Inconsistency and Incompleteness in Relational Databases and Logic Programs

Navin Viswanath (advised by Dr. Rajshekhar Sunderraman)

Georgia State University, Atlanta, GA

1 July 2009
INTRODUCTION

INCOMPLETENESS IN RELATIONAL DATABASES
  • D-RELATIONS
  • OA-TABLES

NEGATION AND NONMONOTONIC REASONING
  • DEFAULT RELATIONS
  • NEGATION IN EXTENDED LOGIC PROGRAMS

INCONSISTENCY IN RELATIONAL DATABASES
  • SOURCE-AWARE REPAIRS
INTRODUCTION

D-RELATIONS (INCOMPLETENESS)

DEFAULT RELATIONS (NONMONOTONIC REASONING)

OA-TABLES (INCOMPLETENESS)

BACKGROUND (LOGIC PROGRAMMING)

INCONSISTENCY IN EXTENDED LOGIC PROGRAMS

INCONSISTENCY IN RELATIONAL DATABASES

CONCLUSIONS
Introduction

- Relational databases may represent incomplete information
- Incomplete information has been studied extensively since the introduction of the relational model
- An application area plagued by the incompleteness problem: sensor databases
- Records information about locations of moving objects, physical quantities like temperature etc.
Close relationship between relational databases and first order logic: a query on a relational database is a formula in first order logic

- First order logic is monotonic
- $\Sigma \vdash \beta$
- $\Sigma \cup \{\alpha\} \vdash \beta$
Relational databases operate under the Closed World Assumption (CWA) of Reiter, a *nonmonotonic* form of reasoning. According to the CWA, if sentence $P$ cannot be proved from the Horn database $DB$, assume $\neg P$.

In the presence of indefinite information (non-Horn clauses), CWA is not appropriate.

Let $DB = \{ P(a) \lor P(b) \}$. $DB \not
\vdash P(a)$ and $DB \not
\vdash P(a)$.

But $\neg P(a) \land \neg P(b)$ is inconsistent with $DB$. 
Motivation

Query: “Find all suppliers who do not supply part p1”

If there is a known list of suppliers, then the answer for the query would be \{s2, s3\}

null values complicates the problem further

If (s3,null) is part of the supply relation we are uncertain whether to include s3 as part of the answer or not

Similar problem occurs when one allows disjunctive information (such as (s3,p1) OR (s3,p2))
Inconsistency and Incompleteness in Relational Databases and Logic Programs

Navin Viswanath
(advised by Dr. Rajshekhar Sunderraman)

INTRODUCTION

D-RELATIONS (INCOMPLETENESS)

DEFAULT RELATIONS (NONMONOTONIC REASONING)

OA-TABLES (INCOMPLETENESS)

BACKGROUND (LOGIC PROGRAMMING)

INCONSISTENCY IN EXTENDED LOGIC PROGRAMS

INCONSISTENCY IN RELATIONAL

Open World Assumption

- It is sometimes important to explicitly include negative information.
- In a medical database, a doctor may be more comfortable knowing that a patient does not show symptoms of a disease by knowing it *explicitly* rather than inferring it, say, by the CWA.
- An *open world* is associated with a first order theory.
- Negative data is explicitly represented in the database.
- When the database complies with this assumption concerning negative data, the database is said to satisfy the *open world assumption*.
- Under the OWA, we “admit” that our knowledge of the world is incomplete.
A d-relation \( R \), over a scheme \( \Sigma \), consists of two components, \( \langle R^+, R^- \rangle \) where \( R^+ \subseteq 2^{\tau(\Sigma)} \) and \( R^- \subseteq 2^{\tau(\Sigma)} \)

- \( R^+ \), the positive component, is a set of tuple sets. Each tuple set represents a disjunctive positive fact
- \( R^- \), the negative component, is also a set of tuple sets. Each tuple set in \( R^- \) represents a disjunctive negative fact
- In the case where the tuple set is singleton, we have a definite fact
An example of a d-relation

```
 supply
 SNUM PNUM
{(s1,p1),(s1,p4)}
{(s1,p3)}
{(s2,p2)}
{(s3,p1),(s3,p4)}
{(s1,p2),(s1,p3)}
{(s2,p3)}
{(s3,p2),(s3,p3)}
```
**Theorem**

1. \( \text{rep}_\Sigma (R \hat{\cup} S) = S(\hat{\cup})(\text{rep}_\Sigma (R), \text{rep}_\Sigma (S)) \).
2. \( \text{rep}_\Sigma (R \hat{\cap} S) = S(\hat{\cap})(\text{rep}_\Sigma (R), \text{rep}_\Sigma (S)) \).
3. \( \text{rep}_{\Sigma_1} (\hat{\sigma}_F (R)) = S(\hat{\sigma}_F)(\text{rep}_{\Sigma_1} (R)) \).
4. \( \text{rep}_{\Sigma_1} (\hat{\pi}_\Delta (R)) = S(\hat{\pi}_\Delta)(\text{rep}_{\Sigma_1} (R)) \).
5. \( \text{rep}_{\Sigma_1 \cup \Sigma_2} (R \hat{\bowtie} S) = S(\hat{\bowtie})(\text{rep}_{\Sigma_1} (R), \text{rep}_{\Sigma_2} (S)) \).
Query Example

- supply
  - \{(s1,p1),(s1,p2)\}
  - \{(s2,p2)\}
  - \{(s3,p1)\}
  - \{(s2,p1),(s1,p1)\}
  - \{(s3,p2)\}

Find all suppliers who do not supply ‘p1’

- Ans = \bar{\pi}_S(\bar{\sigma}_p='p1'(supply)))
- \bar{\sigma}_p='p1'(supply)
- \bar{\pi}_S(\bar{\sigma}_p='p1'(supply))
- Ans

- \{s3\}
- \{s1,s2\}
- \{s1,s2\}
- \{s3\}
### Default Relations

A default relation on scheme $\Sigma$ is a triple $< R^+_e, R^-_e, R^-_d >$ where $R^+_e, R^-_e$ and $R^-_d$ are any subsets of $\tau(\Sigma)$.

- $R^+_e$ is the set of facts for which $R$ is known to hold.
- $R^-_e$ is the set of facts for which $R$ is known not to hold.
- $R^-_d$ is the set of facts for which $R$ is not known to hold.
Inconsistency and Incompleteness in Relational Databases and Logic Programs

Navin Viswanath (advised by Dr. Rajshekhar Sunderraman)

INTRODUCTION

D-RELATIONS (INCOMPLETENESS)

DEFAULT RELATIONS (NONMONOTONIC REASONING)

OA-TABLES (INCOMPLETENESS)

BACKGROUND (LOGIC PROGRAMMING)

INCONSISTENCY IN EXTENDED LOGIC PROGRAMS

INCONSISTENCY IN RELATIONAL

---

Query Example

<table>
<thead>
<tr>
<th>Patient</th>
<th>Disease</th>
</tr>
</thead>
<tbody>
<tr>
<td>pname</td>
<td>symptom</td>
</tr>
<tr>
<td>Tom</td>
<td>Forgetfulness</td>
</tr>
<tr>
<td>Jack</td>
<td>Headache</td>
</tr>
<tr>
<td>Tom</td>
<td>Nausea</td>
</tr>
<tr>
<td>Jack</td>
<td>Nausea</td>
</tr>
<tr>
<td>Jack</td>
<td>Forgetfulness</td>
</tr>
<tr>
<td>Ann</td>
<td>Forgetfulness</td>
</tr>
<tr>
<td>Ann</td>
<td>Sneezing</td>
</tr>
<tr>
<td>Ann</td>
<td>Headache</td>
</tr>
<tr>
<td>Ann</td>
<td>Nausea</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Disease</th>
<th>symptom</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cold</td>
<td>Headache</td>
</tr>
<tr>
<td>Alzheimer’s</td>
<td>Forgetfulness</td>
</tr>
<tr>
<td>Jaundice</td>
<td>Nausea</td>
</tr>
<tr>
<td>Alzheimer’s</td>
<td>Headache</td>
</tr>
<tr>
<td>Jaundice</td>
<td>Forgetfulness</td>
</tr>
</tbody>
</table>

∅

Figure: An instance of a hospital database
## INTRODUCTION

### D-RELATIONS (INCOMPLETENESS)

- **Default Relations (Nonmonotonic Reasoning)**
- **OA-TABLES (Incompleteness)**

### BACKGROUND (LOGIC PROGRAMMING)

- **Inconsistency in Extended Logic Programs**
- **Inconsistency in Relational Databases**

### INCONSISTENCY AND INCOMPLETENESS IN RELATIONAL DATABASES AND LOGIC PROGRAMS

---

<table>
<thead>
<tr>
<th>Temp</th>
<th>pname</th>
<th>symptom</th>
<th>dname</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Tom</td>
<td>Forgetfulness</td>
<td>Alzheimer’s</td>
</tr>
<tr>
<td></td>
<td>Jack</td>
<td>Headache</td>
<td>Cold</td>
</tr>
<tr>
<td></td>
<td>Tom</td>
<td>Nausea</td>
<td>{Alzheimer’s, Cold, Jaundice}</td>
</tr>
<tr>
<td></td>
<td>Jack</td>
<td>Nausea</td>
<td>{Alzheimer’s, Cold, Jaundice}</td>
</tr>
<tr>
<td></td>
<td>Jack</td>
<td>Forgetfulness</td>
<td>{Alzheimer’s, Cold, Jaundice}</td>
</tr>
<tr>
<td></td>
<td>Ann</td>
<td>Forgetfulness</td>
<td>{Alzheimer’s, Cold, Jaundice}</td>
</tr>
<tr>
<td></td>
<td>Ann</td>
<td>Nausea</td>
<td>{Alzheimer’s, Cold, Jaundice}</td>
</tr>
<tr>
<td></td>
<td>Ann</td>
<td>Headache</td>
<td>{Alzheimer’s, Cold, Jaundice}</td>
</tr>
<tr>
<td></td>
<td>Ann</td>
<td>Sneezing</td>
<td>{Alzheimer’s, Cold, Jaundice}</td>
</tr>
<tr>
<td></td>
<td>Jack</td>
<td>Headache</td>
<td>Alzheimer’s</td>
</tr>
<tr>
<td></td>
<td>Tom</td>
<td>Headache</td>
<td>Alzheimer’s</td>
</tr>
<tr>
<td></td>
<td>Tom</td>
<td>Forgetfulness</td>
<td>Jaundice</td>
</tr>
<tr>
<td></td>
<td>Jack</td>
<td>Sneezing</td>
<td>Alzheimer’s, Cold, Jaundice</td>
</tr>
<tr>
<td></td>
<td>Jack</td>
<td>Sneezing</td>
<td>Alzheimer’s, Cold, Jaundice</td>
</tr>
<tr>
<td></td>
<td>Tom</td>
<td>Forgetfulness</td>
<td>Alzheimer’s, Cold</td>
</tr>
<tr>
<td></td>
<td>Tom</td>
<td>Headache</td>
<td>Cold</td>
</tr>
<tr>
<td></td>
<td>Tom</td>
<td>Headache</td>
<td>Cold</td>
</tr>
<tr>
<td></td>
<td>Jack</td>
<td>Headache</td>
<td>Jaundice</td>
</tr>
</tbody>
</table>

**Which patients suffer from Alzheimer’s disease?**

**Answer**
- pname
- Tom
- Ann
- Jack

**Figure:** The result of the query.
Set valued attributes used to denote "disjunctions"

- Information contained in the above database:
  
  - `travel(c1, c2, 4)`
  - `travel(c2, c3, 3) ∨ travel(c2, c3, 4)`
  - `¬travel(c2, c3, 3) ∨ ¬travel(c2, c3, 4)`
  - `travel(c1, c4, 2) ∨ travel(c3, c4, 2)`
  - `¬travel(c1, c4, 2) ∨ ¬travel(c3, c4, 2)`

<table>
<thead>
<tr>
<th>src</th>
<th>dest</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>c2</td>
<td>4</td>
</tr>
<tr>
<td>c2</td>
<td>c3</td>
<td>{3,4}</td>
</tr>
<tr>
<td>{c1,c3}</td>
<td>c4</td>
<td>2</td>
</tr>
</tbody>
</table>
oa-Tables

oa-table scheme: \( R = \langle A_1, \ldots, A_n \rangle \), a list of attribute names.

oa-table \( T \) over the scheme \( R \) is defined as follows:

\[
T \subseteq 2^{\text{dom}(A_1) \times \text{dom}(A_2) \times \ldots \times \text{dom}(A_n)}
\]

- **oa-table** \( T \) consists of oa-tuples; \( T = \{w_1, w_2, \ldots, w_n\} \)
- **oa-tuple** \( w \) consists of tuple-sets; \( w = \{\eta_1, \eta_2, \ldots, \eta_m\} \)
  - each \( \eta_i \) is a possible world part.
- **tuple-set** \( \eta \) consists of tuples; \( \eta = \{t_1, t_2, \ldots, t_k\} \)
- \( \text{NEG}(T) = \bigcup_{i=1}^{n} (\mathcal{P}(\text{atoms}(w_i)) - w_i) \), is the set of impossible world parts.
Example of oa-table

```
<table>
<thead>
<tr>
<th>travel(src,dest,time)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(c1,c2,4)</td>
</tr>
<tr>
<td>(c2,c3,3)</td>
</tr>
<tr>
<td>(c2,c3,4)</td>
</tr>
<tr>
<td>(c1,c4,2)</td>
</tr>
<tr>
<td>(c3,c4,2)</td>
</tr>
<tr>
<td>(c1,c4,2)</td>
</tr>
<tr>
<td>(c3,c4,2)</td>
</tr>
</tbody>
</table>
```

\[ \text{NEG}(\text{travel}) = \{(c2, c3, 3), (c2, c3, 4)\} \]
Both oa-tables are inconsistent. For this discussion, we restrict all oa-tables to be consistent. Handling inconsistency is a separate issue.
Inconsistency and Incompleteness in Relational Databases and Logic Programs

Navin Viswanath (advised by Dr. Rajshekhar Sunderraman)

INTRODUCTION

D-RELATIONS (INCOMPLETENESS)

DEFAULT RELATIONS (NONMONOTONIC REASONING)

OA-TABLES (INCOMPLETENESS)

BACKGROUND (LOGIC PROGRAMMING)

INCONSISTENCY IN EXTENDED LOGIC PROGRAMS

INCONSISTENCY IN RELATIONAL DATABASES

\[
\begin{array}{c}
T \\
| a \\
| b \\
| c \\
| \hline
| a \\
| b \\
| \hline
| c \\
| \hline
| d \\
| \hline
| e \\
| f \\
\end{array}
\]

\[
\begin{array}{c}
COMPACT(T) \\
| a \\
| b \\
| \hline
| c \\
| \hline
| d \\
| \hline
| e \\
| f \\
\end{array}
\]
Inconsistency and Incompleteness in Relational Databases and Logic Programs

Navin Viswanath (advised by Dr. Rajshekhar Sunderraman)

INTRODUCTION
D-RELATIONS (INCOMPLETENESS)
DEFAULT RELATIONS (NONMONOTONIC REASONING)
OA-TABLES (INCOMPLETENESS)
BACKGROUND (LOGIC PROGRAMMING)
INCONSISTENCY IN EXTENDED LOGIC PROGRAMS
INCONSISTENCY IN RELATIONAL DATABASES

\[ T \xrightarrow{REDUCE} (T) \]

\[
\begin{array}{c}
  a \\
  a \\
  b \\
  c \\
  a \\
  a \\
  d \\
  e \\
  a \\
  a \\
  e \\
\end{array}
\]
\[\Gamma_R = \{ T \mid T \text{ is a oa-table over } R \}\]
\[\Sigma_R = \{ U \mid U \text{ is a set of relations over } R \}\]

Let \( T = \{ w_1, w_2, \ldots, w_n \} \). Then,
\[REP(T) = M(REDUCE(T))\]

\( M(T) : \Gamma_R \rightarrow \Sigma_R \) is defined as:

\[M(T) = \{ \eta_1 \cup \eta_2 \ldots \cup \eta_n \mid (\forall i, 1 \leq i \leq n)(\eta_i \in w_i) \land \\
\neg(\exists u \in NEG(T))(u \subseteq \eta_1 \cup \eta_2 \ldots \cup \eta_n)\}\]
Examples of *REP*

\[
M(T_1) = \{ [\text{a} \ c], [\text{b}] \}
\]

\[
M(T_2) = \{ [\text{a} \ b], [\text{a} \ c], [\text{b}] \}
\]
Inconsistency and Incompleteness in Relational Databases and Logic Programs

Navin Viswanath (advised by Dr. Rajshekhar Sunderraman)

INTRODUCTION
D-RELATIONS (INCOMPLETENESS)
DEFAULT RELATIONS (NONMONOTONIC REASONING)
OA-TABLES (INCOMPLETENESS)
BACKGROUND (LOGIC PROGRAMMING)
INCONSISTENCY IN EXTENDED LOGIC PROGRAMS
INCONSISTENCY IN RELATIONAL

\[ \sigma_F(T') = REDUCE(T) \text{ where,} \]
\[ T = \{ \{ \eta_1, \eta_2, \ldots, \eta_m \} \mid (\exists \{ \eta_1', \eta_2', \ldots, \eta_m' \} \in T')(\forall i, 1 \leq i \leq m)(\eta_i = \sigma_F(\eta_i')) \} \]

Drop tuples that do not satisfy the selection condition.

\[ T_{\sigma_1='a1'}(T) \]

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>b1</td>
</tr>
<tr>
<td>a1</td>
<td>b2</td>
</tr>
<tr>
<td>a1</td>
<td>b3</td>
</tr>
<tr>
<td>a1</td>
<td>b4</td>
</tr>
<tr>
<td>a2</td>
<td>b5</td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>b1</td>
</tr>
<tr>
<td>a1</td>
<td>b2</td>
</tr>
<tr>
<td>a1</td>
<td>b3</td>
</tr>
<tr>
<td>a1</td>
<td>b4</td>
</tr>
</tbody>
</table>
Inconsistency and Incompleteness in Relational Databases and Logic Programs

Navin Viswanath (advised by Dr. Rajshekhar Sunderraman)

INTRODUCTION

D-RELATIONS (INCOMPLETE-NES)*

DEFAULT RELATIONS (NONMONOTONIC REASONING)

OA-TABLES (INCOMPLETE-NES)*

BACKGROUND (LOGIC PROGRAMMING)

INCONSISTENCY IN EXTENDED LOGIC PROGRAMS

INCONSISTENCY IN RELATIONAL

PROJECTION

\[ \dot{\pi}_A(T') = REDUCE(T) \text{ where,} \]
\[ T = \{ \{ \eta_1, \eta_2, \ldots, \eta_m \} \mid (\exists \{ \eta_1', \eta_2', \ldots, \eta_m' \} \in T')(\forall i, 1 \leq i \leq m)(\eta_i = \pi(\eta'_i)) \} \]

Project tuples in each tuple-set of each oa-tuple.

\[ T \quad \dot{\pi}_1(T) \]

<table>
<thead>
<tr>
<th>( \dot{\pi}_1(T') )</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
</tr>
<tr>
<td>a1</td>
</tr>
<tr>
<td>a1</td>
</tr>
<tr>
<td>a1</td>
</tr>
<tr>
<td>a2</td>
</tr>
<tr>
<td>a2</td>
</tr>
<tr>
<td>a2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( \pi_1(T) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
</tr>
<tr>
<td>a1</td>
</tr>
<tr>
<td>a1</td>
</tr>
<tr>
<td>a2</td>
</tr>
<tr>
<td>a2</td>
</tr>
<tr>
<td>a2</td>
</tr>
</tbody>
</table>

Projection of tuples in set T.
\[ T_1 \times T_2 = \text{REDUCE}(T) \text{ where,} \]
\[ T = \{ \{ \eta_{11}, \eta_{12}, \ldots, \eta_{mn} \} \mid (\exists \{ \eta_{11}, \eta_{12}, \ldots, \eta_{1m} \} \in T_1) \]
\[ (\exists \{ \eta_{21}, \eta_{22}, \ldots, \eta_{2n} \} \in T_2) \]
\[ (\forall i, 1 \leq i \leq m)(\forall j, 1 \leq j \leq n) \]
\[ (\eta_{ij} = \eta_{1i} \times \eta_{2j}) \} \]

For each pair of oa-tuples from \( T_1 \) and \( T_2 \), take the Cartesian product of all "possible" world combinations.
Cartesian Product - Example

$T_1 \times T_2$
Union/Difference

\[ T_1 \cup T_2 = \text{REDUCE}(T_1 \cup T_2) \]

\[ T_1 \setminus T_2 = \text{REDUCE}(T) \text{ where,} \]
\[ T = \{ \{ \eta_{11}, \eta_{12}, \ldots, \eta_{mn} \} \mid (\exists \{ \eta_{11}, \eta_{12}, \ldots, \eta_{1m} \} \in T_1) \]
\[ (\exists \{ \eta_{21}, \eta_{22}, \ldots, \eta_{2n} \} \in T_2) \]
\[ (\forall i, 1 \leq i \leq m)(\forall j, 1 \leq j \leq n) \]
\[ (\eta_{ij} = \eta_{1i} - \eta_{2j}) \} \]
Inconsistency and Incompleteness in Relational Databases and Logic Programs

Navin Viswanath (advised by Dr. Rajshekhar Sunderraman)

INTRODUCTION

D-RELATIONS (INCOMPLETENESS)

DEFAULT RELATIONS (NONMONOTONIC REASONING)

OA-TABLES (INCOMPLETENESS)

BACKGROUND (LOGIC PROGRAMMING)

INCONSISTENCY IN EXTENDED LOGIC PROGRAMS

Inconsistency and Incompleteness in Relational Databases and Logic Programs

Theorem

1. \( \text{REP}(\hat{\sigma}_F(T)) = \sigma_F(\text{REP}(T)) \) for any reduced oa-table \( T \).
2. \( \text{REP}(\hat{\pi}_A(T)) = \pi_A(\text{REP}(T)) \) for any reduced oa-table \( T \) and list of attributes \( A \).
3. \( \text{REP}(T_1 \dot{\times} T_2) = \text{REP}(T_1) \times \text{REP}(T_2) \) for any reduced oa-tables \( T_1 \) and \( T_2 \).
4. \( \text{REP}(T_1 \dot{\cup} T_2) = \text{REP}(T_1) \cup \text{REP}(T_2) \) for any reduced and compatible oa-tables \( T_1 \) and \( T_2 \).
5. \( \text{REP}(T_1 \dot{-} T_2) = \text{REP}(T_1) - \text{REP}(T_2) \) for any reduced and compatible oa-tables \( T_1 \) and \( T_2 \).
**Query Example**

**Patient**

<table>
<thead>
<tr>
<th></th>
<th>Disease</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>cold</td>
</tr>
<tr>
<td></td>
<td>cold</td>
</tr>
<tr>
<td>tom</td>
<td>cold</td>
</tr>
<tr>
<td></td>
<td>flu</td>
</tr>
<tr>
<td></td>
<td>flu</td>
</tr>
<tr>
<td></td>
<td>flu</td>
</tr>
<tr>
<td></td>
<td>flu</td>
</tr>
<tr>
<td></td>
<td>flu</td>
</tr>
</tbody>
</table>

**Disease**

<table>
<thead>
<tr>
<th></th>
<th>Patient</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>cold</td>
</tr>
<tr>
<td></td>
<td>cold</td>
</tr>
<tr>
<td></td>
<td>cold</td>
</tr>
<tr>
<td></td>
<td>cold</td>
</tr>
<tr>
<td></td>
<td>cold</td>
</tr>
<tr>
<td></td>
<td>cold</td>
</tr>
<tr>
<td></td>
<td>cold</td>
</tr>
<tr>
<td></td>
<td>cold</td>
</tr>
<tr>
<td></td>
<td>cold</td>
</tr>
</tbody>
</table>

**Query: Who has the flu?**

\[ p \models \hat{\pi}_2(\hat{\sigma}_1 = \text{'flu'}(D)) \]

- jim

**Query: Who has the cold?**

\[ p \models \hat{\pi}_2(\hat{\sigma}_1 = \text{'cold'}(D)) \]

- jim
- don
C-tables (Imielinski and Lipski)
I-tables (Liu and Sunderraman)
E-tables (Liu and Zhang)
Minker generalized the CWA to represent such information.

Minker’s Generalized Closed World Assumption (GCWA) is based on the idea of minimal models.

The minimal models of \( \{P(a) \lor P(b)\} \) are \( \{P(a)\} \) and \( \{P(b)\} \).

According to the GCWA, the true sentences are ones that appear in every minimal model and the false sentences are the ones that appear in no minimal model.

The GCWA interprets disjunctions exclusively.
Another related concept is Ross and Topor’s Disjunctive Database Rule (DDR).

- **closed set**: set of ground atoms that can be assumed false.
- $S$ is a closed set of $DB$ if, for every element $A \in S$, and for every instance of a ground clause $C$ such that $A$ is in the head of $C$, there exists an atom $B$ in the body of $C$ such that $B \in S$.
- **greatest closed set** $gcs(DB)$ exists and represents the negative information that can be inferred from the database.
- DDR interprets disjunctions inclusively.
GCWA interprets disjunctions exclusively and DDR interprets disjunctions inclusively.

Consider $DB' = \{P(a) \lor P(b), P(a)\}$. The only minimal model of $DB'$ is $\{P(a)\}$ and GCWA assigns truth value false to $P(b)$.

Consider $D' = \{BG(john, A) \lor BG(john, B), BG(john, A)\}$ where $BG$ denotes the bloodgroup relation.

$DDR$ conclusions: $\{BG(john, A), BG(john, B)\}$ and $gcs(D') = \emptyset$.

The disjunction in the bloodgroup relation is obviously exclusive and DDR fails to negate $BG(john, B)$. 
Possible World Semantics (PWS) was introduced simultaneously by Sakama and Chan to overcome the drawbacks of GCWA and DDR.

Consider the database
\[ D_1 = \{ D \leftarrow; A \lor B \leftarrow D; C \leftarrow A, B; \neg (A \land B) \} \].

The possible worlds of \( D_1 \) are \( \{ D, A \} \) and \( \{ D, B \} \).

Notice the introduction of the negative clause \( \neg (A \land B) \).

This permits the interpretation of the disjunction \( A \lor B \) as exclusive.

Atoms that appear in every possible world are \textit{true}, those in no possible world are \textit{false} and those in some possible world are \textit{possibly true}.

By the PWS, \( D \) is true, \( C \) is false and \( A \) and \( B \) are possibly true.
DDR would ignore the negative clause and treat the disjunction as inclusive.

\[ T_{D_1} \uparrow \omega(\emptyset) = \{A, B, C, D\} \quad \text{and} \quad gcs(D_1) = \emptyset. \]

However, PWS needs the introduction of a negative clause to understand the correct nature of the disjunction.

oa-tables implements the PWS semantics at the EDB level.
Definite Logic Programs

- A term is a constant, a variable or a complex term of the form \( f(t_1, \ldots, t_n) \) where \( t_1, \ldots, t_n \) are terms and \( f \) is a function symbol.
- Atom: A formula of the form \( p(t_1, \ldots, t_n) \) where \( p \) is a predicate symbol.
- A definite logic program is a set of rules of the form \( A \leftarrow B_1, \ldots, B_n \), where \( A, B_1, \ldots, B_n \) are atoms (rules are called Horn clauses).
- Definite logic programs have a unique least model.
- Declarative semantics given by the \( T_P \) operator of van Emden and Kowalski.
The $T_P$ operator computes the least model of the definite logic program $P$
The least model of $P$ is the least fixpoint of $T_P$

$$T_P \uparrow 0 = \emptyset$$
$$T_P \uparrow i + 1 = T_P(T_P \uparrow i)$$
$$T_P \uparrow \omega = \bigcup_{i=0}^{\infty} T_P \uparrow i.$$
The “odd” program

\[\begin{align*}
\text{odd}(s(0)) & \leftarrow . \\
\text{odd}(s(s(X))) & \leftarrow \text{odd}(X).
\end{align*}\]

\[\begin{align*}
T_P \uparrow 0 &= \emptyset \\
T_P \uparrow 1 &= \{\text{odd}(s(0))\} \\
& \vdots \\
T_P \uparrow \omega &= \{\text{odd}(s^n(0)) \mid n \in \{1, 3, 5, \ldots \}\} \]
A general logic program is a set of rules of the form
\[ A \leftarrow B_1, \ldots, B_n, \sim C_1, \ldots, \sim C_m \]
where \( A, B_1, \ldots, B_n, C_1, \ldots, C_m \) are atoms

\( \sim \) denotes a form of negation called “negation by failure” (used in systems like Prolog)

We assume \( \sim A \) when atom \( A \) cannot be proved from the program (similar in spirit to CWA)
General Logic Program Semantics

- General logic programs may have several *minimal models*
- Finding the “intended” model is a difficult problem
- \( a \leftarrow \neg b \) has two *minimal* models \{a\} and \{b\}
- The intended model is \{a\}
- Two of the most popular semantics: stable model semantics (two-valued) and well-founded semantics (three-valued)
- Most semantics agree on a large class of general logic programs - the “stratified” programs (no recursion through negation)
Stable Model Semantics

Definition

Let $\Pi$ be a general logic program. For any set $S$ of atoms, let $\Pi^S$ be the definite logic program obtained from $\Pi$ by deleting

1. each rule that has a formula $\sim L$ in its body with $L \in S$, and
2. all formulas of the form $\sim L$ from the bodies of the remaining rules.

$\Pi^S$ does not contain $\sim$, so that its model is already defined. If this model coincides with $S$, then we say that $S$ is a stable model of $\Pi$.

A general logic program can have several stable models (when it is not stratified).
Example: Stable models

- Consider program $P$:
  
  $a \leftarrow \neg b$
  
  $b \leftarrow \neg a$

- Let $S_1 = \{a\}$
- Then $P^{S_1}$:
  
  $a \leftarrow \neg b$
  
  $b \leftarrow \neg a$

- Similarly, $S_2 = \{b\}$ is also a stable model
Extended Logic Programs

- **Literal**: An atom $p(t_1, \ldots, t_n)$ or its negation $\neg p(t_1, \ldots, t_n)$
- An extended logic program is a set of rules of the form $A \leftarrow B_1, \ldots B_n, \sim C_1, \ldots, \sim C_m$ where $A, B_1, \ldots B_n, C_1, \ldots C_m$ are literals
- $\sim$ denotes explicit negation
- $\sim$ of a literal $L$ is accepted only if it can be proven from the program (like proving $L$)
- $\sim$ of a literal $L$ is accepted as “failure to prove” $L$
- $\neg$ may be thought of as a “stronger” form of negation than $\sim$
Consider program $P$:

\[
\begin{align*}
  a & \leftarrow \sim \neg a \\
  \neg a & \leftarrow \sim a \\
  b & \leftarrow \sim \neg b
\end{align*}
\]

Prime transformation to obtain a general logic program

\[
\begin{align*}
  a & \leftarrow \sim a' \\
  a' & \leftarrow \sim a \\
  b & \leftarrow \sim b'
\end{align*}
\]

Find stable models of transformed program and reverse transformations to get answer sets
Example: Answer set semantics

Let $S_1 = \{a, b\}$

Then $P^{S_1}$:

$$a \leftarrow \neg a'$$
$$a' \leftarrow a$$
$$b \leftarrow \neg b'$$

Similarly, $S_2 = \{a', b\}$ is also a stable model. After reversing, $\{\neg a, b\}$ is an answer set
A feature of extended logic programs: querying the incompleteness

- $\text{Eligible}(X) \leftarrow \text{HighGPA}(X)$. \hspace{1cm} (1)
- $\text{Eligible}(X) \leftarrow \text{Minority}(X), \text{FairGPA}(X)$. \hspace{1cm} (2)
- $\neg \text{Eligible}(X) \leftarrow \neg \text{FairGPA}(X)$. \hspace{1cm} (3)
- $\text{Interview}(X) \leftarrow \sim \text{Eligible}(X), \sim \neg \text{Eligible}(X)$. \hspace{1cm} (4)
Extended logic programs operate under the OWA

A contradictory program

\[ a \leftarrow . \]

\[ \neg a \leftarrow . \]

Is the following program contradictory?

\[ \neg a \leftarrow . \]

\[ a \leftarrow \sim b. \]
Contradictory Extended Logic Programs

- After prime transformation
  
  \[ a' \leftarrow . \]
  
  \[ a \leftarrow \sim b. \]

- Stable model contains both \( a \) and \( a'(\neg a) \)
- Prime transformation loses the semantic connection between \( a \) and \( \neg a \)
- A simple solution: To every rule with \( L \) in the head, add \( \sim \neg L \) to the body
- Problem: Even

  \[ a \leftarrow . \]
  
  \[ \neg a \leftarrow . \]

  is not contradictory
Coherence

- $\neg a \leftarrow \cdot$
- $a \leftarrow \sim b$

- Assume we also had $\neg b \leftarrow \cdot$
- Principle of coherence: Strong negation ($\neg$) implies weak negation ($\sim$) (Alferes et al.)
- Simulating CWA for a predicate $L$ in an extended logic program: $\neg L \leftarrow \sim L$ (Gelfond and Lifschitz)
Coherence *alone* is too “strong” a requirement

The program $a \leftarrow \neg b$ does not entail $a$

Transformation has to include both coherence and some form of negation by failure

Derive a literal $L$ through a rule containing $\neg$ if we are willing to accept the CWA for $L$ (by adding $\neg \neg L$ to the body)
A ← B₁, …, Bₙ, ¬C₁, …, ¬Cₘ
Transformation Example I

The transformed program

\[ a \leftarrow \sim b \]

\[ \neg a \leftarrow \sim b \]
The transformed program

\[ a \leftarrow \sim b \]
\[ \neg a \leftarrow \sim b \]

\[ a \leftarrow \neg b \]
\[ a \leftarrow \sim b, \sim \neg a \]
The transformed program

\[ a \leftarrow \sim b \]
\[ \neg a \leftarrow \sim b \]

\[
\begin{align*}
  a & \leftarrow \neg b \\
  a & \leftarrow \sim b, \sim \neg a \\
  \neg a & \leftarrow \neg b
\end{align*}
\]
INTRODUCTION

D-RELATIONS (INCOMPLETENESS)

DEFAULT RELATIONS (NONMONOTONIC REASONING)

OA-TABLES (INCOMPLETENESS)

BACKGROUND (LOGIC PROGRAMMING)

INCONSISTENCY IN EXTENDED LOGIC PROGRAMS

INCONSISTENCY IN RELATIONAL DATABASES

Transformation Example I

The transformed program

\[
\begin{align*}
  a & \leftarrow \neg b \\
  \neg a & \leftarrow \neg b \\
  a & \leftarrow \neg b \leftarrow b' \\
  a & \leftarrow \sim b, \sim \neg a \\
  \neg a & \leftarrow \neg b \\
  \neg a & \leftarrow \sim b, \sim a \\
  \end{align*}
\]
Inconsistency and Incompleteness in Relational Databases and Logic Programs

Navin Viswanath
(advised by Dr. Rajshekhar Sunderraman)

INTRODUCTION

D-RELATIONS (INCOMPLETE-NESS)

DEFAULT RELATIONS (NONMONOTONIC REASONING)

OA-TABLES (INCOMPLETE-NESS)

BACKGROUND (LOGIC PROGRAMMING)

INCONSISTENCY IN EXTENDED LOGIC PROGRAMS

INCONSISTENCY IN RELATIONAL DATABASES

Transformation Example I

The transformed program

\[
\begin{align*}
    a & \leftarrow \neg b \\
\neg a & \leftarrow \neg b
\end{align*}
\]

\[
\begin{align*}
    a & \leftarrow \neg b \\
    a & \leftarrow \neg b, \neg a \\
\neg a & \leftarrow \neg b \\
\neg a & \leftarrow \neg b, \neg a
\end{align*}
\]

\[
\begin{align*}
    a & \leftarrow b' \\
    a & \leftarrow \neg b, \neg a' \\
\end{align*}
\]

Prime

Navin Viswanath (advised by Dr. Rajshekhar Sunderraman)
Transformation Example I

\[
\begin{align*}
a &\leftarrow \sim b \\
\neg a &\leftarrow \sim b
\end{align*}
\]

The transformed program

\[
\begin{align*}
a &\leftarrow \neg b \\
a &\leftarrow \sim b, \sim \neg a \\
\neg a &\leftarrow \neg b \\
\neg a &\leftarrow \sim b, \sim a
\end{align*}
\]

\[
\begin{align*}
a &\leftarrow b' \\
a &\leftarrow \sim b, \sim a' \\
\prime a &\leftarrow b' \\
a' &\leftarrow b'
\end{align*}
\]
Inconsistency and Incompleteness in Relational Databases and Logic Programs

Navin Viswanath (advised by Dr. Rajshekhar Sunderraman)

INTRODUCTION

D-RELATIONS (INCOMPLETE-NESS)

DEFAULT RELATIONS (NONMONOTONIC REASONING)

OA-TABLES (INCOMPLETE-NESS)

BACKGROUND (LOGIC PROGRAMMING)

INCONSISTENCY IN EXTENDED LOGIC PROGRAMS

Inconsistency and Incompleteness in Relational Databases and Logic Programs

Transformation Example 1

The transformed program

\[
\begin{align*}
  a & \leftarrow \neg b \\
  \neg a & \leftarrow \neg b \\
  a & \leftarrow \sim b, \neg a \\
  \neg a & \leftarrow \neg b \\
  \neg a & \leftarrow \sim b, a
\end{align*}
\]

\[
\begin{align*}
  a & \leftarrow \neg b \\
  \neg a & \leftarrow \neg b \\
  a & \leftarrow \sim b, \neg a \\
  \neg a & \leftarrow \neg b, a
\end{align*}
\]

\[
\begin{align*}
  a & \leftarrow b' \\
  a' & \leftarrow \sim b, a'
\end{align*}
\]
Transformation Example II

The transformed program

\[
\begin{align*}
  c & \leftarrow \neg b \\
  c & \leftarrow \sim b, \sim c \\
  a & \leftarrow \neg b \\
  a & \leftarrow \sim b, \sim a \\
  \neg a & \leftarrow \neg b \\
  \neg a & \leftarrow \sim b, \sim a \\
  c' & \leftarrow b \\
  c' & \leftarrow \sim b, \sim c' \\
  a' & \leftarrow b' \\
  a' & \leftarrow \sim b, \sim a' \\
\end{align*}
\]

- \( c \) is a consequence of the transformed program
- Other semantics would declare the program inconsistent although inconsistency is local to \( a \)
Transformation Properties

- For definite logic programs $T(P) = P$
- For general logic programs (without $\neg$), the transformation has no effect on the semantics of the program
- For extended logic program $P$, if $T(P)$ is inconsistent by a semantics, say $SEM$, then $P$ is inconsistent by $SEM$
Inconsistency and Incompleteness in Relational Databases and Logic Programs

Navin Viswanath (advised by Dr. Rajshekhar Sunderraman)

Inconsistent Databases

Functional dependency \( \text{class} \rightarrow \text{professor} \)

<table>
<thead>
<tr>
<th>Class</th>
<th>Professor</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>p1</td>
</tr>
<tr>
<td>c2</td>
<td>p2</td>
</tr>
<tr>
<td>c3</td>
<td>p3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>Professor</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>p2</td>
</tr>
<tr>
<td>c3</td>
<td>p3</td>
</tr>
</tbody>
</table>

**Figure:** Two relations whose union is inconsistent w.r.t FD
A repair of a database is the set of changes made to the database so that consistency is restored.

We are interested in the minimal repairs, the repairs that involve minimal updates to the original database.

A consistent query answer is the set of tuples that is true in every minimal repair of the database.
A database inconsistent w.r.t FD $\text{Class} \rightarrow \text{Professor}$

<table>
<thead>
<tr>
<th>Class</th>
<th>Professor</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>p1</td>
</tr>
<tr>
<td>c2</td>
<td>p2</td>
</tr>
<tr>
<td>c3</td>
<td>p3</td>
</tr>
<tr>
<td>c1</td>
<td>p2</td>
</tr>
</tbody>
</table>

**Figure:** The integrated database

<table>
<thead>
<tr>
<th>Class</th>
<th>Professor</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>p1</td>
</tr>
<tr>
<td>c2</td>
<td>p2</td>
</tr>
<tr>
<td>c3</td>
<td>p3</td>
</tr>
</tbody>
</table>

**Figure:** The minimal repairs
Logic programming has been used to obtain the repairs of the database.

Construct a *repair program* such that the answer sets of the repair program correspond to the repairs of the database.

The repair program is a disjunctive logic program with two kinds of negation, explicit negation and default negation.

There can be an exponential number of repairs for an inconsistent database.
Preferred Repairs

- Look only for a subset of the possible repairs of the database, the “preferred repairs”
- Repairs may be computed based on a preference for a subset of the sources which we consider “reliable”
- Requires the database to include information regarding what data is confirmed by what source
- Such a data model is the Information Source Tracking Method (IST) of Sadri
Figure: An example of a table in the IST method

<table>
<thead>
<tr>
<th>Class</th>
<th>Professor</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>p1</td>
<td>1 0</td>
</tr>
<tr>
<td>c1</td>
<td>p2</td>
<td>1 0</td>
</tr>
<tr>
<td>c2</td>
<td>p2</td>
<td>1 1</td>
</tr>
<tr>
<td>c3</td>
<td>p3</td>
<td>1 1</td>
</tr>
</tbody>
</table>

Tuple \((c3, p3)\) is in the relation if sources \(s_1\) and \(s_2\) are **correct** and source \(s_3\) is **incorrect**
Source-aware Repairs

- Introduce propositional constants of the form $s_i$ in the repair program to indicate source $s_i$ is believed: called $s$-literals.
- The facts in the database are introduced in the logic program as "conditional facts".
- $\text{teaches}(c3, p3) \leftarrow s_1, s_2, \neg s_3$ should indicate $\text{teaches}(c3, p3)$ is obtained as a consistent answer (it is in every repair) if we believe in sources $s_1$ and $s_2$ and do not believe in $s_3$.
- Modify answer set semantics to obtain preferred repairs.
Let $S_{source} = S \cup slits$.

$slords$ denotes the set of sources we want to believe.(disbelieve).

**Definition**

The transformation $\Pi_{S_{source}}$ of $\Pi$ w.r.t $S_{source}$ is obtained by:

1. Deleting every rule with not $L$ in the body with $L \in S_{source}$ and deleting every s-rule that:
   1. has $\neg s$ in the body with $s \in slits$ OR
   2. does not have every literal from $slits$ in its body

2. Deleting the negative literals from the bodies of the remaining rules and deleting every literal from the bodies of the remaining s-rules

$S_{source}$ is a source-aware answer set of $\Pi$ if it is the least model of $\Pi_{S_{source}}$. 

Source-aware Answer Sets
Source-aware Repairs

**Figure**: Data collected from independent sources \( s_1, s_2 \) and \( s_3 \)

<table>
<thead>
<tr>
<th>( P )</th>
<th>( S_1 )</th>
<th>( S_2 )</th>
<th>( S_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X \ Y )</td>
<td>( X \ Y )</td>
<td>( X \ Y )</td>
<td></td>
</tr>
<tr>
<td>a b</td>
<td>a e</td>
<td>b d</td>
<td></td>
</tr>
<tr>
<td>c d</td>
<td>b c</td>
<td>c d</td>
<td></td>
</tr>
</tbody>
</table>

**Figure**: The integrated database along with source information

<table>
<thead>
<tr>
<th>( P )</th>
<th>( X \ Y )</th>
<th>( I )</th>
</tr>
</thead>
<tbody>
<tr>
<td>a b</td>
<td>1 0 0</td>
<td></td>
</tr>
<tr>
<td>c d</td>
<td>1 0 0</td>
<td></td>
</tr>
<tr>
<td>a e</td>
<td>0 1 0</td>
<td></td>
</tr>
<tr>
<td>b c</td>
<td>0 1 0</td>
<td></td>
</tr>
<tr>
<td>b d</td>
<td>0 0 1</td>
<td></td>
</tr>
<tr>
<td>c d</td>
<td>0 0 1</td>
<td></td>
</tr>
</tbody>
</table>
Inconsistency and Incompleteness in Relational Databases and Logic Programs

Navin Viswanath (advised by Dr. Rajshekhar Sunderraman)

INTRODUCTION

D-RELATIONS (INCOMPLETE-NESS)

DEFAULT RELATIONS (NONMONOTONIC REASONING)

OA-TABLES (INCOMPLETE-NESS)

BACKGROUND (LOGIC PROGRAMMING)

INCONSISTENCY IN EXTENDED LOGIC PROGRAMS

INCONSISTENCY IN RELATIONAL

Figure: The integrated database along with source information

<table>
<thead>
<tr>
<th>P</th>
<th>X</th>
<th>Y</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>d</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>e</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>c</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>d</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>d</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure: The set of all minimal repairs of the database

\[
\begin{align*}
\{ P_{X Y} \} &= \{ \begin{array}{c|c|c}
X & Y & I \\
\hline
a & b & 1 \\
c & d & 1 \\
a & e & 0 \\
b & c & 0 \\
b & d & 0 \\
c & d & 0 \\
\end{array} \}, \\
\{ P_{X Y} \} &= \{ \begin{array}{c|c|c}
X & Y & I \\
\hline
a & b & 1 \\
c & d & 1 \\
a & e & 0 \\
b & c & 0 \\
b & d & 0 \\
c & d & 0 \\
\end{array} \} \\
\{ P_{X Y} \} &= \{ \begin{array}{c|c|c}
X & Y & I \\
\hline
a & b & 1 \\
c & d & 1 \\
a & e & 0 \\
b & c & 0 \\
b & d & 0 \\
c & d & 0 \\
\end{array} \}, \\
\{ P_{X Y} \} &= \{ \begin{array}{c|c|c}
X & Y & I \\
\hline
a & b & 1 \\
c & d & 1 \\
a & e & 0 \\
b & c & 0 \\
b & d & 0 \\
c & d & 0 \\
\end{array} \}
\end{align*}
\]

Figure: The repairs of the database based on a belief in source $s_1$

\[
\begin{align*}
\{ P_{X Y} \} &= \{ \begin{array}{c|c|c}
X & Y & I \\
\hline
a & b & 1 \\
c & d & 1 \\
a & e & 0 \\
b & c & 0 \\
b & d & 0 \\
c & d & 0 \\
\end{array} \} \\
\{ P_{X Y} \} &= \{ \begin{array}{c|c|c}
X & Y & I \\
\hline
a & b & 1 \\
c & d & 1 \\
a & e & 0 \\
b & c & 0 \\
b & d & 0 \\
c & d & 0 \\
\end{array} \}, \\
\{ P_{X Y} \} &= \{ \begin{array}{c|c|c}
X & Y & I \\
\hline
a & b & 1 \\
c & d & 1 \\
a & e & 0 \\
b & c & 0 \\
b & d & 0 \\
c & d & 0 \\
\end{array} \}
\end{align*}
\]
The repair program consists of:

- change program: responsible for the insertions and deletions in order to restore consistency
- persistence rules: enforce the fact that tuples in the database remain intact unless they violate constraints

The change program:

Facts:

\[ p(a, b) \leftarrow . \quad p(a, e) \leftarrow . \quad p(b, c) \leftarrow . \quad p(b, d) \leftarrow . \quad p(c, d) \leftarrow . \]

Triggering rule:

\[ \neg p'(X, Y) \lor \neg p'(X, Z) \leftarrow p(X, Y), p(X, Z), Y \neq Z. \]

Stabilizing rule:

\[ \neg p'(X, Z) \leftarrow p(X, Y), ydom(Z), Y \neq Z. \]
The Repair Program

- The persistence rules:
  s-rules:
  
  \[ p_s(a, b) \leftarrow s_1. \quad p_s(a, e) \leftarrow s_2. \quad p_s(b, c) \leftarrow s_2. \]
  
  \[ p_s(b, d) \leftarrow s_3. \quad p_s(c, d) \leftarrow s_1, s_3. \]

  Persistence defaults:
  
  \[ p'(X, Y) \leftarrow p_s(X, Y). \]
  
  \[ p'(X, Y) \leftarrow p(X, Y), \text{ not } \neg p'(X, Y). \]
  
  \[ \neg p'(X, Y) \leftarrow xdom(X), ydom(Y), \text{ not } p(X, Y), \]
  
  \[ \text{not } p'(X, Y). \]

  Starter rules: For 1 \leq i \leq 3,
  
  \[ s_i \leftarrow s_i. \quad \neg s_i \leftarrow \neg s_i. \]
Theorem

1. For every source-aware answer set $S_{source}$ of $\Pi$, there exists a repair $r'$ of the database instance $r$ w.r.t the integrity constraints $IC$ such that $r' = \{p(\bar{a}) \mid p'(\bar{a}) \in S_{source}\}$

2. For every repair $r'$ of the database instance $r$ w.r.t the integrity constraints $IC$ that is consistent with the set of sources believed(disbelieved), there exists a source-aware answer set $S_{source}$ such that $r' = \{p(\bar{a}) \mid p'(\bar{a}) \in S_{source}\}$
Conclusions

- We have presented data models capable of handling incomplete information under the open world assumption.
- $d$-relations handle disjunctive information in an open world relational database.
- default relations introduce the two-negation concept in a relational database.
- We have developed a translation technique to handle inconsistent information in extended logic programs.
- We have developed a technique to compute “preferred” repairs based on lineage information.
Future Work

- Semantic definition of the repairs of a disjunctive database that is inconsistent
- Efficient computation of the repairs
- Use of data models for incomplete information to represent repairs
- Semantic Web reasoning and its relationship to extended logic programs
# Publications

Navin Viswanath.
Explicit and default negation relational databases and logic programs.

Navin Viswanath and Rajshekhar Sunderraman.
Handling disjunctions in open world relational databases.
One of six best student papers.

Navin Viswanath and Rajshekhar Sunderraman.
Defaults in open world relational databases.
Won a grant of 250 euros.

Navin Viswanath and Rajshekhar Sunderraman.
Degrees of exclusivity in disjunctive databases.

Navin Viswanath and Rajshekhar Sunderraman.
Query processing in paraconsistent databases in the presence of integrity constraints.

Navin Viswanath and Rajshekhar Sunderraman.
A paraconsistent relational data model.

Navin Viswanath and Rajshekhar Sunderraman.
Source-aware repairs for inconsistent databases.

Navin Viswanath and Rajshekhar Sunderraman.
Handling inconsistencies in extended logic programs through program transformation.
To appear.