Logic and Relational Databases

bioactivity

<table>
<thead>
<tr>
<th>mol</th>
<th>ACE</th>
<th>SSA 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>m2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>m3</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

atoms

<table>
<thead>
<tr>
<th>mol</th>
<th>name</th>
<th>type</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1</td>
<td>al</td>
<td>C</td>
<td>2.1</td>
<td>4.3</td>
<td>3.4</td>
</tr>
<tr>
<td>m1</td>
<td>a2</td>
<td>O</td>
<td>5.0</td>
<td>1.0</td>
<td>3.5</td>
</tr>
</tbody>
</table>

bonds

<table>
<thead>
<tr>
<th>mol</th>
<th>atom1</th>
<th>atom2</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1</td>
<td>a1</td>
<td>a2</td>
<td>1</td>
</tr>
<tr>
<td>m1</td>
<td>a1</td>
<td>a3</td>
<td>2</td>
</tr>
</tbody>
</table>

bioactivity (m1, 0, 0).
bioactivity (m2, 1, 0).
bioactivity (m3, 0, 1).

atom (m1, al, C, 2.1, 4.3, 3.4).
atom (m1, a2, O, 5.0, 1.0, 3.5).

bond (m1, a1, a2, 2).
bond (m1, a1, a3, 1).

"Extensional Database"
**DataLog**

Definite program containing no non-nullary function symbols.

\[
\begin{align*}
grandparent (X, Z) &\leftarrow parent (X, Y), parent (Y, Z). \\
parent (X, Y) &\leftarrow father (X, Y). \\
parent (X, Y) &\leftarrow mother (X, Y). \\
father (adam, bill). &\leftarrow father (adam, beth). \\
mother (anne, bill). &\leftarrow mother (anne, beth). \\
father (bill, cathy). &\leftarrow mother (cathy, donald). \\
mother (diana, eric). &\leftarrow mother (diana, eric).
\end{align*}
\]

mother and father are defined extensionally.

grandparent and parent are defined intensionally.

---

**Some Interesting Notes**

- With function symbols, even the "simplest" logic program is undecidable—more specifically, there exist choices of literals \( L_1, L_2, \) \& \( L_3 \) such that it is undecidable if

\[
L_1, \quad L_2 \equiv L_3, \quad L_4
\]

for some literals \( L_4 \).

- Moreover, there exist 2-literal clauses.
• Moreover, there exist 2-literal clauses such that the other 2-literal clauses entailing/entailed by them are undecidable.

• But Datalog is **decidable**. (Not using SLD-resolution but using other approaches such as Magic Sets — Ramakrishnan, Ch.15.)

---

**What are we still missing?**

• **Field names.**
  Syntactic sugar — will live without.

• **Types for fields.**
  Just unary predicates:
  \[ \text{male(tom). person(X) \\& male(X)}. \]

• **Query language.**
  Coming next... will show that Datalog with negation as finite failure has at least the computational power of relational algebra.
Primitive Operations of Relational Algebra

- Union
- Set Difference
- Cartesian (cross-) Product
- Projection
- Selection

Union

\[ R_1 \cup R_2 = \{ \langle x_{1,\ldots,n} \rangle \mid \langle x_{1,\ldots,n} \rangle \in R_1 \lor \langle x_{1,\ldots,n} \rangle \in R_2 \} \]

In Datalog, use predicate symbols \( r_1/n \) & \( r_2/n \) for \( R_1 \) & \( R_2 \), respectively. Use \( r/n \) for their union:

\[ r(X_1,\ldots,X_n) \Leftarrow r_1(X_1,\ldots,X_n). \]
\[ r(X_1,\ldots,X_n) \Leftarrow r_2(X_1,\ldots,X_n). \]

Example: parent\((X,Y)\) \(\Leftarrow\) father\((X,Y)\).

parent\((X,Y)\) \(\Leftarrow\) mother\((X,Y)\).
**Difference**

\[ R_1 \setminus R_2 = \{ \langle x_1, \ldots, x_n \rangle \mid \langle x_1, \ldots, x_n \rangle \in R_1 \land \langle x_1, \ldots, x_n \rangle \notin R_2 \} \]

*In Datalog:*

\[ r(X_1, \ldots, X_n) \leftarrow r_1(X_1, \ldots, X_n), \text{ not } r_2(X_1, \ldots, X_n). \]

**Example:**

\[ \text{father}(X, Y) \leftarrow \text{parent}(X, Y), \text{ not mother}(X, Y). \]

**Cartesian Product**

\[ R_1 \times R_2 = \{ \langle x_1, \ldots, x_m, y_1, \ldots, y_n \rangle \mid \langle x_1, \ldots, x_m \rangle \in R_1 \land \langle y_1, \ldots, y_n \rangle \in R_2 \} \]

*In Datalog:*

\[ r(X_1, \ldots, X_m, Y_1, \ldots, Y_n) \leftarrow r_1(X_1, \ldots, X_m), r_2(Y_1, \ldots, Y_n). \]

**Example:**

\[ \text{mfcouple}(X, Y) \leftarrow \text{male}(X), \text{ female}(Y). \]
**Projection**

\[ \Pi_{i_1, \ldots, i_m} R = \{ \langle x_{i_1}, \ldots, x_{i_m} \rangle \mid \langle x_1, \ldots, x_n \rangle \in R, 1 \leq i_1, \ldots, i_m \leq n \} \]

(Projection deletes and/or rearranges one or more columns from \( R \).)

**In Datalog:**

\[ p \circ_r \langle x_1, \ldots, x_m \rangle \leftarrow r \langle x_1, \ldots, x_n \rangle. \]

**Example:** father \((X)\) \(\leftarrow\) father \((X, Y)\).

---

**Selection**

\[ d_F (R) = \{ \langle x_1, \ldots, x_n \rangle \mid \langle x_1, \ldots, x_n \rangle \in R \text{ and formula } F \text{ is true of } \langle x_1, \ldots, x_n \rangle \} \]

\(F\) is allowed to contain comparisons (=, \(\neq\), etc.) among variables and constants, joined by \(\land, \lor, \Rightarrow\). **Field names** \(\leftrightarrow\) **Values**

Datalog translation depends on \(F\). Where \(O(F)\) is the Datalog representation of \(F\):

\[ d_F R \text{ is represented as} \]

\[ r_S \langle x_1, \ldots, x_n \rangle \leftarrow r \langle x_1, \ldots, x_n \rangle, O(F). \]

**Example:** \(d_{\geq 1,000,000} \text{INCOME}(X, Y)\) is \(\text{millionaire}(X, Y)\) \(\leftarrow\) \(\text{income}(X, Y), Y \geq 1,000,000.\)
Notes on Selection

- Replace field names in \( F \) by corresponding variable names during translation.

- Using \( ; \) for \( V \) and not for \( T \), translate \( F \) directly.

\[
\begin{array}{c}
\begin{array}{cccc}
\text{R} \\
\hline
\text{f}_1 & \text{f}_2 & \text{f}_3 & \text{f}_4 \\
\vdots & \vdots & \vdots & \vdots \\
\end{array}
\end{array}
\]

\[
\delta_{(k_1,k_2)}(v)_{(k_2 \neq f_3 \land f_4 != 10)} \text{ R becomes}
\]

\[
\begin{array}{l}
\sigma_{(k_1,k_2)}(v)_{(k_2 \neq f_3 \land f_4 != 10)} \\
((X_1=Z_1) ; \\
(X_1 > X_2 \land X_3 < 10)).
\end{array}
\]

Can build other relational operators from those relational algebra primitives.

Example: Natural Join

\[
R \bowtie S = \Pi_{A} d_{R \bowtie S} A_{R \bowtie S} (R \times S)
\]

where \( T_1, ..., T_a \) are the attributes (field names) appearing in both \( R \) and \( S \), and \( A \) is the list of all attributes except \( S.T_1, ..., S.T_b \).

\[
\text{father (} X, Y \text{)} \bowtie \text{parent (} Y, Z \text{)}:
\]

\[
\begin{array}{l}
\text{cross (} X, Y_1, Y_2, Z \text{)} \leftarrow \text{father (} X, Y_1 \text{), parent (} Y_2, Z \text{).} \\
\text{sel (} X, Y_1, Y_2, Z \text{)} \leftarrow \text{cross (} X, Y_1, Y_2, Z \text{), } Y_1 = Y_2. \\
\text{proj (} X, Y_2 \text{)} \leftarrow \text{sel (} X, Y_1, Y_2 \text{).}
\end{array}
\]

Could do in one step:

\[
\text{of (} X, Y, Z \text{)} \leftarrow \text{father (} X, Y \text{), parent (} Y, Z \text{).}
\]
Missing Values

<table>
<thead>
<tr>
<th>f1</th>
<th>f2</th>
<th>f3</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
<td>?</td>
</tr>
<tr>
<td>b</td>
<td>?</td>
<td>9</td>
</tr>
<tr>
<td>c</td>
<td>3</td>
<td>f</td>
</tr>
</tbody>
</table>

Like existential variables: we know some value must go there, but we don't know what.

Datalog: no existential quantifiers. Use skolem constants. Every missing value is represented by a distinct skolem constant.

\[
p(a, 1, \text{sk}-1).
p(b, \text{sk}-2, 9).
p(c, 3, f).
\]