Multi-Source Entity Resolution

Christiaan Michiel Punter
MSc. Thesis
January 18, 2010

University of Twente
Department of Computer Science

Abstract

Background The focus of this research was on multi-source entity resolution in the setting of pair-wise data integration. In contrast to most existing approaches to entity resolution this research does not consider matching to be transitive. A consequence of this is that entity resolution on multiple sources is not guaranteed to be associative. The goal of this research was to construct a generic model for multi-source entity resolution in the setting of pair-wise data integration that is associative.
Results The main contributions of this research are: (1) a formal model for multi-source entity resolution and (2) strategies that can be used to resolve matching conflicts in a way that renders multi-source entity resolution to be associative. The possible worlds semantics is used to handle uncertainty originating from possible matches. The presented model is generic enough to allow different match and merge function as well as allowing different strategies to resolve matching conflicts.
Conclusions A formalization of an example of multi-source entity resolution is presented to show the utility of the proposed model. By using small examples in which three sources are integrated it is shown that the strategies resulted in associative behavior of the integrate function.

Acknowledgments

I would like to thank my supervisors for their guidance during this research. Also, I would like to thank my family and close friends for their support. Thanks.

Contents

Chapter 1

Introduction

With the growing volume of information on the Internet it can be expected that we will see increasingly more applications integrating information from multiple sources.

Consider an on-line TV guide that annotates movies with information from The Internet Movie Database (IMDB). In order to accomplish this there should exist a mapping between movie items from the TV guide and movie items from IMDB. When there is no unique identifier to link an item from the TV guide with an item from IMDB, mapping needs to be done on other available information such as title, year and actors. Since such meta information is not always complete or accurate it is likely that some movie items from the TV guide are mapped to the wrong movie items from IMDB.

Entity resolution, which is also known as record linkage or record deduplication, handles the problem of relating *information items* to *entities* in the real world. In the afore mentioned example the information items consist of features such as title, year and actors and refer to a movie in the real world. A movie is a real world entity where an information item refers to. The goal of entity resolution is to find the set of items that best represent the real world entities.

Besides textual features such as title and year in previous example we can also consider entity resolution on biometric features. Consider the integration of information from surveillance camera's on airports and train stations. When a camera on an airport detects a person with certain biometric features, e.g. size or skin color, we can compare the features with biometric features registered by a camera on the nearby train station. With entity resolution we are able to determine if the person seen on the airport is the same person seen on the train station.

Information items originate from a *data source.* We assume that a data source does not contain *duplicates.* With duplicates we mean items that refer to the same entity. As an example think of two movie items from IMDB that refer to the same movie. Figure 1.1 depicts the afore mentioned

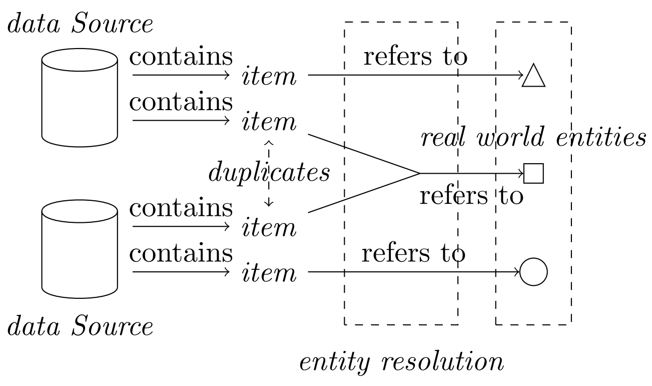


Figure 1.1: Entity Resolution Concepts

concepts.

In order to get meaningful results when we integrate data sources we do not want the integrated result to contain duplicates. In order to prevent duplicates we have to *match* items from different data sources and see if both items refer to the same real world entity. Two items are considered to match if they are so much alike that they probably refer to the same entity. When we encounter items that match we have to *merge* these items in order to get one representation for the entity to which they refer.

Matching items is done by comparing features of items. The examples used in this research consider features consisting of textual information. Comparing these features can be done by a character based similarity metric such as the edit distance. Consider the case in which two names are compared for equality. When we encounter the names John and Mike we can be sure that both names are referring to two distinct persons. However if we encounter the names John and Jon we can argue that someone made a typing error in one of the two names because both names differ by only one character. Using a character based similarity metric such as the edit distance would render both names as very similar. In the case of a typing error both names could actually refer to one person named either John or Jon.

In this research we abstract away from the details of matching items. We assume there exists a match function which relates pairs of items to *matching scores*. These matching scores indicate the degree of similarity and therefore the likelihood that items refer to the same real world entity.

In order to make a distinction between a match and a non-match we need a *threshold value*. When the matching score between two items is equal or greater than the threshold value we consider both items as being the same. When the matching score is below the threshold value we consider both items as being distinct.

When two items are considered to refer to the same entity we cannot add both items to the integrated result since this would lead to duplicates. Either one of the items needs to be used as a representation of the real world entity or we have to combine the information contained by each item. In this research we abstract away from the details of combining items and assume there exists a merge function which relates pairs of items to items that are merged. A *merged item* is assumed to contain all the information of the items from which it is derived. There is no information loss when we merge two items. When we merge John and Jon the merged item will still represent both names but refers to only one real world entity, e.g. a person called either John or Jon.

Most approaches to entity resolution regard two items to be either the same or distinct. There are cases however in which the matching score between two items is very near the threshold value. Making an incorrect decision, such as a match between two items that actually refer to distinct entities, leads to erroneous items in the integrated result.

In "A Theory For Record Linkage" [7] Fellegi and Sunter formalized the concept of entity resolution. Instead of making only a distinction between items that match and items that are distinct they also consider the case in which there is a possible match between two items. Items that possibly match need manual assessment to see if they actually refer to the same entity in the real world.

In order to prevent wrong decisions we use two threshold values: a threshold value to determine if items are distinct from each other and a threshold value to determine if two items are the same. When the matching score is lower or equal to the first threshold value we regard both items as being distinct. When the matching score is greater or equal to the second threshold value we regard both items as being the same. Two items with a matching score that is between both threshold values are regarded as possibly the same.

For two items that are possibly the same we have to consider the case in which both items are distinct and the case in which both items are the same. In order to deal with multiple alternatives we use the *possible worlds semantics.* With the possible worlds semantics we consider multiple alternative representations of the real world in which at most one of the alternatives correctly represents the real world situation.

When we perform entity resolution on multiple sources and consider possible matches between items, the integrated result will contain items for which we have multiple alternatives. Making a choice for each of the possible matches results in a *possible world.* At most one possible world represents the real world situation. In this research we only focus on the possible results without considering confidence scores.

The focus of this research will be on the entity resolution process in the setting of pair-wise integration. We do not consider schema alignment or similarity metrics. We assume that data sources have the same schema and

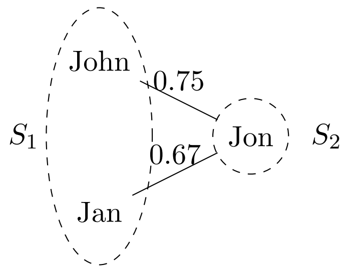


Figure 1.2: Matching conflict example

that matching scores between items are available.

1.1 Motivation

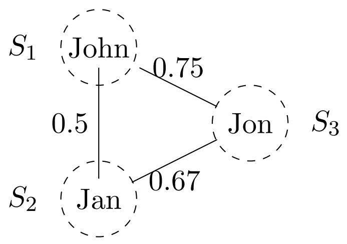
We investigate entity resolution in the setting of pair-wise data integration. When we integrate the two data sources depicted in figure 1.2 and we regard two items with a matching score above 0.6 as being the same we derive at a situation in which we have a *matching conflict*.

Because both matching scores are greater than 0.6 we have to consider the case in which John and Jon are the same and the case in which Jan and Jon are the same. Since we assume that items originating from the same source are distinct from each other there cannot be a match between John and Jan. The situation in which all three names refer to the same person is therefore impossible. Since a match between John and Jon and a match between Jan and Jon is mutual exclusive we have to think of strategies to deal with these situations.

When integrating multiple sources it is possible that more than two items all refer to the same real world entity. Since we integrate data sources in a pair-wise fashion we can consider the data sources in different orders.

Consider the integration of the three sources depicted in figure 1.3a. Using a threshold value of 0.6 and not considering any strategy to resolve matching conflict we get different results depending on the order in which the sources are integrated. All possible results are depicted in figure 1.3b.

When we first see John and Jon we conclude that both items refer to the same person and merge both items. When we later see Jan it is unclear if Jan matches with our merged item since Jan matches with Jon but does not match with John. If we first see John and Jan we conclude that both items refer to distinct persons. When we later see Jan we derive at a matching conflict. If we first see Jan and Jon we conclude that both items refer to the same person. When we later see John we also derive at a matching conflict. This example shows that multi-source entity resolution is not always associative.



(a) Data sources S_1, S_2 and S_3

Integration order	Result after step 1	Result after step 2
$S_{12} = \text{integrate}(S_1, S_2)$ $S_{123} = \text{integrate}(S_{12}, S_3)$	John Jan	John — Jon Jan — Jon
	John — Jon Jan	{John, Jon} | ? Jan
	Jon Jan — Jon	John | ? {Jan, Jon}

(b) Integration orders

Figure 1.3: Integrating three data sources

1.2 Research Question

The main question which we will try to answer with this research is:

How to perform multi-source entity resolution using the possible worlds semantics in the setting of pair-wise data integration that is associative?

As we have seen in the motivation we can encounter matchings involving matching conflicts. We need to think of strategies to resolve these conflicts and guarantee associativity of the integration process.

How to deal with matching conflicts?

As we have seen in the example there are also situations in which entity resolution for three or more sources is not associative. This problem occurs when we have a matching conflict involving a merged item.

How to deal with cases where multi-source entity resolution is not associative?

1.3 Approach

Before we look into the problems that occur during multi-source entity resolution we discuss related research. Most approaches to entity resolution focus on either data quality, i.e. minimize errors made by the entity resolution process, or performance, i.e. minimizing the number of matches and merges to make. We found out that there is not much known about how to perform pair-wise entity resolution on multiple, i.e. more than two, sources. Because we perform entity resolution pair-wise we can integrate multiple source in different orders. In order to get meaningful results the order in which entity resolution is performed on the sources should not have an impact the final outcome, i.e. pair-wise entity resolution should be associative.

One of the assumptions we make in our research is that items originating from the same source can never refer to the same entity in the real world. Because of this assumption we cannot assume that matching is transitive. This differs from approaches found in literature such as [2].

To minimize the number of false positives, i.e. items that are considered to be the same but actually refer to distinct entities, and false negatives, i.e. items that are considered to be distinct but actually refer to the same entity, we also consider some items as being possibly the same. For items that are possibly the same the entity resolution process could not make a decision whether or not the items refer to the same entity. Because it is uncertain if both items refer to the same entity we consider the case in which both items do refer to the same entity and the case in which both items refer to distinct entities. We will use the possible worlds semantics to describe the outcome of the entity resolution process. The same approach has been taken in [5] but they do not consider integration of three or more sources.

To get more insight into the problem domain we will first investigate the problems that might occur when performing entity resolution on multiple sources in the setting of pair-wise data integration. We start by describing the integration of two sources. Then we describe the integration of three sources and investigate the problems that might occur regarding associativity.

For each step we will describe what we expect as input and what we expect to get as output. When we encounter a problematic case with respect to matching conflicts and/or associativity we will first investigate what the desired outcome should be before discussing ways of how to handle these problems.

After investigating the problem domain we will define a formal model for multi-source entity resolution in the setting of pair-wise data integration. This model abstracts away from the match and merge functions as well as from how to handle matching conflicts. We will investigate the problematic cases and will discuss ways to handle these problems that fits the proposed model. We can then describe strategies to handle matching conflicts.

1.4 Overview

The next chapter gives a short overview of the related research. We will then informally introduce the problem domain by discussing an example. From the insights gained in the informal discussion of the problem we will define a generic model for multi-source entity resolution. In appendix we included a formalization of the example using the model.

The formal model describing multi-source entity resolution partly answers our research question with the exception that we abstracted away from the strategies necessary to resolve matching conflicts. Our sub-questions are answered in the strategies chapter in which we describe how to resolve the problematic cases. Figure 1.4 gives an overview of all the chapters.

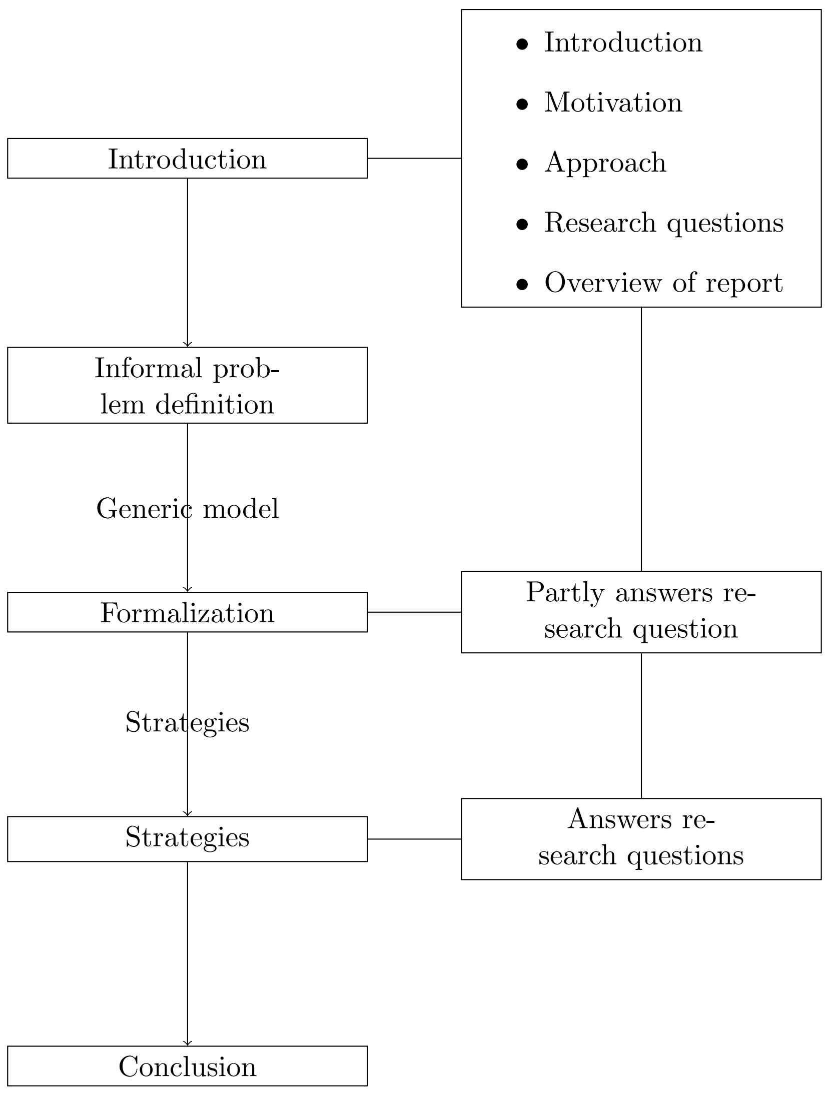


Figure 1.4: Overview of the report

Chapter 2

Related Research

Entity resolution, which is also known as deduplication [13, 6], merge-purge [8], citation matching [9] and record linkage [12] [7], handles the problem of relating information items to real world entities. An information item can be a database record but can also be, in the case of citation matching, a group of records.

Newcombe [12] laid the probabilistic foundation for entity resolution which was formalized by Fellegi and Sunter in "A Theory For Record Linkage" [7]. Fellegi and Sunter consider all pairs of items from two sets $A \times B$ and devide them into three sets: positive link, positive non-link and possible link. Since an item consists of a number of attributes, each attribute of two items has to be compared to each other. This results in a comparison vector. A linkage rule is a set of random decision functions that determine if two items, given their comparison vector, belong to either the set of matches, the set of non-matches or need clerical review, i.e. the items possibly match. Fellegi and Sunter describe a linkage rule that is optimal provided that the comparison attributes are conditionally independent. Many entity resolution applications today are still based on the work of Fellegi and Sunter.

We can divide approaches to entity resolution into two groups: (1) pair-wise approaches [2, 11] and (2) global clustering approaches [1]. In the pair-wise approach we compare items from one source with items from another source by using a match function and combine items that are considered to be the same by using a merge function. Performing entity resolution on multiple sources requires multiple entity resolution steps in which the result of a previous step is matched and merged with a new source.

Global clustering approaches consider multiple sources during an entity resolution step and group items that are deemed to be the same. Since items from multiple source are considered it is possible to make more accurate decisions compared to the pair-wise approach since more information is available. When we merge two items in the pair-wise approach and later discover an item from another source which matches with one of the items involved in the merged item we have to (1) consider all items as being the same or (2) reconsider the merged item in order to guarantee associative behaviors of the overall ER-process. Global clustering approach do not suffer from this problem.

Our approach can be seen as a combination of pair-wise and global clustering approaches. We consider pair-wise entity resolution in the sense that we perform entity resolution on two sources. Performing entity resolution on multiple sources thus takes multiple entity resolution steps. When we encounter a match involving a merged item we consider the all the base items of the merged item and the item with which the merged item matches. In this case we consider items origination from possibly many sources which is comparable to global clustering approaches.

Some approaches associate confidence scores to items [11]. Confidence scores can either be associated with items coming from an unreliable source or can be the result from a comparison between items. This second kind of confidence scores represent how likely it is that the items refer to the same real world entity. Pair-wise entity resolution which incorporates confidence scores may be order critical. With the pair-wise approach all potential derivations of merged items must therefor be considered. In our research we do not associate confidence scores to items, but we will keep track of matching scores between items. These matching scores can be interpreted as confidence scores that represent how likely it is that two items represent the same entity.

An approach to data integration that uses the possible worlds semantics is described in [15, 5]. With this approach data integration can result in multiple alternative integration results which are associated with a confidence score. The benefit of this approach is that it is sufficient to use only a few knowledge rules and to set rough safe threshold values in order to get integration results that can be meaningfully used [14]. *Knowledge rules* prescribe when two items match. An example of such a rule is “two items refer to the same person if the name and address fields are equal”.

Chapter 3

Informal problem definition

Entity resolution deals with the problem of linking items from a data source to entities in the real world. Think of a database containing contact information. It is well possible that two or more items in this database refer to the same person. In most applications it is undesirable to have multiple items referring to the same entity. Entity resolution can be used to find these duplicates. Once found these duplicates can be merged. Finding items that refer to the same entity is also known as duplicate record detection [6].

The setting in which we will use entity resolution is data integration. Data integration handles the problem of combining information from two or more data sources and providing the user with a uniform view of these data [10]. Instead of having a homogeneous logical view of data that is stored at distributed heterogeneous data sources we assume that all data is first collected and integrated before it is stored in a new single data source.

Consider that we have two databases, instead of one, containing contact information that we want to integrate. Now there can be an overlap between the persons to which the items from both databases refer. After integrating both databases it is undesirable to have duplicates in the integrated result. It is therefor necessary to use entity resolution during the integration process in order to find those items that refer to the same person.

We do not perform entity resolution after the integration process in the sense that we first accumulate all data and then perform duplicate record detection. We make the assumption that data sources do not contain duplicate items. This means that two items from the same source can never refer to the same entity. Only items from different sources can potentially refer to the same entity. This differs from duplicate record detection after data integration because we already know which items can never refer to the same entity. This information would be lost if we first accumulated all data and did not track from which source each item originated. Tracking the source from which items originated is also known as the *lineage* of items.

Many existing data integration approaches first collect all items from the

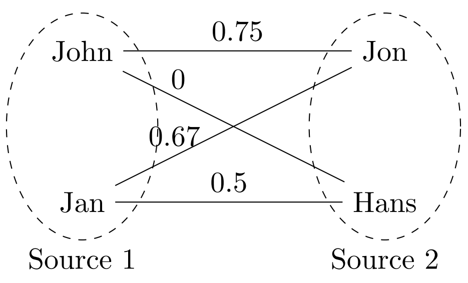


Figure 3.1: Two datasources

different data sources and then merge all duplicates. The problem with this approach is that the lineage of the items is lost. We no longer know if two items came from the same source and may therefor never refer to the same entity.

In this research we perform entity resolution on items originating from different sources. This way we only match items that potentially refer to the same entity. Items that originate from the same source are not matched with each other.

In this chapter we will integrate data sources and see what difficulties regarding entity resolution arise. In the introduction we already discussed a small example in which we encountered a matching conflict. Here we will further discuss this example and think of ways to deal with matching conflicts. Also the problem of associativity will be discussed as we integrate more than 2 data sources.

3.1 Integrating 2 sources

Consider the two data sources, containing names, depicted in figure 3.1. Each data source consists of two items. To simplify our example we consider only one feature, namely the name-feature, for each item. In the next subsection we will elaborate on matching items containing multiple features. The lines between items represent matchings. Each line is associated with a matching score which signifies the likelihood that both items refer to the same entity.

Since we made the assumption that items originating from the same data source can never refer to the same real world entity we already know that John and Jan are two distinct persons and that Jon and Hans are two distinct persons as well. There is however the possibility that a name from source 1 refers to the same person as a name from source 2. In our example we can imagine that John and Jon refer to the same person due to a typing error in one of the names.

3.1.1 Matching items

The matching score between items in our example is based on the edit distance between the names. We use the *Levenshtein distance* to determine the edit distance between two names. The Levenshtein distance considers the number of deletions, insertions and substitutions that are necessary to transform one character string into the other. Take for example John and Jon. The edit distance (Levenshtein distance) between both names is 1 since we can derive at Jon from John by deleting one character. The same is true if we measure the edit distance the other way around. We can derive at John from Jon by inserting one character.

In a more realistic scenario we would have items consisting of multiple features. In our example we could have items containing features such as address, telephone number and email address. When we have multiple features we need to perform matching on each feature. Matching items consisting of multiple features thus results in multiple matching scores.

Some of the features that we want to compare can be more important than other features. Consider two items that both contain a name and address feature. If only the address feature differs we could reason that both items refer to the same person since there is the possibility that this person moved to a new address. If, however, the name features differ it is more likely that the items refer to distinct persons that live or have lived at the same address. As can be seen in this example the name feature is more important than the address feature. Each feature can be associated with a weight which indicates how discriminative this feature is when we perform a match between two items. Matching items constituting of multiple features results in a *comparison vector* [7] consisting of a matching score for each feature.

Comparing features is usually done by using a character based similarity metric such as the Levenshtein distance we used in our example. In some cases we can benefit from *domain knowledge* when we compare two features. An example of this would be the name feature in which the format can vary from source to source or even from item to item. Consider for example "J. Doe", "John Doe", "John D.", "Doe, J.", "Doe, John". All these names probably all refer to a person called John Doe but the format in which the names are expressed differ.

Since we focus on the entity resolution process we abstract away from similarity metrics and assume that all features have the same format. Furthermore we only have one feature and therefor do not consider weights for multiple features.

In our example we calculate the matching score between items by determining the edit distance and calculate the percentage of characters that are the same. The percentage that is different is calculated by dividing the edit distance d by the length l of the longest name. Since we want to know

Item 1	Item 2	Edit Distance	Length	Matching Score
John	Jon	1	4	0.75
John	Hans	4	4	0
Jan	Jon	1	3	0.67
Jan	Hans	2	4	0.5

Table 3.1: Matching scores

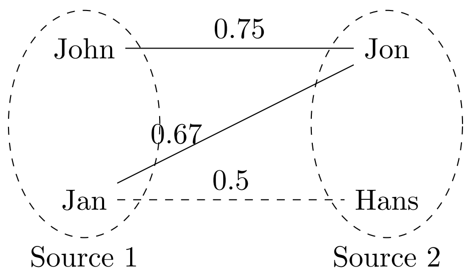


Figure 3.2: Annotated edges

the percentage of characters that is the same, instead of the percentage that is different, we subtract 1.0 by the calculated difference. We calculate the matching score by $1.0 - (d/l)$. All pairs of items with their edit distance, length of the longest name and matching score is shown in table 3.1.

From the matching scores we can determine if two items refer to the same entity or if they refer to distinct entities. To make this distinction we use two threshold values, δ_m and δ_d to classify each match between two items as either distinct, same or possibly the same. When the matching score is lower or equal than δ_d we regard both items as being distinct. When the matching score is greater or equal than δ_m we regard both items as referring to the same entity. When the matching score is between both threshold values we consider both items as possibly the same. When two items are possibly the same we consider the possibility in which both items refer to the same entity and the possibility in which both items are distinct. Since we use the possible worlds approach each possible match potentially constitutes to twice as many possible worlds since all already known possible worlds now have two new possibilities.

When we set threshold value δ_m to 0.65 and threshold value δ_d to 0.45 we derive at the situation in figure 3.2. When there is no edge between two items we regard both items as being distinct. A dashed edge depicts the situation in which both items are possibly the same. A solid edge depicts the situation in which both items are the same.

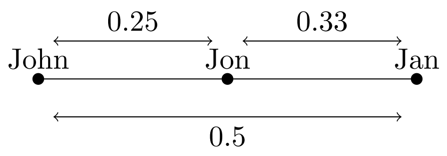


Figure 3.3: Matching distance

3.1.2 Matching conflicts

After matching all items we sometimes derive at situations which are not possible in the real world. Consider the matching between items John, Jon and Jan as shown in figure 3.2. We know that the names John and Jan refer to two distinct persons and yet both names refer to the same person as the name Jon. This situation contradicts with our assumption that names John and Jan refer to two distinct persons since they originate from the same data source.

When an item is considered to be the same as two or more items from the same source we derive at a *matching conflict*.

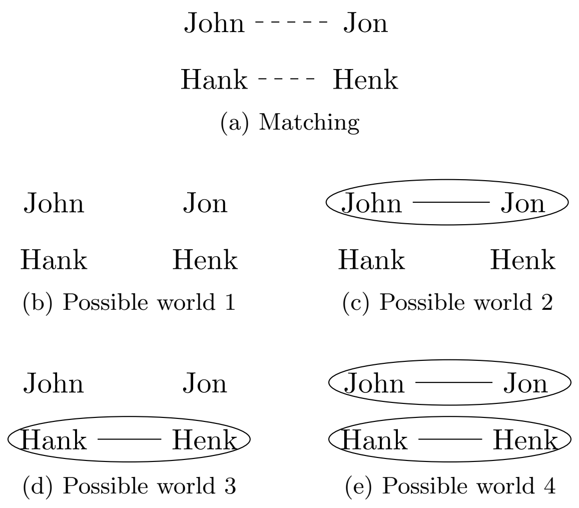
So what should be the integrated result of items that are involved in a matching conflict? Consider figure 3.3 which shows the distances between each items. We see that John and Jan match because Jon is within matching distance from both John and Jan. With matching distance we mean the amount of characters that differ between two items. The distance between John and Jon is 0.25 since 25% of the characters differ. Since we use a threshold value δ_m of 0.65 the maximum distance between two items in order to match is 0.35. The distance between John and Jan is far greater than 0.35. We argue that an item can only match another item if the total distance between both items is within the matching distance.

In the John, Jon, Jan case we have only two alternatives: either John and Jon refer to the same person or Jon and Jan refer to the same person. Other combinations are not possible.

Another question is what should happen with the possible match between Jan and Hans. In the possible world where John and Jon are the same there is the possibility in which Jan and Hans are the same. In the possible world where Jan and Jon are the same this is not possible since this would lead to a matching conflict between Jan, Jon and Hans.

3.1.3 Possible worlds semantics

In the previous section we discussed how we match 2 items. For items that are possibly the same, e.g. refer to the same entity, we have to consider the case in which both items refer to the same entity and the case in which both items refer to distinct entities. Only one of these propositions can reflect the real world situation.



(a) Matching (b) Possible world 1 (c) Possible world 2 (d) Possible world 3 (e) Possible world 4

Figure 3.4: Possible worlds

Consider the matching between 4 items as depicted in figure 3.4a. In this case we have two pairs of items that are possibly the same. Figure 3.4b depicts the possible world in which all four items refer to distinct persons. Figures 3.4c and 3.4d depict the possible worlds in which one pair of items are considered to be the same. Figure 3.4e depicts the possible world in which all pairs of items are considered to be the same.

When we first encounter John and Jon, which are possibly the same, we derive at two possible worlds. When we later see Hank and Henk, which are also possibly the same, for each of our possible worlds we have again two possibilities. This indicates that when we encounter a possible match this potentially leads to twice as many possible worlds.

In some situation we derive at possible worlds which are not possible in the real world. An example of this is a potential matching conflict as shown in figure 3.5a. The possible world shown in figure 3.5e is not possible since it contains a matching conflict. In this case we have only 3 possible worlds.

3.1.4 Merging items

When we merge items we need to think of a representation that best reflects the entity to which both items refer. In the case of a merge between items John and Jon we know that the person to which both items refer is either called John or Jon. There is no way of knowing which name actually reflects the name of the person in the real world.

We assume a *union class* merge function [2] in which each feature of an item maintains all the values seen in its base items. Because of this we need to keep track of the items from which a merged item is derived. The base items from which a merged item is derived is also known as the lineage of

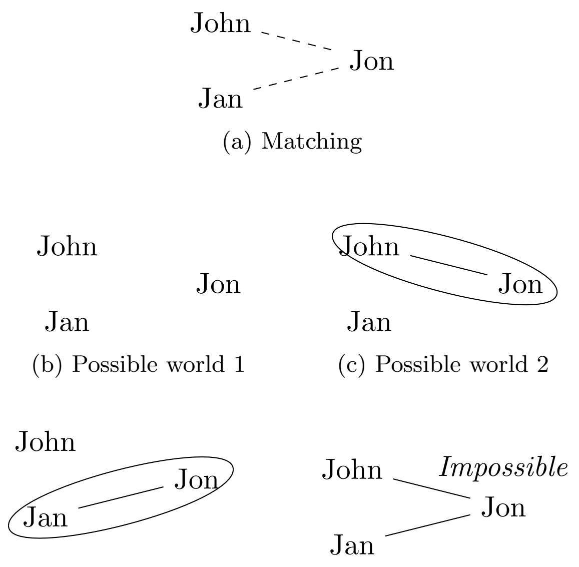


Figure 3.5: Impossible worlds

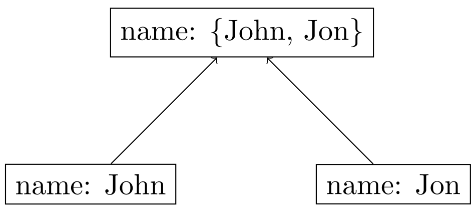


Figure 3.6: Merged item

the merged item. Keeping track of the *lineage* of items will also be necessary when we merge multiple items which will be discussed later.

Consider the case in which items John and Jon are considered to be the same. When we merge both items we construct a new item which refers to both base items John and Jon. Since the merged item contains all values of its base items the name feature consists of the set $\{\text{John}, \text{Jon}\}$. This situation is depicted in figure 3.6.

In the case of multiple features we can use the same approach. The value of a feature consists of the union of all values of that feature in its base items. These values are not actually stored in the merged item but are derived from its base items.

Note that a merged item can be matched and merged with other items as well. Because of this we can have merged items consisting of a multiple base items.

$$\begin{tikzcd}
 & & & S_{1234} & \\
 & & & \text{Integrate} \arrow[u] & \\
 & & S_{123} \arrow[ur] & & S_4 \arrow[ul] \\
 & & \text{Integrate} \arrow[u] & & \\
 & S_{12} \arrow[ur] & & S_3 \arrow[ul] & \\
 & \text{Integrate} \arrow[u] & & & \\
S_1 \arrow[ur] & & S_2 \arrow[ul] & &
\end{tikzcd}$$

(a)

$$\begin{tikzcd}
 & & & S_{1234} & & & \\
 & & & \text{Integrate} \arrow[u] & & & \\
 & S_{12} \arrow[urr] & & & & S_{34} \arrow[ull] & \\
 & \text{Integrate} \arrow[u] & & & & \text{Integrate} \arrow[u] & \\
S_1 \arrow[ur] & & S_2 \arrow[ul] & & S_3 \arrow[ur] & & S_4 \arrow[ul]
\end{tikzcd}$$

(b)

Figure 3.7: Integration process

3.2 Integrating multiple sources

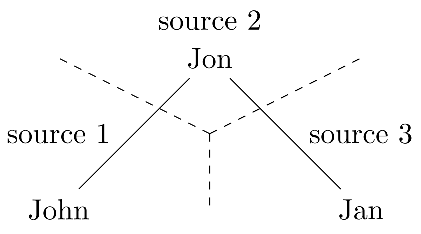
Instead of considering only two sources we now discuss the case in which we integrate three or more sources. We assume that the integration process operates pair-wise. Consider that we want to integrate four sources, S_1, S_2, S_3 and S_4. This means that we first integrate two of the four sources and then continuously add a new sources to the integrated result. Pair-wise integration of four sources is depicted in figure 3.7a.

We do not consider the case in which two previously integrated sources are integrated as depicted in figure 3.7b. At most one of the sources comes from a previous integration steps. A consequence of this is that a merged item can never consist of two previously merged items.

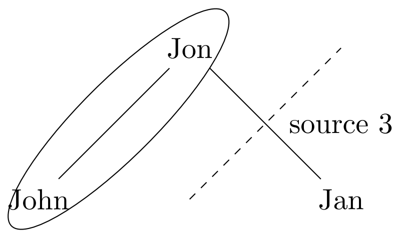
3.2.1 Matching merged items

Consider again the items John, Jon and Jan but now residing on three different sources as depicted in figure 3.8a. When we first integrate the sources 1 and 2, containing items John and Jon, and then integrate the result with the source 3, containing item Jan, we see that Jon is considered to be the same as Jan but that John is considered to be distinct from Jan. Since items John and Jon are one merged item we cannot say how Jan matches with this merged item.

We can take the minimum matching score of all matching scores between item Jan and items Jon and John and then take this score to be the matching score between Jan and the merged item. In this case we see that Jan is considered to be distinct from the merged item consisting of John and Jon.



(a) Matching between items



(b) Matching after integrating source 1 and 2

Figure 3.8: Matching a merged item

If we take the maximum matching score then both items are considered to be the same.

Chapter 4

Formalization

This chapter formalizes each step of the integration process which was informally described in the previous chapter. After constructing a generic model for multi-source entity resolution in the setting of pair-wise data integration we revisit the example of the previous chapter and will use the generic model to formally describe each step of the integration process.

The model that we present in this chapter abstracts away from handling problematic cases such as matching conflicts. In the next chapter we will discuss strategies to resolve the problematic cases.

4.1 Matching two items

When we integrate two data sources we do not want to have duplicates in the integrated result. Because of this we need to perform entity resolution on the items from both sources. As we have seen in the previous chapter we need to compare items from one source with all items from the other source. Items that originate from the same source are assumed to refer to distinct entities.

There is a difference between sources where items originate from and sources that are the result from an integration step. Two items from a source that is the result from integrating two sources can still be considered to be the same in a later integration step, as we will see in the chapter about strategies to resolve conflicts, if the sources where the items originally come from are not the same.

Assumption 1 *We assume there exists a match function which returns the matching score for each pair of items. The matching score represents the likelihood that two items are equal. A high matching score indicates that both items are very similar and therefor probably refer to the same entity. A low matching score indicates that both items differ significantly and are therefor probably distinct.*

$$match : I \times I \rightarrow [0, 1]$$

The match function is expected to be idempotent and commutative. With idempotent we mean that an item always matches itself such that $match(i, i) = 1.0$*. With commutative we mean that the order in which two items are matched does not affect the matching score such that* $match(i_1, i_2) = match(i_2, i_1)$*.*

Note that we assume that items that originate from the same source S are all distinct from each other such that $\forall i_1, i_2 \in S, \text{match}(i_1, i_2) = 0$ where $i_1 \neq i_2$. A source that is the result from an integration step can contain items, i_1, i_2 for which $\text{match}(i_1, i_2) > 0$ holds. In this case the items i_1 and i_2 originally come from different sources but were considered to be distinct in a previous integration step.

4.2 Merging two items

Where the matching score of two items is above threshold value δ_m we need a merge function that combines these items in order to have one representation for the real world entity to which both items refer. When we merge two items we still want to know how this merged item came into existence i.e. from which items it was derived. This is also known as the *lineage* of the merged item. The lineage of a merged item is necessary, as we will later see, when we want to resolve conflicts associated with associativity. We will call the items from which a merged item is derived *base items.*

Assumption 2 *We assume there is a merge function which merges two matching items from the set of all possible items* I *such that*

$$merge : I \times I \rightarrow I. \tag{4.1}$$

It needs to be possible to determine the lineage of a merged item i.e. to know from which base items the merged item is derived. The merge function is expected to be commutative and associative.

The merge function is expected to be commutative and associative. With commutative we mean that the order in which two items are merged does not matter such that $\text{merge}(i_1, i_2) = \text{merge}(i_2, i_1)$. With associative we mean that the order in which we merge multiple items does not matter such that $\text{merge}(i_1, \text{merge}(i_2, i_3)) = \text{merge}(\text{merge}(i_1, i_2), i_3)$.

4.3 Matching merged items

As can be seen in the signatures of our match and merge function a merged item can be matched with other items. Before we can say something about matching merged items we need to consider the way in which we integrate data sources. We integrate sources pair-wise and additionally assume that at most one source originates from a previous integration step. This means that at most one of the items that are being matched is a merged item.

The question that then arises is how we should match an item with a merged item which consists of two or more base items. We state that the matching score between an item and a merged item can never be lower than the lowest matching score of the matchings between the item and the base items of the merged item. Likewise the matching score can never be greater than the maximum matching score between the item and the base items of the merged item such that

$$\begin{aligned} i_4 &= \text{merge}(i_1, \text{merge}(i_2, i_3)) \\ \text{match}(i_5, i_4) &\geq \min(\{\text{match}(i_5, i_1), \text{match}(i_5, i_2), \text{match}(i_5, i_3)\}) \\ \text{match}(i_5, i_4) &\leq \max(\{\text{match}(i_5, i_1), \text{match}(i_5, i_2), \text{match}(i_5, i_3)\}) \end{aligned}$$

Given the two threshold values δ_m and δ_d we can have the situation in which some base items are considered to be the same as the item with which we match while other base items are considered to be distinct from the item with which we match. If we consider the minimum matching score between the base items and the item with which we match to be the overall matching score we can have the situation in which both items are considered to be distinct even though some base items are considered to be the same as the item to which we match. If, on the other hand, we consider the maximum matching score to be the overall matching score we can have the situation in which both items are regarded to be the same even though some base items are considered to be distinct from the item to which we match.

Figure 4.1a depicts the matching between a merged item, consisting of base items a and b, and item c. Base item a is considered to be the same as item c. Base item b is considered to be distinct from item c. The minimum matching score is the matching score between base item b and item c. The maximum matching score is the matching score between base item a and item c.

If we take the minimum matching score to be the overall matching score between the merged item and item c then we regard both items as being distinct from each other which is depicted in figure 4.1b. If we take the maximum matching score to be the overall matching score then we regard both items as being the same which is depicted in figure 4.1c. Both situations are undesirable since we either dismiss the fact that some items are the same or the fact that some items are distinct.

Since we cannot give one overall matching score when we match a merged item with an ordinary item we have to redefine the match function. Instead of returning one matching score the match function has to return a pair of

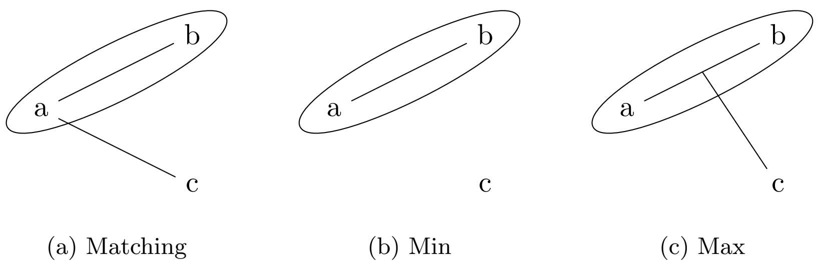


Figure 4.1: Minimum and maximum matching score

matching scores consisting of the minimum matching score and the maximum matching score.

Definition 1 *The match function compares two items from the set of all possible items I such that*

$$match : I \times I \rightarrow [0,1] \times [0,1]. \quad (4.2)$$

The match function relates a pair of items to a pair of matching scores such that $(\min, \max) = match(i_1, i_2)$ *for any* $i_1, i_2 \in I$ *where* $\min \leq \max$. *When i_1 is a merged item then all base items of i_1 are compared to item i_2 which results in a set of matching scores. The match function results in the minimum and maximum matching score of this set. When no merged item is involved the minimum and maximum matching score are the same.*

When the minimum and maximum matching score are the same, such as when we match two ordinary items, we will just give one matching score.

4.4 Matching two sets of items

A data source consists of a set of items. When we perform entity resolution on two such sets we need to match sets of items. We do not match items from the same sources with each other. The reason for this is that items from a source not originating from a previous integration step are assumed to be distinct from each other and items from a source that does originate from a previous integration step have already been compared to each other.

The matching scores between items from different sources can be represented as a graph which we will call a *matching graph.* The vertices in such a graph represent the items. The edges represent matchings between the items. Since we only have edges between items from two different sources this graph is a *bipartite graph.*

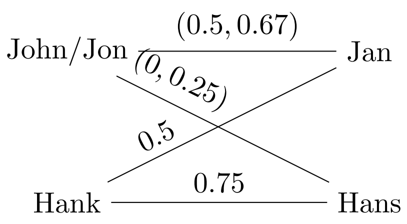


Figure 4.2: Matching graph

Definition 2 *We define a match function which takes two sets of items and results in a matching graph.*

$$Match : \mathbb{P}I \times \mathbb{P}I \rightarrow G. \tag{4.3}$$

A matching graph g is a tuple (V, E, ℓ_m) consisting of a set of vertices V representing all items, a set of edges E between items and match function ℓ_m which relates edges to pairs of matching scores consisting of the minimum and maximum matching score.

The match function for sets of items is defined as

$$Match(I_1, I_2) = (I_1 \cup I_2, I_1 \times I_2, \ell_m). \tag{4.4}$$

The function ℓ_m is defined for all edges connecting two items from different sources such that $\forall i_1 \in I_1 \forall i_2 \in I_2[\ell_m(i_1, i_2) = match(i_1, i_2)]$. Remind that function match is a relation from pairs of items to pairs of matching scores consisting of the minimum and maximum matching score.

Figure 4.2 depicts the resulting matching graph after matching two sources $S_1 = \{\text{John/Jon}, \text{Hank}\}$ and $S_2 = \{\text{Jan}, \text{Hans}\}$. Item John/Jon is a merged item. The matching scores between this merged item and the items of the other sources consists of a minimum and maximum matching score. The minimum matching score between John/Jon and Jan is the matching score between base item John and item Jan which is 0.5. The maximum matching score between John/Jon and Jan is the matching score between base item Jon and item Jan which is 0.67. When there are no merged items involved the minimum and maximum matching scores are the same.

4.5 Annotating edges

Given our threshold values δ_m and δ_d we can make a decision for each edge in our matching graph whether or not two items are the same, distinct or possibly the same. An edge can be annotated as distinct, same, possibly the same or reconsider depending on the matching score between both connected items.

An edge annotated as distinct connects two items which are considered to refer to distinct entities. An edge annotated as same connects two items which are considered to refer to the same entity. An edge annotated as possible connects two items which possibly refer to the same entity but can also refer to distinct entities. When a merged item is matched with an item it is possible that some, but not all, base items are considered to be the same as the item to which we match. In this case we have to reconsider the matching between the base items of the merged item. In the next chapter we will discuss strategies that handle these cases.

Edges are annotated by a function $\text{decision}_{\delta_m,\delta_d} : G \to \widetilde{G}$ which relates a matching graph from the set of all possible matching graphs G to an annotated matching graph from the set of all possible annotated matching graphs $\widetilde{G}$.

Definition 3 *We define a function $decision_{\delta_m,\delta_d} : G \to \widetilde{G}$ which is a relation from matching graphs to annotated matching graphs. This function annotates each edge of a matching graph as either distinct d, same s, possibly the same p or reconsider r given two threshold values δ_m and δ_d such that*

$$decision_{\delta_m,\delta_d}(g) = (V, E, \ell_m, \ell_d). \tag{4.5}$$

The matching graph g is defined as (V, E, ℓ_m). The function $\ell_d : E \to \{d, s, p, r\}$, which is a relation from edges to the set $\{d, s, p, r\}$, is defined for all edges E of graph g such that

$$\ell_d(e) = \begin{cases} d & \textit{if } \max \leq \delta_d \\ s & \textit{if } \min \geq \delta_m \\ p & \textit{if } \max = \min \wedge \delta_d < \max < \delta_m \\ r & \textit{if } \max \neq \min \wedge \min < \delta_m \wedge \max > \delta_d \end{cases} \tag{4.6}$$

where $(\min, \max) = \ell_m(e)$.

Furthermore it is convenient to have separate sets for each kind of edge. For this we define sets E_s, E_d, E_p and E_r which contain all edges annotated as same, distinct, possible or reconsider respectively.

Definition 4 *We define sets E_s, E_d, E_p and E_r which contain all edges annotated as same, distinct, possible or reconsider respectively given an annotated matching graph (V, E, ℓ_m, ℓ_d) such that*

$$E_s = \{e | e \in E \wedge \ell_d(e) = s\} \tag{4.7}$$
$$E_d = \{e | e \in E \wedge \ell_d(e) = d\} \tag{4.8}$$
$$E_p = \{e | e \in E \wedge \ell_d(e) = p\} \tag{4.9}$$
$$E_r = \{e | e \in E \wedge \ell_d(e) = r\} \tag{4.10}$$

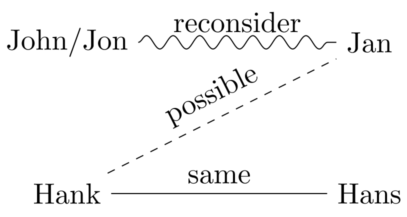


Figure 4.3: Annotated matching graph

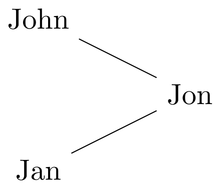


Figure 4.4: Matching conflict

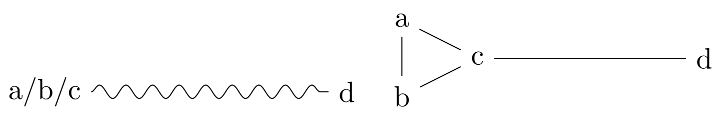
Given our decision function and threshold values $\delta_d = 0.6$ and $\delta_m = 0.7$ we can annotate the matching graph depicted in figure 4.2. The annotated matching graph is depicted in figure 4.3. Edges annotated as distinct are not shown in the graphical representation of the matching graph. A solid line between two items represents an edge annotated as same. A twisting line between two items represents an edge annotated as reconsider. A dashed line between two items represents an edge annotated as possible. Note that an edge annotated as reconsider has to be connected to a merged item.

4.6 Repairing conflicts

There are situations in which we derive at a matching graph that contains matching conflicts. Matching conflicts occur when multiple edges are annotated in such a way that there is a contradiction in the matching graph.

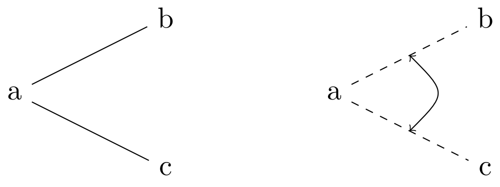
Consider the annotated matching graph depicted in figure 4.4 which is the result of matching items from two sources $S_1 = \{\text{John}, \text{Jan}\}$ and $S_2 = \{\text{Jon}\}$. Assume that item John and item Jan originate from the same source and can therefor never refer to the same entity. Because John is considered to be the same as Jon and Jan is considered to be the same as Jon we derive at a contradiction (John = Jon and Jon = Jan but John $\neq$ Jan.)

Besides matching conflicts between ordinary items we also have to deal with reconsider edges. If we consider the base items of a merged item and the ordinary item connected to an edge annotated as reconsider we also have matching conflicts. Consider figure 4.5 which depicts the matching between the base items of a merged item and an ordinary item. There are matching conflicts between items a, c and d and items b, c and d.



(a) Matching graph (b) Matching between base items and ordinary item

Figure 4.5: Matching conflicts originating from reconsider edge



(a) Matching conflict (b) Dependency between edges

Figure 4.6: Repaired matching graph

We assume there is a function $\text{repair}_s : \widetilde{G} \rightarrow \widehat{G}$ which is a relation from annotated matching graphs to repaired matching graphs. A repaired matching graph does not contain matching conflicts. An edge annotated as reconsider is also considered to be a matching conflict. In the next chapter we define matching conflicts and discuss strategies that can be used to resolve these conflicts.

The repair function can also add *dependencies* between edges in a repaired matching graph. A dependency between some edges means that at least one edge has to be annotated as true in any given possible world. Edges that are involved in a dependency cannot all be distinct. Consider the situation depicted in figure 4.6a. If we state that either items a and b are the same or items a and c are the same we cannot have the situation in which all items are considered to be distinct. A dependency between both edges, which is depicted in figure 4.6b, prevents the situation in which all items are considered to be distinct. The possibility in which both edges are annotated as same is a matching conflict and will not result in a valid possible world.

Assumption 3 *We assume there is a repair_s function which repairs, given a repair strategy s, matching conflicts in a matching graph. The function $\text{repair}_s : \widetilde{G} \rightarrow \widehat{G}$ is a relation from annotated matching graphs to repaired matching graphs. A repaired matching graph is defined as (V, E, ℓ_d, D) in which $D \in \mathbb{PP}E$ represents dependencies between edges. A dependency between edges means that at least one edge should be annotated as same in any given possible world.*

4.7 Enumerate possibilities

When an edge is annotated as possible we have to consider the possibility in which the edge would have been annotated as same and the possibility in which the edge would have been annotated as distinct. For each of these possibilities we will get a new matching graph. A matching graph that contains edges annotated as possible is considered to be an *uncertain matching graph.* A matching graph that does not contain edges annotated as possible is considered to be a *certain matching graph*

Consider the repaired matching graph depicted in figure 4.7a. When we consider all possibilities resulting from considering each edge annotated as possible as either being the same or as being distinct we derive at the eight possibilities depicted in figure 4.7b. Not all possibilities are valid. Some contain matching conflicts whilst others do not satisfy dependencies between edges.

A possibility is valid when is does not contain matching conflicts and satisfies all dependencies between edges. With matching conflicts we mean items that are considered to be the same as two or more other items that are distinct from each other. A certain matching graph satisfies all dependencies when at least one edge, of the edges which are dependent on each other, is annotated as same.

Definition 5 *We define a function conflicting* : $\bar{G} \rightarrow \{\bot, \top\}$ *which is a relation from certain matching graphs to the boolean domain. A certain matching graph is conflicting if it contains three items* i_1, i_2 *and* i_3 *such that* i_1 *is considered to be the same as* i_2 *and* i_3 *but in which* i_2 *is considered to be distinct from* i_3 *i.e.* $i_2 = i_1$ *and* $i_1 = i_3$ *but* $i_2 \neq i_3$.

$$\mathit{conflicting}(\bar{g}) = \forall i_1, i_2, i_3 \in V[\ell_d(i_1, i_2) = s, \ell_d(i_1, i_3) = s, \ell_d(i_2, i_3) = d] \tag{4.11}$$

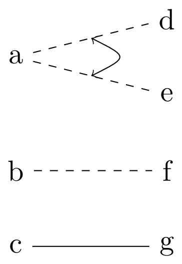
where $\bar{g}$ *is defined as* (V, E, ℓ_d, D).

Definition 6 *We define a function validdeps* : $\bar{G} \rightarrow \{\bot, \top\}$ *which is a relation from certain matching graphs to the boolean domain. A certain matching graph is valid with respect to dependencies when at least one edge, involved in a group of edges which are dependent on each other, is annotated as same such that*

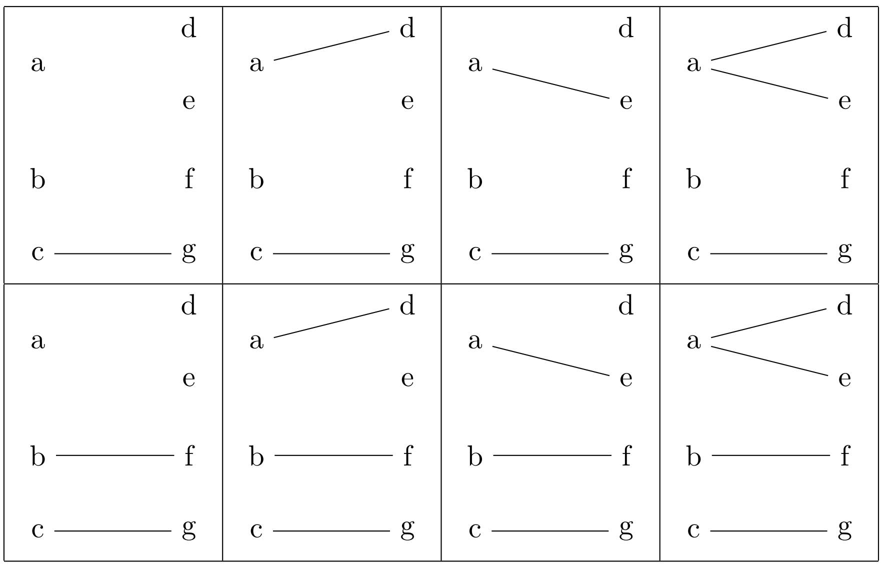
$$\mathit{validdeps}(\bar{g}) = \forall E' \in D \exists e \in E'[e \in E_s] \tag{4.12}$$

where $\bar{g}$ *is defined as* (V, E, ℓ_d, D).

Now we have defined which certain matching graphs are valid we can define the enumerate function. The enumerate function enumerate : $\widehat{G} \rightarrow \mathbb{P}\bar{G}$ is a relation from repaired matching graphs to certain matching graphs and



(a) Repaired matching graph



(b) Enumerated possibilities

Figure 4.7: Enumerating possibilities

enumerates all possible matching graphs by considering each edge annotated as possible as either same or distinct. A certain matching graph is defined as (V, E, ℓ_d, D) and does not contain edges annotated as possible or reconsider.

Definition 7 *We define a function enumerate* $: \widehat{G} \to \mathbb{P}\bar{G}$ *which is a relation from repaired matching graphs to certain matching graphs. This function enumerates all certain matching graphs by considering each edge annotated as possible as either same or distinct, while filtering out graphs that are conflicting or not satisfying edge dependencies, such that*

$$\begin{aligned} \mathit{enumerate}(\hat{g}) = \quad & \{g' | C \in E_p \to \{d, s\} \\ & \wedge g' = (V, E, \ell_d \rceil C, D) \\ & \wedge \neg \mathit{conflicting}(g') \\ & \wedge \mathit{validdeps}(g') \end{aligned} \tag{4.13}$$

where $\hat{g} = (V, E, \ell_d, D)$*. All edges which are annotated as possible are considered to be either same or distinct. This is done by* $\ell_d \rceil C$ *which overwrites the function* ℓ_d *for all edges annotated as possible such that it relates to the set* $\{d, s\}$*.*

4.8 Merge items in a matching graph

After all certain matching graphs have been enumerated we can merge items deemed to represent the same entity i.e. are connected by edges annotated as same. We define a function Merge : $\bar{G} \to \mathbb{P}I$ which is a relation from certain matching graphs to the power set of items.

Definition 8 *We define a function Merge* $: \bar{G} \to \mathbb{P}I$ *which is a relation from certain matching graphs to the power set of items such that*

$$\begin{aligned} E_m &= \{(i_1, i_2) \in \mathit{domain}(\ell_d) | \ell_d(i_1, i_2) = s\} \\ V_m &= \bigcup_{(i_1,i_2) \in E_m} \{i_1, i_2\} \\ V_d &= V \setminus V_m \\ \mathit{Merge}(\bar{g}) &= \{\mathit{merge}(i_1, i_2) | (i_1, i_2) \in E_m\} \cup V_d \end{aligned} \tag{4.14}$$

where $\bar{g}$ *is defined as* (V, E, ℓ_d, D)*.*

4.9 Integrate two sets of items

We can now define the integrate function which integrates two sources. Since it is possible that a source originated from a previous integration step it can consist of many possible worlds. Each of these possible worlds has to be integrated with the items of the other source. At most one of the sources may originate from a previous integration step.

Definition 9 *We define a function integrate* $: S \times S \rightarrow S$ *which is a relation from pairs of sources to sources. A source is from the set* $\mathbb{PP}I$*. Two sources are integrated by considering the items from a possible world from one source and all items from the other source and perform entity resolution on these items such that:*

$$\begin{aligned} &\mathit{integrate}(S_1, S_2) = \{\mathit{Merge}(\bar{g})| \\ &\bar{g} \in \bigcup_{I_1 \in S_1\, I_2 \in S_2} \mathit{enumerate} \circ \mathit{repair}_s \circ \mathit{decision}_{\delta_m, \delta_d} \circ \mathit{Match}(I_1, I_2)\}. \end{aligned} \tag{4.15}$$

Chapter 5

Strategies

As we have seen in the previous chapters there are cases in which entity resolution is problematic due to matching conflicts. This chapter defines which cases are problematic, why they are problematic and discusses strategies to resolve them.

In the formalization of the entity resolution problem we abstracted away from handling problematic situations by defining a repair function. The repair function has to resolve matching conflicts and has to guarantee associative behavior of the overall integration function.

Consider sources S_1, S_2 and S_3 that we want to integrate. Since we integrate pair-wise we start by integrating S_1 and S_2. The result of integrating S_1 and S_2 is a source S_{12} consisting of one or more possible worlds. We then integrate sources S_{12} and S_3. Matching S_{12} and S_3 results in a matching graph g_{123} which is then annotated by the decision function resulting in annotated matching graph $\tilde{g}_{123}$. In order for the integrate function to be associative $\text{repair}_s(\tilde{g}_{123}) = \text{repair}_s(\tilde{g}_{132}) = \text{repair}_s(\tilde{g}_{231})$ has to hold. Two repaired matching graphs $\hat{g}_1, \hat{g}_2$ are considered to be the same if the possible worlds that result after enumerating $\hat{g}_1, \hat{g}_2$ are the same.

This chapter describes four strategies that can be used to resolve matching conflicts and guarantee associative behavior of the integrate function. Before discussing the strategies we first define what the problematic cases are.

5.1 Problematic matching cases

As we have seen in the previous chapter we sometimes derive at situations in which an item is considered to be the same as two or more other items which are themselves distinct.

Definition 10 *A matching conflict is a sub-graph in which one item is connected by edges annotated as same to two or more items which are distinct* *from each other, i.e. are not connected to each other by edges annotated as same.*

A matching conflict consists of at least three items. It is possible that some items originate from the same source. Note that these items can never participate in the same merged item.

Items connected by a reconsider edge can also contain matching conflicts. When an item is considered to be the same as some, but not all, base items of a merged item then both items are connected by a reconsider edge. Consider the matching between an item and the base items of a merged item. Since not all items are considered to be the same it is possible that some of these items are involved in a matching conflict.

The afore mentioned situation can lead to non-associative integration. Consider the three items depicted in figure 5.1a which originate from three different sources. If we first integrate the sources containing items a and b we get a possible worlds in which items a and b are merged. When we then see the source containing items c we derive at the possible world depicted in figure 5.1b in which item c is connected by a reconsider edge to the merged item consisting of items a and b. This possible world differs from the possible world that results if we first integrate the sources containing items b and c, as depicted in figure 5.1c. If we simply change the edge annotated as reconsider to an edge annotated as distinct, i.e. we do not consider a repair strategy for items connected to a reconsider edge, we see that the order in which we integrate sources influences the integrated result i.e. integration is not associative.

For items connected by a reconsider edge we therefore have to consider all base items. We then have to match all items and annotate the edges connecting these items accordingly. We also have to match the item that was connected to the merged item by a reconsider edge with all base items.

Note that base items can be connected by edges annotated as either same, possible or even distinct. To see why base items can be connected by an edge annotated as distinct consider the matching shown in figure 5.1a. Depending on the strategy to resolve the matching conflict we can regard the edge between items a and c, which is annotated as distinct, as incorrect. If we annotate this edge as same the matching conflict is resolved and all items can be merged into one item even though base items a and c are considered to be distinct by the decision function.

When we consider three items origination from three different sources we have ten possible ways in which the items can match with each other. All ten matching cases are depicted in table 5.1. Matching cases 0, 1, 2, 4, 5 and 7 are also possible when items b and c originate from the same source whereas cases 3, 6, 8 and 9 are only possible when all three items originate from different sources. Note that matching case 2 contains a matching conflict.

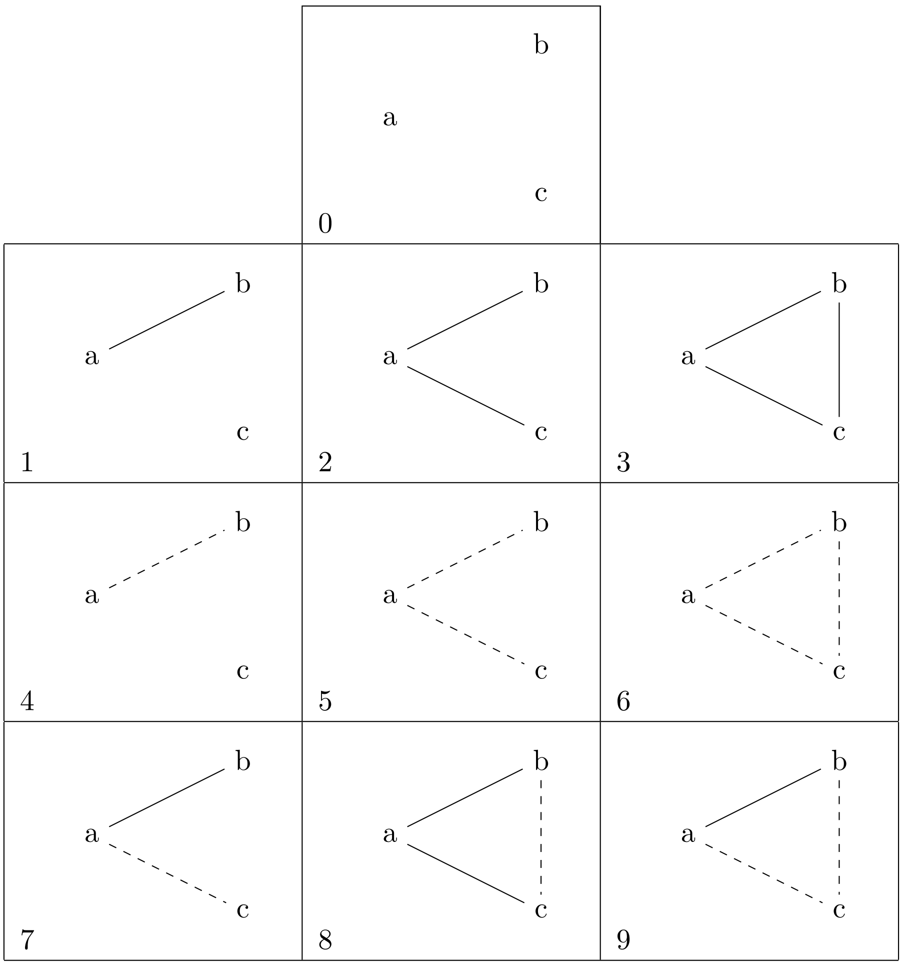


Table 5.1: Matching cases

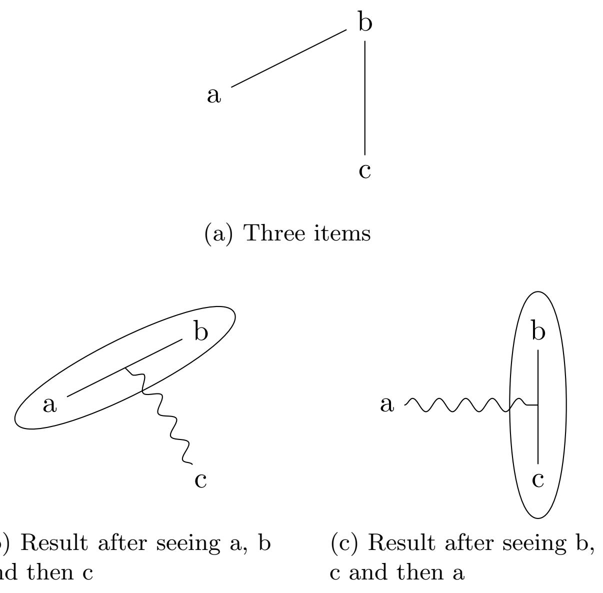

a
b
c
(a) Three items
a
b
c
(b) Result after seeing a, b and then c
a
b
c
(c) Result after seeing b, c and then a

Figure 5.1: Potential non-associative integration

An annotated matching graph can only contain matching cases 0, 1, 2, 4, 5 and 7 since there will always be two items residing in the same source which are therefor considered to be distinct from each other. All matching cases can be found when we consider the matching between base items and the item with which some of the base items match.

To show that a strategy resolves matching conflicts and that they lead to associative behavior of the integrate function we integrate cases 2, 5, 6, 7, 8 and 9 in all possible orders. We can integrate three sources $S_1 = \{a\}, S_2 = \{b\}$ and $S_3 = \{c\}$ in three possible orders

$$\begin{gathered}\text{integrate}(\text{integrate}(S_1, S_2), S_3)\\ \text{integrate}(\text{integrate}(S_1, S_3), S_2)\\ \text{integrate}(\text{integrate}(S_2, S_3), S_1).\end{gathered}$$

5.2 Approach

When only a small amount of items are involved in a matching conflict or are connected by a reconsider edge it is inefficient to use a repair strategy on all items. We only have to apply the repair strategy on clusters of items that are involved in a matching conflict or in which some items are connected by a reconsider edge.

5.2.1 Cluster items

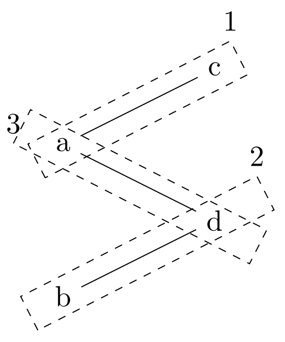
The first step in resolving all matching conflicts is to find all clusters of items that are involved in a matching conflict or contain edges annotated as reconsider. Algorithm 1 shows how clusters of items can be found. The algorithm addresses each edge contained in a decided matching graph. When an edge, connected to items i_1 and i_2, is annotated as either same, possible or reconsider the algorithm searches for clusters containing i_1 or i_2. If it finds two such clusters then these clusters are unioned otherwise a new cluster consisting of items i_1 and i_2 is constructed.

```
$S \leftarrow \emptyset$ {$S$ is a set of clusters, $S : \mathbb{PP}I$}
**for** $(i_1, i_2) \in E$ **do** {$E$ is the set of edges from graph $g : (V, E, \ell_m, \ell_d)$}
  **if** $\ell_d(i_1, i_2) \in \{s, p, r\}$ **then**
    $C_{i_1} \leftarrow \{i_1\}$
    $C_{i_2} \leftarrow \{i_2\}$
    **for** $C \in S$ **do**
      **if** $C \cap C_{i_1} \neq \emptyset$ **then**
        $C_{i_1} \leftarrow C$
      **end if**
      **if** $C \cap C_{i_2} \neq \emptyset$ **then**
        $C_{i_2} \leftarrow C$
      **end if**
    **end for**
    **if** $C_{i_1} \in S$ **then**
      remove $C_{i_1}$ from $S$
    **end if**
    **if** $C_{i_2} \in S$ **then**
      remove $C_{i_2}$ from $S$
    **end if**
    add $C_{i_1} \cup C_{i_2}$ to $S$
  **end if**
**end for**
```

Algorithm 1: Clustering algorithm

To see how the algorithm works consider the matching graph shown in figure 5.2a. The order in which the algorithm encounters the edges is also shown in this figure. Figure 5.2b shows which clusters have been found after each encountered edge.

Some of the clusters contain items connected by a reconsider edge. In these clusters the merged items need to be expanded into their base items in order to resolve matching conflicts between the base items and the item to which the merged item is connected by a reconsider edge. Figure 5.3 depicts how a merged item, which is connected to another item by a reconsider edge,



(a) Matching graph

encountered edge	clusters
(a, c)	$\{\{a, c\}\}$
(b, d)	$\{\{a, c\}\,\{b, d\}\}$
(a, d)	$\{\{a, c, b, d\}\}$

(b) Clusters at each step of the algorithm

Figure 5.2: Construction of clusters

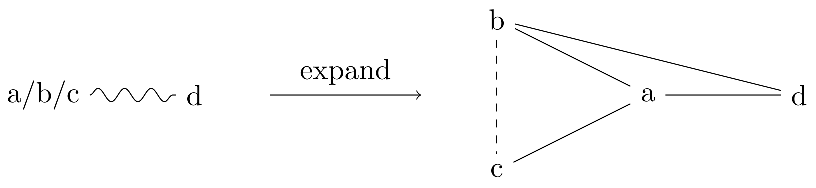


Figure 5.3: Expanding a merged item

is expanded into its base items. The edges between the items are annotated by the decision function ℓ_d.

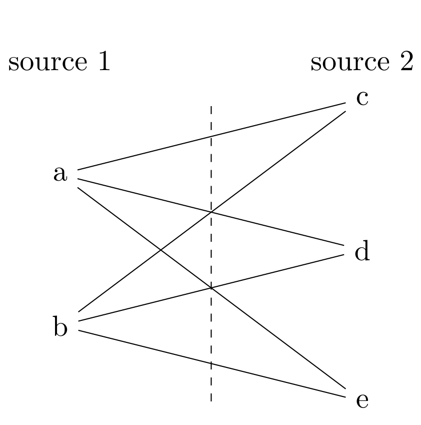
Clusters containing only two items, which are not connected by a reconsider edge, can be disregarded since they can never contain a matching conflict.

After all clusters have been found a repair strategy can be used to transform each cluster into clusters that do not contain matching conflicts.

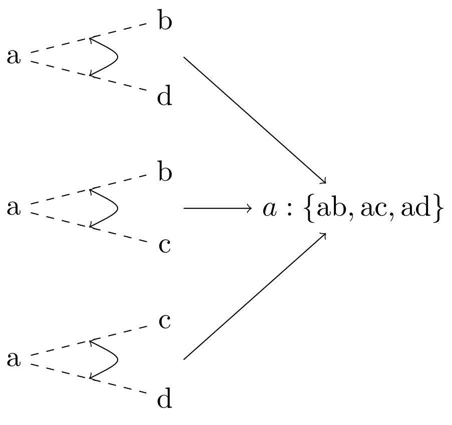
5.2.2 Dependencies between edges

A repair strategy can add dependencies between edges. Consider the matching shown in figure 5.4a. A repair strategy considers all combinations of three items. When a dependency is added between two edges it is possible that one or both edges are already involved in other dependencies. We group dependencies around the item which is connected to both edges. Figure 5.4b shows all dependencies between edges that are connected to item a.

In some cases multiple dependencies prevent a repaired matching graph



(a) Matching



(b) Dependencies from item a

Figure 5.4: Adding dependencies

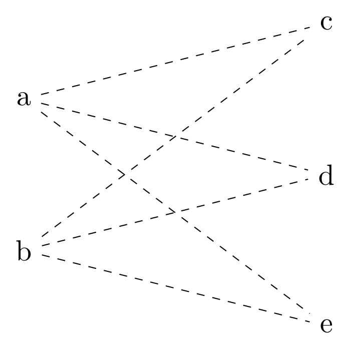
to result in any possible world. Consider the matching in figure 5.5a which contains the dependencies depicted in figure 5.5b. Remember that at least one of the edges involved in a dependency has to be annotated as same in any given possible world.

When we try to annotate at least one edge of all dependencies as same we see that this will always lead to a matching conflict in the certain matching graph. The reason for this is that source 2 contains more items than source 1. There will always be one item of source 2 that is not connected to an item of source 1.

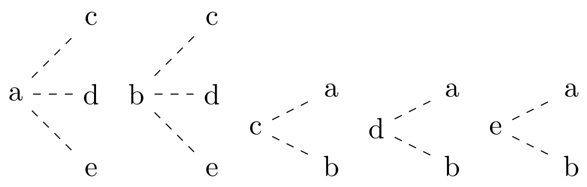
Note that some of the dependencies contain edges already contained in other dependencies. If the edges involved in a dependency are already contained in other dependencies we can discard this dependency. For the dependencies shown in figure 5.5b we can discard the dependencies grouped around item c, d and e, i.e. the dependencies involving two edges. When we only consider the dependencies grouped around items a and b we see that we there are six possible worlds that do consistent with the dependencies and are not conflicting.

5.3 All distinct strategy

The all distinct strategy annotates all edges involved in a matching conflict as distinct. Consider the matching case depicted in figure 5.1a containing three items originating from different sources. After integrating the two sources containing items a and b we derive at a possible world containing a merged item consisting of base items a and b. When we then see the third source, containing item c, we derive at the matching graph depicted in figure 5.1b in which item c is connected to the merged



(a) Repaired matching graph



(b) All dependencies between edges

Figure 5.5: Dependencies between edges

item by a reconsider edge. Note that the reconsider edge resulted because $\ell_m(\text{merge}(a, b), c) = (\min, \max)$ in which $\min \leq \delta_d$ and $\max \geq \delta_m$ resulting in $\ell_d(merge(a, b), c) = r$.

After expanding the merged item into base items a and b we consider the matching graph between all items. This corresponds to the matching depicted in figure 5.1a. Since the matching graph contains a matching conflict we have to change one or more edges to resolve the conflict. The all distinct strategy changes all edges involved in matching conflicts to edges annotated as distinct. Effectively this means that all items involved in a matching conflicts are considered to refer to distinct entities.

Graph transformations made by the all the distinct strategy are shown in table 5.2.

Figure 5.6 shows an example in which the three sources are integrated. The matching between all items is shown in figure 5.6a. When we first see the sources containing items a and b we get the situation depicted in figure 5.6b. If the sources containing items b and c are seen first we derive at the situation depicted in figure 5.6c. The order in which the sources containing items a and c are seen first is the same as the situation in which the sources containing items a and b are seen first. The order in which the sources are integrated has no influence on the resulting possible worlds, i.e. the used strategy is associative.

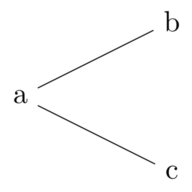
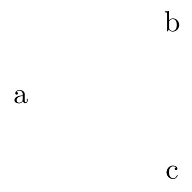
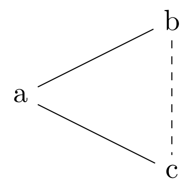
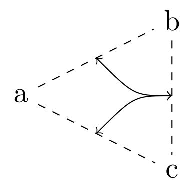
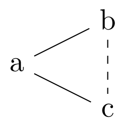
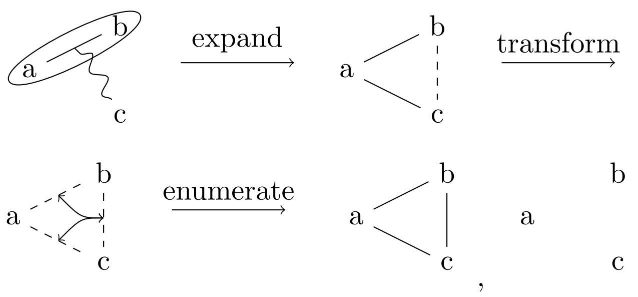
Matching	Transformation
	
	

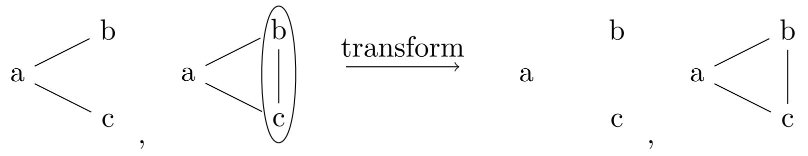
Table 5.2: Graph transformations for the all distinct strategy



(a) matching



(b) items a and b seen first



(c) items b and c seen first

Figure 5.6: Example of the all distinct strategy

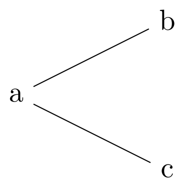
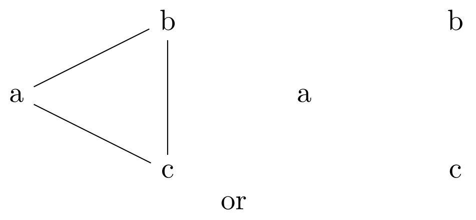
Matching	Transformation
	

Table 5.3: Graph transformations for the all the same strategy

5.4 All the same strategy

Another approach to resolve matching conflicts is to consider all edges involved in a matching conflict as edges annotated as same. This can lead to situations in which two items originating from the same source are involved in a merged item. This conflicts with the assumption that items from the same source are always referring to distinct entities.

When we only look at items that are considered to be the same then this strategy is transitive in the sense that when item a is considered to be the same as item b and item b is considered to be the same as item c then items a and c have to be the same too.

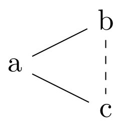
Graph transformation made by the all the same strategy are shown in table 5.3. Note that, based on the assumption that two items originating from the same source can never refer to the same entity, there is the possibility in which items b and c can never be connected by an edge annotated as same. This possibility occurs when both items originate from the same source. When two of the three items originate from the same source all items are considered to be distinct from each other.

The example shown in figure 5.7 shows that the order in which the sources are integrated has no influence on the resulting possible worlds, i.e. the used strategy is associative.

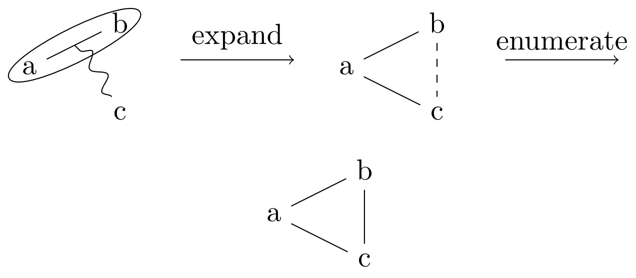
5.5 All possible strategy

If we assume that our assumption always holds, we can never have the situation in which two items originating from the same source are considered to be the same. It is also unlikely that all edges previously annotated as possible or same are all incorrect. Because of these two observations we reject the all distinct and all the same strategies. A better approach is to doubt all edges annotated by the decision function. We do not doubt distinct-edges between items that originate from the same source.

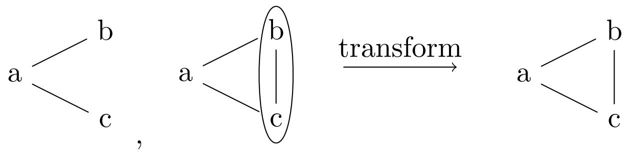
The all possible strategy changes all edges, involved in a conflicting situation, between items originating from different sources to edges annotated



(a) matching



(b) items a and b seen first



(c) items b and c seen first

Figure 5.7: Example of the all the same strategy

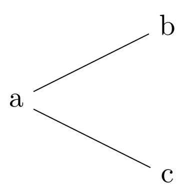
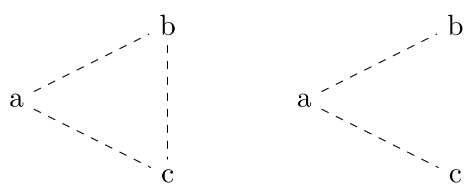
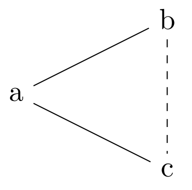
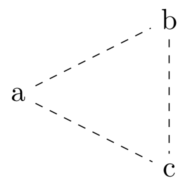
Matching	Transformation
	 or
	

Table 5.4: Graph transformation for the all possible strategy

as possible.

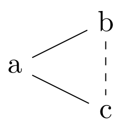
Graph transformation made by the all possible strategy are shown in table 5.4. Note that when items b and c originate from different sources there can never be an edge annotated as possible between both items.

The example shown in figure 5.8 shows that the order in which the sources are integrated has no influence on the resulting possible worlds, i.e. the used strategy is associative.

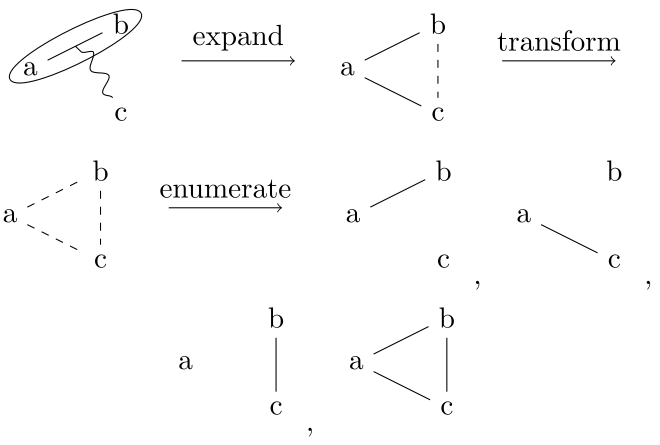
5.6 Reconsider strategy

The problem with the all possible strategy is that edges annotated as same can all become distinct when they are involved in a matching conflict even though it is more likely that at least one edge annotated as same is correct. Consider two source $S_1 = \{John, Jan\}$ and $S_2 = \{Jon\}$ where John and Jon are considered to be the same and Jan and Jon are also considered to be the same. In this case we have a matching conflict since only one of the edges annotated as same can reflect the real world situation. Either John and Jon are the same person or Jan and Jon are the same person. The all possible strategy considers the possibility in which John, Jan and Jon are three distinct persons which is unlikely since this would mean that both edges annotated as same are incorrect. The reconsider strategy adds dependencies between edges so that the case in which all items are considered to be distinct is not possible.

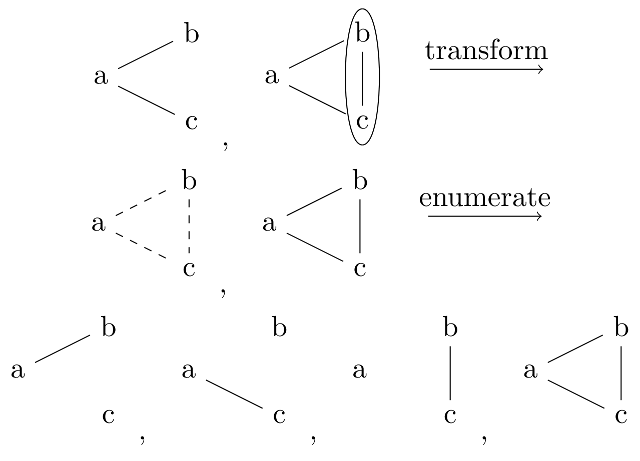
Graph transformation made by the reconsider strategy are shown in table 5.5. Note that when items b and c originate from different sources there can never be an edge annotated as possible between both items.



(a) matching



(b) items a and b seen first



(c) items b and c seen first

Figure 5.8: Example of the all possible strategy

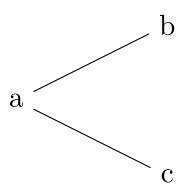
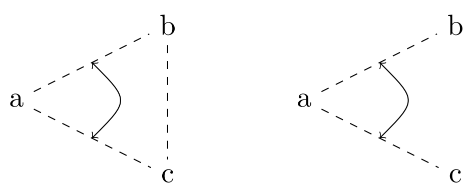
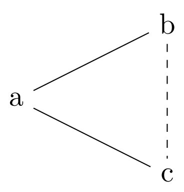
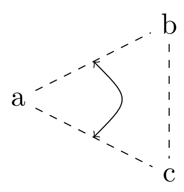
Matching	Transformation
	 or
	

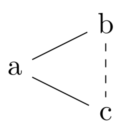
Table 5.5: Graph transformation reconsider strategy

The example shown in figure 5.9 shows that the order in which the sources are integrated has no influence on the resulting possible worlds, i.e. the used strategy is associative.

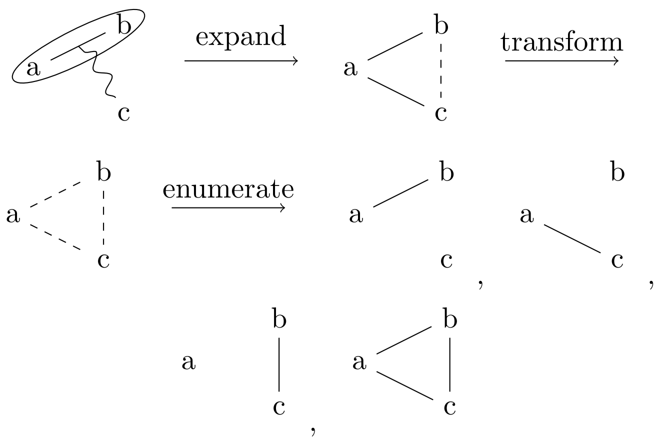
5.7 Comparison between strategies

The all distinct, and the all the same strategy are expected to result in the least number of possible worlds. No uncertainty is introduced when matching conflicts are resolved, i.e. no edges annotated as possible are added to the matching graph.

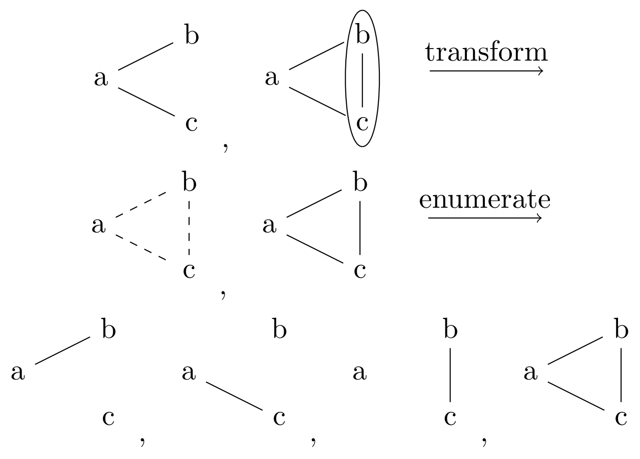
The all possible strategy is expected to result in the most possible worlds since it considers all possible matchings in case of a matching conflict. The reconsider strategy is expected to result in fewer possible worlds compared to the all possible strategy but more than the all distinct strategy and the all the same strategy.



(a) matching



(b) items a and b seen first



(c) items b and c seen first

Figure 5.9: Example of the reconsider strategy

Chapter 6

Conclusions and future work

This research dealt with the problem of multi-source entity resolution in the setting of pair-wise data integration. Since we do not assume matching to be transitive we found that the main problem was to make multi-source entity resolution associative. The main contributions of this research are: (1) a formal model for multi-source entity resolution and (2) strategies that can be used to resolve matching conflicts in a way that renders multi-source entity resolution to be associative.

The main research question that was addressed was

How to perform multi-source entity resolution using the possible worlds semantics in the setting of pair-wise data integration that is associative?

We answered this question by defining a formal model which describes how multi-source entity resolution in the setting of pair-wise data integration works. We used the possible worlds semantics to handle uncertainty originating from possible matches which is based on the approach discussed in [5]. In our model we abstracted away from the details of handling matching conflicts to allow different strategies to resolve matching conflicts. We considered the match and merge functions to be black boxes, which is similar to the approach discussed in [2]. We do, however, expect the match function to be idempotent and commutative and the merge function to be commutative and associative. Our model is generic enough to allow different match and merge function as well as allowing different strategies to resolve matching conflicts.

In our model matches between items are represented in a matching graph. We described four strategies that can be used to resolve matching conflicts in such a graph: (1) all distinct strategy, (2) all the same strategy, (3) all possible strategy and the (4) reconsider strategy. Each strategy resolves matching conflicts by performing simple graph transformations. We used small examples in which we integrated three sources in different orders to show that the strategies resulted in associative behavior of the integrate function.

Existing approaches to entity resolution often consider matching to be transitive. Because of the transitivity property matching conflicts do not occur. This research does not expect the match function to be transitive. Furthermore we also assume items originating from different sources to be distinct from each other, i.e. two items originating from the same source can never refer to the same real world entity.

6.1 Future work

There are still some things left that can be considered to be future work. This research did not formally proof that the integrate function is associative given a strategy to resolve matching conflicts. A formal proof is necessary to guarantee associativity of the integrate function under all circumstances.

Another thing that can be considered to be future work is confidence score calculation. Given that each item is associated with a confidence score and since we assume that information about the lineage of each item is available it should be possible to calculate a confidence score for each item in any given possible world. These confidence scores can be used to calculate an overall confidence score for each possible world which would enable us to rank all possible worlds according to these scores. How to incorporate matching scores into the model that we presented in this research is still an open question. The approach discussed in [5] incorporates confidence scores but does not consider multiple sources. The approach discussed in [11] also incorporates confidence scores but does not use the possible worlds semantics.

It is also interesting to look at the storage of the integrated result. Since we abstracted away from implementation details we did not discuss how items should be merged. Uncertain lineage databases [3] are capable of handling both lineage and confidence score computation. How to use such a database system to store an uncertain integrated result and how it can be used to compute a confidence score for each item is also an open question.

Once there is a prototype system that uses our approach to multi-source entity resolution it is interesting to see how the proposed strategies perform, with respect to precision and recall, compared to existing approaches to entity resolution. An obstacle here is that we use the possible worlds semantics which means that we potentially have more than one integrated result. By using metrics discussed in [4] we might be able to compare our approach to already existing approaches.

Appendix A

Formalization of an example

This appendix contains a formalization of the example discussed in the informal problem definition.

In this example we integrate three sources S_1, S_2 and S_3 each containing one item. The sources, threshold values and matching scores are shown below:

$$\begin{gathered} S_1 = \{\text{John}\}, S_2 = \{\text{Jon}\}, S_3 = \{\text{Jan}\} \\ \delta_d = 0.60, \delta_m = 0.70 \\ \text{match}(\text{John}, \text{Jon}) = (0.75, 0.75) \\ \text{match}(\text{John}, \text{Jan}) = (0.5, 0.5) \\ \text{match}(\text{Jon}, \text{Jan}) = (0.67, 0.67) \end{gathered}$$

$$
\begin{aligned}
g = \mathrm{Match}(\{\mathrm{John}\}, \{\mathrm{Jon}\}) = \; & (\{\mathrm{John}, \mathrm{Jon}\}, \\
& \{(\mathrm{John}, \mathrm{Jon})\}, \\
& \{((\mathrm{John}, \mathrm{Jon}), (0.75, 0.75))\})
\end{aligned}
$$

$$
\begin{aligned}
\tilde{g} = \mathrm{decision}_{\delta_m, \delta_d}(g) = \; & (\{\mathrm{John}, \mathrm{Jon}\}, \\
& \{(\mathrm{John}, \mathrm{Jon})\}, \\
& \{((\mathrm{John}, \mathrm{Jon}), (0.75, 0.75))\}, \\
& \{((\mathrm{John}, \mathrm{Jon}), s)\})
\end{aligned}
$$

$$\hat{g} = \mathrm{repair}_s(\tilde{g}) = \tilde{g}$$

$$\bar{g} = \mathrm{enumerate}(\hat{g}) = \{\hat{g}\}$$

$$\mathrm{integrate}(\{\{\mathrm{John}\}\}, \{\{\mathrm{Jon}\}\}) = \{\mathrm{Merge}(\bar{g}) | \bar{g} \in \bar{g}\} = \{\{\mathrm{merge}(\mathrm{John}, \mathrm{Jon})\}\}$$

$$
\begin{aligned}
& g = \mathrm{Match}(\{\mathrm{merge}(\mathrm{John}, \mathrm{Jon})\}, \{\mathrm{Jan}\}) = \\
& (\{\mathrm{merge}(\mathrm{John}, \mathrm{Jon}), \mathrm{Jan}\}, \\
& \{(\mathrm{merge}(\mathrm{John}, \mathrm{Jon}), \mathrm{Jan})\}, \\
& \{((\mathrm{merge}(\mathrm{John}, \mathrm{Jon}), \mathrm{Jan}), (0.5, 0.5))\})
\end{aligned}
$$

$$
\begin{aligned}
\tilde{g} = \mathrm{decision}_{\delta_m, \delta_d}(g) = \; & (\{\mathrm{merge}(\mathrm{John}, \mathrm{Jon}), \mathrm{Jan}\}, \\
& \{(\mathrm{merge}(\mathrm{John}, \mathrm{Jon}), \mathrm{Jan})\}, \\
& \{((\mathrm{merge}(\mathrm{John}, \mathrm{Jon}), \mathrm{Jan}), (0.5, 0.5)\}, \\
& \{((\mathrm{merge}(\mathrm{John}, \mathrm{Jon}), \mathrm{Jan}), d)\})
\end{aligned}
$$

$$\hat{g} = \mathrm{repair}_s(\tilde{g}) = \tilde{g}$$

$$\bar{g} = \mathrm{enumerate}(\hat{g}) = \{\hat{g}\}$$

$$
\begin{gathered}
\mathrm{integrate}(\{\{\mathrm{merge}(\mathrm{John}, \mathrm{Jon})\}\}, \{\{\mathrm{Jan}\}\}) = \{\mathrm{Merge}(\bar{g}) | \bar{g} \in \bar{g}\} = \\
\{\{\mathrm{merge}(\mathrm{John}, \mathrm{Jon}), \mathrm{Jan}\}\}
\end{gathered}
$$

$$
\begin{aligned}
g = \text{Match}(\{\text{John}\}, \{\text{Jan}\}) = \; & (\{\text{John}, \text{Jan}\}, \\
& \{(\text{John}, \text{Jan})\}, \\
& \{((\text{John}, \text{Jan}), (0.5, 0.5))\})
\end{aligned}
$$

$$
\begin{aligned}
\tilde{g} = \text{decision}_{\delta_m,\delta_d}(g) = \; & (\{\text{John}, \text{Jan}\}, \\
& \{(\text{John}, \text{Jan})\}, \\
& \{((\text{John}, \text{Jan}), (0.5, 0.5))\}, \\
& \{((\text{John}, \text{Jan}), d)\})
\end{aligned}
$$

$$\hat{g} = \text{repair}_s(\tilde{g}) = \tilde{g}$$

$$\bar{g} = \text{enumerate}(\hat{g}) = \{\hat{g}\}$$

$$\text{integrate}(\{\{\text{John}\}\}, \{\{\text{Jan}\}\}) = \{\text{Merge}(\bar{g}) | \bar{g} \in \bar{g}\} = \{\{\text{John}, \text{Jan}\}\}$$

$$
\begin{aligned}
g = \text{Match}(\{\text{John}, \text{Jan}\}, \{\text{Jon}\}) = \; & (\{\text{John}, \text{Jan}, \text{Jon}\}, \\
& \{(\text{John}, \text{Jon}), (\text{Jan}, \text{Jon})\}, \\
& \{((\text{John}, \text{Jon}), (0.75, 0.75)), ((\text{Jan}, \text{Jon}), (0.67, 0.67))\})
\end{aligned}
$$

$$
\begin{aligned}
\tilde{g} = \text{decision}_{\delta_m,\delta_d}(g) = \; & (\{\text{John}, \text{Jan}, \text{Jon}\}, \\
& \{(\text{John}, \text{Jon}), (\text{Jan}, \text{Jon})\}, \\
& \{((\text{John}, \text{Jon}), (0.75, 0.75)), ((\text{Jan}, \text{Jon}), (0.67, 0.67))\}, \\
& \{((\text{John}, \text{Jon}), s), ((\text{Jan}, \text{Jon}), p)\})
\end{aligned}
$$

$$\hat{g} = \text{repair}_s(\tilde{g}) = \tilde{g}$$

$$
\begin{aligned}
\bar{g} = \text{enumerate}(\hat{g}) = \; & (\{\text{John}, \text{Jan}, \text{Jon}\}, \\
& \{(\text{John}, \text{Jon}), (\text{Jan}, \text{Jon})\}, \\
& \{((\text{John}, \text{Jon}), (0.75, 0.75)), ((\text{Jan}, \text{Jon}), (0.67, 0.67))\}, \\
& \{((\text{John}, \text{Jon}), s), ((\text{Jan}, \text{Jon}), d)\})
\end{aligned}
$$

$$
\begin{aligned}
& \text{integrate}(\{\{\text{John}, \text{Jan}\}\}, \{\{\text{Jon}\}\}) = \{\text{Merge}(\bar{g}) | \bar{g} \in \bar{g}\} = \\
& \{\{\text{merge}(\text{John}, \text{Jon}), \text{Jan}\}\}
\end{aligned}
$$

$$
\begin{aligned}
g = \text{Match}(\{\text{Jon}\}, \{\text{Jan}\}) = \ & (\{\text{Jon}, \text{Jan}\},\\
& \{(\text{Jon}, \text{Jan})\},\\
& \{((\text{Jon}, \text{Jan}), (0.67, 0.67))\})\\
\tilde{g} = \text{decision}_{\delta_m,\delta_d}(g) = \ & (\{\text{Jon}, \text{Jan}\},\\
& \{(\text{Jon}, \text{Jan})\},\\
& \{((\text{Jon}, \text{Jan}), (0.67, 0.67))\},\\
& \{((\text{Jon}, \text{Jan}), p)\})\\
\hat{g} = \text{repair}_s(\tilde{g}) = \tilde{g}\\
\bar{g} = \text{enumerate}(\hat{g}) =\\
\{(\{\text{Jon}, \text{Jan}\}, \{(\text{Jon}, \text{Jan})\}, \{((\text{Jon}, \text{Jan}), (0.67, 0.67))\}, \{((\text{Jon}, \text{Jan}), d)\}),\\
(\{\text{Jon}, \text{Jan}\}, \{(\text{Jon}, \text{Jan})\}, \{((\text{Jon}, \text{Jan}), (0.67, 0.67))\}, \{((\text{Jon}, \text{Jan}), s)\})\}\\
\text{integrate}(\{\{\text{Jon}\}, \{\text{Jan}\}\}) = \{\text{Merge}(\bar{g}) | \bar{g} \in \bar{g}\} =\\
\{\{\text{Jon}, \text{Jan}\}, \{\text{merge}(\text{Jon}, \text{Jan})\}\}\\
g_1 = \text{Match}(\{\text{Jon}, \text{Jan}\}, \{\text{John}\}) = \ & (\{\text{Jon}, \text{Jan}, \text{John}\},\\
& \{(\text{Jon}, \text{John}), (\text{Jan}, \text{John})\},\\
& \{((\text{Jon}, \text{John}), (0.75, 0.75)), ((\text{Jan}, \text{John}), (0.5, 0.5))\})\\
\tilde{g}_1 = \text{decision}_{\delta_m,\delta_d}(g_1) = \ & (\{\text{Jon}, \text{Jan}, \text{John}\},\\
& \{(\text{Jon}, \text{John}), (\text{Jan}, \text{John})\},\\
& \{((\text{Jon}, \text{John}), (0.75, 0.75)), ((\text{Jan}, \text{John}), (0.5, 0.5))\},\\
& \{((\text{Jon}, \text{John}), s), ((\text{Jan}, \text{John}), d)\})\\
\hat{g}_1 = \text{repair}_s(\tilde{g}_1) = \tilde{g}_1\\
\text{enumerate}(\hat{g}_1) = \{\hat{g}_1\}\\
g_2 = \text{Match}(\{\text{merge}(\text{Jon}, \text{Jan})\}, \{\text{John}\}) =\\
(\{\text{merge}(\text{Jon}, \text{Jan}), \text{John}\},\\
\{(\text{merge}(\text{Jon}, \text{Jan}), \text{John})\},\\
\{((\text{merge}(\text{Jon}, \text{Jan}), \text{John}), (0.5, 0.5))\})\\
\tilde{g}_2 = \text{decision}_{\delta_m,\delta_d}(g_2) = \ & (\{\text{merge}(\text{Jon}, \text{Jan}), \text{John}\},\\
& \{(\text{merge}(\text{Jon}, \text{Jan}), \text{John})\},\\
& \{((\text{merge}(\text{Jon}, \text{Jan}), \text{John}), (0.5, 0.5))\},\\
& \{((\text{merge}(\text{Jon}, \text{Jan}), \text{John}), d)\})\\
\hat{g}_2 = \text{repair}_s(\tilde{g}_2) = \tilde{g}_2\\
\text{enumerate}(\hat{g}_2) = \{\hat{g}_2\}\\
\bar{g} = \text{enumerate}(\hat{g}_1) \cup \text{enumerate}(\hat{g}_2) =\\
\{(\{\text{Jon}, \text{Jan}, \text{John}\},\\
\{(\text{Jon}, \text{John}), (\text{Jan}, \text{John})\},\\
\{((\text{Jon}, \text{John}), (0.75, 0.75)), ((\text{Jan}, \text{John}), (0.5, 0.5))\},\\
\{((\text{Jon}, \text{John}), s), ((\text{Jan}, \text{John}), d)\}),\\
(\{\text{merge}(\text{Jon}, \text{Jan}), \text{John}\},\\
\{(\text{merge}(\text{Jon}, \text{Jan}), \text{John})\},\\
\{((\text{merge}(\text{Jon}, \text{Jan}), \text{John}), (0.5, 0.5))\},\\
\{((\text{merge}(\text{Jon}, \text{Jan}), \text{John}), d)\})\}\\
\text{integrate}(\{\{\text{Jon}, \text{Jan}\}, \{\text{merge}(\text{Jon}, \text{Jan})\}\}, \{\{\text{John}\}\}) = \{\text{Merge}(\bar{g}) | \bar{g} \in \bar{g}\} =\\
\{\{\text{merge}(\text{Jon}, \text{John}), \text{Jan}\}, \{\text{merge}(\text{Jon}, \text{Jan}), \text{John}\}\}
\end{aligned}
$$

Bibliography

[1] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. *Machine Learning*, 56:89–113, July 2004.

[2] Omar Benjelloun, Hector Garcia-Molina, David Menestrina, Qi Su, Steven Euijong Whang, and Jennifer Widom. Swoosh: a generic approach to entity resolution. *The VLDB Journal*, 18(1):255–276, 2009.

[3] Omar Benjelloun, Anish Das Sarma, Alon Halevy, and Jennifer Widom. Uldbs: Databases with uncertainty and lineage. Technical Report 2005-39, Stanford InfoLab, 2005. A previous version of the paper was titled: "The Symbiosis of Lineage and Uncertainty".

[4] A. de Keijzer and M. van Keulen. Quality measures in uncertain data management. In H. Prade and V. S. Subrahmanian, editors, *Proceedings of the First International Conference on Scalable Uncertainty Management (SUM2007), Washington, DC, USA*, volume 4772 of *Lecture Notes in Computer Science*, pages 104–115, Berlin, October 2007. Springer Verlag.

[5] A. de Keijzer and M. van Keulen. Imprecise: Good-is-good-enough data integration. In *Proceedings of the 24th International Conference on Data Engineering (ICDE2008), Cancun, Mexico*, pages 1548–1551, Los Alamitos, April 2008. IEEE Computer Society Press.

[6] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 19(1):1–16, 2007.

[7] Ivan P. Fellegi and Alan B. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64(328):1183–1210, 1969.

[8] Mauricio A. Hernández and Salvatore J. Stolfo. The merge/purge problem for large databases. In *SIGMOD '95: Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, pages 127–138, New York, NY, USA, 1995. ACM.

[9] Steve Lawrence, C. Lee Giles, and Kurt D. Bollacker. Autonomous citation matching. In *AGENTS '99: Proceedings of the third annual conference on Autonomous Agents*, pages 392–393, New York, NY, USA, 1999. ACM.

[10] Maurizio Lenzerini. Data integration: a theoretical perspective. In *PODS '02: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 233–246, New York, NY, USA, 2002. ACM.

[11] David Menestrina, Omar Benjelloun, and Hector Garcia-Molina. Generic entity resolution with data confidences. Technical Report 2005-35, Stanford InfoLab, 2005.

[12] H.B. Newcombe, J.M. Kennedy, S.J. Axford, and A.P. James. Automatic linkage of vital records. *Science*, 130:954–959, October 1959.

[13] Sunita Sarawagi and Anuradha Bhamidipaty. Interactive deduplication using active learning. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 269–278, New York, NY, USA, 2002. ACM.

[14] M. van Keulen and A. de Keijzer. Qualitative effects of knowledge rules in probabilistic data integration. Technical Report TR-CTIT-08-42, Enschede, June 2008.

[15] M. van Keulen, A. de Keijzer, and W. Alink. A probabilistic xml approach to data integration. In *Proceedings of the 21st International Conference on Data Engineering (ICDE'05), Tokyo, Japan*, IEEE Conference Proceedings, pages 459–470, Washington, DC, USA, April 2005. IEEE Computer Society.