Summary of:
The LLUNATIC Data-Cleaning Framework

Vincent J. Maccio

Department of Computing and Software
McMaster University
Hamilton, Ontario, Canada
The Problem

- The known issues of data cleaning
- What does one mean by clean? Repaired?
- If we know something is dirty, how do we clean it?
## Example - Data

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name</th>
<th>Phone</th>
<th>Conf</th>
<th>Str</th>
<th>City</th>
<th>CC#</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>111</td>
<td>M. White</td>
<td>408–3334</td>
<td>0.8</td>
<td>Red Ave.</td>
<td>NY</td>
</tr>
<tr>
<td>$t_2$</td>
<td>222</td>
<td>L. Lennon</td>
<td>122–1876</td>
<td>0.9</td>
<td>NULL</td>
<td>SF</td>
</tr>
<tr>
<td>$t_3$</td>
<td>222</td>
<td>L. Lennon</td>
<td>000–0000</td>
<td>0.0</td>
<td>Fry Dr.</td>
<td>SF</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SSN</th>
<th>Salary</th>
<th>Insur.</th>
<th>Treat</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_4$</td>
<td>111</td>
<td>10K</td>
<td>Abx</td>
<td>Dental</td>
</tr>
<tr>
<td>$t_5$</td>
<td>111</td>
<td>25K</td>
<td>Abx</td>
<td>Cholest.</td>
</tr>
<tr>
<td>$t_6$</td>
<td>222</td>
<td>30K</td>
<td>Med</td>
<td>Eye Surg.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name</th>
<th>Phone</th>
<th>Str</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_m$</td>
<td>222</td>
<td>F. Lennon</td>
<td>122-1876</td>
<td>Sky Dr.</td>
</tr>
</tbody>
</table>
Example - Constraints

- (SSN, NAME) → (PHONE, CC#)
- (SSN) → (SALARY)
- (INSUR= “Abx”) → (Treat= “Dental”)
- If the customer relation agrees with the Master relation on SSN and PHONE, then they must also agree on NAME, STR, and CITY
- The company Abx only accepts customers from SF
Example - Problems

- Repairing the FD, which phone numbers to resolve to isn’t clear
- Same problem when resolving SALARY
- When resolving the correct CC# there is no information available to make a “good” decision
- If these dependencies are resolved in a different order, you get different results (different definitions of repaired)
Contributions

The authors identify, address and offer solutions to three problems:

- Missing semantics to deal with these complex constraints
- Missing a formal repair algorithm
- Implementation and scalability of such an algorithm, if it were to exist
Let a schema \( S = \{R_1, ..., R_k\} \), where each table/relation \( R_i \) has some arity \( n_i \geq 0 \)

Given discrete sets NULL and CONSTS, an instance of \( S \), \( I = \{I_1, ..., I_k\} \), where for all \( i \), \( I_i \subset (\text{NULL} \cup \text{CONSTS})^{n_i} \)

If \( T \) is also a schema and \( J \) is an instance of it, then \((I, J)\) is an instance of \((S, T)\)

Let an Equality Generating Dependency (EGD) be defined as:
\[
\forall \vec{x} (\phi(\vec{x}) \rightarrow x_i = x_j)
\]
(SSN, NAME) → (PHONE)

- Cust(ssn,n,p,s,c,cc), Cust(ssn,n,p',s',c',cc') → p=p'

(INSUR= "Abx") → (Treat= "Dental")

- Treat(ssn,s,ins,tr,d), ins=Abx → tr=Dental

If the customer and master tables agree on SSN and PHONE, then they should also agree on NAME

- Cust(ssn,n,p,s,c,cc), MD(ssn,n',p',s',c',cc') → n=n'
### LLUNs

Placeholders, the opposite of null

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name</th>
<th>Phone</th>
<th>Str</th>
<th>City</th>
<th>CC#</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>111</td>
<td>M. White</td>
<td>408 – 3334</td>
<td>Red Ave.</td>
<td>NY</td>
</tr>
<tr>
<td>$t_2$</td>
<td>222</td>
<td>F. Lennon</td>
<td>122 – 1876</td>
<td>Sky Dr.</td>
<td>SF</td>
</tr>
<tr>
<td>$t_3$</td>
<td>222</td>
<td>F. Lennon</td>
<td>122 – 1876</td>
<td>Sky Dr.</td>
<td>SF</td>
</tr>
</tbody>
</table>
Repairs and Cell Groups

Cell Group
- \( g = (v \rightarrow \mathcal{C}, \mathcal{C}_s), \ \text{occ}(g) = \mathcal{C}, \ \text{just}(g) = \mathcal{C}_s \)

Examples
- \((L_0(781658, 784659) \rightarrow \{t_2.CC\#, \ t_3.CC\#\}, \ by \ \emptyset)\)
- \((F.Lennon \rightarrow \{t_2.NAME, \ t_3.NAME\}, \ by \ \{t_m.NAME\})\)
- \((Dental \rightarrow \{t_5.TREAT\}, \ by \ \{t_{c3}.TREAT\})\)

Repairs
- \(Rep = \{g_1, ..., g_k\}\)
The Partial Order

Informative Relation
- $v_1 \triangleleft v_2$ if $v_1$ is null and $v_2$ is not, or if $v_2$ is a llun and $v_1$ is not

Partial Order Specification
- $\sqcap$ is a set of assignments of attributes to partial orders
- $\sqsubseteq^\sqcap$ is the partial on cell values
- For two cell values $v_1$ and $v_2$; $v_1 \sqsubseteq^\sqcap v_2$ iff $v_1 = v_2$, $v_1 \triangleleft v_2$, or the corresponding poset values hold under that ordering i.e. $v_1' < v_2'$
• $\leq_{\Pi}$ induces a partial order on cell groups ($\text{val}(g)=\text{lub}$)
• Given a partial order based on the specification, $g_1 \leq_{\Pi} g_2$ if the following hold: $\text{occ}(g_1) \subseteq \text{occ}(g_2)$, $\text{just}(g_1) \subseteq \text{just}(g_2)$, and $\text{val}(g_1) \triangleleft \text{val}(g_2)$ or $g_1$ and $g_2$ are of the same type
The Partial Order - Cell Groups

- \((A) \rightarrow (B), (A=a) \rightarrow (B=x), (A=a) \rightarrow (B=y)\)
- \(R(a,1), R(a, 2)\)

\[
\begin{align*}
&1 \rightarrow \{t_1.B\}, \text{by } \emptyset \quad \subseteq_{\cap} \\
&2 \rightarrow \{t_1.B, t_2.B\}, \text{by } \emptyset \quad \subseteq_{\cap} \\
&x \rightarrow \{t_1.B, t_2.B\}, \text{by } \{t_{c1}.x\} \quad \subseteq_{\cap} \\
&L \rightarrow \{t_1.B, t_2.B\}, \text{by } \{t_{c1}.x, \{t_{c2}.y\} \quad \subseteq_{\cap}
\end{align*}
\]

Recall repairs are just set of cell groups, now one can say what repairs are better.
Definitions

Satisfaction after repairs

- A repair Rep is said to satisfy an EGD w.r.t \( \preceq \) if for all homomorphisms \( h \) of \( \phi(\bar{x}) \), \( g_h(x) \preceq g_h(x') \) or \( g_h(x') \preceq g_h(x) \)

Solution

- Given a cleaning scenario \( CS = \{(S, T), \Sigma, \sqcap\} \) and instance \( (I, J) \) a solution is a repair Rep s.t. \( J \preceq \sqcap \) Rep, and Rep satisfies \( \Sigma \)

Minimal Solution

- Rep is minimal if there does not exist a solution Rep’ s.t. Rep’ \( \preceq \) Rep
Chase Algorithm

- Known algorithm for testing implications of data dependencies
- Modifications: chases forward and backwards, uses the partial order, simplifies data into equivalences classes
Chase Example

FD: (SSN) → (SALARY)

<table>
<thead>
<tr>
<th>SSN</th>
<th>Salary</th>
<th>Insur.</th>
<th>Treat</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>10K</td>
<td>Abx</td>
<td>Dental</td>
<td>10/1/2011</td>
</tr>
<tr>
<td>111</td>
<td>25K</td>
<td>Abx</td>
<td>Cholest.</td>
<td>8/12/2012</td>
</tr>
</tbody>
</table>

\[
Rep = \{(10K \rightarrow \{t_4.\text{SALARY}\}, \text{by } \emptyset), (25K \rightarrow \{t_5.\text{SALARY}\}, \text{by } \emptyset)\}
\]

\[
Rep_{f,f} = (25K \rightarrow \{t_5.\text{SALARY}, t_4.\text{SALARY}\}, \text{by } \emptyset)
\]

\[
Rep_{b,f} = (L_1 \rightarrow \{t_4.\text{SALARY}\}, \text{by } \emptyset)
\]

\[
Rep_{f,b} = (L_2 \rightarrow \{t_5.\text{SALARY}\}, \text{by } \emptyset)
\]
Given a set of EGDs and a dirty data set, this modified chase algorithm terminates, generates a finite set of solutions, and also generates all minimum solutions. What’s the catch?
Cost Manager

- Maximum size, frequency, and forward only

Delta Database

- Only store what you need to know
Conclusion

• Current methods were not complete enough to specify all dependencies
• Repairing methods were too ad-hoc and no real algorithm existed
• LLUNATIC addresses both of these issues from the ground up and delivers a complete and proven algorithm
Thank you.
Discussion - Andrew Leung

- Q: What is the advantage of the EDG notation described in the paper?
- Q: How to translate the CFD into a formula: insurance company “Abs” only offers dental treatments (“Dental”)?
• Treat(ssn, s, ins, tr, d), ins = “Abs” → tr = “Dental”
• Using constant tables,
• Treat(ssn, s, ins, tr, d), Cst(ins, tr’) → tr = tr’
• Q: What are some limitations of the LLUNATIC framework?
• Q: What are some key differences between LLUNATIC and other systems with respect to their data repair model?
• Q: What are some ways to evaluate the data repair in terms of Quality Metrics?
• Q: What are some strengths and weaknesses of the paper?