# Inferring schemata from semi-structured data with Formal Concept Analysis

Bachelor thesis

Luca Liechti
Software Composition Group
Universität Bern

27.6.2017

# Roadmap



- Recap: Structured vs. semi-structured data
- Recap: Formal Concept Analysis
- Recap: The concept lattice
- Goals
- A motivational quote
- Algorithm
- Results
- Future work
- Literature

# Recap: Structured vs. semi-structured data

UNIVERSITÄT BERN

```
<library>
  <item>                (a book)
    <id>1</id>
    <title>The C Programming Language</title>
    <author>Brian W. Kernighan</author>
    <author>Dennis M. Ritchie</author>
    <year>1978</year>
  </item>
  <item>                (an article)
    <id>2</id>
    <title>Inferring NoSQL schema</title>
    <author>John Doe</author>
    <journal>VLDB</journal>
    <year>2016</year>
    <vol>1</vol>
  </item>
  <item>                (a thesis)
    <id>3</id>
    <title>Hacking Evil Corp</title>
    <author>Elliot Alderson</author>
    <date>09.05.2015</date>
    <institution>fsociety</institution>
  </item>
</library>
```

**lib**

| id | title | journal | year | vol | date | inst |
|---|---|---|---|---|---|---|
| 1 | The C Programming Language | NULL | 1978 | NULL | NULL | NULL |
| 2 | Inferring NoSQL schema | VLDB | 2016 | 1 | NULL | NULL |
| 3 | Hacking Evil Corp | NULL | NULL | NULL | 09.05.2015 | fsociety |

**auth**

| id | name |
|---|---|
| 1 | Brian W. Kernighan |
| 2 | Dennis M. Ritchie |
| 3 | John Doe |
| 4 | Elliot Anderson |

**ref**

| lib_id | auth_id |
|---|---|
| 1 | 1 |
| 1 | 2 |
| 2 | 3 |
| 3 | 4 |

# What we would like

UNIVERSITÄT BERN

**book**

| id | title | year |
|---|---|---|
| 1 | The C Programming Language | 1978 |
| 10 | Harry Potter | 1997 |
| 11 | Random book | 2000 |

**article**

| id | title | author | journal | year | vol |
|---|---|---|---|---|---|
| 2 | Inferring NoSQL schema | John Doe | Inferring NoSQL schema | 2016 | 1 |
| 20 | Are You Living In a Computer Simulation? | Nick Bostrom | Philosophical Quarterly | 2003 | 53 |
| 21 | Random article | Random woman | Random journal | 2010 | 20 |

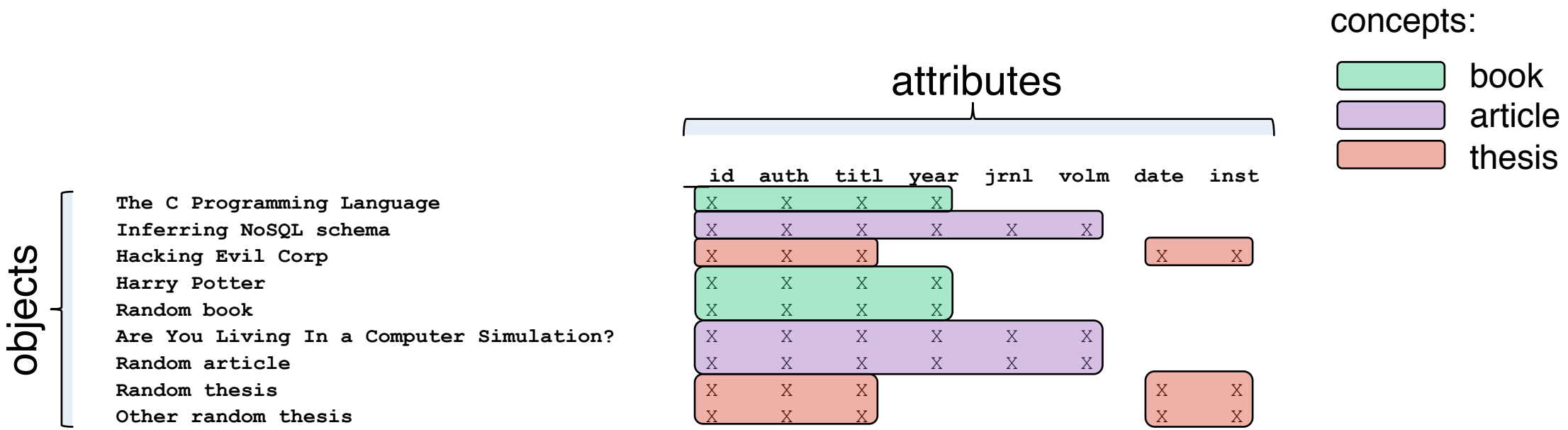
**thesis**

| id | title | author | date | institution |
|---|---|---|---|---|
| 3 | Hacking Evil Corp | Elliot Alderson | 09.05.2015 | fsociety |
| 30 | Random thesis | Random student | 01.01.2010 | Oxford University |
| 31 | Other random thesis | Random man | 02.02.2012 | Bern University |

no NULLs!

BUT: we do not know what an `item` is (`book`, `article`, `thesis`, or something else)!

# Recap: Formal Concept Analysis




- Context := (G,M,I) where G = objects, M = attributes, I = binary relation between G, M
- Concept := (A,B), A⊆G, B⊆M, all As have all attributes in B; these are found in all As

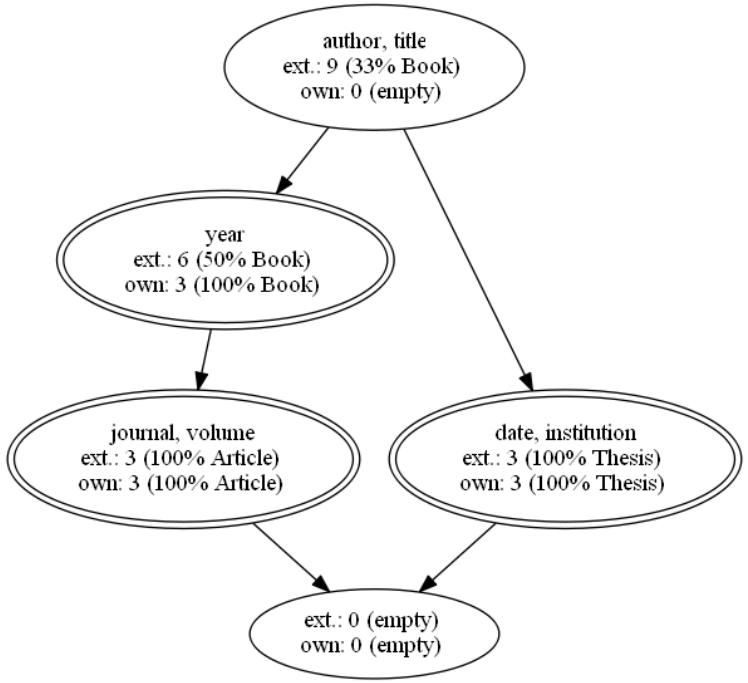
cf. Ganter, Wille: Formal Concept Analysis, p. 18f.

# Recap: The concept lattice

Attribute name in node: This attribute appears only in this node and all its children

Percentages in brackets denote the type majority

Ext denotes all descendants
Own denotes own objects




Two peripheries: Has own objects (objects whose attribute composition is equal to the node's intent)

(This means that there are no objects with only an author and a title, and no objects with all mentioned attributes)

Own visualisation

# Goals



- We want to tranform a semi-structured dataset to a set of relational database tables. We want to optimize this process with regard to two aspects.
  - We want our tables to "wrap around" the data "tightly" (more formal definition given later)
  - Objects of the same type should end up in the same table

# Motivational quote

UNIVERSITÄT BERN

"Rechnen Sie damit, frustriert zu werden" ("Expect to be frustrated")

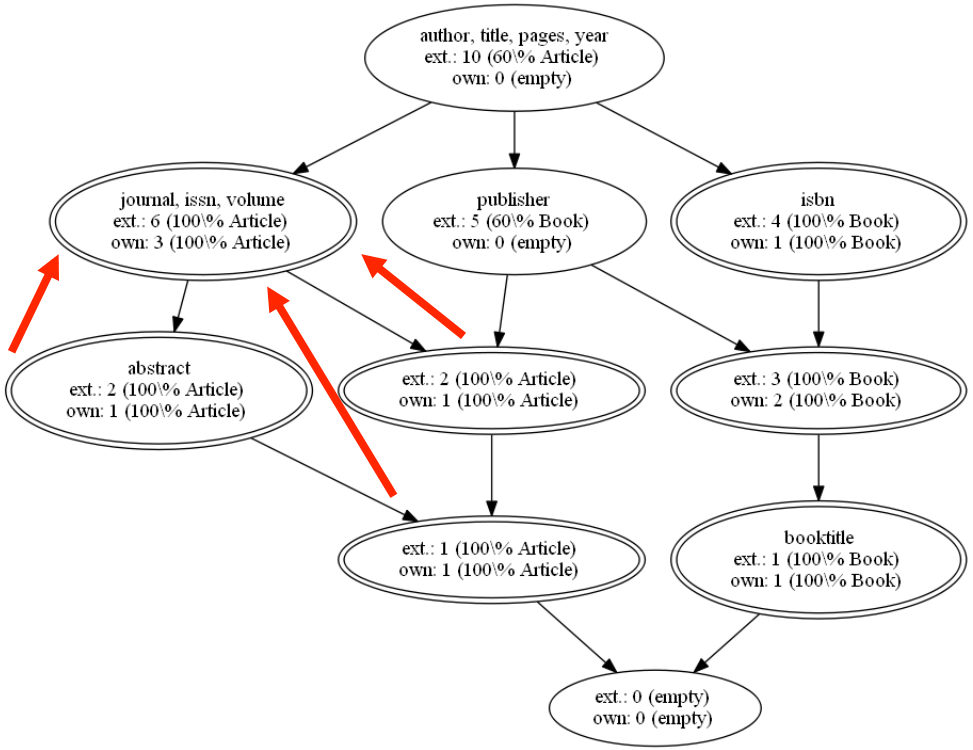
–Bernhard Ganter when we told him what we were trying to do
Bern, 10.2.2017

# Algorithm



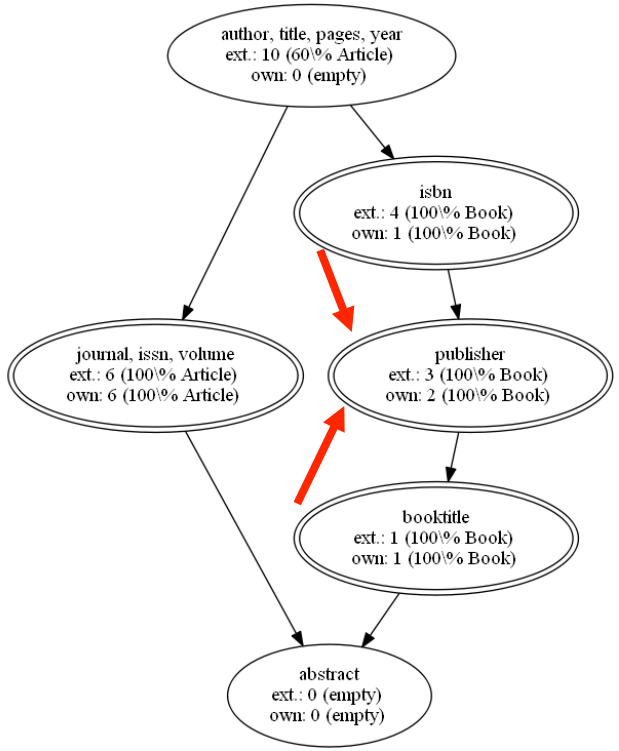
- Iteratively execute the following:
- Calculate the highest **merge score** between any two adjacent lattice nodes that both have own objects
- The **merge score** is defined as ([#objects in bigger node] / [#objects in smaller node]^2)
  - Merge node with 1 object into node with 2 objects: score = 2
  - Merge node with 10 objects into node with 20 objects: score = 0.2
  - Like this, outliers are merged first; nodes close to the archetype last
- Set the intent of the objects in the node with fewer objects to the intent of the objects in the node with more objects (merge the nodes)
- Recompute the lattice. Stop either when there are no such adjacent lattice node pairs, or the highest merge score is under a defined threshold

# Example (continued)



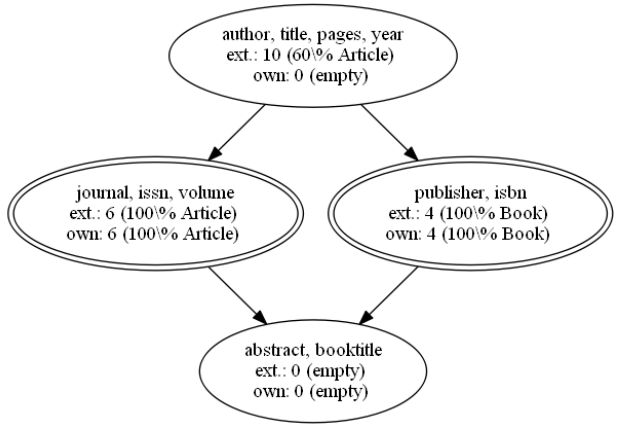

Example context, own visualisation

# Example (continued)




Example context, own visualisation

# Example (continued)




Example context, own visualisation

# Visualising the transformation: We go from this…

| | Auth. | Title | Pages | Year | Journ. | ISSN | Vol. | Publ. | ISBN | Abstr. | Bookt. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Book1 | X | X | X | X | | | | | X | | |
| Book2 | X | X | X | X | | | | X | X | | |
| Book 3 | X | X | X | X | | | | X | X | | |
| Book 4 | X | X | X | X | | | | X | X | | X |
| Article 1 | X | X | X | X | X | X | X | | | | |
| Article 2 | X | X | X | X | X | X | X | | | | |
| Article 3 | X | X | X | X | X | X | X | | | | |
| Article 4 | X | X | X | X | X | X | X | X | | | |
| Article 5 | X | X | X | X | X | X | X | | | X | |
| Article 6 | X | X | X | X | X | X | X | X | | X | |

NB: the crosses are actually values

# … to this



| | Auth. | Title | Pages | Year | Publisher | ISBN | Legacy |
|---|---|---|---|---|---|---|---|
| **Book 1** | name 1 | title 1 | 1230 | 2001 | NULL | 12340 | |
| **Book 2** | name 2 | title 2 | 2340 | 2002 | Publisher 2 | 23450 | |
| **Book 3** | name 3 | title 3 | 3450 | 2003 | Publisher 3 | 34560 | |
| **Book 4** | name 4 | title 4 | 1200 | 2004 | Publisher 4 | 45670 | Booktitle: Booktitle 4 |

5 of 70 values are legacy values

1 of 66 (6x4 + 7x6) cells is NULL

| | Auth. | Title | Pages | Year | Journ. | ISSN | Vol. | Legacy |
|---|---|---|---|---|---|---|---|---|
| **Article 1** | name 11 | title 11 | 123 | 2011 | journal 1 | 1234 | 1 | |
| **Article 2** | name 12 | title 12 | 234 | 2012 | journal 2 | 2345 | 2 | |
| **Article 3** | name 13 | title 13 | 345 | 2013 | journal 3 | 3456 | 3 | |
| **Article 4** | name 14 | title 14 | 12 | 2014 | journal 4 | 4567 | 4 | Publisher: Publisher 14 |
| **Article 5** | name 15 | title 15 | 23 | 2015 | journal 5 | 5678 | 5 | Abstract: Abstract 15 |
| **Article 6** | name 16 | title 16 | 34 | 2016 | journal 6 | 6789 | 6 | Publisher: Publisher 16<br>Abstract: Abstract 16 |

# Measuring success



- 5 of 70 values are legacy values: **Legacy Score** = 5/70 = 7.143%
- 1 of 66 post-transformation cells is NULL: **Null Score** = 1/66 = 1.515%
- "Wrapping" tables "tightly" around the data = have low legacy, null scores

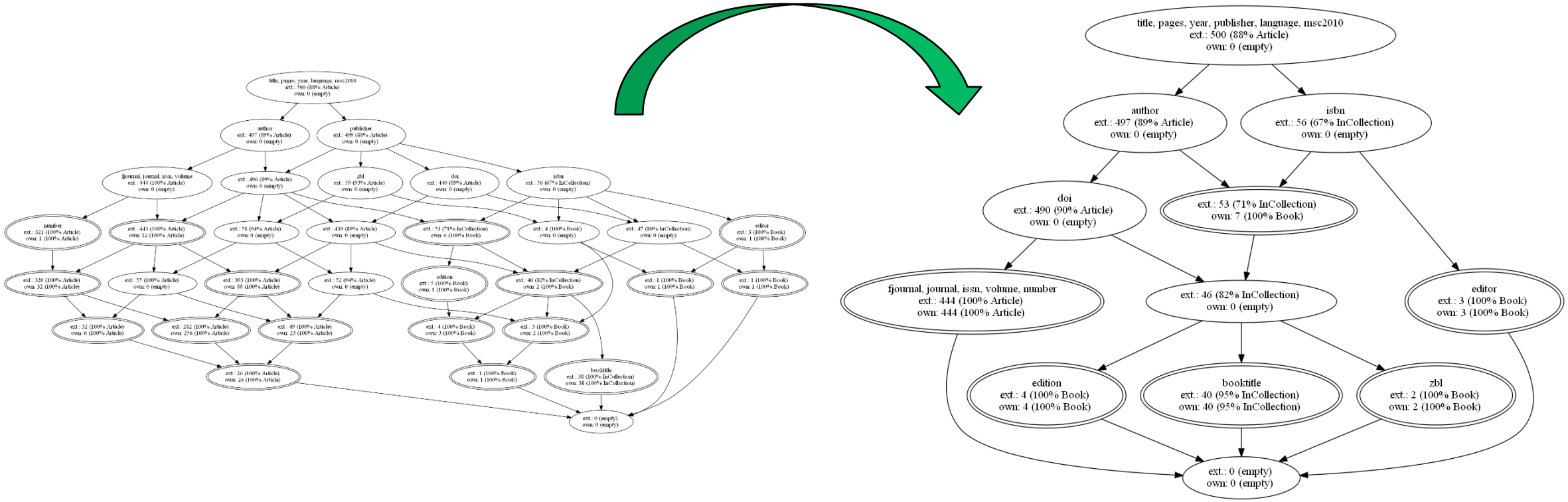
- **Major Score**: Percentage of objects belonging to nodes with a majority of objects of their type
- **Clean Score**: Percentage of objects belonging to nodes with only one object type
- In this (constructed) example, major = clean = 100%

# Results



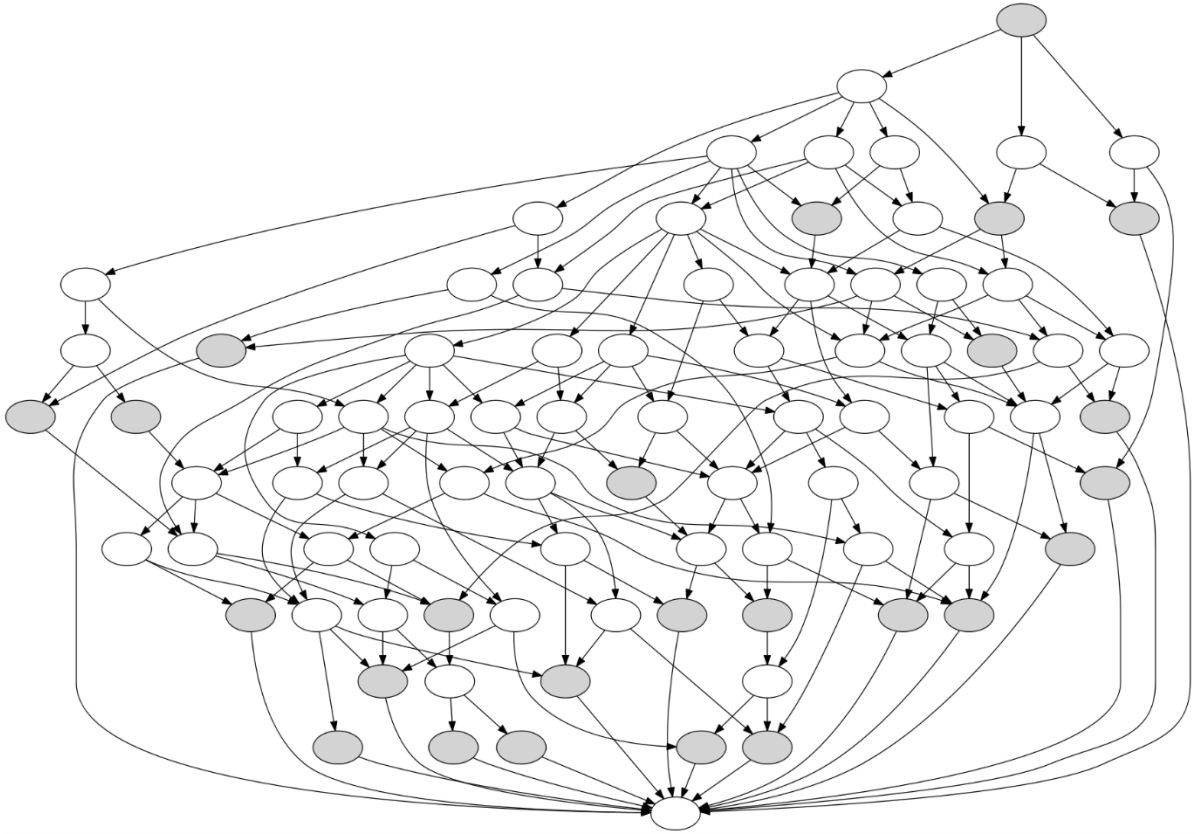
- We tested a few variations of the algorithm, e.g. forbidding certain merges
  - None of them works best on all datasets
  - There might not be a heuristics-based optimal approach to this problem
- For a few datasets, our algorithm produces the optimal result
  - clean = major = 100%, small legacy and null values
- Generally good results on datasets that were **already quite regular**
- However, results for large and diverse datasets are indeed frustrating
- In practice, a **domain expert** would be consulted
  - One of the cornerstones of Formal Concept Analysis
  - This means that we can deal with a certain type of dirty data
  - Namely outliers. These can easily be spotted and presented to the expert

# Results: A good example




12 merge steps; major = 99.6%, clean = 92%, null = 2.804%, legacy = 0.953%
A domain expert will be able to make this result almost optimal

# Results: A bad example



41 merge steps; major = 82%, clean = 35%, null = 2.5%, legacy = 16.2%
Also, there are 25 nodes left, but we only have 9 different types in the data!

# Future Work



- Our method is promising, yet far from mature
- It has two major weaknesses:
  - **Locality:** For every step of the merge process, only adjacent nodes are considered. However, there may be important patterns in the data that cannot be mapped to adjacent concept lattice nodes.
  - **Rigidity:** The algorithm is highly sensitive to small changes in the data, e.g. a single object with a certain intent can make nodes adjacent that otherweise wouldn't be
- We could mine **implications** or use **Fuzzy Formal Concept Analysis**
  - Implications are all order relations in the lattice, not just of neighbouring nodes
  - In Fuzzy FCA, the relation between objects and attributes is not binary

# Future Work



- There are many parameters that can be tweaked:
  - Computation of concept (dis)similarity
    - Similar to graph edit distance
    - much more nuances possible than used here
  - Merge score threshold
- Special break conditions
  - e.g. no merging if the difference in attributes is more than 2
- Instead of simple heuristics, use machine learning
  - Have an algorithm learn different parameters from a training set
  - Learn which parameters are important for which kind of dataset

# Questions?

# Literature


- Ganter, Bernhard and Rudolf Wille: Formal Concept Analysis. Mathematical Foundations. Berlin and Heidelberg 1999 [1996]
- Lindig, Christian: Mining Patterns and Violations Using Concept Analysis. In: Bird, Thomas, Tim Menzies, and Thomas Zimmermann: The Art and Science of Analyzing Software Data. Waltham MA 2015, pp. 17-38