Generating hypotheses and classifications
4. Generating Duquenne-Guigues (stem) base of implications
Algorithms for computing concepts: A history

1. First algorithms:
   - Y. Malgrange, 1962: Recherche des sous-matrices premières d’une matrice à coefficients binaire
   - G. Fay, 1975: Algorithm for finite Galois connections

2. First efficient algorithms
   - E. Norris, 1978
   - B. Ganter, 1984
   - J.-P. Bordat, 1986 (also constructed the graph of the Hasse diagram)

Common feature of the latter: they used some efficient methods for testing whether a concept has been already generated
A context and its concept lattice

<table>
<thead>
<tr>
<th>G \ M</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>△</td>
<td>×</td>
<td></td>
<td>×</td>
</tr>
<tr>
<td>2</td>
<td>△</td>
<td>×</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>□</td>
<td>×</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>□</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
</tbody>
</table>

a stays for “to have exactly three vertices”
b stays for “to have exactly four vertices”
c stays for “to have a right angle”
d stays for “to have equal sizes”
Published comparisons of algorithms

- A. Guénoche, 1990 (4 algorithms: those of Chein, Norris, Ganter, and Bordat)
- R. Godin et al., 1995 (6 algorithms with modifications)
- B. Goetals and M. Zaki, 2003 (6 algorithms from the data mining community)

A new wave of interest (starting from the first APRIORI algorithm of Agrawal 1993) came from Data Mining, where “frequent (closed) itemsets” are being generated. At FIMI workshops competitions for experimental evaluation of algorithms are organized.
Classification of algorithms. 1

- Top-down vs. bottom-up
- Depth-first vs. breadth-first (for algorithms using graph structure)
- Incremental vs. batch
- Usage vs. nonusage of the diagram graph for computing concepts.
  - For example, the algorithms of Carpineto and Romano (1996) and that of Valtchev et al. (2001) use the diagram graph to compute concepts and those of Norris (1978) and Ganter (1984) do not use it.
Classification of algorithms. 2

- The use of lexicographical order on concept generations (attributes) (Norris 1978, Ganter 1984, Kuznetsov 1993)

  This assumes that there is a linear order on the set of objects (attributes). A generation of a concept is considered canonical if at each step $i$ of applying a closure operator there are no objects in $(A \cup \{g_i\})'' \setminus A$ that have smaller number than $i$.

- Some algorithms divide the set of all concepts into disjoint sets, which allows narrowing down the search space. For example, the algorithm of M. Chein stores concepts in layers, each layer corresponding to some step of the algorithm.
Classification of algorithms. 3

- Computing intersections:
  - intersecting a generated intent with some object intent
  - intersecting already obtained intents (Chein 1969, Zabezhailo et al. 1987)
  - intersecting all object intents of the corresponding extent (Ganter 1984)
  - not using intersections at all (Stumme et al. 2000) (attributes are added to already generated intents and some condition on the number of objects with a chosen set of attributes is checked)

- The use of hash functions (algorithm of Godin et al. 1995), which makes it possible to distribute concepts among “buckets” and to reduce the test for uniqueness.
## Classification of algorithms (2001)

<table>
<thead>
<tr>
<th></th>
<th>m1</th>
<th>m2</th>
<th>m3</th>
<th>m4</th>
<th>m5</th>
<th>m6</th>
<th>m7</th>
<th>m8</th>
<th>m9</th>
<th>m10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bordat</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>NextClosure</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Close by One</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lindig</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Chein</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Nourine</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Norris</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Godin</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dowling</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Titanic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

m1—incremental; m2—uses canonicity based on the lexical order; m3—divides the set of concepts into several parts; m4—uses hash function; m5—maintains an auxiliary tree structure; m6—uses attribute cache; m7—computes intents by subsequently computing intersections of object intents (i.e. \{g\}' \cap \{h\}''); m8—computes intersections of already generated intents; m9—computes intersections of nonobject intents and object intents; m10—uses supports of attribute sets.
Classification of algorithms (2001). 5

Line (Hasse) diagram for the concept lattice of algorithms
Theoretical complexity of problems and algorithms. 1

**Problem 1** Given a context \( K = (G, M, I) \), output all its concepts.

The worst-case complexity of computing all concepts: exponential \( 2^{|A|} \) for the context \( (A, A \neq) \)

**Problem 2** Given a context \( K = (G, M, I) \), output the number of all its concepts. This number is useful for efficient resource allocation.

**Definition:** \#P is the class of counting problems related to decision problems in NP: a problem is in \#P if there is a non-deterministic, polynomial time Turing machine that, for each instance \( I \) of the problem, has a number of accepting computations that is equal to the number of distinct solutions for instance \( I \).

A problem is \#P-complete if it is in \#P and it is \#P-hard, i.e., any problem in \#P can be reduced by Turing to it.

In particular, a problem in \#P is \#P-complete if a \#P-complete problem can be reduced to it. Obviously, \#P = P \implies NP = P.
#P-completeness results

The problem of counting all concepts, given a context $K = (G, M, I)$ is #P-complete. Proof by reduction from monotone 2-CNF problem [Kuznetsov 1989, 2001].

Other examples of #P-complete problems:

- Given a matrix, output its permanent
- Given a bipartite graph, output the number of its perfect matchings
- Given a CNF, output the number of its satisfying assignments
- Given a graph, output the number of its vertex covers
- Given a graph, output the number of its cliques
Delay

Def. 1: An algorithm for listing a family of combinatorial structures is said to have **delay** $d$ (Johnson et al. 1988) if it executes at most $d$ many computation steps before either outputting each next structure or halting.

Def. 2: **Polynomial delay** – $d$ is bounded from above by a polynomial from input size.
Theoretical complexity of problems and algorithms. 1

Many batch algorithms have polynomial delay, among them, e.g., those of

- B. Ganter 1984
- J.-P. Bordat 1986
- S. Kuznetsov 1993

and others with polynomial delays $O(|G|^2 \cdot |M|)$ and $O(|G|^3 \cdot |M|)$.

All these algorithms have $O(|G|^2 \cdot |M| \cdot |L|)$ total runtime complexity. The delay of
incremental algorithms for generating lattices is not polynomial, however the
cumulative delay may be polynomial.
Data structures

Two major steps of algorithms:

- generating a concept candidate
- “testing uniqueness” (to avoid repetitive generation of the same concept)

Tests of this kind require efficient data structures, such as

- Spanning trees (Bordat 1986, Ganter and Reuter 1991),
- Ordered lists (e.g., Ganter 1984),
- CbO trees (consistent with order on concepts, but not with the covering relation, Kuznetsov 1993),
- Tries (Nourine and Raynaud 1999)

Possible alternatives:

- 2-3 balanced trees (Aho et al. 1983)
Theoretical complexity of problems and algorithms. 2

Def. 3: An algorithm is said to have a **cumulative delay** $d$ (Johnson et al. 1988, Goldberg 1993) if at any point of time in any execution of the algorithm with any input $i$ the total number of computation steps that have been executed is at most $d(i)$ plus $K \cdot d(i)$, where $K$ is the number of structures that have been output so far. If $d(i)$ can be bounded by a polynomial of $i$, the algorithm is said to have a **polynomial cumulative delay**.

- Most of the well-known incremental algorithms have polynomial cumulative delays, such as those of Dowling (1993), Carpineto and Romano (1996), The algorithm of Godin et al. (1995) does not have a polynomial cumulative delay, however is very fast for real sparse contexts.

- The algorithm with the best known theoretical worst-case complexity bound is that of L. Nourine and O. Raynaud (1999): $O((|G| + |M|) \cdot |G| \cdot |L|)$ total runtime. This algorithm waits still for efficient practical implementation.
Theoretical worst-case vs. experimental comparison

Discrepancy between expectations based on theoretical worst-case complexity analysis and experimental performance is well-known, e.g., for Algorithms of Linear Programming.

L. Khachiyan around 1980 proved that the Linear Programming problem can be solved in polynomial time by the method of ellipsoids, the latter algorithm being much slower in practice than that of Simplex Method, which has exponential worst-case time complexity.
Decision problems about concepts

INSTANCE A context \((G, M, I)\), a constant \(k \leq \max\{|G|, |M|\}\)

QUESTION Is there a concept \((A, B)\) with extent/intent smaller/equal/larger than \(k\)?

<table>
<thead>
<tr>
<th></th>
<th>(\leq)</th>
<th>=</th>
<th>(\geq)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(</td>
<td>A</td>
<td>)</td>
<td>P</td>
</tr>
<tr>
<td>(</td>
<td>B</td>
<td>)</td>
<td>P</td>
</tr>
<tr>
<td>(</td>
<td>A</td>
<td>+</td>
<td>B</td>
</tr>
</tbody>
</table>

NP: problem is NP-complete (reduction from 3-Matching [Kuznetsov 1989, 2001])
P: there is a polynomial algorithm for the problem
The size of a “maximal concept”

INSTANCE A context \((G, M, I)\), a constant \(k \leq \max\{|G|, |M|\}\)

QUESTION Is there a concept \((A, B)\) with \(|A| + |B| \geq k\)?

Consider the context \(\overline{K} = (G, M, \overline{I})\) with \(\overline{I} = G \times M \setminus I\), and the corresponding bipartite graph \(\overline{\Gamma}_B\).

- A concept \((A, B)\) of \(\mathfrak{B}(K)\) corresponds to an inclusion-maximal independent set of vertices of \(\overline{\Gamma}_B\).
- The number of vertices in the largest independent set is \(|V(\overline{\Gamma}_B)| - |M|\), where \(|V(\overline{\Gamma}_B)|\) is the number of vertices in \(\overline{\Gamma}_B\) and \(|M|\) is the number of edges in the maximal matching of \(\overline{\Gamma}_B\) [König, 1930].
- The size of a maximal matching can is found by a polynomial-time algorithm [Karp, Hopcroft 1973].
Further problems in theoretical complexity

- Fast good estimates of the number of concepts, e.g., by methods of Mixing Markov Chains that made it possible to obtain good approximations for the problem of counting matchings, which is also $\#P$-complete.
- How large can be the minimum implication (e.g., that of Duquenne-Guigues) base expressed as a function of both $G$ and $M$ (it can be exponential if we allow for exponential relation between $G$ and $M$)
- A good object-incremental algorithm for computing Duquenne-Guigues implication base.
Difficulties of experimental comparison. 1

1. Algorithms, as described by their authors, often allow for different interpretation of crucial details, such as the test of uniqueness of a generated concept.

2. Authors seldom describe exactly data structures and their realizations.

3. The choice of a programming language and platform strongly affects the performance of an algorithm.

   A possible solution (remedy):
   
   ○ comparing not the time, but the number of specified operations (intersections, unions, closures, etc.) from a certain library. However, here one encounters the difficulty of weighting these operations in order to get the overall performance.

   ○ Much simpler would be comparing algorithms using a single platform.
4. Algorithms behave differently on different databases (contexts).
   
   *A possible solution:*
   - The community reaches a consensus with respect to databases to be used as testbeds.
   - Two types of possible testbeds:
     - “real” (open datasets recognized by the community).
       Here several standards should be specified, e.g., that of the sequence of rows and columns in data tables, since performance of algorithms can be very sensitive on this.
     - “randomly generated contexts”
       Randomization strategies should be specified.
Difficulties of experimental comparison. 3

Main parameters of randomly generated contexts:

- number of objects \(|G|\) (relative to \(|M|\))
- number of attributes \(|M|\) (relative to \(|G|\))
- size of the relation \(I\) relative to \(|G| \cdot |M|\) (we call it \textbf{density})
- average number of attributes per object intent
- average number of objects per attribute extent.

The community can agree on particular type(s) of random context generator(s) that can be tuned by the choice of above (or some other) parameters.
Outline

1. Generating all concepts
2. Generating hypotheses and classifications
4. Generating Duquenne-Guigues (stem) base of implications
Generating hypotheses

- The number of hypotheses can be exponential w.r.t. the sizes of the underlying contexts.

- The problems of determining the number of all (minimal) hypotheses is \( \#P \)-complete.

- Given a context \( K_\pm \) the set \( H_+(K_\pm) \) of all positive hypotheses and the set of all minimal hypotheses can be generated in time \( O(|H_+(K_\pm)||G|^2|M|) \).

**Open Problem.** Can all minimal hypotheses be generated with a polynomial-delay time algorithm?
Complexity of decision problems about hypotheses

<table>
<thead>
<tr>
<th></th>
<th>$\leq$</th>
<th>$=$</th>
<th>$\geq$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>h_+</td>
<td>$</td>
<td>NP</td>
</tr>
<tr>
<td>$</td>
<td>h_+^+</td>
<td>$</td>
<td>P</td>
</tr>
<tr>
<td>$</td>
<td>h_+^+</td>
<td>+</td>
<td>h_+</td>
</tr>
</tbody>
</table>

Here P denotes that there exists a polynomial algorithm for a problem and NP denotes NP-completeness of the problem.

All decision problems are solved in polynomial time if the size of an object intent is bounded by a constant:

$\exists k \forall g \in G_+ \quad |\{g\}^+| \leq k$ and

$\exists k \forall g \in G_- \quad |\{g\}^-| \leq k$. 
Complexity of lazy classification

Challenge Given positive, negative, and undetermined examples, is it possible to classify undetermined ones (keeping strictly to definitions) without generating all (minimal) hypotheses?

Motivating partial case If there is no negative (or no positive) examples, this can be done very quickly.

Unfortunately, the problem

INSTANCE Contexts $K_+ = (G_+, M, I_+), K_- = (G_-, M, I_-), K_\tau = (G_\tau, M, I_\tau)$, an undetermined example $g_\tau \in G_\tau$.

QUESTION Is there a hypothesis for positive classification of $g_\tau$ is NP-complete.

However, if the size of $g_\tau$ is bounded from above by a constant, i.e., $|g'_\tau| \leq const$, then there is a polynomial-time solution for the question (e.g., one can run CbO from attribute concepts, slightly changing the branching condition).
Positive lattice

$$\begin{align*}
\{1,2,3,4\}, \{\bar{f}\}\end{align*}$$

minimal (+)-hypotheses

$$\begin{align*}
\{1,2\}, \{\bar{y}, \bar{f}, r\}\end{align*}$$

$$\begin{align*}
\{2,3\}, \{\bar{f}, \bar{s}\}\end{align*}$$

$$\begin{align*}
\{1,4\}, \{\bar{r}, \bar{s}\}\end{align*}$$

$$\begin{align*}
\{3,4\}, \{\bar{f}, \bar{r}\}\end{align*}$$

$$\begin{align*}
\{4\}, \{4\}^{+}\end{align*}$$

$$\begin{align*}
\{3\}, \{3\}^{+}\end{align*}$$

$$\begin{align*}
\{2\}, \{2\}^{+}\end{align*}$$

$$\begin{align*}
\{1\}, \{1\}^{+}\end{align*}$$

$$\{7\}^{-} = \{w, \bar{f}, \bar{s}, r\}$$

$$\text{falsified (+)-generalizations}$$

<table>
<thead>
<tr>
<th>G \ M</th>
<th>w</th>
<th>y</th>
<th>g</th>
<th>b</th>
<th>f</th>
<th>\bar{f}</th>
<th>s</th>
<th>\bar{s}</th>
<th>r</th>
<th>\bar{r}</th>
<th>fruit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>apple</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>+</td>
</tr>
<tr>
<td>2</td>
<td>grapefruit</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>+</td>
</tr>
<tr>
<td>3</td>
<td>kiwi</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td>+</td>
</tr>
<tr>
<td>4</td>
<td>plum</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td>+</td>
</tr>
<tr>
<td>5</td>
<td>toy cube</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>egg</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>tennis ball</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>mango</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td>\tau</td>
</tr>
</tbody>
</table>

DSS'06 [28]
Classifying undetermined example **mango**

<table>
<thead>
<tr>
<th></th>
<th>G\M</th>
<th>w</th>
<th>y</th>
<th>g</th>
<th>b</th>
<th>f</th>
<th>(\overline{f})</th>
<th>s</th>
<th>(\overline{s})</th>
<th>r</th>
<th>(\overline{r})</th>
<th>fruit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>apple</td>
<td>×</td>
<td></td>
<td>×</td>
<td>×</td>
<td>×</td>
<td></td>
<td>×</td>
<td></td>
<td></td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>grapefruit</td>
<td>×</td>
<td></td>
<td></td>
<td>×</td>
<td>×</td>
<td></td>
<td>×</td>
<td></td>
<td></td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>kiwi</td>
<td>×</td>
<td></td>
<td></td>
<td>×</td>
<td>×</td>
<td></td>
<td>×</td>
<td></td>
<td></td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>plum</td>
<td>×</td>
<td></td>
<td></td>
<td>×</td>
<td>×</td>
<td></td>
<td>×</td>
<td></td>
<td></td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>toy cube</td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>×</td>
<td></td>
<td></td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>egg</td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>×</td>
<td></td>
<td></td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>tennis ball</td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>×</td>
<td></td>
<td></td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>mango</td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
<td>×</td>
<td></td>
<td>×</td>
<td>τ</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The object **mango** is classified positively:

- \(\{\overline{r}, \overline{f}\}\) is a \((+)-\)hypothesis,
  \(\{\overline{r}, \overline{f}\} \subseteq\text{mango}^\tau = \{g, \overline{f}, s, \overline{r}\}\);
- for \((-)-\)hypotheses \(\{w\}\) and \(\{f, s, \overline{r}\}\):
  \(\{w\} \not\subseteq\text{mango}^\tau\),
  \(\{\overline{f}, s, \overline{r}\} \not\subseteq\text{mango}^\tau\).
Outline

1. Generating all concepts
2. Generating hypotheses and classifications
3. Generating Duquenne-Guigues (stem) base of implications
Duquenne-Guigues implication base

Implications satisfy **Armstrong rules:**

\[
\begin{align*}
A \rightarrow A \quad & \quad A \rightarrow B \quad & \quad A \rightarrow B, B \cup C \rightarrow D \\
A \cup C \rightarrow B \quad & \quad A \cup C \rightarrow D
\end{align*}
\]

**Duquenne-Guigues base:** A cardinality-minimal subset of implications, from which all other implications can be deduced by means of Armstrong rules; characterized in terms of **noeuds minimaux de non-redondance (NMNR)** in [J.-L. Guigues, V. Duquenne 1984].

NMNR can be equivalently described as pseudo-intents [Ganter 1984]: a set \( P \subseteq M \) is a **pseudo-intent** if \( P \neq P'' \) and \( Q'' \subseteq P \) for every pseudo-intent \( Q \subseteq P \).
Computing Implications

- The decision problem about the existence of a nontrivial implication (whose premise does not contain the conclusion) is solved in polynomial time.
- Existing algorithms for generating pseudointents, e.g., [Ganter 1984] can terminate in exponential time without outputting any pseudointent.

**Problem:** What is the largest size of the stem base and its computational complexity?
Counting pseudo-intents

**Obvious:** The number of pseudo-intents can be exponential in $|M|$, the number of attributes.

**Example:** A context $K = (G, M, I)$, where object intents are exactly all possible subsets of size $|M|/2$. However, in this case $|G|$, as well as $|I|$, are also exponential in $|M|$, and the number of pseudo-intents is polynomial in $|I|$.

**Hint:** Observe the following fact about functional dependencies, namely that the size of a smallest base (cover) of functional dependencies can be exponential in the size of the relation.
Exponential-large cover of functional dependencies

Here a cover of the set of all dependencies is of size $2^m$ [Mannila, Räihä 1992]

<table>
<thead>
<tr>
<th>$A_1$</th>
<th>$A_2$</th>
<th>$A_3$</th>
<th>$A_4$</th>
<th>$A_5$</th>
<th>$A_6$</th>
<th>…</th>
<th>$A_{2m-1}$</th>
<th>$A_{2m}$</th>
<th>$A_{2m+1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>…</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>…</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>…</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>…</td>
<td>1</td>
<td>1</td>
<td>$m$</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>…</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>…</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>…</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>…</td>
<td>2</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>
D.-G. base of size $2^n$: the context $K_{\exp,n}$

<table>
<thead>
<tr>
<th>$G \setminus M$</th>
<th>$m_0$</th>
<th>$m_1 \ldots m_n$</th>
<th>$m_{n+1} \ldots m_{2n}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$g_1$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\vdots$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$g_n$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$g_{n+1}$</td>
<td>$\times$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\vdots$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\vdots$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\vdots$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$g_{3n}$</td>
<td>$\times$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The set \{\(m_1, \ldots, m_n\)\} is a pseudo-intent. Replacing $m_i$ with $m_{n+i}$ independently for each $i$, one obtains all $2^n$ pseudo-intents.
An example: $K_{\exp,3}$

<table>
<thead>
<tr>
<th>$G \setminus M$</th>
<th>$m_0$</th>
<th>$m_1$</th>
<th>$m_2$</th>
<th>$m_3$</th>
<th>$m_4$</th>
<th>$m_5$</th>
<th>$m_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$g_1$</td>
<td></td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>$g_2$</td>
<td>×</td>
<td></td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>$g_3$</td>
<td>×</td>
<td>×</td>
<td></td>
<td>×</td>
<td>×</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>$g_4$</td>
<td>×</td>
<td></td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>$g_5$</td>
<td>×</td>
<td>×</td>
<td></td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>$g_6$</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td></td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>$g_7$</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td></td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>$g_8$</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td></td>
<td>×</td>
</tr>
<tr>
<td>$g_9$</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td></td>
</tr>
</tbody>
</table>

Here, we have $2^3 = 8$ pseudo-intents: $\{m_1, m_2, m_3\}$, $\{m_1, m_2, m_6\}$, $\{m_1, m_5, m_3\}$, $\{m_1, m_5, m_6\}$, $\{m_4, m_2, m_3\}$, $\{m_4, m_2, m_6\}$, $\{m_4, m_5, m_3\}$, $\{m_4, m_5, m_6\}$. 
#P-hardness of counting pseudo-intents

**Proposition 2.** The following problem is #P-hard.
**INPUT** A formal context $K = (G, M, I)$
**OUTPUT** The number of pseudo-intents of $K$

**Proof:** by reduction from the problem of counting all (inclusion) minimal covers proved to be #P-complete in


For a graph $(V, E)$ a subset $W \subseteq V$ is a **vertex cover** if every edge $e \in E$ is incident to some $w \in W$. 
The reduction context

$\bar{I}$ is the complement of the edge-vertex graph incidence matrix

| $G \setminus M$ | $m_0$ | $m_1, \ldots, m_{|V|}$ |
|-----------------|-------|-------------------------|
| $g_1$           |       |                         |
| $\vdots$       |       |                         |
| $\vdots$       |       |                         |
| $g_{|E|}$       |       | $\bar{I}$               |
| $g_{|E|+1}$     | $\times$ |                     |
| $\vdots$       |       |                         |
| $\vdots$       |       |                         |
| $\vdots$       |       |                         |
| $g_{|E|+|V|}$   | $\times$ |                     |
Quasi-closed

Let $M$ be a set and let $'' : \mathcal{P}(M) \rightarrow \mathcal{P}(M)$ be a closure operator.

**Definition** [Ganter 1984] A set $Q \subseteq M$ is quasi-closed if for any $R \subseteq Q$ one has $R'' \subseteq Q$ or $R'' = Q''$.

For example, closed sets are quasi-closed. A quasi-closed set $Q \subseteq M$ of a context $\mathbb{K} = (G, M, I)$ is called quasi-intent.

**Proposition 3.** [Ganter 1984] A set $Q \subseteq M$ is a quasi-intent iff $Q \cap C$ is closed for every closed set $C$ with $Q \not\subseteq C$.

Equivalently, $Q \subseteq M$ is a quasi-intent iff for every $R \subseteq M$, the set $Q \cap R''$ is closed or $Q \cap R'' = Q$.

Quasi-intents and pseudo-intents

Actually, we can weaken the condition over all subsets of the set $Q$ to a condition over all quasi-closed subsets of $Q$.

**Proposition 5.** Let $K = (G, M, I)$ be a context and $Q \subseteq M$. Then the following two statements are equivalent:
1. $Q$ is quasi-intent;
2. For any quasi-intent $R \subseteq Q$ one has $R'' \subseteq Q$ or $R'' = Q''$.

**Proposition 6.** [Ganter 1984] A quasi-intent $P$ is pseudo-intent if $P'' \neq P$ and for any quasi-intent $Q \subset P$ one has $Q'' \subset P$.

By Proposition 6, a pseudo-intent is a minimal quasi-intent in its closure class, i.e., among quasi-intents with the same closure.
Testing quasi-closedness in polynomial time

From

**Proposition 3.** [Ganter 1984] A set \( Q \subseteq M \) is a quasi-intent iff \( Q \cap C \) is closed for every closed set \( C \) with \( Q \not\subseteq C \).

we have

**Proposition 7.** The set \( S \) is a quasi-intent iff for any object \( g \in G \) either \( S \cap g' \) is closed or \( S \cap g' = S \).

**Corollary.** Testing whether \( S \subseteq M \) is quasi-intent in the context \((G, M, I)\) may be performed in \( O(|G|^2 \cdot |M|) \) time.
Counting nonpseudo-intents is in \( \#P \)

**Proposition 8.** The problem
INSTANCE Given a context \( K = (G, M, I) \) and a set \( S \subseteq M \).
QUESTION Is \( S \) a nonpseudo-intent of \( K = (G, M, I) \)?
is in \( \text{NP} \).

**Corollary.** The problem
INSTANCE Given a context \( K = (G, M, I) \) and a set \( S \subseteq M \).
QUESTION Is \( S \) a pseudo-intent of \( K = (G, M, I) \)?
is in \( \text{coNP} \).

**Corollary.** The problem
INPUT Given a context \( K = (G, M, I) \)
OUTPUT Number of nonpseudo-intents
is \( \#P \)-complete.
Open problems

- Whether the problems
  INSTANCE A context \( K = (G, M, I) \), a subset \( Q \subseteq M \)
  QUESTION Is \( Q \) a pseudo-intent?
  INSTANCE A context \( K = (G, M, I) \), a closed subset \( Q \subseteq M \)
  QUESTION Is there a pseudo-intent \( P: P'' = Q \)?

are solvable in polynomial time? Are these problems in NP?

- Is it possible to compute all pseudo-intents with a polynomial-delay algorithm?
  Recall that for a polynomial-delay algorithm the elapsed time between two consecutive outputs is bounded polynomially.
  Example: Next Closure generating all concepts.
Related problems

Note that the decision problem

INSTANCE A context $K = (G, M, I)$, a natural number $k \leq |M|$.  
QUESTION Is there a pseudo-intent of the context $K$ of size not greater than $k$?  
is NP-hard by the same reduction from the vertex covering problem.

At the same time the problem

INSTANCE A context $K = (G, M, I)$  
QUESTION Is there a pseudo-intent of the context $K$?  
is solvable in polynomial time: Test whether the reduced context of $K$, after respective permutations of objects and attributes, coincides with $(A, A, \neq)$. 

DSS'06  [44]