Conditional Inclusion Dependencies

Jana Bauckmann

Guest lecture in Data Profiling and Data Cleansing
Prof. Dr. Felix Naumann
Outline

- Defining Conditional Inclusion Dependencies (CINDs)
- Fields of Application
- Reasoning on CINDs
  - Consistency
  - Implication
- Discovering CINDs
  - Quality Measures for Conditions
  - Discovering „Good“ Conditions
  - Discovering an Entire Pattern Tableau
Defining Conditional Inclusion Dependencies (CINDs)

- Recall INDs
- What could be Conditional INDs?
- How could we use Conditional INDs?
Running Example

- DBpedia 3.6
  - 296,454 persons in the English DBpedia
  - 175,457 persons in the German DBpedia
  - 74,496 persons in both data sets

- data sets mapped into relations using one attribute per predicate:
  - personID, name, givenname, surname, birthdate, birthplace, deathdate, deathplace, and description
  - extracted the century and the year of birth and death into additional attributes
### DBpedia Persons: German

<table>
<thead>
<tr>
<th>pid</th>
<th>birth cent.</th>
<th>birthplace</th>
<th>deathplace</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cecil Kellaway</td>
<td>18</td>
<td>Kapstadt</td>
<td>Vereinigte Staaten</td>
<td>Schauspieler</td>
</tr>
<tr>
<td>Mel Sheppard</td>
<td>18</td>
<td>Almonesson Lake</td>
<td>Vereinigte Staaten</td>
<td>Leichtathlet</td>
</tr>
<tr>
<td>Sam Sheppard</td>
<td>19</td>
<td></td>
<td>Vereinigte Staaten</td>
<td>Mediziner</td>
</tr>
<tr>
<td>Isobel Elsom</td>
<td>18</td>
<td>Cambridge</td>
<td>Kalifornien</td>
<td>Schauspielerin</td>
</tr>
</tbody>
</table>

This table shows persons not included in English DBpedia.

### DBpedia persons: English

<table>
<thead>
<tr>
<th>pid</th>
<th>birth cent.</th>
<th>birthplace</th>
<th>deathplace</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cecil Kellaway</td>
<td>18</td>
<td></td>
<td>United States</td>
<td>Actor</td>
</tr>
<tr>
<td>Mel Sheppard</td>
<td>18</td>
<td></td>
<td>United States</td>
<td>Athlete</td>
</tr>
<tr>
<td>Buddy Roosevelt</td>
<td>18</td>
<td>Meeker, Colorado</td>
<td>Meeker, Colorado</td>
<td></td>
</tr>
</tbody>
</table>

Persons not included in German DBpedia.

Which conditions distinguish included from non-included persons?

Which conditions describe persons covered in English DBpedia?
Defining Conditional Inclusion Dependencies (CINDs)

- Approximate Inclusion Dependency $R_1[A] \subseteq R_2[B]$
  - an IND that allows a certain amount of violating values in the dependent attribute $A$
  - $\text{Persons}_{\text{DE}}[\text{pid}] \subseteq \text{Persons}_{\text{EN}}[\text{pid}]$
Defining Conditional Inclusion Dependencies (CINDs)

- Pattern tableau $T_P$
  - restricts tuples of $R_1$ over attributes $X_P$ and tuples of $R_2$ over attributes $Y_P$
  - Each pattern tuple $t_p \in T_P$ defines a condition.
  - Tuple $t_1 \in I_1$ matches $t_p \in T_P$ if $\forall A \in X_P: t_p[A] = ('-' \lor t_1[A])$.
  - Definition for tuple $t_2 \in I_2$ matching $t_p \in T_P$ follows analogously over attributes $Y_P$.

<table>
<thead>
<tr>
<th>$T_P:$</th>
<th>birth cent.</th>
<th>deathplace</th>
<th>birth cent.</th>
<th>deathplace</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>18</td>
<td>Vereinigte Staaten</td>
<td>18</td>
<td>United States</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>Kapstadt</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Defining Conditional Inclusion Dependencies (CINDs)

- conditional inclusion dependency (CIND)
  \( \varphi: (R_1[X;X_P] \subseteq R_2[Y;Y_P], T_P) \)
  - embedded approximate IND \( R_1[X] \subseteq R_2[Y] \)
  - pattern tableau \( T_P \) over attributes \( X_P \) and \( Y_P \) defining the restrictions
  - \( X \) and \( X_P \) are disjoint, \( Y \) and \( Y_P \) are disjoint.

- A CIND \( \varphi \) holds for a pair of instances \( I_1 \) of \( R_1 \) and \( I_2 \) of \( R_2 \) if
  - selecting condition on \( I_1 \): Let \( t_1 \in I_1 \) match any tuple \( t_p \in T_P \). Then \( t_1 \) must satisfy the embedded IND.
  - demanding condition on \( I_2 \): Let \( t_1 \in I_1 \) match tuple \( t_p \in T_P \). Further, let \( t_1 \) satisfy the embedded IND with referenced tuple \( t_2 \in I_2 \), i.e., \( t_1[X] = t_2[Y] \). Then \( t_2 \) also must match \( t_p \).
Outline

- Defining Conditional Inclusion Dependencies (CINDs)
- **Fields of Application**
- Reasoning on CINDs
  - Consistency
  - Implication
- Discovering CINDs
  - Quality Measures for Conditions
  - Discovering „Good“ Conditions
  - Discovering an Entire Pattern Tableau
Describe that only those tuples of S are included in book that are of S.type "book":
φ: (S[title, author; type] ⊆ book[title, author;], T_p)

Find better matchings between source and target relations.

Ensure to expect data in correct referenced relation.

<table>
<thead>
<tr>
<th>source relation S</th>
<th>target relations: book, CD</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td>author</td>
</tr>
<tr>
<td>Angela's Ashes</td>
<td>McCourt</td>
</tr>
<tr>
<td>Angela's Ashes</td>
<td>McCourt</td>
</tr>
<tr>
<td>Brave New World</td>
<td>Huxley</td>
</tr>
<tr>
<td>1984</td>
<td>Orwell</td>
</tr>
</tbody>
</table>
Data Quality

- \( \varphi: (S\{\text{title, author; type}\} \subseteq T\{\text{title, author; format}\}, T_P) \)
- Ensure data quality in target relation
  - Matching tuples in S must be included in T
  - Covered tuples in T must conform to given format

**S:**

<table>
<thead>
<tr>
<th>title</th>
<th>author</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angela's Ashes</td>
<td>McCourt</td>
<td>book</td>
</tr>
<tr>
<td>Angela's Ashes</td>
<td>McCourt</td>
<td>a-book</td>
</tr>
<tr>
<td>Brave New World</td>
<td>Huxley</td>
<td>book</td>
</tr>
<tr>
<td>1984</td>
<td>Orwell</td>
<td>book</td>
</tr>
</tbody>
</table>

**T:**

<table>
<thead>
<tr>
<th>title</th>
<th>author</th>
<th>format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brave New World</td>
<td>Huxley</td>
<td>paperback</td>
</tr>
<tr>
<td>1984</td>
<td>Orwell</td>
<td>hardcover</td>
</tr>
<tr>
<td>Angela’s Ashes</td>
<td>McCourt</td>
<td>audio</td>
</tr>
</tbody>
</table>

**\( T_P \):**

<table>
<thead>
<tr>
<th>type</th>
<th>format</th>
</tr>
</thead>
<tbody>
<tr>
<td>book</td>
<td></td>
</tr>
<tr>
<td>a-book</td>
<td></td>
</tr>
</tbody>
</table>

Conditional Inclusion Dependencies |
Link Discovery in Linked Open Data

- Idea: find missing sameAs links in linked open data, based on existing sameAs links for similar objects
  1. identify characteristics of persons with sameAs links
  - persons in German DBpedia that are also included in English DBpedia (or vice versa)
  - idea: only a certain set of persons in the German DBpedia are interesting to English readers
    - real example: deathplace = United States ^ birthcentury = 18
  2. characteristics also match a small amount of persons without sameAs link → good candidates for missing sameAs links
- Defining Conditional Inclusion Dependencies (CINDs)
- Fields of Application

**Reasoning on CINDs**
- Consistency
- Implication

- Discovering CINDs
  - Quality Measures for Conditions
  - Discovering „Good“ Conditions
  - Discovering an Entire Pattern Tableau
Given a set $\Sigma$ of CINDs over a relational schema $\mathcal{R}$. Exists a non-empty database instance $D$ of $\mathcal{R}$ such that $D$ satisfies $\Sigma$?

In other words:

- Are there conflicts or inconsistencies in $\Sigma$?

If set $\Sigma$ of CINDs is dirty in themselves, we could not use them for any applications, but: there always exists an instance $D$ that satisfies $\Sigma$.

(idea: Build a relational schema of all attributes in the pattern tableau, use their values plus at most one additional distinct value in a cross-product. The result is $D$.)

Complexity: $O(1)$
Reasoning on CINDs: Implication

- Given a set $\Sigma$ of CINDs and a single CIND $\phi$ over a relational schema $R$. Determine for all instances $D$ of $R$: If $D$ satisfies $\Sigma$, then $D$ satisfies $\phi$.

- Remove redundancies.

- Complexity: EXPTIME-complete, i.e., $O(2^{p(n)})$
Reasoning on CINDs: Inference rules for CINDs

- Reflexivity
- Projection-Permutation
- Transitivity – additional requirement on pattern tableaux
- Move Attributes from embedded IND to pattern tableau
- Add Attributes to $X_P$
- Remove Attributes from $Y_P$
- Only for finite domains: Merge CINDs
  - If CINDs only differ in values in attribute A in $X_P$ (or $Y_P$) and cover all values of domain of A, then remove A from $X_P$ or $Y_P$
Outline

■ Defining Conditional Inclusion Dependencies (CINDs)
■ Fields of Application
■ Reasoning on CINDs
  □ Consistency
  □ Implication
■ Discovering CINDs
  □ Quality Measures for Conditions
  □ Discovering „Good“ Conditions
  □ Discovering an Entire Pattern Tableau
■ Sources:
Discovering CINDs: Quality Measures for “Good” Conditions

- Valid condition matches *only* included tuples
- Covering or completeness condition matches *all* included tuples
### DBpedia Persons: German

<table>
<thead>
<tr>
<th>pid</th>
<th>birth cent.</th>
<th>birthplace</th>
<th>deathplace</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cecil Kellaway</td>
<td>18</td>
<td>Kapstadt</td>
<td>Los Angeles</td>
<td>Schauspieler</td>
</tr>
<tr>
<td>Cecil Kellaway</td>
<td>18</td>
<td>Kapstadt</td>
<td>Kalifornien</td>
<td>Schauspieler</td>
</tr>
<tr>
<td>Cecil Kellaway</td>
<td>18</td>
<td>Kapstadt</td>
<td>Vereinigte Staaten</td>
<td>Schauspieler</td>
</tr>
<tr>
<td>Cecil Kellaway</td>
<td>18</td>
<td>Südafrika</td>
<td>Los Angeles</td>
<td>Schauspieler</td>
</tr>
<tr>
<td>Cecil Kellaway</td>
<td>18</td>
<td>Südafrika</td>
<td>Kalifornien</td>
<td>Schauspieler</td>
</tr>
<tr>
<td>Cecil Kellaway</td>
<td>18</td>
<td>Südafrika</td>
<td>Vereinigte Staaten</td>
<td>Schauspieler</td>
</tr>
<tr>
<td>Mel Sheppard</td>
<td>18</td>
<td>Almonesson Lake</td>
<td>Vereinigte Staaten</td>
<td>Leichtathlet</td>
</tr>
<tr>
<td>Sam Sheppard</td>
<td>19</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Isobel Elsom</td>
<td>18</td>
<td>Cambridge</td>
<td>Los Angeles</td>
<td>Schauspielerin</td>
</tr>
<tr>
<td>Isobel Elsom</td>
<td>18</td>
<td>Cambridge</td>
<td>Kalifornien</td>
<td>Schauspielerin</td>
</tr>
</tbody>
</table>

### DBpedia persons: English

<table>
<thead>
<tr>
<th>pid</th>
<th>birth cent.</th>
<th>birthplace</th>
<th>deathplace</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cecil Kellaway</td>
<td>18</td>
<td>South Africa</td>
<td>United States</td>
<td>Actor</td>
</tr>
<tr>
<td>Mel Sheppard</td>
<td>18</td>
<td>United States</td>
<td>United States</td>
<td>Athlete</td>
</tr>
<tr>
<td>Buddy Roosevelt</td>
<td>18</td>
<td>Meeker, Colorado</td>
<td>Meeker, Colorado</td>
<td>Actor</td>
</tr>
<tr>
<td>pid</td>
<td>birth cent.</td>
<td>birthplace</td>
<td>deathplace</td>
<td>description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
<td>------------------</td>
<td>-----------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Cecil Kellaway</td>
<td>18</td>
<td>Kapstadt</td>
<td>Los Angeles</td>
<td>Schauspieler</td>
</tr>
<tr>
<td>Cecil Kellaway</td>
<td>18</td>
<td>Kapstadt</td>
<td>Kalifornien</td>
<td>Schauspieler</td>
</tr>
<tr>
<td>Cecil Kellaway</td>
<td>18</td>
<td>Kapstadt</td>
<td>Vereinigte Staaten</td>
<td>Schauspieler</td>
</tr>
<tr>
<td>Cecil Kellaway</td>
<td>18</td>
<td>Südafrika</td>
<td>Los Angeles</td>
<td>Schauspieler</td>
</tr>
<tr>
<td>Cecil Kellaway</td>
<td>18</td>
<td>Südafrika</td>
<td>Kalifornien</td>
<td>Schauspieler</td>
</tr>
<tr>
<td>Cecil Kellaway</td>
<td>18</td>
<td>Südafrika</td>
<td>Vereinigte Staaten</td>
<td>Schauspieler</td>
</tr>
<tr>
<td>Mel Sheppard</td>
<td>18</td>
<td>Almonesson Lake</td>
<td>Vereinigte Staaten</td>
<td>Leichtathlet</td>
</tr>
<tr>
<td>Sam Sheppard</td>
<td>19</td>
<td>-</td>
<td>-</td>
<td>Mediziner</td>
</tr>
<tr>
<td>Isobel Elsom</td>
<td>18</td>
<td>Cambridge</td>
<td>Los Angeles</td>
<td>Schauspielerin</td>
</tr>
<tr>
<td>Isobel Elsom</td>
<td>18</td>
<td>Cambridge</td>
<td>Kalifornien</td>
<td>Schauspielerin</td>
</tr>
</tbody>
</table>

- Cope with embedded INDs if dependent attributes are no key
  - Mapping RDF data to relational data
  - Joining Relations to provide more potential condition attributes

- Tuples whose projection on the inclusion attributes is equal are called a *group*.
<table>
<thead>
<tr>
<th>pid</th>
<th>birth cent.</th>
<th>birthplace</th>
<th>deathplace</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cecil Kellaway</td>
<td>18</td>
<td>Kapstadt</td>
<td>Los Angeles</td>
<td>Schauspieler</td>
</tr>
<tr>
<td>Cecil Kellaway</td>
<td>18</td>
<td>Kapstadt</td>
<td>Kalifornien</td>
<td>Schauspieler</td>
</tr>
<tr>
<td>Cecil Kellaway</td>
<td>18</td>
<td>Kapstadt</td>
<td>Vereinigte Staaten</td>
<td>Schauspieler</td>
</tr>
<tr>
<td>Cecil Kellaway</td>
<td>18</td>
<td>Südafrika</td>
<td>Los Angeles</td>
<td>Schauspieler</td>
</tr>
<tr>
<td>Cecil Kellaway</td>
<td>18</td>
<td>Südafrika</td>
<td>Kalifornien</td>
<td>Schauspieler</td>
</tr>
<tr>
<td>Cecil Kellaway</td>
<td>18</td>
<td>Südafrika</td>
<td>Vereinigte Staaten</td>
<td>Schauspieler</td>
</tr>
<tr>
<td>Mel Sheppard</td>
<td>18</td>
<td>Almonesson Lake</td>
<td>Vereinigte Staaten</td>
<td>Leichtathlet</td>
</tr>
<tr>
<td>Sam Sheppard</td>
<td>19</td>
<td>-</td>
<td>-</td>
<td>Mediziner</td>
</tr>
<tr>
<td>Isobel Elsom</td>
<td>18</td>
<td>Cambridge</td>
<td>Los Angeles</td>
<td>Schauspielerin</td>
</tr>
<tr>
<td>Isobel Elsom</td>
<td>18</td>
<td>Cambridge</td>
<td>Kalifornien</td>
<td>Schauspielerin</td>
</tr>
</tbody>
</table>

### Valid Condition

\[
\text{valid} (t_P) = \frac{|\text{included matching tuples}|}{|\text{all matching tuples}|} = \frac{|I_{\varphi[t_P]}|}{|I_{1[t_P]}|} = 2 \quad \frac{|G_{\varphi[t_P]}|}{|G_{1[t_P]}|} = 1
\]

\[
\text{valid}_g (t_P) = \frac{|\text{included matching groups}|}{|\text{all matching groups}|} = \frac{|G_{\varphi[t_P]}|}{|G_{1[t_P]}|} = \frac{1}{2}
\]
<table>
<thead>
<tr>
<th>pid</th>
<th>birth cent.</th>
<th>birthplace</th>
<th>deathplace</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cecil Kellaway</td>
<td>18</td>
<td>Kapstadt</td>
<td>Los Angeles</td>
<td>Schauspieler</td>
</tr>
<tr>
<td>Cecil Kellaway</td>
<td>18</td>
<td>Kapstadt</td>
<td>Kalifornien</td>
<td>Schauspieler</td>
</tr>
<tr>
<td>Cecil Kellaway</td>
<td>18</td>
<td>Kapstadt</td>
<td>Vereinigte Staaten</td>
<td>Schauspieler</td>
</tr>
<tr>
<td>Cecil Kellaway</td>
<td>18</td>
<td>Südafrika</td>
<td>Los Angeles</td>
<td>Schauspieler</td>
</tr>
<tr>
<td>Cecil Kellaway</td>
<td>18</td>
<td>Südafrika</td>
<td>Kalifornien</td>
<td>Schauspieler</td>
</tr>
<tr>
<td>Cecil Kellaway</td>
<td>18</td>
<td>Südafrika</td>
<td>Vereinigte Staaten</td>
<td>Schauspieler</td>
</tr>
<tr>
<td>Mel Sheppard</td>
<td>18</td>
<td>Almonesson Lake</td>
<td>Vereinigte Staaten</td>
<td>Leichtathlet</td>
</tr>
<tr>
<td>Sam Sheppard</td>
<td>19</td>
<td>-</td>
<td>-</td>
<td>Mediziner</td>
</tr>
<tr>
<td>Isobel Elsom</td>
<td>18</td>
<td>Cambridge</td>
<td>Los Angeles</td>
<td>Schauspielerin</td>
</tr>
<tr>
<td>Isobel Elsom</td>
<td>18</td>
<td>Cambridge</td>
<td>Kalifornien</td>
<td>Schauspielerin</td>
</tr>
</tbody>
</table>

- **Completeness Condition**: \( \text{birthcentury} = 18 \) and \( \text{deathplace} = \text{Los Angeles} \)

  \[
  \text{complete} (t_p) = \frac{|\text{included matching tuples}|}{|\text{all included tuples}|} = \frac{|I_{\varphi[t_p]}|}{|I_{[t_p]}|} = \frac{2}{7}
  \]

- **Covering Condition**

  \[
  \text{covering} (t_p) = \frac{|\text{included matching groups}|}{|\text{all included groups}|} = \frac{|G_{\varphi[t_p]}|}{|G_{\Pi[t_p]}|} = \frac{1}{2}
  \]
Discovering CINDs: Challenges of CIND discovery

1. Which (and how many) attributes should be used for the conditions?

2. Which attribute values should be chosen for the conditions?

3. Which conditions should be chosen for the pattern tableau?

- Algorithms CINDERELLA / PLI answer 1. and 2.
- Algorithm for entire tableau discovery answers 2. and 3.
Discovering CINDs: Discovering Conditions Using CINDERELLA

- Idea: use association rule mining algorithms to discover conditions

- Association rule mining was introduced for market basket analysis: Find rules of type “Who buys X and Y often buys Z.”

- We apply this concept to identify conditions like: “Whose century of birth is 18 and place of death is Vereinigte Staaten often is INCLUDED (in the English DBpedia).

- Two challenges:
  - Map problem of condition discovery to association rule mining
  - Improve efficiency based on characteristics of condition discovery
The Apriori Algorithm

- Two steps
  - Find all frequent itemsets that occur in at least a given number of baskets, i.e., hold a given support. 
    \{X, Y, Z\} and all subsets as frequent itemsets
  - Use these frequent itemsets to derive association rules – that hold a given confidence.
    \{X, Y\} \rightarrow \{Z\}
    with confidence = support (XYZ) / support \{XY\}

- Search space is pruned using support and confidence
- In terms of condition discovery
  - covering / completeness of a condition is measure for support
  - validity of a condition is measure for confidence
We need to build baskets as input to the algorithm.

Each group of the dependent relation forms a basket as a set of:

- Each attribute’s value(s)
- an inclusion indicator (Is this group INCLUDED or NOT INCLUDED?) – use a left outer join over embedded IND to get this information

To distinguish values from different attributes: prefix each value with the attribute name (or a shortcut)

Example Cecil Kellaway:
{INCLUDED, A18, BKapstadt, BSüdafrika, CLos_Angeles, CKalifornien, CVereinigte_Staaten, DSchauspieler}
We need only a special case of association rules:
- Rules with right-hand side item \texttt{INCLUDED}
- Left-hand side of these rules builds selecting condition.

We only need to find frequent itemsets with item \texttt{INCLUDED}
- Largely reduce search space
- But: We need extra scan over data to derive association rules, i.e., to compute the validity of a condition.

- frequent itemset: \{\texttt{INCLUDED}, A18, CVereinigte_Staaten\}
- Validity = \text{support (\texttt{INCLUDED}, A18, CVereinigte_Staaten)} / \text{support (A18, CVereinigte_Staaten)}
Find frequent itemsets

**input**: Included tuples as baskets: baskets

**output**: frequent itemsets with item INCLUDED

/* single scan over baskets to get \( L_1 \)

1 \( L_1 = \{ \text{frequent 1-itemsets} \} \);

2 \( L_2 = \{(\text{INCLUDED}, l_1) \mid l_1 \in L_1 \} \);

3 \textbf{for } k=3; \ L_{k-1} \neq \emptyset; \ k++ \ \textbf{do}

4 \quad C_k = \text{aprioriGen-Constrained}(L_{k-1}) ;

5 \textbf{foreach basket } b \in \text{baskets} \ \textbf{do}

6 \quad C_t = \text{subset}(C_k, b) ;

7 \textbf{foreach } c \in C_t \ \textbf{do}

8 \quad \quad \text{c.count}++ ;

9 \quad L_k = \{ c \in C_k \mid \text{c.count} \geq \lambda * |\text{baskets}| \} ;

10 \textbf{return } (\bigcup k L_k) \bigcup L_2 ;
Create item sets of size $k$

**input**: frequent itemsets of size $k-1$: $I_{k-1}$

**output**: candidates for frequent itemsets of size $k$: $C_k$

1. insert into $C_k$
2. select $p[item_1], p[item_2], \ldots, p[item_{k-1}], q[item_{k-1}]$
3. from $L_{k-1} p$, $L_{k-1} q$
4. where $p[item_1] = q[item_1] \land \ldots \land p[item_{k-2}] = q[item_{k-2}] \land$
   $p[item_{k-1}] < q[item_{k-1}]$

6. foreach candidate $c \in C_k$ do
   7.  
   8.     foreach $(k-1)$-subsets $s$ of $c$ containing item Included do
   9.         if $s \notin L_{k-1}$ then
   10.            delete $c$ from $C_k$

10. return $C_k$
Discovering CINDs:
Discovering Conditions Using PLI

- **CINDERELLA** algorithm traverses powerset lattice of condition combinations breadth-first
- PLI traverses this powerset lattice depth-first (recursively)

Idea is twofold:

- Use a special position list for included groups (called *includedPositions*)
- Cross-intersect position lists of attributes to test value combinations (i.e., conditions) for the intersected attributes (Intersect all position lists of attribute A with all position lists of attribute B.)
<table>
<thead>
<tr>
<th>pid</th>
<th>birth cent.</th>
<th>birthplace</th>
<th>deathplace</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cecil Kellaway</td>
<td>18</td>
<td>Kapstadt</td>
<td>Los Angeles</td>
<td>Schauspieler</td>
</tr>
<tr>
<td>Cecil Kellaway</td>
<td>18</td>
<td>Kapstadt</td>
<td>Kalifornien</td>
<td>Schauspieler</td>
</tr>
<tr>
<td>Cecil Kellaway</td>
<td>18</td>
<td>Kapstadt</td>
<td>Vereinigte Staaten</td>
<td>Schauspieler</td>
</tr>
<tr>
<td>Cecil Kellaway</td>
<td>18</td>
<td>Südafrika</td>
<td>Los Angeles</td>
<td>Schauspieler</td>
</tr>
<tr>
<td>Cecil Kellaway</td>
<td>18</td>
<td>Südafrika</td>
<td>Kalifornien</td>
<td>Schauspieler</td>
</tr>
<tr>
<td>Cecil Kellaway</td>
<td>18</td>
<td>Südafrika</td>
<td>Vereinigte Staaten</td>
<td>Schauspieler</td>
</tr>
<tr>
<td>Mel Sheppard</td>
<td>18</td>
<td>Almonesson Lake</td>
<td>Vereinigte Staaten</td>
<td>Leichtathlet</td>
</tr>
<tr>
<td>Sam Sheppard</td>
<td>19</td>
<td>-</td>
<td>-</td>
<td>Mediziner</td>
</tr>
<tr>
<td>Isobel Elsom</td>
<td>18</td>
<td>Cambridge</td>
<td>Los Angeles</td>
<td>Schauspielerin</td>
</tr>
<tr>
<td>Isobel Elsom</td>
<td>18</td>
<td>Cambridge</td>
<td>Kalifornien</td>
<td>Schauspielerin</td>
</tr>
</tbody>
</table>

- \( \text{includedPositions} = \{1, 2\} \)
- \( \text{Deathplace.Los\_Angeles} = \{1,4\}; \ \text{deathplace.VereinigteStaaten} = \{1,2\}; \ldots \)
- \( \text{Birthcent.18} = \{1,2,4\}; \ldots \)

For cross-intersection deathplace with birthcent: Intersect all position lists of attribute deathplace with all position lists of attribute birthcent
Covering of a condition, i.e., a position list $P$
  - $|P \cap \text{includedPositions}| \div |\text{includedPositions}|

Validity of a condition, i.e., a position list $P$
  - $|P \cap \text{includedPositions}| \div |P|$

Prune position lists with covering less than given threshold.
Results on Condition Discovery

- Used DBpedia 3.6 person data set
- German DBpedia persons included in English DBpedia
- Validity threshold:
  - Twice the validity of an empty condition
  - here: 0.84
- Covering threshold
  - 0.008 (600 persons) leads to useful amount of conditions
Results on Condition Discovery

- description = american actor
- description = american actress
- birthcentury = 18 and description = American politician
- birthcentury = 19 and deathplace = California
- birthcentury = 19 and deathplace = Los Angeles
- birthcentury = 19 and deathplace = New York City

- But also birthyear = X for X in 1900 to 1926 and 1945 to 1947
Results CINDERELLA vs. PLI
Varying the Number of Conditions to be identified

![Graph showing runtime vs. number of conditions for Cinderella and PLI]
Results Cinderella vs. PLI
Varying the Number of Conditions to be identified

![Graph showing memory consumption vs. lambda-coversing for Cinderella and PLI](image)
Results CINDERELLA vs. PLI
Varying the Number of Attributes

Experiments on generated data: 300,000 tuples with 150,000 included tuples; each 300 tuples build a group; 5 conditions of size 3, 10 conditions of size 2, 20 conditions of size 1

![Graph showing runtime vs. number of attributes for CINDERELLA and PLI with different attribute distributions.](image-url)
Results CINDERELLA vs. PLI
Varying the Number of Attributes

![Graph showing memory consumption vs. number of attributes for CINDERELLA and PLI conditions.]

- CINDERELLA; condition: attributes distributed over all attributes
- CINDERELLA; condition: attributes distributed over first 10 attributes
- PLI; condition: attributes distributed over all attributes
- PLI; condition: attributes distributed over first 10 attributes
Discovering CINDs:
Discovering Entire Pattern Tableau

- Find a pattern tableau with given support and confidence
  - Support relates to previous validity
  - Confidence relates to previous completeness
- Additional requirement: parsimony, i.e., produce the smallest possible pattern tableaux
- Finding an optimal tableau is NP-complete
- Greedy algorithm
- Discovers only completeness conditions, i.e., no covering conditions
Discovering CINDs: Discovering Entire Pattern Tableau

- Starting from all-wildcards pattern
- Traverse different pattern in top-down manner...
- ...while inserting pattern into the pattern tableau that
  - meet the confidence threshold and
  - match the most tuples that have not already been matched
    -> “marginal local support”

- requirement: confidence of any pattern can be computed in a single scan over the data

- requirement: set of condition attributes to use must be given
Wikipedia use case

- table Image with attributes name, size, width, height, bits, media_type, major_mime, user, user_text, timestamp, sha1
- table ImageLinks with attributes il_from and il_to
- Embedded IND Image[name] ⊆ ImageLinks.il_to

- Pre-selected attributes bits, media_type, user_text
- Validity 0.85, completeness = 0.003
CINDERELLA/PLI also discover all identified conditions, but additionally...

CINDERELLA/PLI discover more detailed conditions
- e.g., media_type = audio and bits = 0 with exact same validity and completeness as media_type = audio
- Stricter conditions
  - give more insight into the dataset
  - prevent from wrongly generalizing identified conditions
CINDERELLA/PLI discover even more interesting conditions on other than pre-selected condition attributes

- Width = 200 and major_mime = image
- Width = 300 and major_mime = image
- Both with completeness 0.04, instead of all previously discovered conditions with completeness between 0.003 and 0.008
- Height = 200, Height = 300 (both with completeness 0.02)
- Height = 240, width = 240 (both with completeness 0.01)

Removing restriction to pre-select condition attributes leads to ability to build better pattern tableaux

- (my) conclusion: First find good conditions, then build pattern tableau with Greedy algorithm
Summary

- Defining Conditional Inclusion Dependencies (CINDs)
- Fields of Application
- Reasoning on CINDs
  - Consistency
  - Implication
- Discovering CINDs
  - Quality Measures for Conditions
  - Discovering „Good“ Conditions
  - Discovering an Entire Pattern Tableau