On entity resolution in probabilistic data
Ayat, S.N.

Citation for published version (APA):
Ayat, S. N. (2014). On entity resolution in probabilistic data

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: http://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.
Appendix A

Uncertain Data Models

In this chapter, we review a number of uncertain data models. In the first three sections, we restrict our attention to the uncertain relational models, then in Section A.4, we explain some other models.

A.1 Incomplete Relational Models

The early uncertain data models mostly focus on representing incomplete information without caring about their possible existence probabilities. As in the literature, we refer to these models as incomplete data models. The notion of null values is perhaps the earliest attempt to model the incomplete information in the relational model. In this section, we begin by applying this way of modeling uncertain data, and then we present the more expressive incomplete relational models.

A.1.1 Codd tables

The aim of Codd tables, proposed by Codd [47, 48], is to model missing information in relational tables. The idea is to represent missing attribute values by a symbol, called null, which is different from all other data values. Let for instance symbol \( \perp \) denote the null. For illustration, consider the Codd table \( T \), shown in Figure A.1(a), where the missing age of Mary and John, and Bob’s phone number are represented using null values.

Codd has adopted the three-valued logic for evaluating the queries over Codd tables. According to this logic, there are three logical values indicating true, false, and unknown. Besides the rules of the boolean logic, three-valued logic has a number of additional rules, which are shown in Figure A.1(b). In the Codd’s proposed query evaluation semantics, comparing any value (including null itself) with null results in the unknown logical value. This semantics is currently in use in most commercial database management systems. However, this approach
Appendix A. Uncertain Data Models

<table>
<thead>
<tr>
<th></th>
<th>name</th>
<th>age</th>
<th>phone-no</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>Bob</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>$t_2$</td>
<td>Mary</td>
<td>$\bot$</td>
<td>789526</td>
</tr>
<tr>
<td>$t_3$</td>
<td>John</td>
<td>$\bot$</td>
<td>5256661</td>
</tr>
</tbody>
</table>

(a)

\[ \land \quad \text{unknown} \quad \lor \quad \text{unknown} \]

\[ \begin{array}{cccc}
\text{true} & \text{unknown} & \text{true} & \text{true} \\
\text{false} & \text{false} & \text{false} & \text{unknown} \\
\text{unknown} & \text{unknown} & \text{unknown} & \text{unknown}
\end{array} \]

\[ \neg \text{unknown} = \text{unknown} \]

(b)

Figure A.1: a) Table $T$: an example Codd table, b) The additional rules in three-valued logic

may return counter-intuitive results [66]. For example, consider evaluating the following query over table $T$, shown in Figure A.1(a):

\[
\begin{align*}
\text{SELECT * FROM } T \\
\text{WHERE } \text{age} = 25 \text{ OR } \text{age} \neq 25
\end{align*}
\]

The query returns $t_1$ as the result, while one might expect all tuples in $T$ as the query result [66].

Codd tables are very limited in representing incomplete information. They can only represent the fact that some attribute values are missing, and cannot represent any other kind of additional information about the missing attribute values. Consider, for instance, the case that we do not know the ages of Mary and Bob, but we know that Mary is younger than Bob. We cannot represent this kind of information using Codd tables. On the other hand, Codd tables are closed only under selection and projection operations [78]. To overcome the limitations of Codd tables, $c$-tables (short for conditional tables) [78] have been proposed.

A.1.2 C-tables

A c-table can represent the uncertainty at both the attribute and tuple level. At the attribute level, it allows constants and variables as attribute values, and at the tuple level, a logical expression, indicating the existence condition of the tuple in the table, is associated with each tuple. A possible world is built by an instantiation of variables with constants, where each possible world contains the tuples whose condition are satisfied by the instantiation. For illustration, Figure A.2 shows a c-table.
The c-tables model is a complete model, and closed under the whole relational algebra database language. The query evaluation however is computationally expensive [9]. Moreover, the use of variables in the model makes it hard for users to read and reason with it [107, 117]. While the c-tables model is a very powerful formalism, these shortcomings make it an impractical uncertainty model.

A number of variations of c-tables have been proposed in the literature. The c-table associated with a global condition has been proposed in [65]. Furthermore, removing the tuples’ conditions from c-table, Abiteboul et al. [9] has proposed the following three models: e-table that is a table where only the equality of variables are allowed; i-table that is a table with a global condition, where the condition is restricted to a conjunction of inequalities on variables; and g-table that is a combination of e-table and i-table. For better intuition, Figure A.3 shows examples of these models. To illustrate, repetitive use of variable $x$ in Figure A.3(a) poses the condition $t_1.A = t_2.C$ on the shown e-table. In the i-table, shown in Figure A.3(b), no variable is used repetitively, which thus shows no condition is posed by the variables on the table, but the table is associated with the global condition $x \neq z \land y \neq 0$. The combination of repetitive use of variables $x$ and $y$ (representing their equality), and the global condition $x \neq 0$ is shown in the g-table in Figure A.3(c).

A.2 Fuzzy Relational Models

Fuzzy relational models use the fuzzy set theory [144] to represent the uncertainty in data. In this section, we first briefly present the concept of fuzzy set, and then
explain how it is used to represent uncertainty in the relational model.

A.2.1 Fuzzy Set

In ordinary set theory, the membership of elements in a set are binary values, meaning that an element either belongs or does not belong to the set. In contrast, the membership of elements in a fuzzy set is represented by real numbers from the range \([0..1]\). More precisely, a fuzzy set \(F\) in a universe \(U\) is identified by a membership function \(\Phi_F: U \rightarrow [0..1]\), where \(U\) is a set of elements, called universe, and \(\Phi_F(u)\) for each \(u \in U\) denotes the degree of membership of \(u\) in the fuzzy set \(F\).

Fuzzy sets mostly are used to represent the imprecise concepts. For instance, the high salary concept is an imprecise concept, which might be represented with the fuzzy set \(HS\), in universe \(Salaries\), with the following membership function:

\[
\Phi_Hs(s) = \begin{cases} 
0 & \text{if } s \leq 30,000 \\
\frac{1}{1 + |s - 70,000|} & \text{if } 30,000 < s < 70,000 \\
1 & \text{if } s \geq 70,000
\end{cases}
\]

where the universe \(Salaries\) is the set of integer numbers in range \([10,000..100,000]\).

The fundamental ordinary set operations, i.e. union, intersection, and complement, have fuzzy counterparts defined as follows. Let \(A\) and \(B\) two fuzzy sets in the universe \(U\) with membership function \(\Phi_A\) and \(\Phi_B\), respectively. Then, the membership functions of \(A \cup B\), \(A \cap B\), and \(A'\) are defined as

\[
\Phi_{A \cup B}(u) = \max(\Phi_A(u), \Phi_B(u)) \\
\Phi_{A \cap B}(u) = \min(\Phi_A(u), \Phi_B(u)) \\
\Phi_{A'}(u) = 1 - \Phi_A(u)
\]

Let \(A_1, \ldots, A_n\) be fuzzy sets in \(U_1, \ldots, U_n\), respectively, and \(U = U_1 \times \cdots \times U_n\). Then, the cartesian product of \(A_1, \ldots, A_n\) is defined to be a fuzzy set in universe \(U\) with the following membership function:

\[
\Phi_{A_1 \times \cdots \times A_n}(u_1, \ldots, u_n) = \min(\Phi_{A_1}(u_1), \ldots, \Phi_{A_n}(u_n))
\]  

(A.1)

where \(u_i \in U_i\) and \(\Phi_i\) is the membership function of fuzzy set \(A_i\).

A.2.2 Fuzzy Relations

A fuzzy relation \(R\) on schema \(S(A_1, \ldots, A_n)\) is a fuzzy set in universe \(\text{dom}(A_1) \times \cdots \times \text{dom}(A_n)\), where \(\text{dom}(A_i)\) is the domain of attribute \(A_i\), which might itself be an ordinary or a fuzzy set [109]. Similar to ordinary relations, a fuzzy relation can be represented as a table with an additional attribute \(\Phi\) which shows the
degree of membership of each tuple to the relation. The $\Phi$ attribute, for each tuple, is determined using the definition of cartesian product for fuzzy sets, i.e. equation (A.1). The table only contains tuples for which $\Phi > 0$.

The main purpose of a fuzzy relation is to establish a fuzzy association between its attributes using a fuzzy proposition, i.e. a proposition whose truth value can be a number in range $[0..1]$, in contrast to an ordinary proposition whose truth value is either zero or one. To illustrate, consider the fuzzy relation $Likes$ in Figure A.4, which establishes a fuzzy association between $person$ and $movie$ attributes using the fuzzy proposition “$person x$ likes movie $y$”. According to relation $Likes$, the truth value for the fuzzy proposition “Bob likes the Godfather” is 0.8.

<table>
<thead>
<tr>
<th>person</th>
<th>movie</th>
<th>$\Phi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Heat</td>
<td>0.7</td>
</tr>
<tr>
<td>Bob</td>
<td>The Godfather</td>
<td>0.8</td>
</tr>
<tr>
<td>Mary</td>
<td>Inception</td>
<td>0.5</td>
</tr>
<tr>
<td>Alice</td>
<td>The Truman Show</td>
<td>0.9</td>
</tr>
</tbody>
</table>

Figure A.4: The example fuzzy relation $Likes$.

The domain of attributes in the $Likes$ fuzzy relation are ordinary sets. However, it is possible to have fuzzy relations of attributes with fuzzy domains. For instance, consider the fuzzy relation $R$ in Figure A.5 which lists movies that are popular and have good screenplays. In this relation, the domain of attribute $movie$ is an ordinary set, but those of $popular$ and $good-screenplay$ are fuzzy sets with the following membership functions:

$$
\Phi_{popular}(x) = \frac{x}{10}, \quad x \in \{1, \ldots, 10\}
$$

$$
\Phi_{good-screenplay}(y) = \left(\frac{y}{5}\right)^2, \quad y \in \{1, \ldots, 5\}
$$

The fuzzy relation $R$ can be interpreted as the truth value for the fuzzy proposition “$movie m$ is popular and has a good screenplay”. For example, according to $R$, the truth value for the fuzzy proposition “The Godfather is popular and has a good screenplay” is 0.64.

<table>
<thead>
<tr>
<th>movie</th>
<th>popular</th>
<th>good-screenplay</th>
<th>$\Phi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Godfather</td>
<td>9</td>
<td>4</td>
<td>0.64</td>
</tr>
<tr>
<td>Heat</td>
<td>8</td>
<td>3</td>
<td>0.36</td>
</tr>
<tr>
<td>In Time</td>
<td>6</td>
<td>4</td>
<td>0.6</td>
</tr>
<tr>
<td>Rain Man</td>
<td>8</td>
<td>5</td>
<td>0.8</td>
</tr>
</tbody>
</table>

Figure A.5: The example fuzzy relation $R$. 


A large number of other fuzzy data models has been proposed in the literature. A discussion of these models is however out of the scope of this thesis, and the interested reader is referred to [36, 37, 35, 128, 129, 105, 106].

### A.3 Probabilistic Relational Models

Modeling uncertainty using incomplete data models is not enough for many applications, where, for instance, we need to represent the likelihood of possible worlds or the confidence that we have in a piece of information. This need has been addressed by introducing the probabilistic data models. In these models, an uncertain database, referred to as a probabilistic database, represents a probability distribution over a set of possible worlds.

ProbView [83] is one of the early probabilistic database systems. In ProbView, each uncertain attribute is represented using a random variable over a set of finite values, and independence is assumed between different random variables. For instance, probabilistic database $D$, shown in Figure 2.1(a), uses ProbView’s data model.

Another early probabilistic data model is that of Fuhr et al. [63]. Fuhr’s model manages uncertainty at the tuple level. In this model, each tuple is associated with a probability value indicating the likelihood of its occurrence in the database, and the occurrence of tuples are assumed to be independent from each other. The Fuhr’s model has been recently extended to the x-relation data model (see Section 2.1.2).

There are probabilistic models which are a direct extension of an incomplete data model. Consider, for instance, probabilistic c-tables [72], where each variable is associated with a discrete probability distribution. As another example, world set decompositions [13] have a probabilistic version in which probabilities can be attached to components.

In recent years, a number of probabilistic relational database systems have been developed. The aim of these database systems is to provide built-in support for probabilistic data as first-class citizens, and efficient implementation of uncertain data management concepts and algorithms. These database systems include MayBMS [3], ORION [4], and Trio [8]. MayBMS has been built on top of the PostgreSQL and implements the world set decompositions model. ORION supports both attribute and tuple level uncertainty with arbitrary correlations, and can represent attributes by both discrete and continuous probability distributions. ORION’s underlying data model is closed under basic relational algebra operations. Trio implements the x-relation model, and supports data provenance in order to trace the origin of query results.
A.4 Probabilistic Graphical Models

Some uncertain data models combine the relational model with a Probabilistic Graphical Model (PGM for short). In this way, they can capture complex correlations both at tuple and attribute levels, and use well-studied probabilistic reasoning techniques for query evaluation over the uncertain database. BayesStore [1] and PrDB [7] are two examples of these models. In this section, we present the basics of PGMs.

The aim of a PGM is to efficiently represent and process a joint probability distribution over a set of random variables. To achieve these goals, PGMs extensively rely on exploiting conditional independencies among random variables to compactly encode their joint distribution. In general, a PGM is composed of two components: a graph, and a set functions, also called factors, each over a subset of random variables. The nodes of the graph are random variables, while there exist an edge between the variables that directly interact with each other, and variables with no edge between them are conditionally independent. Depending on the type of the edges between random variables, we can distinguish between two classes of PGMs: directed models, also known as Bayesian networks, and undirected models, also known as Markov networks.

A directed PGM uses a directed acyclic graph to encode three types of dependency between random variables, i.e. dependent, independent, and conditionally independent. In a directed PGM, each variable is dependent to its parents; connected variables (except through an edge) are conditionally independent; and disconnected variables are independent. Each variable $X_i$ is associated with a probability distribution function, denoted by $P(X_i \mid \text{parents}(X_i))$, which shows the distribution of probabilities for the values of $X_i$ given the values of its parents. In general, the joint probability distribution that is encoded by a directed PGM is as follows:

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid \text{parents}(X_i)),$$

where $X_1, \ldots, X_n$ are random variables.

Let us illustrate directed PGMs using an example from [55]. Figure A.6 shows a bayesian network for modeling the location, degree, age, experience, and salary of a person. Typically, location is independent of all other variables. That is why location is not connected to any other attribute. Age affects experience, and salary is affected by both experience and degree. Although age affects salary, its effect is indirect through experience, meaning that for salary we do not need the age of a person when his experience is known. Figure A.6 also shows the probability functions, each associated to a variable, and the joint probability distribution that is modeled by the network.

In the undirected PGM, an undirected graph is used to encode the dependency between variables. Let $G$ denote the undirected graph over random variables...
Appendix A. Uncertain Data Models

<table>
<thead>
<tr>
<th>Location</th>
<th>P(L)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NY</td>
<td>0.1</td>
</tr>
<tr>
<td>CA</td>
<td>0.2</td>
</tr>
<tr>
<td>FL</td>
<td>0.4</td>
</tr>
<tr>
<td>Other</td>
<td>0.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Degree</th>
<th>P(D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSc</td>
<td>0.6</td>
</tr>
<tr>
<td>MSc</td>
<td>0.2</td>
</tr>
<tr>
<td>PhD</td>
<td>0.1</td>
</tr>
<tr>
<td>Other</td>
<td>0.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Experience</th>
<th>P(E)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-10</td>
<td>0.1</td>
</tr>
<tr>
<td>0-11</td>
<td>0.1</td>
</tr>
<tr>
<td>0-12</td>
<td>0.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Salary</th>
<th>P(S)</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; $50K</td>
<td>0.2</td>
</tr>
</tbody>
</table>

\[ P(L, D, A, E, S) = P(L)P(A)P(D)P(E \mid A)P(S \mid D, E) \]

Figure A.6: An example Bayesian network (directed PGM)

\[ P(X_1, \ldots, X_n) = \frac{1}{Z} \prod_{g \in C} F_g(X_g), \]

where for each subgraph \( g \in C \), \( F_g \) is a probability function over the variables in \( g \), denoted by \( X_g \), and \( Z = \sum_X \prod_{g \in C} F_g(X_g) \) is the normalization constant.

To illustrate, consider the example of undirected PGM in Figure A.7, which uses the same variables as our directed PGM example. The complete subgraphs are \( \{\text{Location}\}, \{\text{Age, Experience}\}, \) and \( \{\text{Degree, Experience, Salary}\} \), and probability functions are defined on the variables in these subgraphs.

The encoded dependencies by an undirected PGM are as follows: Considering \( X, Y, \) and \( Z \) as subsets of variables, then, \( X \) and \( Z \) are conditionally independent given \( Y \) if \( X \) is separated from \( Z \) by \( Y \), where separation, here, means that each path from a variable in \( X \) to a variable in \( Z \) includes a variable from \( Y \).

In general, directed PGMs are more intuitive and easier to use than undirected PGMs.
\[ P(L, D, A, E, S) = \frac{1}{2} F_1(L) F_2(A, E) F_3(D, E, S) \]

Figure A.7: An example Markov network (undirected PGM)