Managing Uncertainty: The Road Towards Better Data Interoperability

Verwaltung von Unsicherheit: Der Weg zu besserer Interoperabilität

Maurice van Keulen, University of Twente, Enschede, The Netherlands

Summary Data interoperability encompasses the many data management activities needed for effective information management in anyone's or any organization's everyday work such as data cleaning, coupling, fusion, mapping, and information extraction. It is our conviction that a significant amount of money and time in IT that is devoted to these activities, is about dealing with one problem: "semantic uncertainty". Sometimes data is subjective, incomplete, not current, or incorrect, sometimes it can be interpreted in different ways, etc. In our opinion, clean correct data is only a special case, hence data management technology should treat data quality problems as a fact of life, not as something to be repaired afterwards. Recent approaches treat uncertainty as an additional source of information which should be preserved to reduce its impact. We believe that the road towards better data interoperability, is to be found in teaching our data processing tools and systems about all forms of doubt and how to live with them. In this paper, we show for several data interoperability use cases (deduplication, data coupling/fusion, and information extraction) how to formally model the associated data quality problems as semantic uncertainty. Furthermore, we provide an argument why our approach leads to better data interoperability in terms of natural problem exposure and risk assessment, more robustness and automation, reduced development costs, and potential for natural and effective feedback loops leveraging human attention.

►►► **Zusammenfassung** Dateninteroperabilität umfasst die zahlreichen Datenverwaltungsaktivitäten, die für effektives Informationsmanagement nötig sind, z. B. Datenreinigung, Kopplung, Fusion, Mapping oder Informationsextraktion. Wir beobachten, dass ein erheblicher Anteil monetärer und zeitlicher Ressourcen in der IT, die auf diese Bereiche entfallen, der Lösung eines einzelnen Problems gewidmet werden: der „semantischen Unsicherheit". Manchmal sind Daten subjektiv, unvollständig, nicht aktuell oder nicht korrekt, manchmal können sie unterschiedlich interpretiert werden, usw. Wir sind der Meinung, dass saubere und korrekte Daten nur einen Spezialfall von Daten darstellen und so sollten Datenmanagementtechnologien Datenqualitätsprobleme als eine Tatsache behandeln statt diese im Nachhinein zu reparieren. Neuere Ansätze betrachten Unsicherheit als eine zusätzliche Informationsquelle, die erhalten werden sollte, um Auswirkungen der Unsicherheit zu reduzieren. Wir glauben, dass der Weg zu einer besseren Interoperabilität von Daten darin besteht, unseren Werkzeugen und Systemen zur Datenverwaltung beizubringen, welche Formen der Unsicherheit es gibt und wie man diese handhabt. In diesem Beitrag zeigen wir für mehrere Fallbeispiele der Dateninteroperabilität (Deduplizierung, Datenkopplung/Fusion und Informationsextraktion), wie die entsprechenden Datenqualitätsprobleme als semantische Unischerheit modelliert werden können. Desweiteren motivieren wir, warum unser Ansatz zu einer besseren Interoperabilität bezüglich Risikobeurteilung, Robustheit und Automatisierung, Entwicklungskosten und Potenzial für effektive Feedbackschleifen unter Nutzung menschlicher Interaktion führt.

Keywords D.2.12 [Software: Software Engineering: Interoperability]; H.1.1 [Information Systems: Models and Principles: Systems and Information Theory]; H.2.5 [Information Systems: Database Management: Heterogeneous Databases] ►►►
Schlagwörter Interoperabilität, Systeme und Informartionstheorie, heterogene Datenbanken

1 Introduction

Data interoperability encompasses the many data management activities needed for effective information management in any organization's everyday work. Examples of such activities are data cleaning, coupling, fusion, evolution, migration, but also schema matching and mapping, information extraction from un/semi-structured text, etc. A significant amount of money and time in IT is devoted to these activities. For example, web shops are confronted with customers creating a new account while they already had one. Hence its customer database needs to be deduplicated regularly, ordinarily requiring, even with tool-assistance, significant amounts of mundane manual data checking. The result is inherently unclean, even with human inspection, because humans make mistakes as well and customers may move or die without informing the company.

Effectively dealing with data quality (DQ) in data interoperability activities becomes a more important problem in many areas. Enterprises often have quality assurance processes involving data auditing and profiling. In data warehousing part of the 'T' in ETL is commonly devoted to data cleaning. Many methods for effective DQ management have been proposed, such as Master Data Management (MDM). Also in science, data interoperability is increasingly problematic. Data-intensive science is considered to be a new, fourth paradigm for scientific exploration [7] where scientists re-use and combine data of others to answer previously unforeseen research questions. One activity is emphasized to be of utmost importance: curation, the active and on-going management of data through its life cycle of interest and usefulness, including quality maintenance, enrichment with metadata, trust management, etc. More examples can be given that not only indicate the importance of DQ, but also the struggle of enterprises in controlling it.

It is our conviction that most of the money and time in data interoperability activities is devoted to dealing with one problem: *semantic uncertainty*. A deduplication algorithm can never always perfectly determine whether or not two records refer to the same customer; even we humans can not perfectly do this without sometimes explicitly asking them. Data may be subjective, not current, open for multiple interpretations, etc. We have observed that current technology supports users in manipulating huge volumes of data, but that it does not support users well in handling data with problems.

In our opinion, clean correct data is only a special case, so technology should fundamentally treat data as partially flawed; And it should treat DQ problems as a fact of life, not as something to be repaired afterwards; or as [10] puts it: "*While traditional data integration methods more or less explicitly consider uncertainty as a problem, as something to be avoided, recent approaches treat uncertainty as an additional source of information, something that is precious and that should be preserved […] One of the objectives of data quality processing is to reduce the amount and impact of imperfect data*".

By modeling DQ problems as semantic uncertainty and managing this uncertainty explicitly, we believe there is much potential for better support for handling data with problems. Therefore, we believe that the road towards better data interoperability, is to be found in teaching our data processing tools and systems about all forms of doubt and how to live with them.

In Sect. 2, we provide a method for formalizing DQ problems as uncertainty in the data. Section 3 investigates three data interoperability problems: deduplication, data coupling/fusion, and information extraction. Section 5 argues why our approach of modeling DQ problems as semantic uncertainty leads to better interoperability in terms of natural problem exposure and risk assessment, more robustness and automation, reduced development costs, and potential for natural and effective feedback loops leveraging human attention.

2 A Model of Uncertain Data

In this section, we provide a formalization of a probabilistic database. We have chosen this particular formalization, because it is rather close to the data models of concrete probabilistic databases while being abstract enough to be sensible in the context of other data models and models of uncertainty (see Sect. 4). Table 1 gives an overview of our notation.

Table 1 Notation.

var	universe	description
d	$\mathcal{D}$	data item
D	$\mathcal{DB}$	database/possible world
o	$\mathcal{RW}$	real-world object
ω	$\mathrm{D} \to \mathcal{RW}$	reference relation between data items and the real world
r	$\mathcal{R}$	random variable
R	$\mathbb{P}\mathcal{R}$	set of random variables
v	$\mathcal{V} = \mathbb{N}$	value
W	$\mathcal{R} \rightsquigarrow \mathcal{V}$ $\rightsquigarrow [0..1]$	world set
$(r \mapsto v)$		random variable assignment
θ_W	$\mathcal{R} \rightsquigarrow \mathcal{V}$	valuation for W
Θ_W $(\bar{\Theta}_W)$		all possible (total) valuations for W
$\widetilde{\mathrm{D}}$	$\mathcal{PDB}$	probabilistic database
φ_W	$\mathcal{WSD}_W$	world set descriptor (wsd)
$\varphi(\theta)$		evaluation of φ under θ
$\bar{\mathrm{D}}$		compact probabilistic database
$\dot{\mathrm{D}}$	$\mathbb{P}\mathcal{D} \times \mathcal{WSD}_W$	set of alternative data items each with its wsd

2.1 Preliminaries

We model a *database* $\mathrm{D} \in \mathcal{DB}$ in an abstract way as a set of *data items* $\mathcal{DB} = \mathbb{P}\mathcal{D}$. Typically, a data item $d \in \mathcal{D}$ would be

- a tuple for a relational database,
- an XML element for an XML database, and
- a triple for an RDF store.

To be able to talk about semantic issues, we model real-world objects $o \in \mathcal{RW}$ and that data items *refer* to them with a reference relation ω, which of course are both unavailable to a computer and its algorithms.

2.2 Example of Data with a Problem: Duplicates

Figure 1 shows an example database and a part of the real world it is supposed to represent. It belongs to a company selling car components and contains sales for car brands. A preferred customer program is meant to avoid important customers switching to a competitor. A preferred customer is one with sales over 100.

Observe that looking only at the data, at first glance there is no preferred customer. But the database has a problem with *duplicates*. The "same" car brand may occur more than once under different names. A common cause is a merge of data from autonomous sources using their own conventions and standards. Although there are 6 data items $d_1 \ldots d_6$, it contains data on only 3 car brands o_1, o_2, o_4. Data items d_i and d_j are duplicates iff $i \neq j \wedge \omega(d_i) = \omega(d_j)$, e. g., d_3 and d_6 refer to the same car brand and their combined sales is 106, so 'Mercedes-Benz' is a preferred customer after all.

Typical data cleaning solutions support duplicate removal. Based on advanced string similarity and other techniques, data items are automatically matched and clustered. Each cluster contains data items that most likely all refer to the same real-world object. These data items are then merged or *fused* into one data item. This technology would readily be able to find duplicates d_1 and d_5 and fuse them into a new data item d_{15}; $d_3, d_6 \mapsto d_{36}$ analogously. But, it is quite possible that an algorithm would not detect that also d_2 refers to 'BMW'. Note that this seemingly small technical glitch has a profound business consequence, namely it determines if 'BMW' is considered a preferred customer or not, risking loosing them to a competitor which the preferred customer program is designed to prevent.

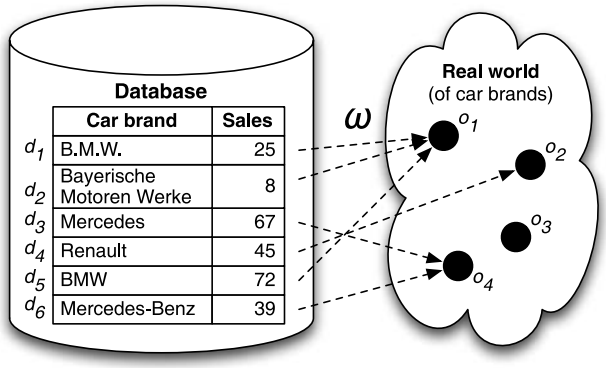

Figure 1 The relationship between data items in a database and the real world.

Be honest, did you as a human know that 'BMW' stands for 'Bayerische Motoren Werke'? This is semantic knowledge. What do we as humans do if we do not know but suspect that this is the case? *We are in doubt*; in the terminology of this paper, we are uncertain about the semantics of 'BMW'. Consequently, we simply consider both cases, reason that in case d_2 also refers to 'BMW', then its sales are over 100; hence we decide that 'BMW' *might* be a preferred customer. Moreover, we decide that it is important and likely enough to act upon. It is this behavior of 'doubting' and 'probability and risk assessment' that we propose to mimic.

2.3 Modeling Semantic Uncertainty as Random Events

We propose to model the uncertainty about whether or not $\omega(d_i) = \omega(d_j)$ as a random event, i. e., we introduce a random variable $r \in \mathcal{R}$ with two possible assignments $(r \mapsto 0)$ representing $\omega(d_i) \neq \omega(d_j)$ and $(r \mapsto 1)$ representing $\omega(d_i) = \omega(d_j)$. Observe that the uncertainty around 'BMW' and 'Mercedes' are independent of each other. Therefore, they can be modelled with two different random variables. We call the collection of all possible random variable assignments (rvas for short) with their probabilities our *world set* $W \in \mathcal{R} \rightsquigarrow \mathcal{V} \rightsquigarrow [0\,..\,1]$. We denote with $\mathrm{P}(r \mapsto v) = W(r)(v)$ the probability of a rva. It should hold that $\forall r \in \mathrm{dom}(W)\colon \sum_{v \in \mathrm{dom}(W(r))} \mathrm{P}(r \mapsto v) = 1$.

'Considering a case' means that we choose a value for one or more random variables and reason about the consequences of this choice. We call such a choice a *valuation* θ_W where $\forall r \in \mathrm{dom}(\theta_W)\colon \theta_W(r) \in \mathrm{rng}(W(r))$. If the choice involves all the variables of the world set, the valuation is total. Θ_W and $\bar{\Theta}_W$ are the sets of all possible valuations resp. total valuations for W. Random variables are independent, hence the *probability of a valuation* $\mathrm{P}(\theta_W) = \prod_{(r \mapsto v) \in \theta_W} \mathrm{P}(r \mapsto v)$. We often omit the W in θ_W if it is clear from the context.

2.4 Probabilistic Databases

A *probabilistic database* $\widetilde{\mathrm{D}} \in \mathcal{PDB}$ is a database capable of handling huge volumes of data items and possible alternatives for these data items while still being able to efficiently query and update. As a foundation for an intuitive and simple semantics, *possible world theory* views a probabilistic database as a set of possible databases each with a likelihood, i. e., $\mathcal{PDB} = \{\widetilde{\mathrm{D}} \in \mathbb{P}\mathcal{DB} \times [0\,..\,1] \mid \sum_{(\mathrm{D},p) \in \widetilde{\mathrm{D}}} p = 1\}$. D is called a *possible world* with probability $\mathrm{P}(\mathrm{D})$ iff $(\mathrm{D}, \mathrm{P}(\mathrm{D})) \in \widetilde{\mathrm{D}}$.

Obviously, an implementation would not store the possible worlds individually, but as a compact representation capable of representing vast numbers of possible worlds in limited space. Possible world theory prescribes

that a query Q on a compact representation should result in a compact answer representing all possible answers (equal to evaluating Q in each world individually).

The compact representation is based on linking alternative data items and the world set by means of world set descriptors (wsd) $\varphi_W \in \mathcal{WSD}_W$. It is a conjunction [1] of rvas $(r_i \mapsto v_i)$ such that $\forall i : v_i \in \mathrm{rng}(W(r_i))$. The wsd determines which data items exist for which rvas. The compact representation $\bar{\mathrm{D}} = (\dot{\mathrm{D}}, W)$ is a set of data items each with a wsd and a world set, where $\dot{\mathrm{D}} \in \mathbb{P}\mathcal{D} \times \mathcal{WSD}_W$. A wsd φ can be *evaluated under a valuation* θ denoted by $\varphi(\theta)$:

$$\begin{cases} \mathit{false} & \text{if } \exists i,j : i \neq j \wedge r_i = r_j \wedge v_i \neq v_j \\ \mathit{true} & \text{if } \forall i : (r_i \mapsto v_i) \in \theta \\ \mathit{false} & \text{otherwise} \end{cases}$$

$\mathrm{P}(\varphi)$ is defined as the sum of all valuations under which it is true: $\mathrm{P}(\varphi) = \sum_{\theta \in \bar{\Theta}_W \wedge \varphi(\theta)} \mathrm{P}(\theta)$, using independence $\mathrm{P}(\varphi) = \prod_{(r_i \mapsto v_i) \in \varphi} \mathrm{P}(r_i \mapsto v_i)$.

A total valuation *induces* one possible world in a compact probabilistic database: $\theta(\bar{\mathrm{D}}) = \{d \mid (d, \varphi) \in \dot{\mathrm{D}} \wedge \varphi(\theta)\}$. In this way, the concept of valuation bridges the gap between the compact representation and possible world theory. The set of all possible worlds $\mathrm{PWS}(\bar{\mathrm{D}}) = \{\mathrm{D} \mid \theta \in \bar{\Theta}_W \wedge \mathrm{D} = \theta(\bar{\mathrm{D}})\}$. The probability of a world D is $\mathrm{P}(\mathrm{D}) = \sum_{\theta \in \bar{\Theta}_W \wedge \mathrm{D} = \theta(\bar{\mathrm{D}})} \mathrm{P}(\theta)$.

Figure 2 illustrates the given concepts for our example. We distinguish three possible cases for the possibility of duplicates among d_1, d_2, and d_5; and two cases for d_3 and d_6. We have introduced random variables r_1 and r_2 for both respectively. There are $3 \cdot 2 = 6$ possible valuations, hence 6 possible worlds. A choice for either $(r_1 \mapsto 0)$ or $(r_1 \mapsto 1)$ has no effect on the answer to Q, so there are ultimately 4 different possible answers.

Typical data cleaners only produce a best effort result, which can be observed here as the most probable world D_5. We already determined that this is not the correct world. Typically a business analyst would first ask for the most likely answer which would allow him/her to work almost completely in the same way as with an ordinary database. The difference now is that despite the unreliability of the data cleaner, he will not make a costly mistake, because he is made aware that the answer may not be 106 but 211.

$\dot{\mathrm{D}}$

	car	sales	φ
d_1	...	25	$(r_1 \mapsto 0)$
d_2	...	8	$(r_1 \mapsto 0)$
d_5	...	72	$(r_1 \mapsto 0)$
d_{15}	...	97	$(r_1 \mapsto 1)$
d_2	...	8	$(r_1 \mapsto 1)$
d_{125}	...	105	$(r_1 \mapsto 2)$
d_4	...	45	
d_3	...	67	$(r_2 \mapsto 0)$
d_6	...	39	$(r_2 \mapsto 0)$
d_{36}	...	106	$(r_2 \mapsto 1)$

W

rva	P	
$(r_1 \mapsto 0)$	0.1	'd_1, d_2, d_5 different'
$(r_1 \mapsto 1)$	0.6	'd_1, d_5 same'
$(r_1 \mapsto 2)$	0.3	'd_1, d_2, d_5 same'
$(r_2 \mapsto 0)$	0.2	'd_3, d_6 different'
$(r_2 \mapsto 1)$	0.8	'd_3, d_6 same'

$Q =$
```
SELECT SUM(sales)
FROM carsales
WHERE sales≥ 100
```
'sales of preferred customers'

All possible worlds with their answer to Q

	θ	$\theta(\mathrm{D})$	$\mathrm{P}(\theta)$	Q_θ
D_1	$\{(r_1 \mapsto 0), (r_2 \mapsto 0)\}$	$\{d_1, d_2, d_3, d_4, d_5, d_6\}$	$0.1 \cdot 0.2 = 0.02$	0
D_2	$\{(r_1 \mapsto 1), (r_2 \mapsto 0)\}$	$\{d_{15}, d_2, d_3, d_4, d_6\}$	$0.6 \cdot 0.2 = 0.12$	0
D_3	$\{(r_1 \mapsto 2), (r_2 \mapsto 0)\}$	$\{d_{125}, d_3, d_4, d_6\}$	$0.3 \cdot 0.2 = 0.06$	105
D_4	$\{(r_1 \mapsto 0), (r_2 \mapsto 1)\}$	$\{d_1, d_2, d_{36}, d_4, d_5\}$	$0.1 \cdot 0.8 = 0.08$	106
D_5	$\{(r_1 \mapsto 1), (r_2 \mapsto 1)\}$	$\{d_{15}, d_2, d_{36}, d_4\}$	$0.6 \cdot 0.8 = 0.48$	106
D_6	$\{(r_1 \mapsto 2), (r_2 \mapsto 1)\}$	$\{d_{125}, d_{36}, d_4\}$	$0.3 \cdot 0.8 = 0.24$	211

Possible answers

sum(sales)	P
0	0.14
105	0.06
106	0.56
211	0.24

Other derivable figures

description	sum(sales)	P
Minimum	0	0.14
Maximum	211	0.24
Answer most likely world	106	0.48
Most likely answer	106	0.56
Second most likely answer	211	0.24
Expected value	116.3	*N.A.*

Figure 2 Example of a probabilistic database (resulting from indeterministic deduplication of Figure 1) with a typical query and its answer.

[1] Theoretically an arbitrary propositional formula with $\wedge$, $\vee$, and $\neg$ is possible, but here a simple conjunction suffices.

3 Modeling Data Interoperability

In this section, we provide a formalization of the three use cases in terms of data in a probabilistic database. Our modeling of the first data operability problem, semantic duplicates, is rather precise and thorough, while the other two, data fusion and information extraction, are presented in a more sketchy way for space reasons.

3.1 Deduplication and Data Coupling the Indeterministic Way

In the previous sections, we saw a glimpse of indeterministic deduplication, i. e., deduplication with an uncertain result. In [12] we proposed a general approach for indeterministic deduplication. It starts with the traditional steps of matching pairs of data items on their attributes and classifying them in certain matches (M), certain non-matches (U), and possible matches (P) typically involving two thresholds distinguishing between M and P, and U and P, respectively. The tuples and their pair-wise decisions can be seen as a graph which is partitioned into clusters C_i by removing the U-edges.

From here onward, [12] deviates from traditional deduplication and proposes adaptations to generate data associated with multiple possible deduplications. Analogous to finding possible worlds, the P-edges in C_i give rise to many possible *matching graphs* G_i^j by replacing each P-edge with either a M- or U-edge. Some of the matching graphs are inconsistent due to transitivity: $d_1, d_2, d_3 \in G_i^j \wedge (d_1, d_2) \in M \wedge (d_2, d_3) \in M$, then $(d_1, d_3) \in M$. We generate a compact probabilistic database $\bar{\mathrm{D}} = (\dot{\mathrm{D}}, W)$ as follows. For each cluster C_i, we introduce one random variable r_i with a different value v_i^j for each consistent matching graph: [2] $W = \{(r_i, v_i^j, p) \mid \exists G_i^j : p = \mathrm{P}(G_i^j)\}$. We refer to [12] for how to calculate the associated probabilities $\mathrm{P}(G_i^j)$. Each

[2] [12] is based on the uncertain data model of Trio, which does not have random variables. The concept of 'indicator', however, plays the role of random variable, and 'indicator table' of world set.

matching graph G_i^j represents a possible deduplication for the cluster of data items. We can generate the associated data $\dot{D}$ simply edge-by-edge by inserting both tuples d' and d'' separately if $(d', d'') \in U$ and a fused tuple $d = \mu(d', d'')$ (see Sect. 3.2) if $(d', d'') \in M$.

Example

For our example of Sect. 2.2, this approach forms 3 clusters. We consider no certain matches (M) to keep the example interesting.

$$C_1 = (\{d_1, d_2, d_5\}, \{(d_1, d_2, \mathrm{P}), (d_1, d_5, \mathrm{P}), (d_2, d_5, \mathrm{P})\})$$
$$C_2 = (\{d_4\}, \emptyset)$$
$$C_3 = (\{d_3, d_6\}, \{(d_3, d_6, \mathrm{P})\})$$

There are 8 possible matching graphs for C_1 of which only 5 are consistent. C_2 and C_3 produce only 1 and 2 matching graphs, respectively.

Figure 3 shows the result. The generic approach slightly differs from Fig. 2 in the following two ways:

(a) It generates a random variable for C_2 with only one value. Although not harmful, it is easy to avoid this.

(b) It considers two more possibilities for C_1. Although these have the lowest probabilities and one could apply a threshold to delete them, observe that even without that, they do not influence query results much, hence the result is 'good enough' as it is.

$\dot{D}$

	car	sales	φ
d_1	...	25	$(r_1 \mapsto 0)$
d_2	...	8	$(r_1 \mapsto 0)$
d_5	...	72	$(r_1 \mapsto 0)$
d_{12}	...	33	$(r_1 \mapsto 1)$
d_5	...	72	$(r_1 \mapsto 1)$
d_{15}	...	97	$(r_1 \mapsto 2)$
d_2	...	8	$(r_1 \mapsto 2)$
d_{25}	...	80	$(r_1 \mapsto 3)$
d_1	...	25	$(r_1 \mapsto 3)$
d_{125}	...	105	$(r_1 \mapsto 4)$
d_4	...	45	$(r_2 \mapsto 0)$
d_3	...	67	$(r_3 \mapsto 0)$
d_6	...	39	$(r_3 \mapsto 0)$
d_{36}	...	106	$(r_3 \mapsto 1)$

W

rva	P	
$(r_1 \mapsto 0)$	...	'd_1, d_2, d_5 different'
$(r_1 \mapsto 1)$	...	'd_1, d_2 same'
$(r_1 \mapsto 2)$	...	'd_1, d_5 same'
$(r_1 \mapsto 3)$	...	'd_2, d_5 same'
$(r_1 \mapsto 4)$	...	'd_1, d_2, d_5 same'
$(r_2 \mapsto 0)$	1.0	'd_4'
$(r_3 \mapsto 0)$	...	'd_3, d_6 different'
$(r_3 \mapsto 1)$	...	'd_3, d_6 same'

Figure 3 Result of indeterministic deduplication of Figure 1.

Data Coupling

We speak of data coupling if we have two databases D_1 and D_2 with data on the same real-world objects, and we try to find the ones that 'belong to each other', i. e., (d_1, d_2) where $d_1 \in D_1, d_2 \in D_2$ and $\omega(d_1) = \omega(d_2)$. Typically, these data items contain overlapping sets of 'attributes'. This activity is very similar to deduplication and can be handled analogously. The result is an uncertain table with data item identifier pairs. Combining the information can simply be accomplished by performing a query with a three-way join of both databases with this 'coupling' table.

3.2 Data Fusion with Conflicts

A good survey on data fusion is [4]. It describes *data fusion* as "*the process of fusing multiple records representing the same real-world object into a single, consistent, and clean representation.*" For a large part, the survey is about handling conflicts. We already saw a typical example of a conflict: if d' and d'' are duplicates, their representations should be merged into $\mu(d', d'')$; but what is the value of the "car brand" for $\mu(d_3, d_6)$ in Fig. 1? Is it "Mercedes", "Mercedez-Benz", or something else?

We view this problem as a form of uncertainty: it is uncertain which value is the correct one. For each attribute with a conflict, we introduce a new random variable r allowing storage of all possible merged representations in one probabilistic database. The approach can be combined with the indeterministic deduplication approach (see Fig. 4).

$\dot{D}$

	car	sales	φ
d_3	Mercedes	67	$(r_3 \mapsto 0)$
d_6	Mercedes-Benz	39	$(r_3 \mapsto 0)$
d_{36}	Mercedes	106	$(r_3 \mapsto 1) \wedge (r_4 \mapsto 0)$
d_{36}	Mercedes-Benz	106	$(r_3 \mapsto 1) \wedge (r_4 \mapsto 1)$

W

rva	P	
$(r_3 \mapsto 0)$	...	'd_3, d_6 different'
$(r_3 \mapsto 1)$	...	'd_3, d_6 same'
$(r_4 \mapsto 0)$	...	'Car brand value for $\mu(d_3, d_6)$ is given by d_3'
$(r_4 \mapsto 1)$	...	'Car brand value for $\mu(d_3, d_6)$ is given by d_6'

Figure 4 Fusing possible duplicates d_3 and d_6.

This conflict handling strategy is called 'CONSIDER ALL POSSIBILITIES' in [4]. Note, however, that the strategy is only an *initial* or *default* approach. It is a good basis for both immediate use and further refinement, because it is automatic as well as information and uniqueness preserving (in all possible worlds where d_3 and d_6 are duplicates, there exists exactly one data item d_{36}). The other conflict handling strategies of [4] can be implemented as subsequent operations on this default. Although for many applications, the default is 'good enough', this is not always the case. For instance, while merging tuples in our example, the 'sales' values do not conflict, but should be summed. In Sect. 5 we elaborate on the importance of striving for an automatic 'good enough' initial result containing uncertainty.

3.3 Information Extraction

Unstructured text is another major data interoperability obstacle. (Too) much data is still inaccessible for data processing, because it is textually embedded in documents, webpages, or text fields. Information extraction (IE) is a technology capable of extracting entities, facts, and relations from unstructured text. Because natural language is highly ambiguous and computers are still incapable of 'real' semantic understanding, IE is a highly

imperfect process. For example, it is ambiguous how to interpret the word "Paris": it could be a first name, a city, etc. Even resolving it to a city, note that there are over 60 other places called "Paris" besides the capital of France and that 46 % of toponyms[3] are ambiguous [6].

We sketch here how to model ambiguousness in natural language interpretation as semantic uncertainty. We base ourselves on [15]. In IE results of the various kinds of analyses are represented as *annotations*. For example, the phrase "Paris Hilton" could be annotated with an entity type 'Hotel' and its precise location. Annotations can simply be stored in a table with a begin and end point in the text. Annotations can 'overlap': there may be another annotation on "Paris".

An information extractor, however, may not be able to immediately attach these annotations to the text, because there are underlying ambiguities. To name three,

(a) The phrase may refer to a person (the media personality), the Hilton hotel in Paris, or even a fragrance.
(b) Since there are over 60 places called Paris, there may be more than one with a Hilton hotel (for illustration purposes, say that there are two).
(c) There is more than one Hilton hotel in the capital of France (there seem to be five, but let us restrict ourselves to the one near the Eiffel Tower and the one at Charles de Gaulle Airport).

All these ambiguities can be viewed as semantic uncertainty, the former regarding the entity type, the latter two regarding the location at two levels of granularity: city level determined by the annotation for phrase 1–5 and hotel level. The example ultimately leads to five possibilities for annotating "Paris Hilton" and two possibilities for annotating "Paris". In terms of our formalization, an annotation is a data item d where the different ambiguities concern different semantic choices. Figure 5 illustrates the resulting probabilistic database. Observe how the dependencies between the annotations are naturally encoded in the wsd's. Also observe that annotation ambiguity is the dual of a duplicate: with d_1, d_2 possibly being duplicates we have $\omega(d_1) \stackrel{?}{=} \omega(d_2)$, with annotation d being ambiguous we have $\exists o_1, o_2 : o_1 \stackrel{?}{=} \omega(d) \wedge o_2 \stackrel{?}{=} \omega(d)$.

$\dot{D}$

pos	entity	location	φ
1–12	Person	NULL	$(r_1 \mapsto 0)$
1–12	Hotel	N 48° 51' E 2° 18'	$(r_1 \mapsto 1) \wedge (r_2 \mapsto 0) \wedge (r_3 \mapsto 0)$
1–12	Hotel	N 49° 0' E 2° 34'	$(r_1 \mapsto 1) \wedge (r_2 \mapsto 1) \wedge (r_3 \mapsto 0)$
1–12	Hotel	...	$(r_1 \mapsto 1) \wedge (r_3 \mapsto 1)$
1–12	Fragrance	NULL	$(r_1 \mapsto 2)$
1–5	City	N 48° 51' E 2° 21'	$(r_3 \mapsto 0)$
1–5	City	...	$(r_3 \mapsto 1)$

W

rva	P	
$(r_1 \mapsto 0)$	...	'Phrase 1–12 has entity type Person'
$(r_1 \mapsto 1)$	...	'Phrase 1–12 has entity type Hotel'
$(r_1 \mapsto 2)$	...	'Phrase 1–12 has entity type Fragrance'
$(r_2 \mapsto 0)$	...	'Hotel 1–12 refers to one near Eiffel Tower'
$(r_2 \mapsto 1)$	...	'Hotel 1–12 refers to one at CdG airport'
$(r_3 \mapsto 0)$	...	'City 1–5 refers to capital of France'
$(r_3 \mapsto 1)$	...	'City 1–5 refers to the other city'

Figure 5 Annotating the phrase "Paris Hilton".

[3] A *toponym* is any name that refers to a location including, e. g., names of buildings

4 Related Work

Probabilistic Relational Databases

In recent years, the database community produced several prototypes of scalable uncertain *relational* databases, each with its own way of modelling uncertainty in the relational data model. The model that we generalized for our formalization is the U-relation [2] underlying the MayBMS system. It supports wsd's stored as a set of additional columns in an otherwise ordinary table; each random variable assignment is represented by three columns: an ID for the random variable, an ID for the value, and the probability. These columns are typically hidden to the general user, who can 'create' uncertain data with the `REPAIR KEY` statement. MayBMS supports efficient querying using a slightly extended variant of SQL.

Another well-developed model and system is the ULDB model [3] underlying the system Trio. Trio supports *x-tuples* containing alternative values and probabilities for uncertain attributes. It has a different but equivalent way of representing dependencies: alternatives contain *lineage*: propositional formulas referencing the alternatives that gave rise to them.

One more system deserves mentioning, namely MCDB [8] which is geared towards what-if analyses and risk analysis. It does not directly store uncertainty, but one specifies variable generation (VG) functions which are used to pseudorandomly generate values for uncertain attributes. By storing parameters and not probabilities, and by estimating rather than exactly computing the probability distribution over possible query answers, MCDB avoids many limitations: it handles arbitrary joint probability distributions over discrete or continuous attributes, arbitrarily complex SQL queries, and arbitrary functionals of the query-result distribution such as means, variances, and quantiles.

Other Data Models

Uncertainty in data is an orthogonal aspect of the data independent of the actual data model. The Probabilistic XML model of [14] introduces special node kinds representing possible subtrees. [1] introduces families of Probabilistic XML models with varying degrees of expressiveness. The semantic web community also addresses the issue of representing uncertainty. There is a W3C incubator group URW3-XG[4]. It has investigated use cases, methodologies, and benefits for reasoning with uncertainty. An example of an uncertain RDF model is [13].

[4] http://www.w3.org/2005/Incubator/urw3/XGR-urw3/

It supports the specification of soft rules, first-order logic rules over RDF facts which introduce weights for inferred facts, and hard rules which define mutual exclusive sets of facts.

Other Data Interoperability Problems Tackled with Explicit Management of Uncertainty

The survey [10] gives an overview of data integration approaches that use uncertainty management. An important one is schema heterogeneity, e. g., [5] views ambiguity in schema matching as uncertainty to which type an instance belongs which gives rise to probabilistic mappings producing data befitting our framework.

We like to emphasize that uncertainty management can be applied in advanced ways. To illustrate, [16] effectively detects problems in workflows handling physical objects by correlating workflow instances with sensor observations despite granularity differences, sensor malfunctions, and bypassed workflow steps or sensors.

5 Why Data Quality and Robustness are Improved

5.1 Exposure and Risk Assessment

The first important aspect of explicitly expressing semantic uncertainty in data is that it naturally exposes DQ problems in query answers, i. e., to end users. Straightforward risk assessment based on

$$Risk = Impact \times Probability$$

can easily notify high-risk problems while suppressing others. In our preferred customer example, the most likely answer 106 had a high risk problem behind it: with a significant probability of 24% the answer could be twice as large (high impact).

5.2 Robust Automation

Not only in the use of data by end users we observe a natural robustness, it also makes it easier to develop robust automatic processes for data interoperability and cleaning. Human intervention is not needed for producing an intial result, because software need only detect problems, not solve them. Wherever a semantic decision is required, software can *postpone* the decision by generating data for all possible cases.

As we have seen with data fusion and information extraction, different names for the same thing can naturally co-exist in the database. This provides robustness against the problem of conventions. For example, some gazetteers refer to Lake Como with "Lake Como", others with "Como". By storing alternative annotations, lookups do not fail because of different conventions [6].

This robustness is only achieved with 'safe' thresholds favoring a bit more recall at the expense of precision. For example, a deduplication process should not miss a duplicate to ensure that the correct answer to a query will be among the possible answers. [1] showed that DQ is rather insensitive to moderate variations in threshold settings on side of a 'safe' setting. Furthermore, more precision can be achieved later (see Sect. 5.4), but not the other way around.

5.3 Trade-off Between Development Effort and Data Quality

A rule of thumb states that 90% of the development effort in data interoperability and cleaning, is devoted to solving the 10% of hard cases. By striving for a less perfect, but near-automatic probabilistic result, they are less of a development obstacle. The result is "good enough", because it is accomodating to immediate use, notably already after 10% of the development effort.

A thorough experimental investigation of the effects and sensitivity of rule definition, threshold tuning, and user feedback on data integration quality can be found in [1]. For a sizeable use case of enriching data from a TV guide (100 movies) with IMDB (250 000 movies), we showed that the approach works for larger dirty data sets, and that indeed 85% to 92% of the cases were easy to resolve with a few simple rules and rough safe thresholds. Furthermore, subjecting the probabilistic result to a series of user feedback from a user interacting with the application showed this feedback to be effective in quickly improving the quality of the enriched data, i. e., in resolving the 10% hard cases the developer did not try to solve (see Sect. 5.4).

In effect, a developer can trade development time for DQ. He would add and refine rules until the result is considered good enough even though it still contains problems. "Good enough" is meant in terms of DQ. Note the difference between the notions of uncertainty and quality. Uncertainty is an indication of how much a system "doubts" its own data, a figure derivable from the data itself. Data quality is the degree in which the data corresponds with "the truth" (the real world). Measuring DQ typicially involves a 'ground truth' for a sample [1].

Some applications can naturally handle a probabilistic result directly, e. g., data mining. Most data mining techniques are statistical in nature, hence allow a straightforward adaptation: roughly speaking, whenever things are counted, one takes the sum of probabilities instead. The intuition is that data items only count for as much they are likely to exist or for as much the database dares to claim that they are true. Quality metrics for data mining results can be adapted analogously.

5.4 Leveraging Human Attention in Feedback Loops

The business analyst we mentioned earlier naturally investigated the situation further. This offers a powerful opportunity to allow him/her to give *feedback*, for example, that the second most likely answer 211 was actually the correct answer to *Q*. It is possible to use this naturally appearing evidence to improve the DQ by conditioning the database according to this correct query answer [9]. In fact it would resolve both issues

of our example, because 211 can only be the answer if $P(r_1 \mapsto 2) \wedge P(r_2 \mapsto 1)$. All data items in $\dot{D}$ with a φ inconsistent with $(r_1 \mapsto 2)$ and $(r_2 \mapsto 1)$ can be deleted, leaving only possible world D_6.

As mentioned earlier, we showed that user feedback was effective in quickly improving the quality of the enriched TV guide [1], hence in resolving the remaining semantic problems. We took an enriched TV guide generated under very bad threshold conditions. We subsequently ran a series of 100 consecutive feedbacks made by a simulated user who randomly picks a query answer and gives feedback according to whether or not the answer was correct. By storing data interoperability problems as semantic uncertainty, feedback can automatically be translated into combinations of random events becoming impossible; the probabilistic database can be updated accordingly. We measured DQ with expected precision and recall of the answers of 43 queries. Even with this limited form of feedback, both measures showed gradual improvement with occasional jumps.

This shows that our approach not only provides development benefits. As argued by [11] *"the only way to truly improve data quality is to increase the use of that data"*. An additional important aspect of our approach is that we created an earlier opportunity for getting the end user in the loop and we effectively deferred the resolution of the remaining 10% data interoperability problems to the end users who are more knowledgeable than a developer and their involvement can be naturally embedded in their everyday's work.

6 Conclusions

In this paper, we propose a generic approach for modeling data interoperability problems as semantic uncertainty. The essence is to determine where semantic choices are being made. If an automatic process cannot make an absolute decision on a choice with enough certainty, it should make that choice at all, but that it should enumerate several sufficiently likely cases and estimate their probabilities. Those cases can be stored simultaneously and without human intervention in a probabilistic database. In this way, the uncertainty is preserved and stored with the data as an additional source of information, fit for both immediate use as well as further refinement. We demonstrated our approach in detail for three use cases: deduplication, data coupling/fusion, and information extraction.

We furthermore provided an argument why our approach leads to better data interoperability. First, it naturally exposes DQ problems during normal use and allows for effective risk assessment. Second, it reduces costs because it provides more robustness and automation, and because one can trade DQ for development effort. Finally, it offers more potential for natural and effective feedback loops leveraging human attention for continuous improvement of DQ.

In our future research, we plan to apply our ideas to more applications to demonstrate its power. The area of information extraction has our special attention, because much valuable information is locked away in unstructured text inaccessible for automated processes. Because of its potential for continuous improvement and adaptation, we furthermore plan to focus on feedback loops, not only ones involving end users, but also involving automated processes based on machine learning.

References

[1] S. Abiteboul, B. Kimelfeld, Y. Sagiv, and P. Senellart. On the expressiveness of probabilistic XML models. In: *The Int'l Journal on Very Large Data Bases (VLDB)*, 18(5):1041–1064, 2009.

[2] L. Antova, T. Jansen, C. Koch, and D. Olteanu. Fast and simple relational processing of uncertain data. In: *Proc. of the 2008 IEEE 24th Int'l Conf. on Data Engineering (ICDE)*, pages 983–992, 2008.

[3] O. Benjelloun, A. D. Sarma, A. Halevy, and J. Widom. ULDBs: Databases with uncertainty and lineage. In: *Proc. of the 32nd Int'l Conf. on Very large Data Bases (VLDB)*, pages 953–964, 2006.

[4] J. Bleiholder and F. Naumann. Data fusion. In: *ACM Computing Surveys*, 41(1):1–41, 2009.

[5] X. Dong, A. Y. Halevy, and C. Yu. Data integration with uncertainty. In: *Proc. of the 33rd Int'l Conf. on Very large Data Bases (VLDB)*, pages 687–698, 2007.

[6] M. B. Habib and M. van Keulen. Named entity extraction and disambiguation: The reinforcement effect. In: *Proc. of the 5th Int'l Workshop on Management of Uncertain Data (MUD)*, pages 9–16, 2011.

[7] T. Hey, S. Tansley, and K. Tolle. *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research, 2009. ISBN 978-0-9825442-0-4.

[8] R. Jampani, F. Xu, M. Wu, L. L. Perez, C. Jermaine, and P. J. Haas. MCDB: a monte carlo approach to managing uncertain data. In: *Proc. of the 2008 ACM SIGMOD Int'l Conf. on Management of Data*, pages 687–700, 2008.

[9] C. Koch and D. Olteanu. Conditioning probabilistic databases. In: *Proc. of the VLDB Endowment*, 1(1):313–325, 2008.

[10] M. Magnani and D. Montesi. A survey on uncertainty management in data integration. In: *Journal of Data and Information Quality (JDIQ)*, 2(1):5:1–5:33, 2010.

[11] K. Orr. Data quality and systems theory. In: *Communications of the ACM*, 41(2):66–71, 1998.

[12] F. Panse, N. Ritter, and M. van Keulen. Indeterministic handling of uncertain decisions in duplicate detection. In: *Journal Data and Information Quality (DIQ)*, Accepted for publication, 2012.

[13] M. Theobald, M. Sozio, F. Suchanek, and N. Nakashole. URDF: Efficient reasoning in uncertain RDF knowledge bases with soft and hard rules. Technical Report MPI-I-2010-5-002, Max-Planck Institut für Informatik, Saarbrücken, 2010.

[14] M. van Keulen and A. de Keijzer. Qualitative effects of knowledge rules and user feedback in probabilistic data integration. In: *The International Journal on Very Large Data Bases (VLDB)*, 18(5):1191–1217, 2009.

[15] M. van Keulen and M. B. Habib. Handling uncertainty in information extraction. In: *Proc. of 7th Int'l Workshop on Uncertainty Reasoning for the Semantic Web (URSW)*, CEUR Workshop Proceesings, volume 778, pages 109–112, 2011.

[16] A. Wombacher. How physical objects and business workflows can be correlated. In: *Proc. of the 2011 IEEE Int'l Conf. on Services Computing (SCC)*, pages 226–233, 2011.

Received: November 21, 2011, accepted: March 22, 2012

Dr. ir. Maurice van Keulen is Associate Professor in data management technology at the University of Twente, The Netherlands. He received his PhD from this university in 1997. He was an Information Architect with the company Ordina until 1999. Since 1999 he has been affiliated with the University of Twente again. He was co-founder of the international team who developed the XML database MonetDB/XQuery. His current research interests include data interoperability, probabilistic databases, information extraction, data quality, and business intelligence. He is a member of the IFIP WG 2.6 on Databases, ACM SIGMOD, and the EUSFLAT WG on Soft Computing in Database Management and Information Retrieval (SCDMIR).

Address: Faculty of EEMCS, University of Twente, P. O. Box 217, 7500 AE Enschede, The Netherlands, Tel.: +31-53-4893688, Fax: +31-53-4892927, e-mail: m.vankeulen@utwente.nl