Constraint Acquisition:

You can Chase but you cannot find

Sven Hartmann, Sebastian Link, Thu Trinh

Massey University, New Zealand
Attributes of Semantic Knowledge Acquisition

- assumption in database design:
  - semantic knowledge has been completely captured

- acquisition of integrity constraints far from trivial
  - but quality of target database crucially depends on it

- requires:
  - high abstraction and modelling capabilities
  - advanced understanding of logic
  - good communication skills
Our favourite tool: Armstrong DBs

- example-based deduction mechanism that results in sound and complete inferences

- Armstrong db satisfies $\varphi$ precisely when $\Sigma \models \varphi$
  - “user-friendly” representations of constraint sets
  - every $\varphi$ not implied by $\Sigma$ is violated
  - consequences of not specifying $\varphi$ can be identified in $db$

- Armstrong DBs may have many tuples
  - Demetrovics, Gyepesi, Katona, Tichler:
    minimum size of Armstrong relation for an arbitrary system of minimal keys over $n$ attributes has lower bound $\frac{1}{n^2} \binom{n}{\lfloor \frac{n}{2} \rfloor}$
  - difficult to focus on candidate constraints under consideration
  - domains of low cardinality may prohibit existence of Armstrong db
Example: Bad Sample Data

- relation schema \{CourseNo, RoomNo, Time\}, no FDs specified

- suppose we show following sample relation to designer:

<table>
<thead>
<tr>
<th>CourseNo</th>
<th>RoomNo</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>157266</td>
<td>SSLB 3</td>
<td>Monday, 9am</td>
</tr>
<tr>
<td>157357</td>
<td>SSLB 3</td>
<td>Wednesday, 1pm</td>
</tr>
</tbody>
</table>

- legal relation, fails to identify suggested design as unacceptable
  - different courses may be given at same room at same time
  - same course may be given at different rooms at same time
Example: Armstrong Database

better relation is following:

<table>
<thead>
<tr>
<th>CourseNo</th>
<th>RoomNo</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>157357</td>
<td>SSLB 3</td>
<td>Monday, 9am</td>
</tr>
<tr>
<td>157266</td>
<td>SSLB 3</td>
<td>Monday, 9am</td>
</tr>
<tr>
<td>157266</td>
<td>SST 4.42</td>
<td>Monday, 9am</td>
</tr>
<tr>
<td>157266</td>
<td>SST 4.42</td>
<td>Wednesday, 1pm</td>
</tr>
</tbody>
</table>

relation indicates design problems:

157266 meets in two different rooms on Monday at 9am
room SSLB 3 used by both 157266 and 157357 on Monday at 9am
Constraint Acquisition by Elimination

\[ \beta \text{ [e.g. Entity-Relationship Modelling, Springer 2000]}: \]

- **Initial Step**
  - Do not specify \( R = \emptyset \)
  - candidate \( \varphi \)
  - Specify \( \Sigma = \emptyset \)

- **Intermediates**
  - Do not specify \( R \)
  - candidate \( \psi \)
  - Specify \( \Sigma \)

- **Final Step**
  - Do not specify \( R \)
  - Specify \( \Sigma \)
A weaker Approach: Sample Databases

► for candidate $\varphi$ generate sample dbs satisfying $\Sigma$ and violating $\varphi$
  $\leftrightarrow$ in case that $\varphi$ is not implied by $\Sigma$
► still shows explicitly the consequences of not specifying $\varphi$
► not concerned with violating other constraints
► sample databases are smaller than Armstrong databases
  $\leftrightarrow$ focus on particular candidate $\varphi$
► really fit process of elimination
► if violations of $\varphi$ acceptable in future db instances, then discard $\varphi$
► if all violations of $\varphi$ unacceptable, then specify $\varphi$
  $\leftrightarrow$ how to represent all counterexamples?
  $\leftrightarrow$ how to generate these (semi)-automatically?
► the title of this talk:
  $\leftrightarrow$ you can Chase (a counterexample) but you cannot find (all of them)
The Stars of this Talk
The Stars of this Talk

Gerhard Gentzen (1909-1945)  
Evert Willem Beth (1908-1964)

Jaakko Hintikka (1929)  
Raymond M. Smullyan (1919)
Utilize Propositional Tableaux Techniques

- focus on Boolean dependencies

- implication of Boolean dependencies corresponds to logical implication of Boolean propositional formulae

- apply propositional tableaux to generate all 2-tuple counterexample relations for implication of $\varphi$ by $\Sigma$

- where $\Sigma \cup \{\varphi\}$ constitutes a set of Boolean dependencies
Example

- relation schema $\text{FILM} = \{ \text{Movie}, \text{Writer}, \text{Actor} \}$
  $\rightarrow \Sigma = \{ \text{Movie } \rightarrow \text{Writer} \}, \varphi = \text{Movie } \rightarrow \text{Writer}$
- $V_M \Rightarrow V_W \lor V_A$, and $\varphi$ as $V_M \Rightarrow V_W$:

$$
\begin{align*}
T(V_M \Rightarrow V_W \lor V_A) \\
F(V_M \Rightarrow V_W) \\
| \\
T(V_M) \\
F(V_W) \\

F(V_M) \\
T(V_W \lor V_A) \\

T(V_W) \\
T(V_A)
\end{align*}
$$
Example continued

- right-most branch not closed, hence \( \Sigma \cup \{\neg \varphi\} \) is satisfiable
- open branch defines a model for \( \Sigma \cup \{\neg \varphi\} \)
  \( \rightarrow \) a counterexample truth evaluation for implication of \( \varphi \) by \( \Sigma \)
- translate into database relation (automatically):

<table>
<thead>
<tr>
<th>Movie</th>
<th>Writer</th>
<th>Actor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

- edit using real-world entries (manually):

<table>
<thead>
<tr>
<th>Movie</th>
<th>Writer</th>
<th>Actor</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Godfather</td>
<td>M.Puzzo</td>
<td>M.Brando</td>
</tr>
<tr>
<td>The Godfather</td>
<td>F.F.Coppola</td>
<td>M.Brando</td>
</tr>
</tbody>
</table>

- pinpoints consequences of not specifying \( Movie \rightarrow Writer \)
Boolean Dependencies

Boolean dependencies on schema $R$ is smallest set with:

- every $A \in R$ is a BD,
- if $\varphi$ is a BD, then $\neg \varphi$ is a BD,
- if $\varphi_1, \varphi_2$ are BDs, then $(\varphi_1 \land \varphi_2)$ is a BD

$R$-relation $\{t_1, t_2\}$ satisfies BD $\varphi$ on $R$ ($\models_{\{t_1, t_2\}} \varphi$) iff:

- $\varphi = A \in R$: $t_1[A] = t_2[A]$,
- $\varphi = \neg \psi$: not $\models_{\{t_1, t_2\}} \psi$,
- $\varphi = (\varphi_1 \land \varphi_2)$: $\models_{\{t_1, t_2\}} \varphi_1$ and $\models_{\{t_1, t_2\}} \varphi_2$

$R$-relation $r$ satisfies BD $\varphi$ on $R$ ($\models_r \varphi$) iff:

for all distinct $t_1, t_2 \in r$ we have $\models_{\{t_1, t_2\}} \varphi$
Correspondence to Propositional Formulae

- let $\varphi'$ denote propositional formulae corresponding to BD $\varphi$
  $\Gamma \rightarrow$ and $\Sigma' = \{\sigma' : \sigma \in \Sigma\}$ for set $\Sigma$ of BDs

- $\phi_R = \neg A_1 \lor \cdots \lor \neg A_n$ where $R = \{A_1, \ldots, A_n\}$

- Theorem [Sagiv, Delobel, Parker jr., Fagin]:
  Let $\Sigma \cup \{\varphi\}$ be a set of Boolean dependencies on the relation schema $R$, and let $\Sigma' \cup \{\varphi'\}$ denote the set of propositional formulae that correspond to $\Sigma \cup \{\varphi\}$. Then the following are equivalent:
  - $\Sigma$ implies $\varphi$,
  - $\Sigma$ implies $\varphi$ in the world of two-element instances,
  - $\Sigma' \cup \{\phi_R\}$ logically implies $\varphi'$.
Propositional Tableaux in a Nutshell

- **signed** formulae is expression $T\varphi$ or $F\varphi$ where $\varphi$ is formula
- conclusions of a rule appended vertically or horizontally at a leaf
  if premise matches formula anywhere on path from root to leaf

$$
\begin{array}{c}
\alpha \\
\mid \\
\alpha_1 \\
\alpha_2 \\
\end{array}
\quad
\begin{array}{c}
\beta \\
\mid \\
\beta_1 \\
\beta_2 \\
\end{array}
$$

($\alpha$-expansion)  ($\beta$-expansion)

- expansion rules:

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>$\alpha_1$</th>
<th>$\alpha_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T(X \land Y)$</td>
<td>$TX$</td>
<td>$TY$</td>
</tr>
<tr>
<td>$F(X \lor Y)$</td>
<td>$FX$</td>
<td>$FY$</td>
</tr>
<tr>
<td>$F(X \Rightarrow Y)$</td>
<td>$TX$</td>
<td>$FY$</td>
</tr>
<tr>
<td>$T(\neg X)$</td>
<td>$FX$</td>
<td>$FX$</td>
</tr>
<tr>
<td>$F(\neg X)$</td>
<td>$TX$</td>
<td>$TX$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\beta$</th>
<th>$\beta_1$</th>
<th>$\beta_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F(X \land Y)$</td>
<td>$FX$</td>
<td>$FY$</td>
</tr>
<tr>
<td>$F(X \lor Y)$</td>
<td>$TX$</td>
<td>$TY$</td>
</tr>
<tr>
<td>$T(X \Rightarrow Y)$</td>
<td>$TX$</td>
<td>$TY$</td>
</tr>
<tr>
<td>$T(X \Rightarrow Y)$</td>
<td>$FX$</td>
<td>$TY$</td>
</tr>
</tbody>
</table>
Finding Counterexamples

- **Theorem:**
  Every complete open branch of a tableau for \( \{ T\sigma : \sigma \in \Sigma \} \cup \{ F\varphi \} \) defines a truth assignment \( \theta \) that satisfies \( \Sigma \) and violates \( \varphi \).

- **\( S \) set of signed formulae in a complete open branch \( \mathcal{B} \):**
  - No signed variable and its conjugate are both in \( S \)
  - If \( \alpha \in S \), then \( \alpha_1 \in S \) and \( \alpha_2 \in S \)
  - If \( \beta \in S \), then \( \beta_1 \in S \) or \( \beta_2 \in S \)

- known as **Hintikka sets** which are satisfiable as follows:
  - If \( Tp \in S \), then give \( p \) the value **true**.
  - If \( Fp \in S \), then give \( p \) the value **false**.
  - If neither \( Tp \) nor \( Fp \) is in \( S \), then give \( p \) value **true** or **false** at will.
Example: All Models

- attempt to refute $T((p \lor q) \lor r)$ and $F(p \land q)$

\[
\begin{array}{ccc}
T((p \lor q) \lor r) & & T(p \lor q) \\
F(p \land q) & & Tr \\
Tp & & Tq \\
& Fp & Fq \\
& Fp & Fq \\
Fx & Fq & Fp \\
\end{array}
\]

- the following models can be read off the tableau:

<table>
<thead>
<tr>
<th></th>
<th>p</th>
<th>q</th>
<th>r</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>true</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>2</td>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>3</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>4</td>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>5</td>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
</tbody>
</table>
Step 1: Attempt Refutation

- so far we have gathered the set $\Sigma$ of constraints to be specified
- $\varphi$ is the new candidate constraint
- translate $\Sigma \cup \{\varphi\}$ into $\Sigma' \cup \{\varphi'\}$
- attempt to refute $\{T(\sigma') \mid \sigma' \in \Sigma'\} \cup \{F(\varphi')\}$
- if completed tableau closed then $\varphi'$ implied by $\Sigma'$
  - $\implies$ and $\varphi$ is implied by $\Sigma$
  - $\implies$ $\varphi$ does not need to be specified
- otherwise $\varphi'$ is not implied by $\Sigma'$
  - $\implies$ read off all truth assignments $\theta$ that satisfy $\Sigma'$ and violate $\varphi'$
Step 2: Generate Counterexamples

- translate $\theta$s into 2-tuple counterexample relation $r_{\theta}$ for $\Sigma$ implying $\varphi$
  - first tuple of $r_{\theta}$ consists of 1’s only
  - second tuple has the truth value of $V_A$ as entry in column $A$
  - two tuples agree on attributes whose variables are true in $\theta$

- designer may substitute 1’s and 0’s by actual domain values
  - any reasonable attribute has at least two distinct values
Step 3: To Specify or not to Specify

- participants inspect all of counterexample relations
  \( \rightarrow \) these conform to rules represented by \( \Sigma \) and violate \( \varphi \)

- if there is some acceptable relation, then \( \varphi \) must not be specified

- otherwise we cannot find any acceptable 2-tuple violation of \( \varphi \)
  \( \rightarrow \) cannot find any acceptable counterexample since any such relation includes some unacceptable 2-tuple counterexample subrelation
  \( \rightarrow \) it is unacceptable to satisfy \( \Sigma \) without satisfying \( \varphi \)
  \( \rightarrow \) as \( \varphi \) not implied by \( \Sigma \) we specify \( \varphi \)
Example

\[ \text{BANK} = \{ C_{ID}, C_{Name}, Acc_{No}, Acc_{Type}, In_{Pol} \} \]

- customers have an id and a name, bank accounts with a no and a certain type, and insurances with an insurance policy

- determine a key for \text{BANK}, maybe \text{C}_{ID}?

\[ \text{candidate FD } C_{ID} \rightarrow C_{Name}, Acc_{No}, Acc_{Type}, In_{Pol} \]

\[
F(C_{ID} \Rightarrow C_{Name} \land Acc_{No} \land Acc_{Type} \land In_{Pol})
\]

\[
\mid
\]

\[
T(C_{ID})
\]

\[
F(C_{Name} \land Acc_{No} \land Acc_{Type} \land In_{Pol})
\]

\[
F(C_{Name}) \quad F(Acc_{No}) \quad F(Acc_{Type}) \quad F(In_{Pol})
\]

Wollongong, 24 January
only look at truth assignments defined by variables that occur
replace FD above by 4 separate FDs

we derive the following counterexamples

<table>
<thead>
<tr>
<th>C_ID</th>
<th>C_Name</th>
<th>C_ID</th>
<th>Acc_No</th>
<th>C_ID</th>
<th>Acc_Type</th>
<th>C_ID</th>
<th>In_Pol</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>Sylvester</td>
<td>001</td>
<td>0553-0331410-38</td>
<td>001</td>
<td>Cheque</td>
<td>001</td>
<td>House</td>
</tr>
<tr>
<td>001</td>
<td>Tweety</td>
<td>001</td>
<td>0553-0331410-40</td>
<td>001</td>
<td>Savings</td>
<td>001</td>
<td>Contents</td>
</tr>
</tbody>
</table>

left: same customer id must not be associated with different names
specify \( C_{ID} \rightarrow C_{Name} \)

second left: two accounts can be associated with the same c_id
counterexample acceptable: do not specify \( C_{ID} \rightarrow Acc_{No} \).

same customer may have different account types (insurance policies)
do not specify \( C_{ID} \rightarrow Acc_{Type} \) nor \( C_{ID} \rightarrow In_{Pol} \).
check suitability of all remaining attributes of BANK as a unary key
no attribute is suitable

when checking Acc_No we obtain following counterexample relation:

<table>
<thead>
<tr>
<th>Acc_No</th>
<th>Acc_Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0553-0331410-38</td>
<td>Cheque</td>
</tr>
<tr>
<td>0553-0331410-38</td>
<td>Savings</td>
</tr>
</tbody>
</table>

different account types are associated with the same account
unacceptable, hence specify Acc_No → Acc_Type

Is \{ C_ID, Acc_No, In_Pol \} a reasonable minimal key?

decide whether C_ID, Acc_No, In_Pol → C_Name, Acc_Type should be specified on top of

\[ \Sigma = \{ C_ID \rightarrow C_Name, Acc_No \rightarrow Acc_Type \} \]
The following tableau is closed:

\[
\begin{align*}
T(C_{ID} \Rightarrow C_{Name}) \\
T(Acc_{No} \Rightarrow Acc_{Type}) \\
F(C_{ID} \land Acc_{No} \land In_{Pol} \Rightarrow C_{Name} \land Acc_{Type}) \\
| \\
T(C_{ID}) \\
T(Acc_{No}) \\
T(In_{Pol}) \\
F(C_{Name} \land Acc_{Type}) \\
F(C_{Name}) & F(Acc_{Type}) \\
F(C_{ID}) & T(C_{Name}) & F(Acc_{No}) & T(Acc_{Type}) \\
\times & \times & \times & \times
\end{align*}
\]

Key is already implied by \( \Sigma \)
accounts independent from insurances: $C\_ID, C\_Type \rightarrow Acc\_No$?

$$
\begin{align*}
T(C\_ID & \Rightarrow C\_Name) \\
T(Acc\_No & \Rightarrow Acc\_Type) \\
F(C\_ID \land C\_Name & \Rightarrow Acc\_No \lor (Acc\_Type \land In\_Pol)) \\
\quad & T(C\_ID) \\
\quad & T(C\_Name) \\
\quad & F(Acc\_No) \\
\quad & F(Acc\_Type \land In\_Pol) \\
F(Acc\_Type) & F(In\_Pol) \\
F(C\_ID) & T(C\_Name)F(C\_ID) & T(C\_Name) \\
\times & F(Acc\_No) & T(Acc\_Type)F(Acc\_No)T(Acc\_Type)
\end{align*}
$$

tableau not closed
read off the following counterexample relations

\[
\begin{array}{|c|c|c|c|c|}
\hline
C_{ID} & C_{Name} & Acc\_No & Acc\_Type & In\_Pol \\
\hline
001 & Sylvester & 0331410 & Cheque & Life \\
001 & Sylvester & 1240501 & Savings & House \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|c|c|c|}
\hline
C_{ID} & C_{Name} & Acc\_No & Acc\_Type & In\_Pol \\
\hline
001 & Sylvester & 0331410 & Cheque & Life \\
001 & Sylvester & 1240501 & Cheque & House \\
\hline
\end{array}
\]

▶ top two unacceptable: policies are not independent of accounts

▶ bottom relation acceptable
  ⟷ constraint does not capture intended semantics

▶ Is \( C\_ID, C\_Type \rightarrow Acc\_No, Acc\_Type \) meaningful?
\( T(C_ID \Rightarrow C_Name) \)
\( T(Acc_No \Rightarrow Acc_Type) \)
\( F(C_ID \land C_Name \Rightarrow (Acc_No \land Acc_Type) \lor In_Pol) \)

\( T(C_ID) \)
\( T(C_Name) \)
\( F(In_Pol) \)
\( F(Acc_No \land Acc_Type) \)

\( F(Acc_No) \)
\( F(Acc_Type) \)

\( F(C_ID) \)
\( T(C_Name) \)
\( F(C_ID) \)
\( T(C_Name) \)

\( F(Acc_No) \)
\( T(Acc_Type) \)
\( F(Acc_No) \)
\( T(Acc_Type) \)

► read off the following counterexample relations

<table>
<thead>
<tr>
<th>C_ID</th>
<th>C_Name</th>
<th>Acc_No</th>
<th>Acc_Type</th>
<th>In_Pol</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>Sylvester</td>
<td>0331410</td>
<td>Cheque</td>
<td>Life</td>
</tr>
<tr>
<td>001</td>
<td>Sylvester</td>
<td>1240501</td>
<td>Savings</td>
<td>House</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C_ID</th>
<th>C_Name</th>
<th>Acc_No</th>
<th>Acc_Type</th>
<th>In_Pol</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>Sylvester</td>
<td>0331410</td>
<td>Cheque</td>
<td>Life</td>
</tr>
<tr>
<td>001</td>
<td>Sylvester</td>
<td>1240501</td>
<td>Cheque</td>
<td>House</td>
</tr>
</tbody>
</table>
both unacceptable: specify $C\_ID, C\_Type \rightarrow Acc\_No, Acc\_Type$

we have acquired the following constraint set $\Sigma$:

$\rightarrow C\_ID \rightarrow C\_Name$

$\rightarrow Acc\_No \rightarrow Acc\_Type$

$\rightarrow C\_ID, C\_Type \rightarrow Acc\_No, Acc\_Type.$

derive the following faithful 4NF-decomposition of BANK:

$\rightarrow C\_INFO=\{C\_ID,C\_Name\}$ with key $\{C\_ID\}$,

$\rightarrow C\_ACCOUNT=\{C\_ID,Acc\_No\}$ with key $C\_ACCOUNT$,

$\rightarrow A\_INFO=\{Acc\_No,Acc\_Type\}$ with key $\{Acc\_No\}$, and

$\rightarrow C\_INSURANCE=\{C\_ID,In\_Pol\}$ with key $C\_INSURANCE$
Conclusion

- Constraint acquisition crucial for quality of target database
- Boolean and multivalued dependencies: very common in database practice
- Semantic implication equivalent to logical implication of propositional fragments
- Analytical tableaux can be used to generate all minimal sample relations that satisfy all the constraints acquired so far but violate the current candidate
- Reject if single of these sample relations is acceptable as future instance
- Otherwise candidate should be specified