Armstrong databases:
Validation, Communication and Consolidation of Conceptual Models with Perfect Test Data

Some Pain, Huge Gain

Sebastian Link
Department of Computer Science, The University of Auckland, New Zealand

joint work with
Sven Hartmann, Henning Koehler, Van Le, Mozhgan Memari, Bernhard Thalheim

This research is supported by the Marsden Fund Council from Government funding, administered by the Royal Society of New Zealand.
Example is the school of mankind, and they will learn at no other.
– Edmund Burke (1729-1797)
Context

- define conceptual models as relational database schemata that result from the transformation of a data model in some conceptual language:
  - Unified Modeling Language
  - Entity-Relationship Diagram
  - Description Logic Specification

- leading database design tools, such as ERWin, promote the creation of data samples to validate, communicate, and consolidate the conceptual models they produce

- What constitutes good data samples?
Main topics

- methodology to capture the semantics of an application domain within a conceptual model guided by perfect data samples

- semantics is captured by integrity constraints such as keys, functional dependencies, cardinality constraints, and NOT NULL constraints

- structural and computational properties of perfect data samples including existence of perfect samples, characterization of perfect samples in terms of integrity constraints, computation of perfect samples given integrity constraints, and investigation of complexities
Running example

- courses given by lecturers at a certain time in a certain location

```
CREATE TABLE Schedule (
  C_ID VARCHAR,
  L_Name VARCHAR,
  Time VARCHAR,
  Room VARCHAR,
  PRIMARY KEY (C_ID, TIME));
```

- DBMS explicitly enforces additional business rules:
  - combinations of C_ID and Time values are unique
  - values in columns C_ID and Time are not nullable

- data engineers would like to consolidate set of business rules
  - challenge: understand interaction of constraints
  - challenge: communicate with domain experts, and with managers
A perfect data sample

The following table is a perfect data sample for business rules given:

<table>
<thead>
<tr>
<th>C_ID</th>
<th>Time</th>
<th>L_Name</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>11301</td>
<td>Mon, 10am</td>
<td>Chen</td>
<td>Red</td>
</tr>
<tr>
<td>11301</td>
<td>Tue, 02pm</td>
<td>Chen</td>
<td>Red</td>
</tr>
<tr>
<td>78200</td>
<td>Mon, 10am</td>
<td>Chen</td>
<td>Red</td>
</tr>
<tr>
<td>99120</td>
<td>Wed, 04pm</td>
<td><strong>ni</strong></td>
<td><strong>ni</strong></td>
</tr>
</tbody>
</table>

What important business rules have not been captured?

Perfect data samples are useful:
- visualize abstract constraint sets
- pinpoint flaws with current design perceptions
- communication tool between engineers, experts, managers
A new perfect data sample

- revised set of business rules:
  - PRIMARY KEY \((C_{ID}, \text{TIME})\),
  - UNIQUE\((\text{Time}, L_{Name}, \text{Room})\)

- perfect sample data for the revised set:

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C_{ID}</td>
<td>Time</td>
<td>L_{Name}</td>
<td>Room</td>
</tr>
<tr>
<td>11301</td>
<td>Mon, 10am</td>
<td>Chen</td>
<td>Red</td>
</tr>
<tr>
<td>11301</td>
<td>Tue, 02pm</td>
<td>Chen</td>
<td>Red</td>
</tr>
<tr>
<td>78200</td>
<td>Mon, 10am</td>
<td>Chen</td>
<td>ni</td>
</tr>
<tr>
<td>99120</td>
<td>Mon, 10am</td>
<td>ni</td>
<td>Red</td>
</tr>
</tbody>
</table>

- is this ok now?
Revised Conceptual Model

the analysis leads us to:

```sql
CREATE TABLE Schedule

(C_ID CHAR[5], L_Name VARCHAR,
 Time CHAR[15], Room VARCHAR,
PRIMARY KEY (C_ID, Time),
UNIQUE (L_Name, Time),
UNIQUE (Room, Time));
```

we used sample data that was perfect with respect to
uniqueness, and
NOT NULL constraints
Different perfect data samples

► which table is a better perfect data sample?

<table>
<thead>
<tr>
<th>C_ID</th>
<th>Time</th>
<th>L_Name</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>11301</td>
<td>Mon, 10am</td>
<td>Chen</td>
<td>Red</td>
</tr>
<tr>
<td>11301</td>
<td>Tue, 02pm</td>
<td>Chen</td>
<td>Red</td>
</tr>
<tr>
<td>78200</td>
<td>Mon, 10am</td>
<td>Chen</td>
<td>Red</td>
</tr>
<tr>
<td>99120</td>
<td>Wed, 04pm</td>
<td>ni</td>
<td>ni</td>
</tr>
</tbody>
</table>

► complexities and trade-offs in modeling and maintenance
  → degree of capturing semantics vs complexity of perfect samples
Outline

► An enhanced database design methodology
► The data model
  ‣ Structure and semantics, perfect data samples
► Structural and computational properties
  ‣ existence of perfect data samples
  ‣ characterization of perfect data samples
  ‣ computation of perfect data samples from constraints
  ‣ investigation of complexities and trade-offs
► Applications
  ‣ Efficient processing of updates and queries
  ‣ Inference control
  ‣ Implementing relational approximations of domains in SQL
► Conclusion, open problems and literature
An Enhanced Database Design Methodology
The Data Model
Tables and their definitions

- table definition is a finite non-empty set $T$ of column headers
  - each column header $H$ has domain $\text{dom}(H) \cup \{\text{ni}\}$ of values
  - $\text{ni}$ denotes the no information marker (null "value")
  - example: $T = \{C\_ID, \ Time, \ L\_Name, \ Room\}$

- row $r$ over $T$ is a function
  - $r : T \rightarrow \bigcup_{H \in T} \text{dom}(H) \cup \{\text{ni}\}$ such that $r(H) \in \text{dom}(H) \cup \{\text{ni}\}$

- row $r$ is $X$-total iff for all $H \in X$, $r(H) \neq \text{ni}$

- table over $T$ is a finite multi-set $t$ of rows over $T$

<table>
<thead>
<tr>
<th>$C_ID$</th>
<th>$\text{Time}$</th>
<th>$L_Name$</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>11301</td>
<td>Mon, 10am</td>
<td>Chen</td>
<td>Red</td>
</tr>
<tr>
<td>11301</td>
<td>Mon, 10am</td>
<td>Chen</td>
<td>Red</td>
</tr>
<tr>
<td>78200</td>
<td>Tue, 02pm</td>
<td>Turing</td>
<td>$\text{ni}$</td>
</tr>
<tr>
<td>99120</td>
<td>Wed, 04pm</td>
<td>$\text{ni}$</td>
<td>Blue</td>
</tr>
</tbody>
</table>
Integrity constraints

- null-free subschema over $T$ is expression $\text{nfs}(T_s)$ where $T_s \subseteq T$
  $\iff$ table $t$ satisfies $\text{nfs}(T_s)$ iff for all $r \in t$, $r$ is $T_s$-total

- uniqueness constraint (UC) over $T$: expression $u(X)$ with $X \subseteq T$
  $\iff$ table $t$ satisfies $u(X)$ iff for all $r_1, r_2 \in t$:

  if $r_1(X) = r_2(X)$ and $r_1, r_2$ are $X$-total, then $r_1 = r_2$

- functional dependency (FD) over $T$: expression $X \rightarrow Y$ with $XY \subseteq T$
  $\iff$ table $t$ satisfies $X \rightarrow Y$ iff for all $r_1, r_2 \in t$:

  if $r_1(X) = r_2(X)$ and $r_1, r_2$ are $X$-total, then $r_1(Y) = r_2(Y)$

<table>
<thead>
<tr>
<th>C_ID</th>
<th>Time</th>
<th>L_Name</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>11301</td>
<td>Mon, 10am</td>
<td>Chen</td>
<td>Red</td>
</tr>
<tr>
<td>11301</td>
<td>Mon, 10am</td>
<td>Chen</td>
<td>Red</td>
</tr>
<tr>
<td>78200</td>
<td>Tue, 02pm</td>
<td>ni</td>
<td>Yellow</td>
</tr>
<tr>
<td>78200</td>
<td>Tue, 02pm</td>
<td>ni</td>
<td>Blue</td>
</tr>
</tbody>
</table>
The implication problem

\[ \text{implication problem:} \]

\[ \text{input: } T, \text{ NFS } nfs(T_s) \text{ and set } \Sigma \cup \{ \varphi \} \text{ of UCs and FDs over } T \]

\[ \text{output: } \begin{cases} \text{yes, if } \Sigma \models_{T_s} \varphi \\ \text{no, if } \Sigma \not\models_{T_s} \varphi \end{cases} \]

\[ \text{here: } \Sigma \models_{T_s} \varphi \text{ holds, if} \]

\[ \text{every table over } T \text{ that satisfies } \Sigma \text{ and } nfs(T_s) \text{ also satisfies } \varphi \]

\[ \text{closure of a set of column headers:} \]

\[ X_{\Sigma, T_s}^* := \{ H \in T \mid \Sigma \models_{T_s} X \rightarrow H \} \]
Examples of the implication problem

$\mathcal{T} = \{C_{ID}, \text{Time}, L_{Name}, \text{Room}\}, \mathcal{T}_s = \{C_{ID}, \text{Time}\}$:

$\Sigma = \{C_{ID}, \text{Time} \rightarrow L_{Name}; \text{Time},L_{Name} \rightarrow \text{Room}\}$

$\Sigma \not\models_{\mathcal{T}_s} C_{ID}, \text{Time} \rightarrow \text{Room}$

<table>
<thead>
<tr>
<th>$C_{ID}$</th>
<th>$\text{Time}$</th>
<th>$L_{Name}$</th>
<th>$\text{Room}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>78200</td>
<td>Tue, 02pm</td>
<td>ni</td>
<td>Yellow</td>
</tr>
<tr>
<td>78200</td>
<td>Tue, 02pm</td>
<td>ni</td>
<td>Blue</td>
</tr>
</tbody>
</table>

$\mathcal{T} = \{C_{ID}, \text{Time}, L_{Name}, \text{Room}\}, \mathcal{T}_s = \{L_{Name}\}$:

$\Sigma \models_{\mathcal{T}_s} C_{ID}, \text{Time} \rightarrow \text{Room}$, but $\Sigma \not\models_{\mathcal{T}_s} u(C_{ID}, \text{Time})$

<table>
<thead>
<tr>
<th>$C_{ID}$</th>
<th>$\text{Time}$</th>
<th>$L_{Name}$</th>
<th>$\text{Room}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>11301</td>
<td>Mon, 10am</td>
<td>Chen</td>
<td>Red</td>
</tr>
<tr>
<td>11301</td>
<td>Mon, 10am</td>
<td>Chen</td>
<td>Red</td>
</tr>
</tbody>
</table>
Perfect sample tables

- given $T$, $nfs(T_s)$, and set $\Sigma$ of constraints in class $C$ over $T$:
- a table $t$ over $T$ is said to be $C$-perfect for $\Sigma$ and $nfs(T_s)$ iff
  1. $t$ satisfies every $\sigma \in \Sigma$, and
  2. $t$ violates every constraint $\varphi$ in $C$ over $T$, if $\Sigma \not\models_{T_s} \varphi$, and
  3. $t$ satisfies $nfs(T'_s)$ iff $T'_s \subseteq T_s$

- given $T = \{ C(_ID), Ti(me), L(_Name), R(oom) \}$, $T_s = \{ C, Ti \}$:
  Which is perfect for $\Sigma = \{ u(C, Ti), u(Ti, L), u(Ti, R) \}$ and $nfs(T_s)$?

<table>
<thead>
<tr>
<th>$C_ID$</th>
<th>Time</th>
<th>L_Name</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>11301</td>
<td>Mon, 10am</td>
<td>Chen</td>
<td>Red</td>
</tr>
<tr>
<td>11301</td>
<td>Tue, 12pm</td>
<td>Chen</td>
<td>Red</td>
</tr>
<tr>
<td>11301</td>
<td>Wed, 02pm</td>
<td>Chen</td>
<td>Blue</td>
</tr>
<tr>
<td>11301</td>
<td>Thu, 04pm</td>
<td>Turing</td>
<td>Blue</td>
</tr>
<tr>
<td>22412</td>
<td>Fri, 06pm</td>
<td>Turing</td>
<td>Blue</td>
</tr>
<tr>
<td>33523</td>
<td>Fri, 06pm</td>
<td>ni</td>
<td>ni</td>
</tr>
</tbody>
</table>
Previous work

- perfect sample tables deeply studied in Codd’s relational model of data, no duplicate rows and no null value occurrences are allowed
- W. W. Armstrong: FDs over relations enjoy perfect sample relations
- Fagin termed perfect sample databases: Armstrong databases
- structural and computational properties of Armstrong relations for FDs studied by Beeri, Dowd, Fagin and Stateman, this is our special case where $T_s = T$ and no duplicate rows allowed
- extensions to other classes of constraints over relations, inclusion dependencies and Fagin’s implicational dependencies
- in presence of null values: hardly any work on Armstrong databases, existence by Levene/Loizou for FDs under exists but unknown null
Structural and Computational Properties
Perfect sample tables do not always exist

► \( T = \{ C, Ti, L, R \}, T_s = \{ C, Ti \}, \Sigma = \{ \emptyset \rightarrow L \} \)

► Is there a table \( t \) over \( T \) that is perfect for \( \Sigma \) and \( \text{nfs}(T_s) \)?

\( \rightarrow \) suppose \( t \) is perfect...

\( \rightarrow \Sigma \nmid_{T_s} \text{CLR} \rightarrow Ti : t \) must violate \( \text{CLR} \rightarrow Ti \)

\( \rightarrow \) hence, there must be two rows \( r_1, r_2 \) that are \( L \)–total

\( \rightarrow \) as \( t \) violates \( \text{nfs}(L) \), there must be a row \( r_3 \) with \( r_3(L) = \text{ni} \)

<table>
<thead>
<tr>
<th>C_ID</th>
<th>Time</th>
<th>L_Name</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>11301</td>
<td>Mon, 10am</td>
<td>Chen</td>
<td>Red</td>
</tr>
<tr>
<td>11301</td>
<td>Tue, 12pm</td>
<td>Chen</td>
<td>Red</td>
</tr>
<tr>
<td>......</td>
<td>..........</td>
<td>ni</td>
<td>...</td>
</tr>
</tbody>
</table>

\( \rightarrow \) if \( \text{CLR} \rightarrow Ti \) and \( \text{nfs}(L) \) are violated, then \( \emptyset \rightarrow L \)

► Theorem: The class of FDs over table definitions does not enjoy perfect sample tables.
Agree sets

for two rows \( r, r' \) over \( T \) let
\[
\begin{align*}
\rightarrow ag(r, r') & := (X, Y) \text{ where} \\
X & := \{H \in T \mid r(H) = r'(H) \land r(H) \neq \text{ni}\}, \text{ and} \\
Y & := \{H \in T \mid r(H) = r'(H)\}
\end{align*}
\]

for a table \( t \) over \( T \) let
\[
\begin{align*}
\rightarrow ag(t) & = \{ ag(r, r') \mid r, r' \in t \land r \neq r' \} \\
\rightarrow ag^s(t) & = \{ X \mid (X, Y) \in ag(t) \}, \\
\rightarrow ag^w(t) & = \{ Y \mid (X, Y) \in ag(t) \} \\
\rightarrow \text{for } X \in ag^s(t) \text{ let } w(X) = \bigcap \{ Y \mid (X, Y) \in ag(t) \}
\end{align*}
\]
Example for agree sets

consider the following table $t$:

<table>
<thead>
<tr>
<th>$C_ID$</th>
<th>$Time$</th>
<th>$L_Name$</th>
<th>$Room$</th>
</tr>
</thead>
<tbody>
<tr>
<td>11301</td>
<td>Mon, 08am</td>
<td>Chen</td>
<td>Red</td>
</tr>
<tr>
<td>11301</td>
<td>Mon, 08am</td>
<td>Chen</td>
<td>Red</td>
</tr>
<tr>
<td>11301</td>
<td>Tue, 10am</td>
<td>Chen</td>
<td>Red</td>
</tr>
<tr>
<td>11301</td>
<td>Wed, 12pm</td>
<td>Turing</td>
<td>Red</td>
</tr>
<tr>
<td>11301</td>
<td>Thu, 02pm</td>
<td>Turing</td>
<td>Blue</td>
</tr>
<tr>
<td>22412</td>
<td>Fri, 04pm</td>
<td>Turing</td>
<td>Blue</td>
</tr>
<tr>
<td>33513</td>
<td>Fri, 04pm</td>
<td>Zuse</td>
<td>Yellow</td>
</tr>
<tr>
<td>44624</td>
<td>Sat, 10am</td>
<td>Gödel</td>
<td>\textit{ni}</td>
</tr>
<tr>
<td>55725</td>
<td>Sat, 10am</td>
<td>Gödel</td>
<td>\textit{ni}</td>
</tr>
<tr>
<td>66836</td>
<td>Sun, 02pm</td>
<td>\textit{ni}</td>
<td>Green</td>
</tr>
<tr>
<td>66836</td>
<td>Sun, 02pm</td>
<td>\textit{ni}</td>
<td>White</td>
</tr>
</tbody>
</table>

some agree sets are:

$\leftarrow TiL, CTi \in ag^s(t)$ with $w(TiL) = TiLR$ and $w(CTi) = CTiL$
Families of maximal sets for column headers

- perfect samples must violate all $X \rightarrow H$ not implied by $\Sigma$ and $nfs(T_s)$
  - violate those where $X$ is maximal with this property
- maximal sets of a column header $H \in T$ wrt $\Sigma$ and $nfs(T_s)$:
  \[
  \text{max}_{\Sigma,T_s}(H) := \{ \emptyset \neq X \subseteq T \mid \Sigma \not\models_{T_s} X \rightarrow H \land \forall H' \in T - X (\Sigma \models_{T_s} X H' \rightarrow H) \}\n  \]
  - $\text{max}_{\Sigma,T_s}(T) = \bigcup_{H \in T} \text{max}_{\Sigma,T_s}(H)$
- $\Sigma = \{ CTi \rightarrow L, TiL \rightarrow R, TiR \rightarrow C \}$ and $T_s = \{ CTiLR \}$
  - $\text{max}_{\Sigma,T_s}(C)$ contains $Ti$ and $LR$,
  - $\text{max}_{\Sigma,T_s}(Ti)$ contains $CLR$,
  - $\text{max}_{\Sigma,T_s}(L)$ contains $CR$ and $Ti$, and
  - $\text{max}_{\Sigma,T_s}(R)$ contains $Ti$ and $CL$
- for $T_s = \{ CTi \}$
  - $\text{max}_{\Sigma,T_s}(C)$ contains $TiL$ and $LR$,
  - $\text{max}_{\Sigma,T_s}(Ti)$ contains $CLR$,
  - $\text{max}_{\Sigma,T_s}(L)$ contains $CR$ and $Ti$, and
  - $\text{max}_{\Sigma,T_s}(R)$ contains $CTi$ and $CL$
Duplicate sets

- perfect samples must also violate \( u(X) \) not implied by \( \Sigma \) and \( nfs(T_s) \)
  - in particular those where \( \Sigma \models_{T_s} X \rightarrow T \)
- the duplicate sets \( dup_{\Sigma,T_s}(T) \) of \( T \) wrt \( \Sigma \) and \( nfs(T_s) \) are
  \[
dup_{\Sigma,T_s}(T) := \{ X \subseteq T \mid \Sigma \models_{T_s} X \rightarrow T \land \Sigma \not\models_{T_s} u(X) \land \forall H' \in T - X (\Sigma \models_{T_s} u(XH')) \}\]

- for set \( \Sigma \) of UCs and FDs let
  \[
  \Sigma[FD] := \{ X \rightarrow Y \mid X \rightarrow Y \in \Sigma \} \cup \{ X \rightarrow T \mid u(X) \in \Sigma \}
  \]
- to compute \( dup_{\Sigma,T_s}(T) \):
  \[
  \text{generate } \mathcal{H} = (V, E) \text{ with } V = T \text{ and } E = \{ X - T_s \mid u(X) \in \Sigma \}
  \]
  \[
  \text{compute } \text{dup}_{\Sigma,T_s}(T) = \{ T - X \mid X \in \text{Tr}(\mathcal{H}) \land \forall M \in \text{max}_{\Sigma[FD],T_s}(T)(T - X \not\subseteq M) \}
  \]
  \[
  \text{Tr}(\mathcal{H}) \text{ are the minimal transversals of } \mathcal{H}
  \]
- \( \Sigma = \{ CTi \rightarrow L, TiL \rightarrow R, TiR \rightarrow C \} \) and \( T_s = \{ CTi \} \)
  \[
  \text{compute } \text{dup}_{\Sigma,T_s}(\text{SCHEDULE}) = \{ CTiLR \} 
  \]
Characterization of perfect sample tables

Let $T$ be a table definition, $\Sigma$ a set of UCs and FDs, and $nfs(T_s)$ an NFS over $T$. For all tables $t$ over $T$ it holds that $t$ is a perfect sample table for $\Sigma$ and $nfs(T_s)$ if and only if all of the following conditions are satisfied:

1. $\forall H \in T \forall X \in \text{max}_{\Sigma,T_s}(H)(X \in ag^s(t) \land H \notin w(X))$,
2. $\forall X \in ag^s(t)(X^*_\Sigma,T_s \subseteq w(X))$,
3. $\forall X \in \text{dup}_{\Sigma,T_s}(T)(X \in ag^s(t))$,
4. $\forall X \in ag^s(t)\forall u(Z) \in \Sigma(Z \not\subseteq X)$,
5. $\text{total}(t) = \{H \in T \mid \forall r \in T(r(H) \neq \text{ni})\} = T_s$
Example of characterization

Example of characterization

the table

<table>
<thead>
<tr>
<th>$C_ID$</th>
<th>$Time$</th>
<th>$L_Name$</th>
<th>$Room$</th>
</tr>
</thead>
<tbody>
<tr>
<td>11301</td>
<td>Mon, 08am</td>
<td>Chen</td>
<td>Red</td>
</tr>
<tr>
<td>11301</td>
<td>Mon, 08am</td>
<td>Chen</td>
<td>Red</td>
</tr>
<tr>
<td>11301</td>
<td>Tue, 10am</td>
<td>Chen</td>
<td>Red</td>
</tr>
<tr>
<td>11301</td>
<td>Wed, 12pm</td>
<td>Turing</td>
<td>Red</td>
</tr>
<tr>
<td>11301</td>
<td>Thu, 02pm</td>
<td>Turing</td>
<td>Blue</td>
</tr>
<tr>
<td>22412</td>
<td>Fri, 04pm</td>
<td>Turing</td>
<td>Blue</td>
</tr>
<tr>
<td>33513</td>
<td>Fri, 04pm</td>
<td>Zuse</td>
<td>Yellow</td>
</tr>
<tr>
<td>44624</td>
<td>Sat, 10am</td>
<td>Gödel</td>
<td>\text{ni}</td>
</tr>
<tr>
<td>55725</td>
<td>Sat, 10am</td>
<td>Gödel</td>
<td>\text{ni}</td>
</tr>
<tr>
<td>66836</td>
<td>Sun, 02pm</td>
<td>\text{ni}</td>
<td>Green</td>
</tr>
<tr>
<td>66836</td>
<td>Sun, 02pm</td>
<td>\text{ni}</td>
<td>White</td>
</tr>
</tbody>
</table>

is perfect for

$\Sigma = \{CTi \rightarrow L, TiL \rightarrow R, TiR \rightarrow C\}$ and $T_s = \{CTi\}$
Computation of perfect sample tables

Input: table definition $T$, a set $\Sigma$ of standard UCs and FDs, NFS $nfs(T_s)$ over $T$
Output: perfect sample table $t$ for $\Sigma$ and $nfs(T_s)$

Method: for all $H \in T$ let $c_{H,1}, c_{H,2}, \ldots \in dom(H)$ be distinct

(A0) for all $H \in T$ compute $\max_{\Sigma[FD], T_s}(H)$;

(A1) $t := \emptyset$; $i := 1$;

(A2) for all $X \in \max(T) \cup \text{dup}_{\Sigma, T_s}(T)$ do

(A3) if $X \in \max(T)$, then $Z := \{H \in T \mid X \in \max(H)\}$; else $Z := \emptyset$; endif;

(A4) $t := t \cup \{r_i, r_{i+1}\}$ where $\forall H \in T$

$r_i(H) := \begin{cases} c_{H,i}, & \text{if } H \in XZ T_s \\ \text{ni}, & \text{else} \end{cases}$

& $r_{i+1}(H) := \begin{cases} c_{H,i}, & \text{if } H \in X \\ c_{H,i+1}, & \text{if } H \in Z(T_s - X) \\ \text{ni}, & \text{else} \end{cases}$

(A5) $i := i + 2$;

(A6) enddo;

(A7) if $\text{total}(t) - T_s \neq \emptyset$, then return $t := t \cup \{r_i\}$ where $\forall H \in T$

$r_i(H) := \begin{cases} \text{ni}, & \text{if } H \in \text{total}(t) - T_s \\ c_{H,i}, & \text{else} \end{cases}$

else return $t$ endif;
Analysis of complexity

- **Theorem:** The complexity of finding a perfect sample table, given a set of UCs and FDs and an NFS, is precisely exponential.
  - there is an algorithm exponential in the number of column headers
  - there is a set $\Sigma$ of UCs and FDs and an NFS $nfs(T_s)$ in which the number of rows in each minimum-sized perfect sample table for $\Sigma$ and $nfs(T_s)$ is exponential

- The perfect sample table $t$ for $\Sigma$ and $nfs(T_s)$ is *minimum-sized* if there is no perfect sample table $t'$ for $\Sigma$ and $nfs(T_s)$ such that $|t'| < |t|$

- **Theorem:** On input $(T, \Sigma, nfs(T_s))$ our algorithm computes a perfect sample table for $\Sigma$ and $nfs(T_s)$ whose size is at most quadratic in the size of a minimum-sized perfect sample table for $\Sigma$ and $nfs(T_s)$

- **Theorem:** There is some table definition $T$, some set $\Sigma$ of UCs and FDs and some NFS $nfs(T_s)$ over $T$ such that $\Sigma$ has size $O(n)$, and the size of a minimum-sized perfect sample table for $\Sigma$ and $nfs(T_s)$ is $O(2^{n/2})$. There is some table definition $T$, some set $\Sigma$ of UCs and FDs and some NFS $nfs(T_s)$ over $T$ such that there is a perfect sample table for $\Sigma$ and $nfs(T_s)$ where the number of rows is in $O(n)$, and the optimal cover of $\Sigma$ with respect to $nfs(T_s)$ has size $O(2^n)$. 
Applications
Application 1:
Efficient processing of updates and queries

- discovery of constraints improves normalization (hence updates)
- $T = CTiLR, \ T_s = CT_i, \ \Sigma = \{CTi \rightarrow L, TiL \rightarrow R, TiR \rightarrow C\}$
- retrieve distinct combinations of courses and times for 'Chen'

```sql
SELECT DISTINCT T.C_ID, T.Time
FROM T
WHERE T.L_Name = 'Chen'
```

- perfect sample table may reveal semantic meaningfulness of $u(C, Ti)$
  - expensive `DISTINCT` clause is unnecessary

```sql
SELECT T.C_ID, T.Time
FROM T
WHERE T.L_Name = 'Chen'
```
Application 2: Inference control

- attacker may infer secrets without violating access control policies
- $T = C T i L R$, $T_s = C T i$, $\Sigma = \{C T i \rightarrow L, T i L \rightarrow R, T i R \rightarrow C\}$:
- some users prohibited access to

$$\Psi = (\exists X_R) T(12345, 'Wed, 2pm', 'Chen', X_R)$$

- attacker asks the following queries without violating access policy $\Psi$:

$$\Phi_1 = (\exists X_L) T(12345, 'Wed, 2pm', X_L, 'Red')$$

$$\Phi_2 = (\exists X_C) T(X_C, 'Wed, 2pm', 'Chen', 'Red')$$

- attacker exploits fact that $\Sigma \models_{C T i} T i R \rightarrow C L$ holds:

- applying $T i R \rightarrow C L$ to $\Phi_1$ and $\Phi_2$ reveals $\Phi$

- if $C \notin T_s$, attacker cannot draw that conclusion

- discovery of constraints assists in preventing inference attacks
Application 3:
From relational approximations to SQL table definitions

\[ R = \{A(\text{address}), C(\text{ity}), Z(\text{IP})\} \text{ with } \Sigma = \{AC \rightarrow Z, Z \rightarrow C\} \]

\[ \text{famous: there is no faithful lossless BCNF-decomposition} \]

How would we implement this relational approximation in SQL?

\[ \text{compute a perfect sample table } (T_s = \emptyset) \]

<table>
<thead>
<tr>
<th>Address</th>
<th>City</th>
<th>ZIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>03 Hudson St</td>
<td>Manhattan</td>
<td>10001</td>
</tr>
<tr>
<td>03 Hudson St</td>
<td>Manhattan</td>
<td>10001</td>
</tr>
<tr>
<td>70 King St</td>
<td>Manhattan</td>
<td>10001</td>
</tr>
<tr>
<td>70 King St</td>
<td>San Francisco</td>
<td>94107</td>
</tr>
<tr>
<td>ni</td>
<td>San Francisco</td>
<td>94129</td>
</tr>
<tr>
<td>15 Maxwell St</td>
<td>ni</td>
<td>ni</td>
</tr>
</tbody>
</table>

Should we really admit total duplicate rows?
Application 3:
From relational approximations to SQL table definitions

replace $AC \rightarrow Z$ by uniqueness constraint $u(AC)$
compute a perfect sample table for $\Sigma' = \{u(AC'), Z \rightarrow C\}$

<table>
<thead>
<tr>
<th>Address</th>
<th>City</th>
<th>ZIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>03 Hudson St</td>
<td>Manhattan</td>
<td>10001</td>
</tr>
<tr>
<td>70 King St</td>
<td>Manhattan</td>
<td>10001</td>
</tr>
<tr>
<td>70 King St</td>
<td>San Francisco</td>
<td>94107</td>
</tr>
<tr>
<td>ni</td>
<td>San Francisco</td>
<td>ni</td>
</tr>
<tr>
<td>15 Maxwell St</td>
<td>ni</td>
<td>60609</td>
</tr>
<tr>
<td>15 Maxwell St</td>
<td>ni</td>
<td>60609</td>
</tr>
</tbody>
</table>

What about the last two rows?
Application 3:  
From relational approximations to SQL table definitions

 perfect table shows that $\Sigma' = \{u(AC'), Z \rightarrow C\} \not= u(AZ)$
 compute perfect sample table for $\Sigma'' = \{u(AC'), u(AZ), Z \rightarrow C\}$
 and $nfs(AZ)$

<table>
<thead>
<tr>
<th>Address</th>
<th>City</th>
<th>ZIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>03 Hudson St</td>
<td>Manhattan</td>
<td>10001</td>
</tr>
<tr>
<td>70 King St</td>
<td>Manhattan</td>
<td>10001</td>
</tr>
<tr>
<td>70 King St</td>
<td>San Francisco</td>
<td>94107</td>
</tr>
<tr>
<td>35 Lincoln Blvd</td>
<td>San Francisco</td>
<td>94129</td>
</tr>
<tr>
<td>15 Maxwell St</td>
<td>ni</td>
<td>60609</td>
</tr>
</tbody>
</table>

that looks pretty good now, doesn’t it?
 what is the resulting SQL table definition?
Application 3: From relational approximations to SQL table definitions

- from $\Sigma = \{AC \rightarrow Z, Z \rightarrow C\}$ to an SQL table definition:

```sql
CREATE TABLE Contact (
    Address VARCHAR,
    City VARCHAR,
    ZIP INT,
    UNIQUE(Address,City),
    PRIMARY KEY(Address,ZIP),
    CHECK($Q = 0$));
```

where state assertion $Q$ is:

```sql
SELECT COUNT(*)
FROM Contact c1
WHERE c1.ZIP IN (
    SELECT ZIP
    FROM Contact c2
    WHERE c1.ZIP = c2.ZIP
    AND (c1.City <> c2.City
         OR (c1.City IS NULL AND c2.City IS NOT NULL)
         OR (c1.City IS NOT NULL AND c2.City IS NULL)));
```

- sample-based transfer of relational approximations to SQL
Conclusion, Future Work, and Literature
Conclusion

- Armstrong databases: a foundation for acquiring data semantics
  - to validate, communicate, and consolidate
  - conceptual models between stake-holders
- extension of previous theory on Armstrong relations to toolbox of perfect data samples
  - accommodate duplicate and partial information
  - only small space and no time penalties for more general toolbox
- perfect sample tables illustrate delicate interactions of constraints
  - useful for discovery of semantically meaningful constraints
  - demonstrated applications in schema design and security
- classes $C$ determine complexity of modeling and maintenance
  - perfect data sample may not exist
  - more expressive classes result in larger perfect data samples
Future work

- extension to classes of inclusion dependencies

- extension to other interpretations of null markers
  or approaches to capture partial information

- extensions of design tools
  including commercial tools which emphasize the need for good test data to validate models they produce

- constraint mining

- empirical evidence for the usefulness of perfect sample databases
Some literature

  - introduced Armstrong axioms for FDs over relational databases
  - implicit proof that FDs enjoy Armstrong relations

  - survey of Armstrong databases for several classes of constraints over relations

  - structural and computational properties of FDs over relations

  - characterization and computation of Armstrong relations for FDs using maximal set families

- Promotion of prototype databases to improve data-intensive applications
- Distinction between test (synthetic values) and sample (existing values) databases


- Introduction of the concept of informative Armstrong databases
- Some empirical studies based on the class of FDs over relations


- Formally measures the usefulness of Armstrong relations to acquire semantically meaningful FDs
- Some empirical studies are conducted


- Studies the implication problem for keys that have total and unique values
- Structural and computational properties of corresponding Armstrong relations
- Extremal problems are studied, e.g., what is the maximal number of minimal keys

V. Le, S. Link, M. Memari: *Discovery of keys from SQL tables*, The 17th International Conference on Database Systems for Advanced Applications (DASFAA), Lecture Notes in Computer Science, Springer, 2012. → studies Armstrong tables for the class of uniqueness and NOT NULL constraints → develops algorithms to determine covers of these constraints satisfied by a given table

S. Hartmann, H. Koehler, S. Link, B. Thalheim: *Armstrong databases and Reasoning for functional dependencies and cardinality constraints over partial bags*, The 7th International Symposium on Foundations of Information and Knowledge Systems (FoIKS), Lecture Notes in Computer Science, Volume 7153, pp. 165-184, Springer, 2012. → extends the results of this talk to the combined class of functional dependencies, cardinality and NOT NULL constraints → further shows trade-off between size of Armstrong tables and extent of modeling data semantics