Indeterministic Handling of Uncertain Decisions in Duplicate Detection

Fabian Panse
University of Hamburg
Vogt-Koelln Straße 33, 22527
Hamburg, Germany
panse@informatik.uni-hamburg.de

Maurice van Keulen
Faculty of EEMCS
University of Twente
POBox 217, 7500 AE
Enschede, The Netherlands
m.vankeulen@utwente.nl

Norbert Ritter
University of Hamburg
Vogt-Koelln Straße 33, 22527
Hamburg, Germany
ritter@informatik.uni-hamburg.de

ABSTRACT

In current research, duplicate detection is usually considered as a deterministic approach in which tuples are either declared as duplicates or not. However, most often it is not completely clear whether two tuples represent the same real-world entity or not. In deterministic approaches, however, this uncertainty is ignored, which in turn can lead to false decisions. In this paper, we present an indeterministic approach for handling uncertain decisions in a duplicate detection process by using a *probabilistic* target schema. Thus, instead of deciding between multiple possible worlds, all these worlds can be modeled in the resulting data. This approach minimizes the negative impacts of false decisions. Furthermore, the duplicate detection process becomes almost fully automatic and human effort can be reduced to a large extent. Unfortunately, a full-indeterministic approach is by definition too expensive (in time as well as in storage) and hence impractical. For that reason, we additionally introduce several semi-indeterministic methods for heuristically reducing the set of indeterministically handled decisions in a meaningful way.

1. INTRODUCTION

In last decades data integration has became an important area of research [8, 15, 16, 21]. The data sets to be integrated may contain data on the same real-world entities. Often it is even the purpose of integration: to combine data on these entities. In order to integrate two or more data sets in a meaningful way, it is necessary to identify representations belonging to the same real-world entity. Therefore, duplicate detection [13] (also known as entity resolution [4], the merge-purge problem [17] or record linkage [14]) is an important component in an integration process. Due to deficiencies like missing data, typos, data obsolescence or misspellings, real-life data is often incorrect and/or incomplete. This principally hinders duplicate detection and is a crucial source of uncertainty.

In current duplicate detection approaches defined for relational data many kinds of uncertainty arising in duplicate decisions are ignored and detecting duplicates is defined as a deterministic approach, where two tuples are either declared as duplicates or not. By using *probabilistic* data models like ULDB [5] or MayBMS [19] for target schemas, however, such a determinism is not necessary. Instead any kind of uncertainty arising in the duplicate detection process can be modeled in the resulting data. This concept may protect against negative impacts resulting from false duplicate decisions. Furthermore, an expensive identification of adequate thresholds and a high number of clerical reviews can be averted.

As an example, we consider two tuples t_1 and t_2 which are duplicates (denoted as $t_1 =_{id} t_2$) with a certainty of 60%. Instead of deciding whether both tuples are duplicates or not, we can consider two possible worlds. One world in which both tuples are determined to be duplicates having a probability of 60% and one world in which both tuples are determined to represent different real-world entities, having a probability of 40%. Nevertheless, for representing the mutual exclusion of the tuples in these two worlds, representations of tuple dependencies are required. In this paper, we show in which way such tuple dependencies can be modeled with the ULDB model by using data lineage. Moreover, we present an indeterministic approach for modeling ambiguous duplicate decisions using x-relations. For reasons of generality and illustration, we use a graph-based approach to model the fundamental part of the indeterministic duplicate detection within the possible world semantics. The main contributions of this paper are:

- A full-indeterministic approach for duplicate detection based on the possible world semantics. This approach (a) minimizes the negative impact (loss of data quality) resulting from ambiguous decisions, (b) avoids human effort during the duplicate detection process — clerical reviews become unnecessary and an expensive identification of decision based configurations, e.g., thresholds, is not required anymore, and (c) enables the usage of existing and established *probabilistic* data models (e.g. ULDB), which increases the reusability of the resulting data (e.g. for further integrations).
- Several semi-indeterministic approaches which make indeterministic duplicate detection feasible in practice.
- Techniques for proper *probabilistic* interpretations of similarity values.

The paper is structured as follows. First we present current techniques of duplicate detection and tuple merging (Section 2). Then, we present *probabilistic* data models (esp. the ULDB model) and demonstrate techniques for modeling tuple dependencies in Section 4. In Section 5.1 we propose our full-indeterministic approach. Then we introduce several semi-indeterministic approaches in Section 5.2. Since indeterministic duplicate detection is based on probabilities, we discuss sources of probabilities in Section 6. Finally, we examine related work in Section 7. Section 8 concludes the paper and gives an outlook on future research.

.

2. DEDUPLICATION

Deduplication consists of two steps. First duplicates are identified (duplicate detection), and second multiple representations of one real-world entity are merged into a single one (tuple merging).

2.1 Deterministic Duplicate Detection

After data preparation [23], a duplicate detection process most often consists of five phases [3]:

1. **Search Space Reduction:** Since a comparison of all combinations of tuples is mostly too inefficient, the search space is usually reduced using heuristic methods such as the sorted neighborhood method, pruning or blocking [3].

2. **Attribute Value Matching:** Similarity of tuples is usually based on the similarity of their corresponding attribute values. Despite data preparation, syntactic as well as semantic irregularities remain. Thus, attribute value similarity is quantified by syntactic (e.g. q-grams, edit- or jaro distance [13]) and semantic (e.g. glossaries or ontologies) means. From comparing two tuples, we obtain a *comparison vector* $\vec{c} = [c_1, \ldots, c_n]$, where c_i represents the similarity of the values from the ith attribute.[1]

3. **Decision Model:** The *comparison vector* is input to a decision model which determines to which set a tuple pair (t_1, t_2) is assigned: matching tuples (M) or unmatching tuples (U). Common decision models (see [13]) are based on probability theory [14, 25], identification rules [17, 32], distance measures [20] or learning techniques [27].

Input: tuple pair (t_i, t_j), *comparison vector* $(\vec{c}_{ij} = [c_1^{ij}, \ldots, c_n^{ij}])$
1. Execution of the *combination fucntion* $\varphi(\vec{c}_{ij})$
   $\Rightarrow$ Result: $sim(t_i, t_j) \in \mathbb{R}$
2. Classification of (t_i, t_j) into $\{M, U\}$ based on $sim(t_i, t_j)$
   $\Rightarrow$ Result: $(t_i, t_j) \rightarrow \{M, U\}$

Output: Decision whether (t_i, t_j) is a duplicate or not

Figure 1: General representation of decision models

In general, the decision whether a tuple pair (t_i, t_j) is a match or not, can be decomposed into two steps (see Figure 1). In the tuple matching step (Step 1), based on the *comparison vector* a single similarity degree $sim(t_i, t_j)$ is determined by a *combination function*:

$$\varphi : [0,1]^n \rightarrow \mathbb{R} \qquad sim(t_i, t_j) = \varphi(\vec{c}_{ij}) \qquad (1)$$

In the classification step (Step 2), based on the similarity $sim(t_i, t_j)$ the tuple pair is assigned to M or U. To minimize the number of ambiguous decisions, in some approaches a third set of possibly matching tuples (P) is intermediately introduced. Each tuple pair originally classified to P is later manually assigned to M or U by domain experts (clerical reviews). Often, the classification is based on two tuple similarity thresholds T_λ and T_μ that demarcate the boundaries between the sets M, P, and U (see Figure 2).

4. **Duplicate Clustering:** A globally consistent duplicate detection is achieved from the individual decisions by using a clustering technique. The clustering's goal is to cluster all representations of a same real-world entity into one group. Simplest, clustering can be achieved by using the transitive closure of detected duplicates. More complex, but also more promising approaches are proposed in [11, 22]. Techniques of duplicate clustering can also be used during the classification step for reducing the set of possible matches and hence to reduce human effort.

5. **Verification:** The effectiveness of the applied identification is evaluated in terms of recall, precision, false negative percentage, false positive percentage and F_1-measure [3]. If the effectiveness is not satisfactory, duplicate detection is repeated with other, better suitable thresholds or methods (e.g. other comparison functions or decision models).

[1] If multiple comparison functions are used, we even obtain a matrix. Without loss of generality, we restrict ourselves to a normalized *comparison vector* ($\Rightarrow \vec{c} \in [0,1]^n$).

2.2 Tuple Merging

After detecting multiple tuples representing a same real-world entity, these various representations have to be combined into a single one. In the literature, the process of combining two or more tuples is usually denoted as tuple merging [4] or data fusion [7].

In our work, we focus on handling uncertainty in duplicate detection and abstract from merging details. In the following we assume an ideal merging function μ, where $\mu(T)$ represents the result from merging the tuples of the set T. An ideal merging function is associative. Thus, the tuple resulting from merging the tuples t_1, t_2 and t_3 is independent of the merging order ($\mu(\{\mu(\{t_1, t_2\}), t_3\}) = \mu(\{\mu(\{t_1, t_3\}), t_2\}) = \mu(\{t_1, t_2, t_3\})$). Furthermore, μ is idempotent ($\mu(\{t\}) = t$)

For reasons of clarity and comprehensibility, in following examples, the index of a merged tuple is an ordered concatenation of the indexes of the tuples it is merged from. For example, $\mu(\{t_1, t_2, t_3\})$ is denoted by t_{123}.

3. PROBLEM DESCRIPTION

The problem resulting from using a decision model as presented in Section 2.1, is illustrated in Figure 2. The greater the distance between the two thresholds T_λ and T_μ, the lower is the number of false decisions (sum of yellow areas), but the higher is the number of possible matches which have to be resolved by domain experts (red area). In general, for financial and processing-time-based reasons, clerical reviews have to be reduced to a minimum. Nevertheless, only from an effective duplicate detection data of high quality results. As a consequence, in existing approaches a trade-off between the effectiveness of the duplicate detection process and the human effort resulting from clerical reviews has to be accepted.

Such a trade-off, however, is not required if a *probabilistic* target schema is used. In this case, ambiguous decisions can be handled indeterministically and both, the number of false decisions as well as human effort, can be reduced to a large extent.

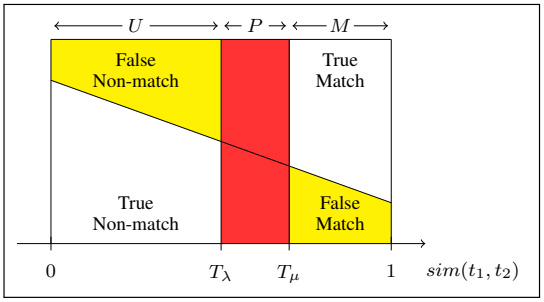


Figure 2: Trade-off between effectiveness and human effort

Furthermore, in many applications (e.g. dynamic data integration) a full-automatic duplicate detection is required ($T_\lambda = T_\mu$). By using an indeterministic approach the deduplication process can be fully automatized without accepting such a high rate of false (non-) matches as it results from a deterministic one.

Finally, the whole integration process need not become blocked because of a small amount of ambiguous matches that need clerical review. By using an indeterministic handling of *uncertain* decisions, the uncertainty of the ambiguous matches is intermediately modeled in the resulting data and can be resolved later after the integration process is finished (see the concept of good-is-good-enough integration in [12]).

4. PROBABILISTIC DATAMODELS

Theoretically, a *probabilistic* database is defined as $PDB = (W, P)$ where $W = \{I_1, \ldots, I_n\}$ is the set of possible worlds and $P : W \to (0, 1]$, $\sum_{I \in W} P(I) = 1$ is the probability distribution over these worlds. Because the data of individual worlds often considerably overlaps and it is sometimes even impossible to store them separately, a succinct representation has to be used.

In *probabilistic* relational models, uncertainty is modeled on two levels: (a) each tuple t is assigned with a probability $p(t) \in (0, 1]$ denoting the likelihood that t belongs to the corresponding relation (tuple level), and (b) alternatives for attribute values are given (attribute value level).

In earlier approaches, alternatives of different attribute values are considered to be independent (e.g. [2]). In these models, each attribute value can be considered as a separate random variable with its own probability distribution. Newer models like ULDB [1, 5, 24] or MayBMS [18, 19] support dependencies by introducing new concepts like ULDB's x-tuple and MayBMS's U-relation. As a representative for modeling uncertainty resulting from an indeterministic duplicate detection we consider the ULDB model. Nevertheless, using another model, e.g., MayBMS, is also possible.

4.1 A Model for Uncertainty and Lineage

For modeling dependencies between attribute values, in the ULDB model [5, 24] the concept of x-tuples is introduced. An x-tuple t consists of one or more alternatives $(t^1, \ldots, t^n)$ which are mutually exclusive. *Maybe* x-tuples (tuples for which non-existence is possible, i.e., for which the probability sum of the alternatives is smaller than 1) are indicated by '?'. Relations containing one or more x-tuples are called x-relations (as an example, see the x-relations $\mathcal{R}_1$ and $\mathcal{R}_2$ in Figure 3).

	name	company	$p(t)$
t_1	John	Nokia	0.7
	Johan	Oracle	0.3
t_2	Tim	Nokia	1.0
t_3	Jim	Nokia	0.3
	Jim	Sony	0.4

?

	company	location	$p(t)$
t_4	Vodafone	G*	1.0
t_5	Oracle	USA	1.0
t_6	Nokia	Finland	0.8
	Nokia	Japan	0.2
t_7	Sony	Japan	1.0

Figure 3: X-relations $\mathcal{R}_1$ (left) and $\mathcal{R}_2$ (right)

Furthermore, the ULDB model supports the concept of data lineage (also known as data provenance [9]). The lineage of a data item contains information about its derivation and can be of an internal (referring to data inside the database) as well as an external (referring to data outside the database) nature. For convenience, we restrict ourselves to internal lineage. In the ULDB model, internal lineage is considered at the granularity of x-tuple alternatives and is defined as a boolean function λ over the presence of other alternatives. Disjunctions in a lineage formula result, if the corresponding alternative can be derived from different source alternatives.

An example of internal lineage is shown in Figure 4. The relation $\mathcal{R}_3$ results from a natural join of $\mathcal{R}_1$ with $\mathcal{R}_2$ and a subsequent projection on the attributes *name* and *location*. Let (i, j) denote the jth alternative tuple of the x-tuple t_i. The lineage formula $\lambda(8, 1) = (1, 1) \wedge (6, 1)$ for the first alternative of t_8 expresses the information that this alternative is derived from the first alternatives of t_1 and t_6. The alternative t_{10}^2 results from joining t_2^1 with t_6^2 as well as from joining t_2^2 with t_7^1. Thus, the lineage formula $\lambda(10, 2)$ is a disjunction.

	name	location	$p(t)$	
t_8	John	Finland	0.56	$\lambda(8,1) = (1,1) \wedge (6,1)$
	John	Japan	0.14	$\lambda(8,2) = (1,1) \wedge (6,2)$
	Johan	USA	0.3	$\lambda(8,3) = (1,2) \wedge (5,1)$
t_9	Tim	Finland	0.8	$\lambda(9,1) = (2,1) \wedge (6,1)$
	Tim	Japan	0.2	$\lambda(9,2) = (2,1) \wedge (6,2)$
t_{10}	Jim	Finland	0.24	$\lambda(10,1) = (3,1) \wedge (6,1)$
	Jim	Japan	0.46	$\lambda(10,2) = ((3,1) \wedge (6,2)) \vee ((3,2) \wedge (7,1))$

?

Figure 4: X-relation $\mathcal{R}_3$

An interesting and useful feature of internal lineage is that the probability of a value can be computed from the probabilities of the data items in its lineage. Furthermore, an x-tuple alternative with lineage can belong to a possible world only, if its lineage formula is satisfied by the presence of the referenced alternatives in the considered world. For example, if the alternative t_8^1 is present in the possible world I_1 then alternative 1 must be chosen for tuple t_6, and hence the alternative t_9^1 must be present as well. As a consequence, lineage imposes restrictions on possible worlds. As we will see in the following section, this property can be effectively used for modeling dependencies between individual sets of tuples.

4.2 Modeling Tuple Dependencies

In the ULDB model tuple dependencies can be represented by using the concept of lineage and by creating a specific catalog relation which in the following is denoted as *tuple dependency-indicator* (short $\mathcal{I}_{td}$). For modelling the dependency $A \subseteq \mathcal{R} \leftrightarrow B \not\subseteq \mathcal{R}$ between two x-tuple sets A and B one indicator x-tuple having two alternatives (0 and 1) is required. While the x-tuples of the first set have a lineage to the first alternative of the indicator tuple, the x-tuples of the second set have a lineage to its second alternative. Since the alternatives of the indicator are mutually exclusive, this dependency holds for the two x-tuple sets, too. In general, for representing a dependency between n mutually exclusive sets, an indicator x-tuple with n alternatives is required. As mentioned before, in the ULDB model, lineage is considered on the granularity of x-tuple alternatives. However, for modeling dependencies between x-tuples, the new lineage conditions hold for the whole x-tuple and hence for all of its alternatives. For that reason, we consider lineage on tuple granularity.

In *certain* source data, if a tuple t already has a lineage, the new lineage results from the conjunction of the prior one and the new lineage condition representing the tuple dependency. As an example, we consider two *certain* tuples (x-tuples with exactly one alternative) t_1 and t_2 of a relation $\mathcal{R}$, which are duplicates with a probability of 80%. Each tuple has a prior lineage ($\lambda'(t_1)$ and $\lambda'(t_2)$) referencing one or more *certain* tuples of other relations. To model the two possible worlds resulting from the *uncertain* duplicate decision, we have to ensure that either the tuples t_1 and t_2 or the merged tuple $t_{12} = \mu(t_1, t_2)$ belong to the resulting x-relation $\mathcal{R}_X$. For representing this tuple dependency, we need an indicator x-tuple i_1 of the catalog relation $\mathcal{I}_{td}$ having two alternatives: One ($i_1^1 = 1$) having a probability of 20% and another

($i_1^2 = 2$) having a probability of 80%. By creating the lineages $\lambda(t_1) = \lambda'(t_1) \wedge (i_1, 1)$, $\lambda(t_2) = \lambda'(t_2) \wedge (i_1, 1)$ and $\lambda(t_{12}) = \lambda'(t_1) \wedge \lambda'(t_2) \wedge (i_1, 2)$, we can exclude that both x-tuple sets belong to a same possible world. Note, t_{12} is derived from t_1 and t_2. As a consequence, the lineage of t_{12} includes the prior lineages of t_1 and t_2. The probability of a tuple results from the probabilities of the alternatives referenced in its lineage. Since in our case all source tuples are *certain*, the probabilities of t_1, t_2 and t_{12} result in $p(t_1) = p(t_2) = p(i_1^1)$ and $p(t_{12}) = p(i_1^2)$.

x-tuple	lineage
t_1	$\lambda(t_1) = \lambda'(t_1) \wedge (i_1, 1)$
t_2	$\lambda(t_2) = \lambda'(t_2) \wedge (i_1, 1)$
t_{12}	$\lambda(t_{12}) = \lambda'(t_1) \wedge \lambda'(t_2) \wedge (i_1, 2)$

indicator		$p(i)$
i_1	1	0.2
	2	0.8

Figure 5: Modeling tuple dependencies in $\mathcal{R}_X$ (left) with the indicator relation $\mathcal{I}_{td}$ (right)

5. INDETERMINISTIC DUPLICATE DETECTION

In decision models as presented in Section 2, uncertainty is ignored during the classification of tuple pairs into M, U (or P) (Step 2). Such decisions, however, are not enforced, if a *probabilistic* target schema is used. In contrast, if similarity between tuples can be mapped to the probability that both tuples are duplicates (matching probability), probabilities of possible worlds can be derived. Due to the fact that no decisions are made, we denote the approach as an indeterministic duplicate detection.

As we will show in Section 5.1.4, the complexity of the computation as well as the storage requirements of a full-indeterministic approach are just too high, as that such an approach is practical. For that reason, semi-indeterministic strategies are required (see Section 5.2). Since such strategies can be seen as restrictions on the full-indeterministic approach, we present the latter first.

5.1 Full-Indeterministic Approach

In the full-indeterministic approach the decision model and the duplicate clustering phases are replaced by three other phases. Similar to the first decision model step, initially for each tuple pair a tuple matching is applied, where after similarity calculation a matching probability is determined (Phase 1). Based on the matching probabilities a set of possible worlds is derived (Phase 2). Finally, depending on the used target model, a *probabilistic* result relation representing all these worlds needs to be created (Phase 3).

5.1.1 Extended Tuple Matching (Phase 1)

In the tuple matching phase, two tuples are matched by calculating tuple similarity (Figure 6, Step 1). As known from the first decision model step (see Figure 1), the similarity of two tuples t_i and t_j results from applying a *combination function* $\varphi(\vec{c}_{ij})$.

Since matching results should be interpreted as the probability that both tuples are duplicates ($p(t_i, t_j)$), a mapping from tuple similarity to matching probability (*sim2p-mapping*) is required (Figure 6, Step 2). In the following, the function used for the *sim2p-mapping* is denoted as ρ:

$$\rho : R \to [0, 1] \qquad p(t_i, t_j) = \rho(sim(t_i, t_j)) \tag{2}$$

In approaches based on identification rules (see [17]), the similarity of two tuples is defined as the certainty that both tuples are duplicates. Thus, in these cases, tuple similarity can be directly used as matching probability. Other sources of probabilities are discussed in Section 6.

Input: tuple pair (t_i, t_j), *comparison vector* ($\vec{c}_{ij} = [c_1, \ldots, c_n]$)

1. Calculation of tuple similarity $sim(t_i, t_j) = \varphi(\vec{c}_{ij})$
   $\Rightarrow$ Result: $sim(t_i, t_j) \in \mathbb{R}$
2. Mapping from similarity to probability by $\rho(sim(t_i, t_j))$
   $\Rightarrow$ Result: $p(t_i, t_j) \in [0, 1]$

Output: Probability whether (t_i, t_j) is a duplicate

Figure 6: General representation of the tuple matching phase

5.1.2 Possible World Creation (Phase 2)

In the second phase, a set of possible worlds is derived from the matching probabilities. For reasons of representation, we define possible world creation as a graph-based process. For this purpose, we define two kinds of graphs: a *matching-graph* representing tuple matching results and *world-graphs* each representing a conceivable world.

a) Generation of an Initial Matching-graph.

A *matching-graph* is a weighted graph, where each node represents one base-tuple. Two nodes are connected with an edge, if the corresponding tuples have been matched during the duplicate detection process[2]. The weight of an edge denotes the probability that the connected tuples represent the same real-world entity.

DEFINITION 1. *A* matching-graph (M-graph) *is a triple $M = (N, E, \gamma)$ where N is a set of nodes, E is a set of edges each connecting two nodes and γ is a weighting function $\gamma : E \to [0, 1]$ denoting matching probabilities.*

In the following, an edge is called *uncertain*, if its weight is between 0 and 1 ($0 < \gamma < 1$). The set of *definite* edges ($\gamma = 1$) is denoted by E^1 and the set of *uncertain* edges is denoted by $E^?$.

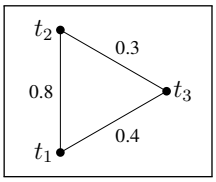


Figure 7: The *M-graph* $M = (N, E, \gamma)$ with $N = \{t_1, t_2, t_3\}$, $E = \{(t_1, t_2), (t_1, t_3), (t_2, t_3)\}$, $\gamma = \{(t_1, t_2) \to 0.8, (t_1, t_3) \to 0.4, (t_2, t_3) \to 0.3\}$

b) Generation of World-graphs.

A *world-graph* is an unweighted graph representing one conceivable world where edges denote that the associated tuples are declared to be duplicates.

DEFINITION 2. *A* world-graph (W-graph) *is a triple $G = (N, E, P)$ where N is a set of nodes, E is a set of edges each connecting two nodes and P is the probability of the corresponding world.*

Based on a given *M-graph* a set of *W-graphs* can be derived by eliminating all *uncertain* edges by either removing it or replacing it by a *certain* edge. For a full-indeterministic approach, the process of *W-graph* generation can be formalized by the mapping

[2]In processes without search space reduction each pair of nodes is connected with an edge.

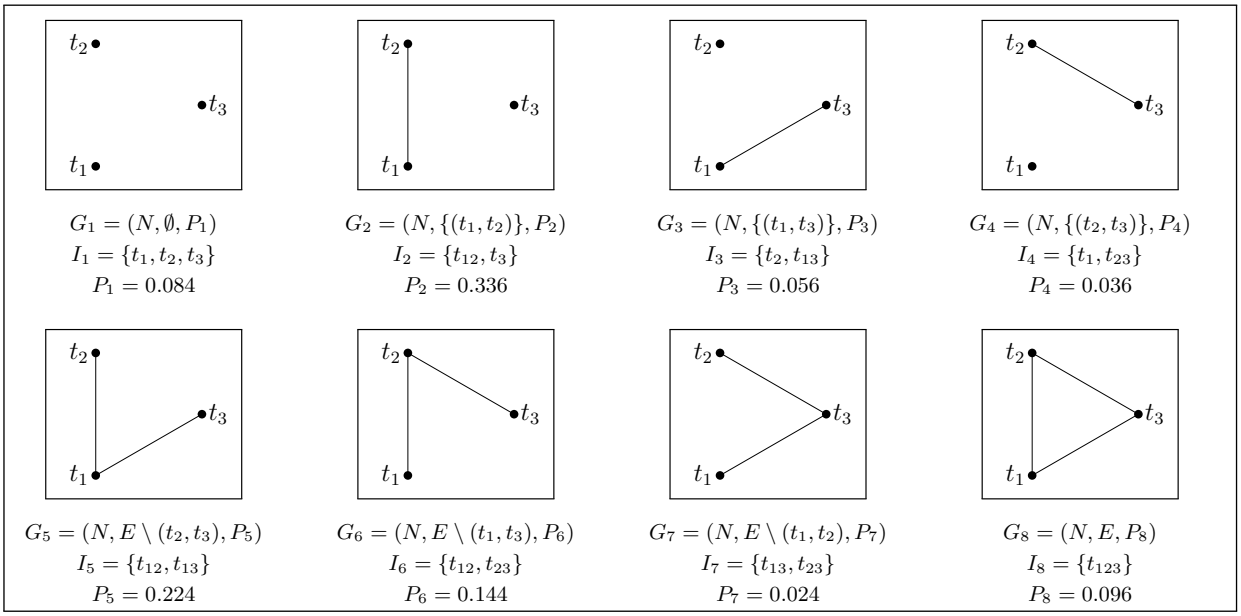


Figure 8: The worlds $I_1, \ldots, I_8$ with their corresponding *W-graphs*

$\nu : \mathcal{M} \rightarrow \mathcal{P}(\mathcal{G})$, where $\mathcal{M}$ is the set of all possible *matching-graphs* and $\mathcal{P}(\mathcal{G})$ is the power set of all possible *world-graphs*. Given an *M-graph* $M = (N, E, \gamma)$, the mapping ν is defined as:

$$\nu(M) = \bigcup_{K \in \mathcal{P}(E^?)} \{(N, E^1 \cup K, \prod_{e \in K} \gamma(e) \prod_{e \notin K} (1 - \gamma(e))\} \quad (3)$$

As an example, we consider the *M-graph* $M = (N, E, \gamma)$ representing the results of matching the three tuples t_1, t_2 and t_3 of an input relation $\mathcal{R}$ (see Figure 7). All three tuples are pairwise compared with each other and have the matching probabilities $p(t_1, t_2) = 0.8$, $p(t_1, t_3) = 0.4$ and $p(t_2, t_3) = 0.3$. Based on these probabilities eight worlds can be derived (see these worlds with their corresponding *W-graphs* in Figure 8).

c) Removing Inconsistent World-graphs.

By definition identity is a transitive relation. Worlds in which transitivity is not respected are considered impossible.

DEFINITION 3. *A world I is possible, if and only if* $(\forall t_1, t_2, t_3 \in I) : t_1 =_{id} t_2 \wedge t_1 =_{id} t_3 \Rightarrow t_2 =_{id} t_3$.

A *W-graph* is called consistent, if it represents a possible world.

THEOREM 1. *A* W-graph $G = (N, E, P)$ *is consistent, if and only if G is equivalent to its transitive closure: $G = G^*$.*

PROOF. ($\Rightarrow$) Assumption: $G \neq G^*$, but G is consistent.
$\Rightarrow (\exists t_1, t_2, t_3 \in N) : (t_1, t_2), (t_1, t_3) \in E \wedge (t_2, t_3) \notin E$
$\Rightarrow$ the world $I = \{t_1, t_2, t_3, \ldots\}$ is impossible
$\Rightarrow G$ is inconsistent $\square$

PROOF. ($\Leftarrow$) Assumption: G is inconsistent, but $G = G^*$.
$\Rightarrow$ the world $I = N$ is impossible
$\Rightarrow (\exists t_1, t_2, t_3 \in N) : (t_1, t_2), (t_1, t_3) \in E \wedge (t_2, t_3) \notin E$
$\Rightarrow G \neq G^*$ $\square$

An *M-graph* M is consistent, if at least one consistent *W-graph* can be derived from M.

THEOREM 2. *An* M-graph $M = (N, E, \gamma)$ *is consistent, if and only if* $(\forall t_1, t_2, t_3 \in N) : \gamma(t_1, t_2) = \gamma(t_1, t_3) = 1 \Rightarrow \gamma(t_2, t_3) > 0$.

PROOF. ($\Rightarrow$) Assumption: $(\exists t_1, t_2, t_3 \in N) :$
$\gamma(t_1, t_2) = \gamma(t_1, t_3) = 1 \wedge \gamma(t_2, t_3) = 0$, but M is consistent.
$\Rightarrow (\forall G{=}(N, E, P){\in}\nu(M)) : (t_1, t_2), (t_1, t_3) \in E{\wedge}(t_2, t_3) \notin E$
$\Rightarrow (\forall G = (N, E, P) \in \nu(M)) : G$ is inconsistent
$\Rightarrow M$ is inconsistent $\square$

PROOF. ($\Leftarrow$) Assumption: M is inconsistent, but
$(\forall t_1, t_2, t_3 \in N) : \gamma(t_1, t_2) = \gamma(t_1, t_3) = 1 \Rightarrow \gamma(t_2, t_3) > 0$.
$\Rightarrow (\forall G = (N, E, P) \in \nu(M)) : G$ is inconsistent
$\Rightarrow (\exists t_1, t_2, t_3 \in N) : (\forall G = (N, E, P) \in \nu(M)) :$
$(t_1, t_2), (t_1, t_3) \in E \wedge (t_2, t_3) \notin E$
$\Rightarrow (\exists t_1, t_2, t_3 {\in} N) : \gamma(t_1, t_2){=}\gamma(t_1, t_3){=}1 {\Rightarrow} \gamma(t_2, t_3){=}0$ $\square$

In the tuple matching phase each tuple pair is considered independently. Thus, worlds are created from independent considerations and hence can be impossible. Since each inconsistent *W-graph* represents an impossible world, inconsistent *W-graphs* are removed from the set of considered graphs.

We consider the example from Figure 8. Due to the transitivity of identity is violated, three ($\{I_4, I_5, I_6\}$) of the eight worlds are impossible. For instance, if t_1 and t_2 as well as t_1 and t_3 are duplicates, the tuples t_2 and t_3 also have to be duplicates. Worlds (I_5) in which this fact is not given are definitely not the true world. As a consequence, the worlds $\{I_4, I_5, I_6\}$ and hence the *W-graphs* $\{G_4, G_5, G_6\}$ have to be removed from further considerations.

After removing inconsistent *W-graphs* (impossible worlds), the probabilities of the remaining *W-graphs* (worlds) no longer sum up to 1. Therefore, the probabilities of the remaining *W-graphs* are conditioned with the event B that the true world must be a possible world (the probability of B is the overall probability of all remaining *W-graphs*). For instance, in our example, the conditioned probability of G_1 (and hence I_1) results in:

$$P(G_1 \mid B) = P(G_1)/P(B) = 0.084/0.608 = 0.138$$

d) Generating Possible Worlds.

Finally, from each *W-graph* exactly one possible world has to be derived. Since all considered *W-graphs* are consistent, each *W-graph* $G = (N, E, P)$ can be divided into m maximally connected components $\{G_1, \ldots, G_m\}$. A component with only one node represents a base-tuple that is apparently not a duplicate, hence it is included in the resulting world as it is. The tuples associated with a component consisting of multiples nodes have to be merged into one result tuple by using the merging function μ. Thus, given a component $G_i = (N_i, E_i)$ with $N_i = \{t_1, \ldots, t_k\}$, the tuple $t_{G_i} = \mu(\{t_1, \ldots, t_k\})$ is derived.

Input: Set of consistent *W-graphs* *WSet*
1. $W = \emptyset$
2. For each graph $G = (N, E, P) \in$ *WSet*
   - 2.1 $I = \emptyset$
   - 2.3 For each component $G_i = (N_i, E_i)$ $I = I \cup \{\mu(N_i)\}$
   - 2.4 $W = W \cup \{I\}$
   - 2.5 $P(I) = P$

Output: **Set of possible worlds** $W = \{I_1, I_2, \ldots, I_K\}$, **Probability distribution** $P : W \rightarrow [0, 1]$

Figure 9: Algorithm for possible world generation

An algorithm for possible world generation is shown in Figure 9. The input of the algorithm is a set of consistent *W-graphs* (*WSet*). Based on these *W-graphs*, a set of possible worlds (denoted as W) is generated (one world for each consistent *W-graph*). Considering a single *W-graph* an initially empty world (I) is defined (Step 2.1). For each of the *W-graph*'s component a tuple is added to the possible world by merging the tuples belonging to the component's nodes (Step 2.2). Finally, the resulting world is added to the set of possible worlds (Step 2.3) and its probability is defined as the probability of the corresponding *W-graph* (Step 2.4).

At the end of the possible world creation phase, duplicate tuples already have been merged. Thus, the set of possible worlds can be further reduced by checking these worlds against a set of domain depending rules based on operational data (e.g. two persons must not have the same social security number). Usually, such rules need to be extracted from domain knowledge. A similar approach is described in [33].

5.1.3 Generation of Probabilistic Data (Phase 3)

In the last phase, the created possible worlds have to be represented by a single *probabilistic* relation. Generating a single result relation, however, depends on the used target model. As already mentioned before, we use the ULDB model as a representative.

For representing the set of possible worlds in a single x-relation, an indicator tuple with $|W|$ alternatives of the relation $\mathcal{I}_{td}$ is required. The resulting x-relation $\mathcal{R}_X$ contains each tuple belonging to at least one possible world. The additional lineage of each of these tuples results in the disjunction of the indicator's alternatives representing the worlds this tuple belongs to. Finally, this lineage is conjugated with prior lineage if existing. As described in Section 4.2, the prior lineage of a merged tuple results from the conjugation of the prior lineages of the base-tuples it is merged from.

For the purpose of demonstration, we consider the example already used before. We create an indicator x-tuple i_1 with one alternative for each of the five possible worlds $\{I_1, I_2, I_3, I_4, I_8\}$ and generate the lineage as described above (prior lineage is assumed to be not existent). The resulting x-relations $\mathcal{R}_X$ and $\mathcal{I}_{td}$ are shown in Figure 10.

x-tuple	lineage
t_1	$\lambda(t_1) = (i_1, 1) \vee (i_1, 4)$
t_2	$\lambda(t_2) = (i_1, 1) \vee (i_1, 3)$
t_3	$\lambda(t_3) = (i_1, 1) \vee (i_1, 2)$
t_{12}	$\lambda(t_{12}) = (i_1, 2)$
t_{13}	$\lambda(t_{13}) = (i_1, 3)$
t_{23}	$\lambda(t_{23}) = (i_1, 4)$
t_{123}	$\lambda(t_{123}) = (i_1, 5)$

indicator		$p(i)$
i_1	1	0.138
	2	0.553
	3	0.092
	4	0.059
	5	0.158

Figure 10: X-relations $\mathcal{R}_X$ (left) and $\mathcal{I}_{td}$ (right)

A complete algorithm for x-relation generation is shown in Figure 11. The input of the algorithm is W, a set of possible worlds and P a probability distribution over these worlds. First, a new indicator tuple is created (Step 1). Second, for each possible world an alternative of the indicator tuple is generated (Step 2). Then we iterate over all possible worlds (Step 3). If a tuple of a considered world already belongs to the output x-relation $\mathcal{R}_X$, the lineage and probability of this tuple is adapted. Otherwise, the tuple is inserted into $\mathcal{R}_X$ (Step 3.1). Finally (Step 4), prior lineage is taken into account. For merged tuples, we consider the prior lineage generation as a part of the tuple merging step.

Input: Set of possible worlds $W = \{I_1, I_2, \ldots, I_k\}$, **Probability distribution** $P : W \rightarrow [0, 1]$
1. Create an indicator tuple $i \in \mathcal{I}_{td}$
2. For each world $I_j \in W$
   - 2.1 Create the alternative i^j with probability $p(i^j) = P(I_j)$
3. For each world $I_j \in W$
   - 3.1 For each tuple $t \in I_j$
     - If $t \in \mathcal{R}_X$
       - $\lambda(t) = \lambda(t) \vee (i, j)$
       - $p(t) = p(t) + P(I_j)$
     - Else
       - $\mathcal{R}_X = \mathcal{R}_X \cup t$
       - $\lambda(t) = (i, j)$
       - $p(t) = P(I_j)$
4. For each tuple $t \in \mathcal{R}_X$
   - 4.1 $\lambda(t) = \lambda'(t) \wedge \lambda(t)$

Output: X-relation $\mathcal{R}_X$

Figure 11: Algortihm for x-relation generation

5.1.4 Complexity

As known from other techniques based on the possible world semantics, the complexity of a full-indeterministic approach theoretically can be tremendous. The number of *W-graphs* which can be generated from an *M-graph* with k *uncertain* edges is:

$$N_{\textit{W-graph}}(k) = 2^k$$

Given a fully connected *M-graph* with n nodes, its number of edges is $k = (n(n-1))/2$. As a consequence, given a source relation with n tuples, the maximal number of resulting *W-graphs* is:

$$N_{\textit{W-graph}}(max) = 2^{(n(n-1))/2}$$

The number of consistent possible worlds resulting from a *certain* source relation with n tuples, where each tuple matching is *uncertain*, is equal to the number of possible partitions of the relation's tuples. Thus, the maximal number of resulting possible worlds

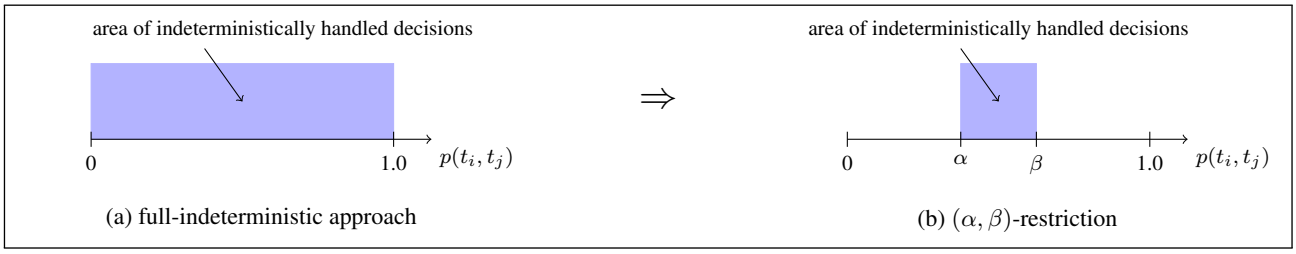


Figure 12: Reduction of the area of indeterministically handled decisions using an (α, β)-restriction

can be reduced to the complexity of set partitioning and results in:

$$N_{PW}(max) = B_n = \frac{1}{e}\sum_{k=1}^{\infty}\frac{k^n}{k!}$$

where B_n is the nth bell number [28].

If each tuple matching is *uncertain*, the resulting x-relation can be mapped to the power set of the source relation without the empty set. Thus, in the worst case, the number of resulting x-tuples is:

$$N_{|\mathcal{R}_X|}(max) = |\mathcal{P}(|\mathcal{R}|)| - 1 = 2^{|\mathcal{R}|} - 1$$

In order to get an idea of the dramatic complexity scale, we assume a source relation $\mathcal{R}$ with 10 tuples. The number of *W-graphs* which can be generated from the initial *M-graph* is maximal:

$$N_{W\text{-}graph}(max) = 2^{45} \simeq 3.5184 \cdot 10^{13}$$

The number of resulting possible worlds and hence the number of indicator alternatives is at most:

$$N_{PW}(max) = B_{10} = 115975$$

Finally, the maximal number of resulting x-tuples is:

$$N_{|\mathcal{R}_X|}(max) = 2^{10} - 1 = 1023$$

Independent from the complexity of the indeterministic duplicate detection algorithm, the size of the resulting data increases dramatically with the number of *uncertain* edges. As a consequence a full-indeterministic approach is generally not feasible. However, by using a semi-indeterministic approach as presented in the following section, the number of *uncertain* tuple matching can be rigorously reduced. How far an adequate reduction can be achieved is demonstrated in Section 5.2.3 using a real data set.

5.2 Semi-Indeterministic Approaches

As already mentioned above, the number of possible worlds resulting from a full-indeterministic duplicate detection is often too vast. For that reason, we propose four semi-indeterministic approaches in which only some of the most probable worlds are taken into account. In the first three approaches, the initial *matching-graph* is modified. The number of resulting worlds is downsized by reducing the set of *uncertain* edges and hence by reducing the set of indeterministically handled decisions. In contrast, in the fourth approach, the number of *W-graphs* is reduced by modifying the *W-graph* generation mapping ν.

In the end, the probability of all worlds must sum up to 1. Thus, the actual probabilities of the resulting worlds are conditioned and hence may be distorted. However, the result is still more accurate than the one world resulting from a deterministic approach.

1) (α, β)-Restrictions.

In order to filter out the most improbable worlds, only the most ambiguous duplicate decisions have to be considered in an indeterministic way. Decisions of high certainty (e.g. two tuples are duplicates with a certainty of 90%) are made deterministically. The uncertainty whether two tuples are duplicates is maximal, if their matching probability is 0.5. As a consequence, we define two thresholds $\alpha \leq 0.5$ and $\beta \geq 0.5$ for reducing the space of indeterministically handled decisions in a meaningful way. Decisions with probabilities between α and β are considered to be most ambiguous and hence are handled indeterministically (see Figure 12). In contrast, decisions with probabilities outside this range are quite evident and can be deterministically handled without running a high risk of failure. Thus, probabilities lower than α are considered to be 0 and probabilities greater or equal than β are considered to be 1. On the whole, depending on α and β, the number of *uncertain* tuple matching (and hence the number of *uncertain* edges in corresponding *M-graphs*) can be effectively downsized in this way.

2) P-Restrictions.

In this approach, we limit the indeterministic duplicate detection on tuple pairs classified into the set of possible matches (P). Matching probability can be suitably calculated by regarding T_λ and T_μ (see Section 6). Note, by considering tuple similarity as matching probability, a P-restriction is a special kind of an (α, β)-restriction, where $\alpha = T_\lambda$ and $\beta = T_\mu$. Naturally, the effectiveness and correctness of a P-restriction is lower than evaluating the tuple pairs in P by clerical reviews. However, a P-restriction is a full-automatic approach and hence no effort of domain experts is required.

3) Manual-Restrictions.

During clerical reviews it could happen that responsible experts do not know with certainty whether two tuples are duplicates or not. In such cases, experts can consider both possibilities by handling the decision indeterministically. In this way, the indeterministic approach is only applied for individual tuple pairs and the number of resulting worlds remains low.

4) HC-Restrictions.

Restrictions on hierarchical tuple clustering are already known from [6]. In our approach, such restrictions can be achieved by modifying the original *W-graph* generation mapping ν presented in Equation 3. For example, given an *M-graph* $M = (N, E, \gamma)$, instead of generating one *W-graph* for each possible combination of *uncertain* edges (power set $\mathcal{P}(E^?)$), the generation can be modified such that an *uncertain* edge is only considered, if all other edges having a weight greater or equal than the edge's weight have been considered, too. This *W-graph* generation can be achieved by introducing the parameter $\alpha \in \{\gamma(e) | e \in E\}$. For each α a *W-graph* is generated by only regarding edges having a weight greater

or equal than α. Using this HC-restriction strategy, from the *M-graph* M shown in Figure 7 only the *W-graphs* $\{G_1, G_2, G_5, G_8\}$ are derived. As a consequence, the hierarchical clustering with the three consistent *W-graphs* $\{G_1, G_2, G_8\}$ as illustrated in Figure 13 results. Besides this strategy, other HC-restrictions are possible. Moreover, a HC-restriction can be combined with other restriction techniques, as for example an (α, β)-restriction.

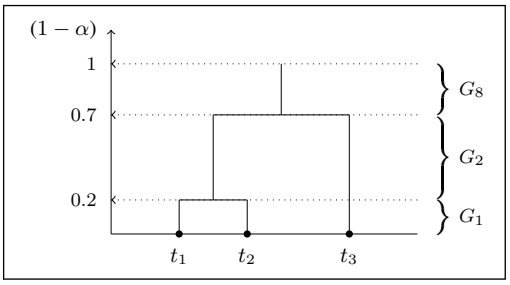


Figure 13: Hierachical Tuple Clustering

5.2.1 Decomposition of Matching-graphs

The more the indeterministic area is restricted, the larger is the proportion of edges weighted with 0. As a consequence, the usage of a semi-indeterministic approach enables a splitting of the initial *M-graph* into multiple independent subgraphs (called *partial M-graphs*). In this case, for each of the *partial M-graphs* the mapping ν can be applied independently. Thus, the number of resulting *W-graphs* can be dramatically downsized and hence the resulting possible worlds are represented in a more succinct way. This in turn extremely reduces the number of required indicator alternatives.

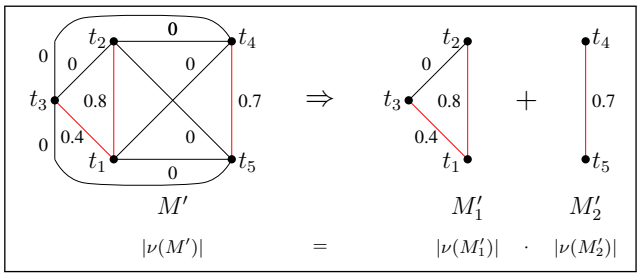


Figure 14: Decomposition of an *M-graph* M' in its independent *partial M-graphs* M'_1 and M'_2

As an example we consider the *M-graph* M' shown in Figure 14. M' can be decomposed into the two independent subgraphs M'_1 and M'_2. From both *partial M-graphs* multiple consistent *W-graphs* can be derived (3 for M'_1, 2 for M'_2). Altogether, from the three possible matches six possible worlds result. However, since the decisions of both *partial M-graphs* are independent, instead one indicator tuple with six alternatives only two indicator tuples with three or two alternatives respectively are required (one tuple for each subgraph). Moreover, since in data lineage the presence of alternatives can be negated (e.g. $\neg(i_2, 1)$), for *partial M-graphs* with only one *uncertain* edge instead of two alternatives a single alternative is sufficient. The x-relations $\mathcal{R}_X$ and $\mathcal{I}_{td}$ resulting from an indeterministic duplicate detection starting from the initial *M-graph* M' by using *M-graph* decomposition are shown in Figure 15.

The decomposition of an *M-graph* $M = (N, V, \gamma)$ into a set of independent *partial M-graphs* is formalized by the mapping $\delta(M)$:

$$\delta(M) = \{M_i = (N_i \subseteq N, V_i \subseteq V, \gamma) \mid A \wedge B \wedge C\}$$

where A is a condition specifying that each subgraph is minimal:

$$A = (\forall n_k \in N_i) : (\exists n_l \in N_i) : (n_k, n_l) \in (V_i^+)^*$$

B is a condition specifying that M is only decomposed into independent subgraphs (no incorrect decomposition has been applied):

$$B = (\forall n_k \in N_i) : (\nexists n_l \in N \setminus N_i) : \gamma((n_k, n_l)) > 0$$

and C is a condition specifying that each subgraph contains all required edges:

$$C = (\forall n_k, n_l \in N_i) : (n_k, n_l) \in V \Rightarrow (n_k, n_l) \in V_i$$

x-tuple	lineage
t_1	$\lambda(t_1) = (i_1, 1)$
t_2	$\lambda(t_2) = (i_1, 1) \vee (i_1, 3)$
t_3	$\lambda(t_3) = (i_1, 1) \vee (i_1, 2)$
t_{12}	$\lambda(t_{12}) = (i_1, 2)$
t_{13}	$\lambda(t_{13}) = (i_1, 3)$
t_4	$\lambda(t_4) = \neg(i_2, 1)$
t_5	$\lambda(t_5) = \neg(i_2, 1)$
t_{45}	$\lambda(t_{45}) = (i_2, 1)$

indicator		$p(i)$
i_1	1	0.176
	2	0.706
	3	0.118
i_2	1	0.7

Figure 15: X-relations $\mathcal{R}_X$ (left) and $\mathcal{I}_{td}$ (right) resulting from decomposing the *M-graph* M'

5.2.2 Consistency

By using a semi-indeterministic approach, deterministically taken decisions can be contradictory. Therefore, in an (α, β)-restriction, the closer α and β, the higher is the probability that the initial *M-graph* is inconsistent and hence all resulting worlds are per se impossible. In such cases, repair operations are required for ensuring the consistency of the resulting *W-graphs* with minimal effort and minimal decision modifications (see future goals in Section 8).

5.2.3 Usability

In order to demonstrate the usability of semi-indeterministic approaches, we consider an (α, β)-restriction on an online cd dataset[3] with 7000 items. For getting matching probabilities, we split the data into two parts. The first part (5000 items) was used as labeled sample data for determining an adequate *sim2p-mapping* (see Section 6). In contrast, the second part (2000 items) was used as actual source data. For attribute value matching, we used the normalized edit distance. For calculating tuple similarity we applied an ordinary distance function based on the similarities of the values of the three attributes c_1=*title*, c_2=*artist* and c_3=*category*:

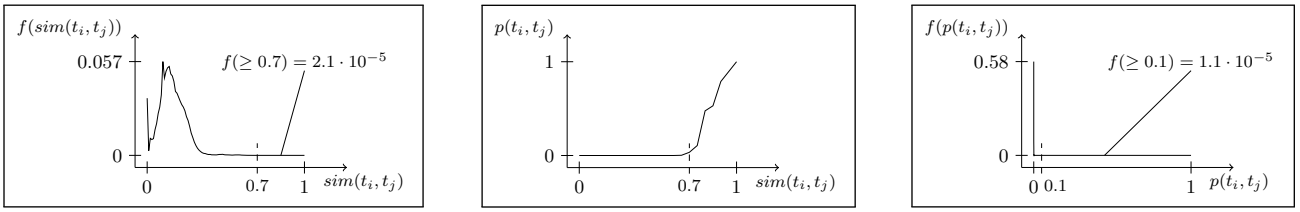
$$sim(t_i, t_j) = 0.5 \cdot c_1^{ij} + 0.4 \cdot c_2^{ij} + 0.1 \cdot c_3^{ij}$$

The results of the experimental evaluations are shown in the Tables 1 and 2, and graphically presented in Figure 16 and 17.

(α, β)	#unc.edges	#*W-graphs*	#poss.worlds	#res.tuples
$(0, 1)$	1033665	$\rightarrow \infty^4$	$\rightarrow \infty^4$	$\rightarrow \infty^4$
$(.05, .95)$	30	1073741824	254803968	2023
$(.1, .9)$	14	16384	6912	2006
$(.2, .8)$	10	1024	768	2000
$(.3, .7)$	8	256	256	1998
$(.4, .6)$	5	32	32	1995
$(.5, .5)$	0	1	1	1987

Table 1: Statistical results of (α, β)-restrictions

[3] http://www.hpi.uni-potsdam.de/naumann/projekte/repeatability/datasets/cd_datasets.html



(i) rel. frequency of similarity values

(ii) *sim2p-mapping* ρ

(iii) rel. frequency of matching probabilities

Figure 16: Statistical characteristics of the sample data

As depicted in Figure 16(i), the similarity of most tuple pairs is very low (98.7% are lower than 0.35 and only 0.002% are higher than 0.7). Moreover, as shown in Figure 16(ii), only high similarity implies an appreciable size of matching probability (almost all duplicates of the labeled sample data have a similarity higher than 0.7). Consequently, the most matching probabilities (99.999%) are lower than 0.1 (see Figure 16(iii)). Thus, the number of considered worlds (*W-graphs*) can be drastically downsized by only taking the most ambiguous decisions into account. For example, only a small restriction of the area of indeterministically handled decisions from $(0, 1)$ to $(.1, .9)$ is required for decreasing the number of *uncertain* edges by almost a factor of one hundred thousand (see Figure 17(i)).

As we expected, the complexity decreases with a shrinking area of indeterministically handled decisions. A $(0, 1)$-restriction is a full-indeterministic approach having an unmanageable complexity, even if not so complex as the worst case predicted in Section 5.1.4 (only each second edge is *uncertain*). In contrast, a $(.5, .5)$-restriction is equal to a full-deterministic approach. Therefore, naturally no *uncertain* edge and hence only one *W-graph* as well as only one possible world result. Since 13 duplicates were detected, the resulting x-relation contains 1987 tuples (see Table 1).

In general, the most edges have low weights. Thus, the number of *uncertain* edges decreases dramatically, if the area of indeterministically handled decision is marginally reduced. In contrast, a restriction of this area from $(.05, .95)$ to $(.4, .6)$ only insignificantly reduces the number of *uncertain* edges further on. The number of resulting *W-graphs* and resulting possible worlds implodes exponentially with a shrinking indeterministic area (see Figures 17(ii) and 17(iii)). In contrast, the number of resulting tuples and the number of required indicator alternatives decrease proportional with a decreasing number of *uncertain* edges (see Tables 1 and 2, Figures 17(iv) and 17(vi)).

As demonstrated by these statistical results, the number of edges weighted with 0 enormously increases, if a semi-indeterministic approach is used. As mentioned in Section 5.2.1, the more edges are weighted with 0, into more *partial M-graphs* the initial *M-graph* can be decomposed. Thus, only a small restriction of the indeterministic area is required to benefit from an *M-graph* decomposition (see Figure 17(v)). For example, already a restriction to $(.05, .95)$ suffices for decomposing the initial *M-graph* into a high number of subgraphs (1963 *partial M-graphs*). The most of these *partial M-graphs* (1931) are single nodes. As a consequence, instead of $1 \cdot 10^9$ *W-graphs* only 2003 *partial W-graphs* result (see Tables 1 and 2). This in turn reduces the required number of indicator tuple alternatives from $2.5 \cdot 10^8$ (the number of possible worlds) to 55. This number can be further reduced to 37, if *partial M-graphs* only having one *uncertain* edge are represented by a single indicator alternative (see Figure 17(vi)). In contrast, in a full-indeterministic approach instead of 1931 only 28 tuples can definitely be excluded to be duplicates. Generally, in a full-indeterministic approach, a decomposition of the initial *M-graph* is most often not useful.

(α, β)	***part.*M-graphs**	***part.*W-graphs**	**#ind.alternatives**
$(0, 1)$	$ca.50$	$\rightarrow \infty$[4]	$\rightarrow \infty$[4]
$(.05, .95)$	1963	2003	55 (37)
$(.1, .9)$	1978	1995	25 (17)
$(.2, .8)$	1980	1991	19 (11)
$(.3, .7)$	1982	1990	16 (8)
$(.4, .6)$	1985	1990	10 (5)
$(.5, .5)$	1987	1987	0

Table 2: Statistical results of (α, β)-restrictions by using *M-graph* decompositions

In conclusion, these results demonstrate that the complexity of an indeterministic approach is already manageable, if the area of indeterministically handled decisions is marginally restricted.

6. SOURCES OF PROBABILITIES

The effectiveness of an indeterministic duplicate detection essentially depends on the taken matching probabilities. Nevertheless, most often deriving adequate probabilities from tuple similarities is not trivial. In many cases, tuple similarity is directly derived from the similarities of their attribute values. The similarity $sim(a_1, a_2) = 0.5$ of two attribute values a_1 and a_2, however, does not necessarily imply that both values represent the same real-world property with a probability of 50%. In contrast, often two real-world properties represented by two attribute values with a similarity of 0.5 are actually, absolutely dissimilar. For example, it is very unlikely that the two names 'Sabine' and 'Janina' both represent the firstname of a same person. Using the normalized hamming-distance, however, the similarity of both names is 0.5.

Nevertheless, the more similar two tuples are, the higher is the probability that both tuples are duplicates. Thus, a mapping function must be monotonically nondecreasing. Moreover, w.r.t. the most similarity measures, the matching probability of two tuples t_i and t_j is lower or equal than their similarity ($p(t_i, t_j) \leq sim(t_i, t_j)$).

As we think, in order to receive adequate mappings from tuple similarity to matching probability, statistics can be used. For example, the probability that the tuples t_i and t_j are duplicates can be defined as the conditional probability $P(t_i =_{id} t_j | sim(t_i, t_j))$ which can be result from empirical analyses on labeled sample data. An example of such a mapping function resulting from empirical analyses of a part of the cd dataset is depicted in Figure 16(ii).

Moreover, as known from estimating or calculating *m-* and *u-probabilities* in the fellegi and sunter theory [14], besides statisti-

[4]Due to our limited resources, processing a full-indeterministic approach was not feasible.

cal analyses, other methods for defining the required conditional probabilities are possible.

Using a P-restriction, only tuple pairs classified as possible matches are considered ($(\forall(t_i, t_j) \in P) : sim(t_i, t_j) \in [T_\lambda, T_\mu]$). In this case, matching probability can be automatically derived from the distance of the tuple similarity to the two thresholds T_λ and T_μ:

$$p(t_i, t_j) = 1 - \frac{T_\mu - sim(t_i, t_j)}{T_\mu - T_\lambda} \quad (4)$$

In manual restrictions, in cases domain experts do not certainly know whether tuples are duplicates or not, the matching probabilities can be manually specified by these experts.

7. RELATED WORK

In general, duplicate detection is already handled in several works [4, 10, 13, 14, 17, 26]. However, even though in the most of these works uncertainty in tuple matching is considered by using different measures of similarity, the decision whether two tuples are duplicates or not is always made in a deterministic way.

Furthermore, there are several approaches using *probabilistic* data models for handling uncertainties in deduplication. In [30, 31] a semi-structured *probabilistic* model is used for handling ambiguities arising during deduplication in XML data. Tseng [29] already used *probabilistic* values in order to resolve conflicts between two or more *certain* relational values. None of the studies, however, handle the uncertainty of ambiguous decisions in detecting relational duplicates.

A *probabilistic* handling of *uncertain* duplicate decisions is proposed in [6]. In this approach, deduplication is considered as a data cleaning task and uncertainty in duplicate decisions is handled by using a set of possible repairs. In contrast to our graph-based approach using the possible world semantics, the authors use hierarchical clustering techniques. This in turn restricts to worlds resulting from hierarchical tuple clustering. Thus, our approach is more general, which can be specialized to the hierarchical clustering approach by using a HC-restriction. Moreover, for the representation of possible repairs, in [6] a new and specific uncertain data model is defined. In contrast, since our approach is based on the possible world semantics, any existing *probabilistic* data model as ULDB or MayBMS can be used. As we think, this increases the reusability of the resulting data, especially if deduplication is considered as a step in a data integration process.

8. CONCLUSION

Due to deficiencies in data collection, data modeling or data management, real-life data is often incorrect and/or incomplete. As a consequence, detecting multiple representations of same real-world entities often comes with a high degree of uncertainty. For that reason, current duplicate detection techniques are designed for properly handling dissimilarities due to typos, data obsolescence or misspellings, in attribute value and tuple matching. Nevertheless, decisions whether two tuples are duplicates or not are still made in a deterministic way. By using a *probabilistic* target schema, however, *uncertain* decisions can be avoided and multiple possible worlds can be taken into account. For that purpose, we introduce a graph-based approach for an indeterministic handling of *uncertain* decisions in duplicate detection. Our approach is based on the possible world semantics and increases the correctness of the resulting data. Moreover, human effort can be reduced to a minimum.

Unfortunately, if any kind of uncertainty is taken into account, the number of resulting possible worlds is just too high and the indeterministic handling becomes impractical. For that reason, we additionally introduce several semi-indeterministic approaches which reduce the number of resulting worlds to a large extent and make the concept of indeterministic duplicate detection more feasible.

In general, an algorithm that needs to iterate over all possible worlds is not scalable. Therefore, one of the directions of future research is a direct and more scalable algorithm for indeterministic deduplication. Another direction is to identify effective and efficient repair strategies for dealing with an inconsistent initial *M-graph* (see Section 5.2.2). Furthermore, if deduplication is to be a step in a larger data integration process, it needs to be extended to *probabilistic* source data. Moreover it should respect fundamental properties such as idempotence if data is duplicate-free.

An essential point of future work are new well-defined quality metrics for *probabilistic* data. These are required for (1) capturing the benefits and drawbacks of *probabilistic* data w.r.t. *certain* data, (2) for working out the most effective parameter settings (e.g. used similarity measures, *combination functions* or *sim2p-mappings*), and (3) for comparing the effectiveness of full-indeterministic, semi-indeterministic (e.g. (α, β)-restriction vs. HC-restriction), and deterministic approaches for duplicate detection. Existing adaptations to recall and precision, such as [30], insufficiently capture what is intuitively better for these applications.

9. REFERENCES

[1] P. Agrawal, O. Benjelloun, A. D. Sarma, C. Hayworth, S. U. Nabar, T. Sugihara, and J. Widom. Trio: A system for data, uncertainty, and lineage. In *VLDB*, pages 1151–1154, 2006.

[2] D. Barbará, H. Garcia-Molina, and D. Porter. The Management of Probabilistic Data. *IEEE Trans. Knowl. Data Eng.*, 4(5):487–502, 1992.

[3] C. Batini and M. Scannapieco. *Data Quality: Concepts, Methodologies and Techniques*. Data-Centric Systems and Applications. Springer, 2006.

[4] O. Benjelloun, H. Garcia-Molina, D. Menestrina, Q. Su, S. E. Whang, and J. Widom. Swoosh: a generic approach to entity resolution. *VLDB J.*, 18(1):255–276, 2009.

[5] O. Benjelloun, A. D. Sarma, A. Y. Halevy, and J. Widom. Uldbs: Databases with uncertainty and lineage. In *VLDB*, pages 953–964, 2006.

[6] G. Beskales, M. A. Soliman, I. F. Ilyas, and S. Ben-David. Modeling and querying possible repairs in duplicate detection. *PVLDB*, 2(1):598–609, 2009.

[7] J. Bleiholder and F. Naumann. Data fusion. *ACM Comput. Surv.*, 41(1), 2008.

[8] O. Brazhnik and J. F. Jones. Anatomy of data integration. *Journal of Biomedical Informatics*, 40(3):252–269, 2007.

[9] P. Buneman and W. C. Tan. Provenance in databases. In *SIGMOD Conference*, pages 1171–1173, 2007.

[10] S. Chaudhuri, V. Ganti, and R. Motwani. Robust Identification of Fuzzy Duplicates. In *ICDE*, pages 865–876, 2005.

[11] W. W. Cohen and J. Richman. Learning to match and cluster large high-dimensional data sets for data integration. In *KDD*, pages 475–480, 2002.

[12] A. de Keijzer and M. van Keulen. Imprecise: Good-is-good-enough data integration. In *ICDE*, pages 1548–1551, 2008.

[13] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate Record Detection: A Survey. *IEEE Trans. Knowl. Data Eng.*, 19(1):1–16, 2007.

[14] I. Fellegi and A. Sunter. A Theory for Record Linkage. *Journal of the American Statistical Association*, 64:1183–1210, 1969.

[15] A. Y. Halevy, M. J. Franklin, and D. Maier. Principles of Dataspace Systems. In *PODS*, pages 1–9, 2006.

[16] A. Y. Halevy, A. Rajaraman, and J. J. Ordille. Data Integration: The Teenage Years. In *VLDB*, pages 9–16, 2006.

[17] M. A. Hernández and S. J. Stolfo. The Merge/Purge Problem for Large Databases. In *SIGMOD Conference*, pages 127–138, 1995.

[18] J. Huang, L. Antova, C. Koch, and D. Olteanu. MayBMS: a probabilistic database management system. In *SIGMOD Conference*, pages 1071–1074, 2009.

[19] C. Koch. MayBMS: A System for Managing Large Uncertain and Probabilistic Databases. In *Managing and Mining Uncertain Data*. Springer, 2009.

[20] N. Koudas, A. Marathe, and D. Srivastava. Flexible String Matching Against Large Databases in Practice. In *VLDB*, pages 1078–1086, 2004.

[21] M. Lenzerini. Data Integration: A Theoretical Perspective. In *PODS*, pages 233–246, 2002.

[22] A. McCallum and B. Wellner. Conditional Models of Identity Uncertainty with Application to Noun Coreference. In *NIPS*, 2004.

[23] H. Müller and J. Freytag. Problems, Methods, and Challenges in Comprehensive Data Cleansing. Technical report, Humboldt Universität Berlin, 2003.

[24] M. Mutsuzaki, M. Theobald, A. de Keijzer, J. Widom, P. Agrawal, O. Benjelloun, A. D. Sarma, R. Murthy, and T. Sugihara. Trio-One: Layering Uncertainty and Lineage on a Conventional DBMS (Demo). In *CIDR 2007, Third Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 7-10, 2007, Online Proceedings*, pages 269–274, 2007.

[25] H. B. Newcombe, J. M. Kennedy, S. J. Axford, and A. P. James. Automatic Linkage of Vital Records. *Science*, 130:954–959, Oct. 1959.

[26] F. Panse, M. van Keulen, A. de Keijzer, and N. Ritter. Duplicate Detection in Probabilistic Data. In *Proceedings of the 2nd Workshop on New Trends in Information Integration (NTII 2010) co-located with ICDE 2010*, pages 179–182, 2010.

[27] P. D. Ravikumar and W. W. Cohen. A Hierarchical Graphical Model for Record Linkage. In *UAI*, pages 454–461, 2004.

[28] G. Rota. The Number of Partitions of a Set. *The American Mathematical Monthly*, 71(5):498–504, 1964.

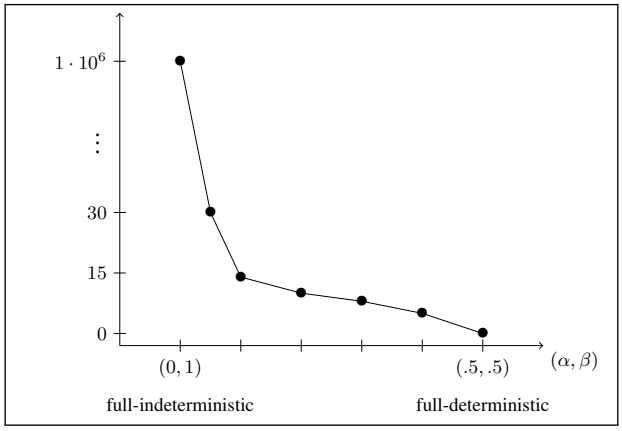
[29] F. S.-C. Tseng, A. L. P. Chen, and W.-P. Yang. Answering Heterogeneous Database Queries with Degrees of Uncertainty. *Distributed and Parallel Databases*, 1(3):281–302, 1993.

[30] M. van Keulen and A. de Keijzer. Qualitative effects of knowledge rules and user feedback in probabilistic data integration. *VLDB J.*, 18(5):1191–1217, 2009.

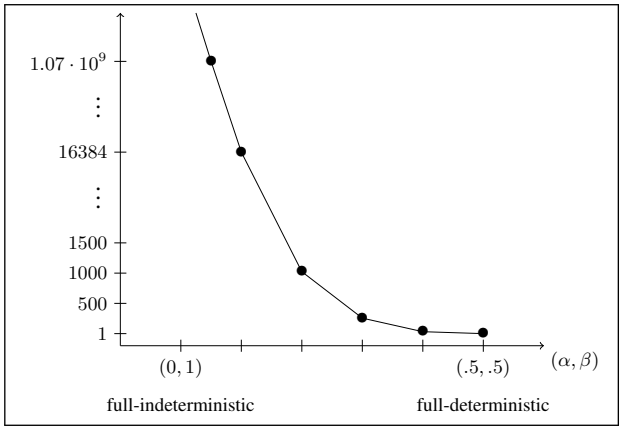
[31] M. van Keulen, A. de Keijzer, and W. Alink. A Probabilistic XML Approach to Data Integration. In *ICDE*, pages 459–470, 2005.

[32] Y. R. Wang and S. E. Madnick. The Inter-Database Instance Identification Problem in Integrating Autonomous Systems. In *Proceedings of the Fifth International Conference on Data Engineering, February 6-10, 1989, Los Angeles, California, USA*, pages 46–55. IEEE Computer Society, 1989.

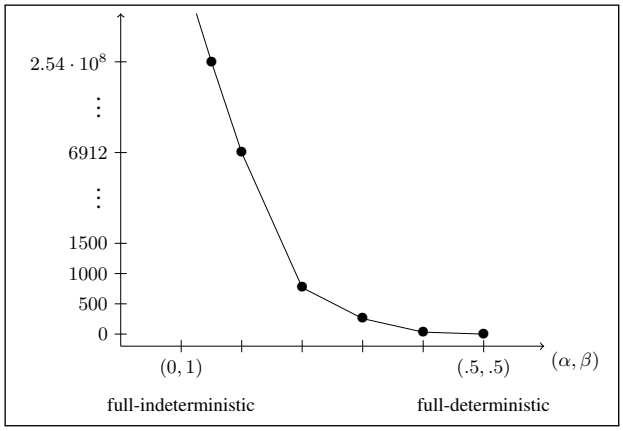
[33] S. E. Whang, O. Benjelloun, and H. Garcia-Molina. Generic entity resolution with negative rules. *VLDB J.*, 18(6):1261–1277, 2009.



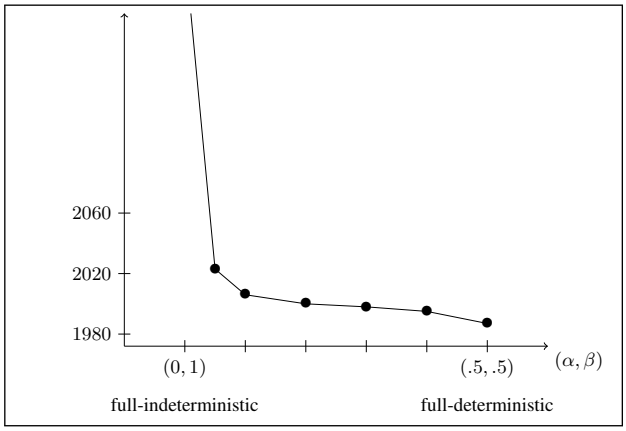
(i) Number of uncertain edges



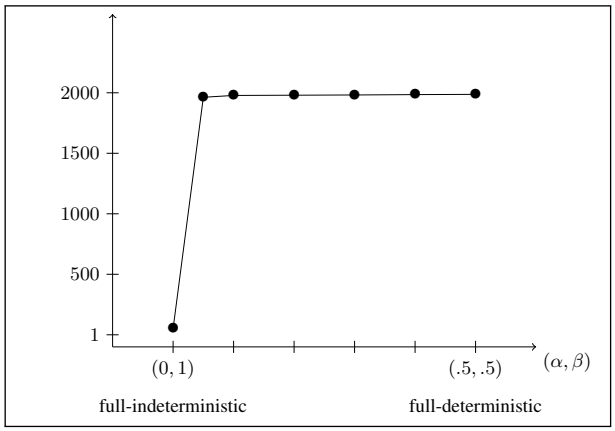
(ii) Number of *W-graphs*



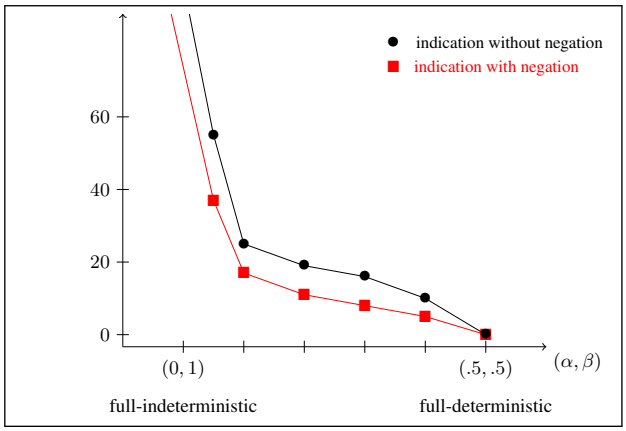
(iii) Number of possible worlds



(iv) Number of resulting tuples



(v) Number of *partial M-graphs*



(vi) Number of required indicator alternatives

Figure 17: Statistical results of (α, β)-restrictions