THE NETWORK IS THE BUSINESS

D3.1 –Micro-mapping composition

Author:	Zoltan Miklos – EPFL
Contributors:	Quoc Viet Hung Nguyen, Nguyen Thanh Tam, Duong Chi Thang, Do Son Thanh – EPFL
Dissemination:	Public
Contributing to:	WP 3
Date:	October 25, 2011
Revision:	V1.4

NisB Consortium Contacts

Organization	Name	Phone	E-Mail
SAP	Victor Shafran	+972-52-3854883	victor.shafran@sap.com
IIT	Avigdor Gal	+972-54-5370811	avigal@ie.technion.ac.il
EPFL	Miklós Zoltán	+41 79 723 3682	zoltan.miklos@epfl.ch
HSG	Boris Otto	+41 79 219 0582	boris.otto@unisg.ch
TXT	Enrico Del Grosso	+39 02 25771230	enrico.delgrosso@txt.it
CRF	Giorgio Sobrito	+39 011 9080542	giorgio.sobrito@crf.it
Momentum	Ian Graham	+44-2890-450101	ian.Graham@momentumni.org

Table of Contents

1 Introduction

Our work in NisB project focuses on interoperability in business networks, in B2B settings. This task is closely related to the widely-studied database schema matching problem. In fact, we do not only rely on this body of work, but we further improve existing schema matching methods. Our setting is somewhat more general, we do not only focus on database schemas, rather B2B interfaces and other settings where enterprises exchange (semi)-structured business documents. Nevertheless, we consider these business descriptions as schemas and apply schema matching techniques to them. We even often refer to them as schemas.

Schema matcher tools establish attribute correspondences between independently developed database schemas. There exists a large body of research on schema matching techniques, and as a result the schema matching tools often achieve impressive performance. Nevertheless, there are a number of situations, when the heuristic techniques fail and the matchers deliver erroneous correspondences or some correspondences are missing from the derived mappings.

In our work we accept that the schema matchers are not perfect and we try to identify possible errors. Once we identify these situations, we would also like to correct these errors.

Our overall strategy for improving initially generated mappings relies on micro-mappings, their composition in particular in the NisB network. This deliverable deals with the following questions:

- How can micro-mappings be used to detect errors in initially created mappings and how can we eliminate these errors?
- How can micro-mappings be composed?
- What is the role of micro mappings in interoperability establishment?

The NisB network is not static; it evolves based on user input and internal improvement methods. These techniques are described in detail in Deliverable D3.2.

The deliverable is structured as follows. Section 2 clarifies basic concepts, such as schema matching and attribute correspondences and micro-mappings, for the purpose of this deliverable. Section 3 discusses the relation of schema covering, concept repository to schema matching and proposes ways to improve initially-created schema matchings. Section 5 presents an experimental analysis of these improvement methods. Section 4 elaborates on possible improvements to schema covering. Section 6 presents some initial work on user feedback, in particular, for improving mappings. Finally, Section 7 concludes the deliverable.

2 Micro mappings

2.1 Schema matching and attribute correspondences

In the following we introduce some terminology. While our terminology relies on schema matching literature, we apply these techniques in a more general setting. We consider business descriptions as database schemas and apply schema matching to them. While we often refer to schemas, they could mean descriptions of enterprise systems or B2B standards.

Schema matching is the process of establishing connections between database schemas. The schema matching techniques consider two input database schemas, which are designed independently and they would like to find the correspondences between the schema attributes. Figure 1 depicts such a situation, where a matching tool has generated correspondences between the attributes of *purchase_order* and *p_order*.

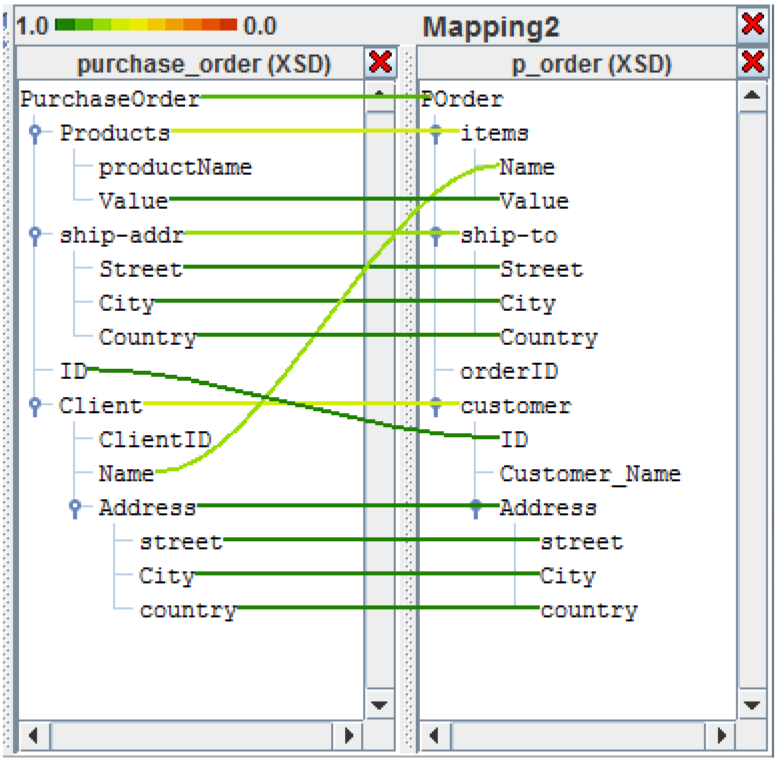


Figure 1 Schema matching

Attribute correspondences are relations between the schema elements (i.e. attributes of the schema). They can be used to define schema mappings which are more complex rules that explain how the data stored in the databases is related to each other. Such mappings are needed for example to exchange data items or reformulate queries (i.e. translate the queries posed to one of the databases to a query against the other database). More precisely, mappings express the exact logical relations, while mapping expressions express the data transformation rules (see [Rahm and Bernstein, 2001]). Our focus in this project and in this deliverable in particular, to find the correspondences, the mappings themselves and the mapping expressions are beyond the scope of this work.

Unfortunately, there is no precise formal definition in the literature what an attribute correspondence actually is. The correctness of an attribute correspondence highly depends on the business context, the goal of data integration and many other factors. This essential auxiliary information is often not present in the schema matching process, and also often not made explicit. In the literature the authors measure the quality of schema matching in terms of information retrieval measures, such as precision, recall, F-measure, etc. This approach implicitly assumes that a ground truth exists, i.e. given two schemas, there is a correct mapping. In other words, this assumption implies that there is a unique way of establishing attribute correspondences and the matcher's goal is to find this unique set. (The availability of the ground truth is clearly a different question, for practical evaluation tasks one often lacks the set of golden mappings.) In the remaining of this document we stick to this uniqueness assumption. However, we believe that this uniqueness assumption –that is implicitly present in the schema matching literature- is not correct, and we will present in a later phase of the project a rigorous treatment of this question. More precisely, we will discuss this in D3.3.

Very often the attribute names are similar, though not identical and the goal is to find these similar attributes. If one tries to match two database schemas, then one has the implicit assumption that the data behind the schemas corresponds to each other, they just have different representations. In fact, in business settings often the precise semantic of the attributes is different, though exchanging data having these attributes has business benefits. For example, in the above figure one establishes a connection between *Client* and *customer*. They have certainly different meaning, but the given business context might make it necessary to exchange *Client* and *customer* records. This situation is rather common: the attribute correspondences are induced by a business setting, and there is a lot of hidden and implicit knowledge behind which attributes shall be put in relation. Exactly this lack of knowledge about the context, about the precise semantic of data makes schema matching difficult.

The schema matchers are software tools which construct a set of attribute correspondences. These attribute correspondences are constructed via heuristic techniques. The tools consider various string similarity measures (as often similar or identical names should correspond) and other information, such as for example the structure of the schemas. Schema matchers often also rely on external information, such as dictionaries, business documents etc. For an overview of the heuristic techniques used by matchers, see [Bernstein & Rahm, 2001]. As a result of the large body of research on schema matching, the performance of the matching tools has increased significantly in the recent years. As schema matching involves a lot of uncertainties about e.g. the business context, similarities, the output of schema matchers often contains errors. Schema matchers can still save time, especially in the case of large schemas: humans need only check the generated correspondences and include those, which are missed by the matcher.

Schema matchers distinguish *source* and *target* schemas. While the attribute correspondences themselves are not directional, the mapping expressions or transformation rules are asymmetric.

2.2 Micro mappings

Database schemas are often not just a homogeneous set of attributes, but they can be decomposed to smaller units or building blocks. In business settings these building blocks often correspond to business concepts. It is very natural to consider not only correspondences between individual attributes, but rather complete business concepts. We call such correspondences between concepts micro mappings. Micro mappings are groups of attribute correspondences that relate to concepts from the involved business schemas.

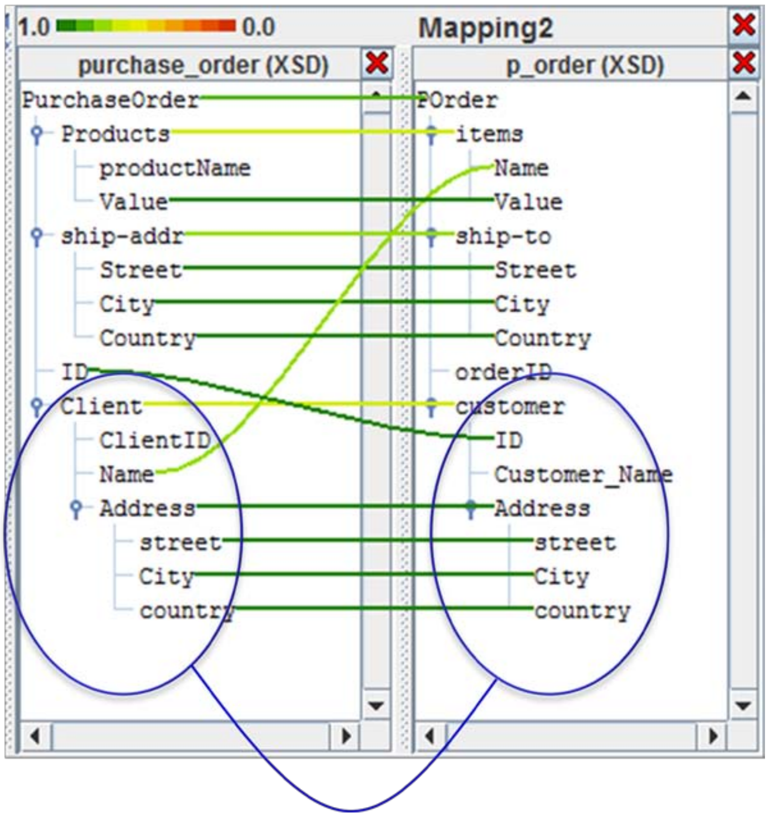


Figure 2 Micro mapping

Figure 2 depicts a micro mapping between the business concepts *Client* and *customer*. Micro mappings shall help to improve automatically generated correspondences.

3 Improving initially generated mappings

In this section we summarize ways in which micro-mappings can help to improve initially generated schema correspondences. We also elaborate on the potential problems that can be detected based on the structure of micro-mappings in the NisB network. The NisB network is in fact the manifestation of micro mapping composition.

The NisB network consists of concepts, interlinked by the micro-mappings between them. There are two different approaches to how the concepts are populated to the NisB network:

1. Concepts-first approach: In this setting there are predefined concepts coming from business standards or quasi-standards, which are well established and widely used. For example, some business standards might be adopted by a large-number of companies.
2. Schemas-first approach: where we decompose the schemas of participants, who are engaging in business with other NisB network participants.

(These are not necessary alternatives; one can imagine also a hybrid strategy.) In both cases, one needs to establish a connection between schema attributes (and groups of attributes, which are essentially also concepts) and concepts in the NisB network. In the schemas-first approach the decomposed schemas keep a back-link (or provenance) to the originating schema. In the concept-first approach there are no such connections. One possibility is to apply the schema covering techniques developed in WP2 (see deliverable D2.1). Note that in this case there is some uncertainty involved, as we do not know a priory, what are the right parameters for the schema covering (i.e. ambiguity constraints). Moreover, the schema covering algorithms are guided by the alignment scores, which are computed using similarity functions, i.e. heuristic techniques. Moreover there is a number other sources for uncertainties and errors, as we discuss in Section 4.

In the following, we will discuss improvements for the schema matching process. The cases are in fact potential improvements; one would need human input to decide for sure whether the improvement suggestions indeed lead to improvements. Alternatively, one could adopt these suggested improvements in a probabilistic way, for example, in a given situation; one could apply several possible improvement strategy, each with a given probability. However, one would need statistical and data mining techniques to estimate these probabilities. We plan to do such investigations once the size and maturity of our schema and concept repository (or the NisB network) permits it.

In Section 5 we analyze the improvement we could achieve using the situations explained in this section. Unfortunately, the improvements are only moderate on the very small datasets we are currently working with. To make the improvements more reliable, we need strategies for more reliably identifying concepts, that is discussed in Section 4.

3.1 Concept structure and matching

In the schema matching process we consider a schema as a set of attributes. The set of attributes in business schemas can often be grouped according to specific business concepts, such as *Person*, *Address*, *PurchaseOrder*, etc. , see Figure 3.

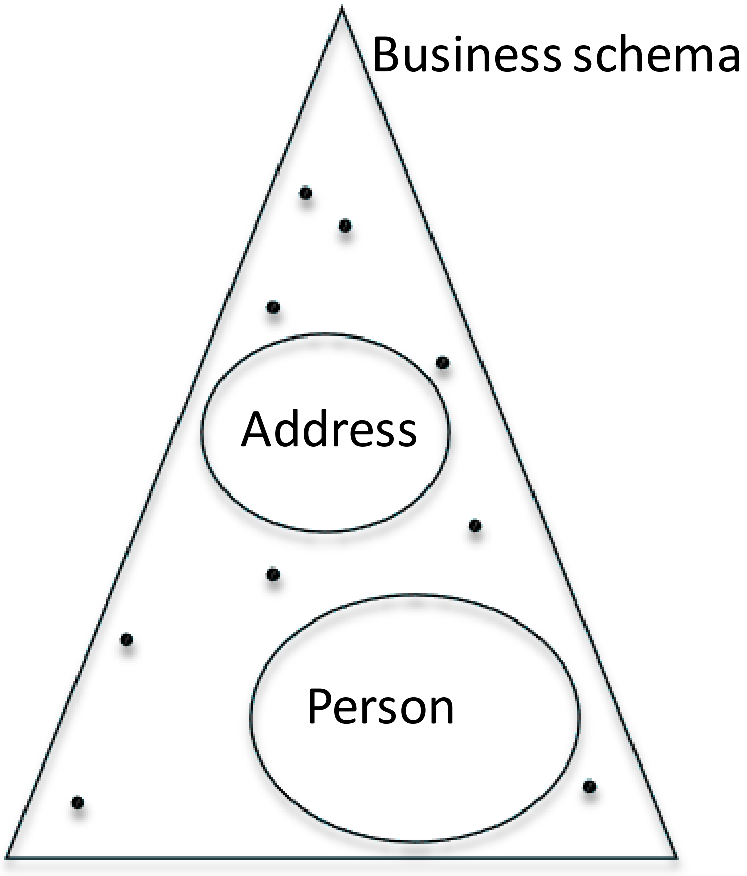


Figure 3 Business schema

This grouping of attributes is natural for human observers, however it is not necessary trivial to identify such concepts automatically and explain the business schemas in terms of concepts. We have studied this problem; the joint efforts of the consortium on this problem are reported in Deliverable 2.1 and in forthcoming publications.

The schema covering process might explain two different schemas with the same concepts, see Figure 4. The bold lines on the figure correspond to explanations through schema covering. Such common explanations give rise to micro-mappings (between the concepts residing within the schemas).

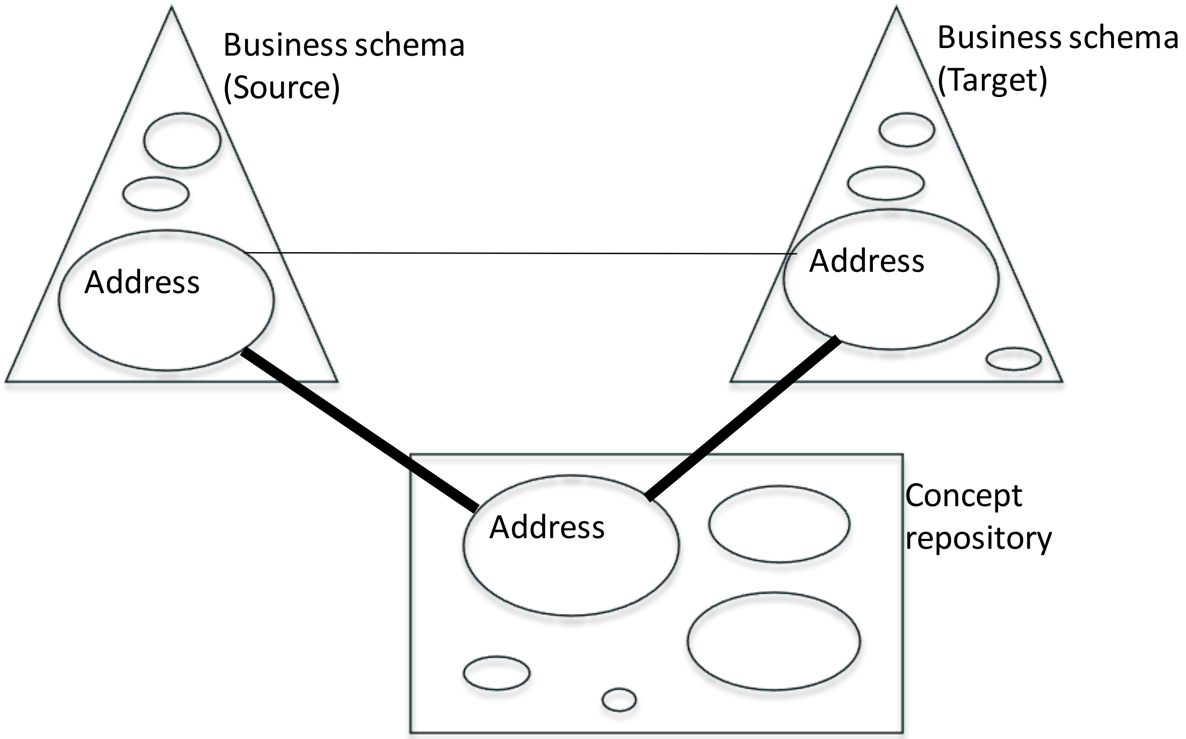


Figure 4 Micro-mapping through common explanation

Such direct common explanations are not always easy to find. We might establish relations through more complex paths in the NisB network, see Figure 5.

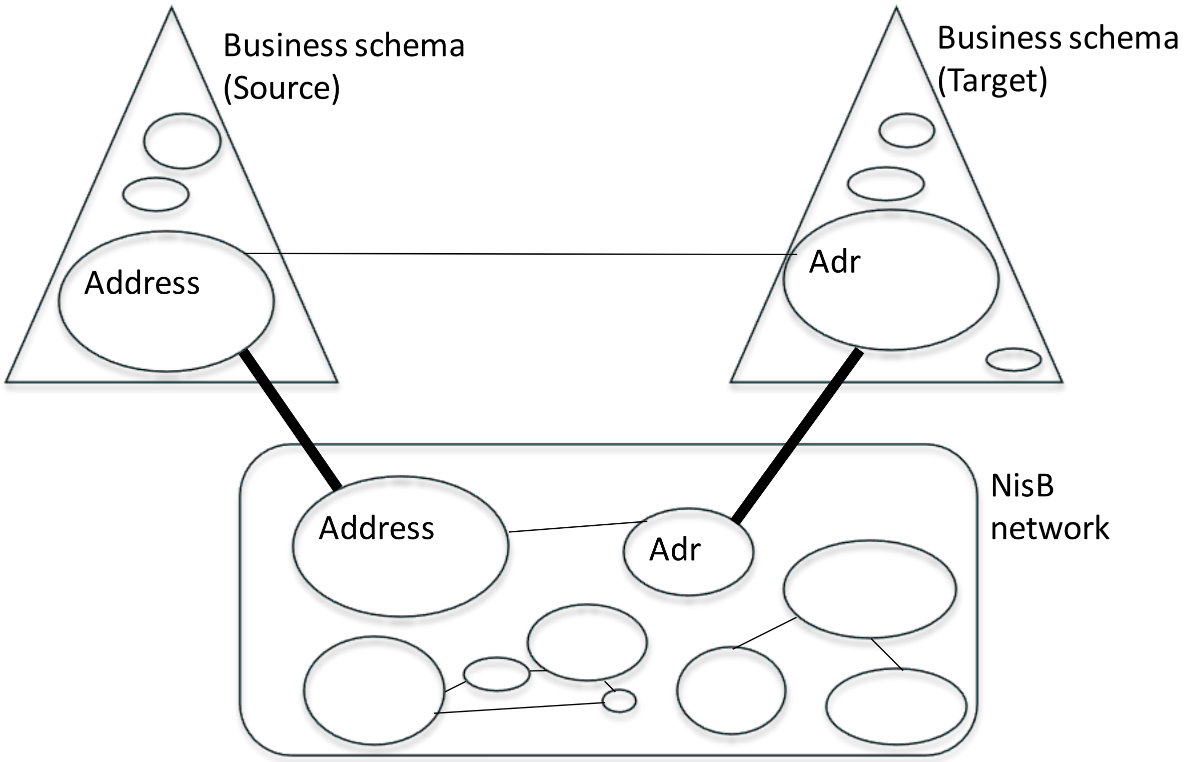


Figure 5 Connection through the NisB network

We give more details on how the NisB network is constructed in Deliverable D1.4, while the evolution of the NisB network is discussed in the Deliverable D3.2. Note that while the schema covering, schema decomposition and the related modules are already integrated, there are still minor differences between the NisB software and the scenarios described here.

3.1.1 Attribute correspondence inconsistent with micro mappings

On Figure 2 the attribute *Name* (from the source schema, on the left-hand side) is mapped to the attribute *Name* in the target schema (right-hand side). This correspondence was generated by a matching tool (Coma++). This correspondence is most likely false, as it does not respect the concept structure and the “Client-customer” micro-mapping, depicted in Figure 6. The schema matcher in this case was relying on the string similarity. Indeed, the string similarity between (*Name,Name*) is higher (in fact, it is 1, as the two strings are identical) as the similarity between (*Name,Customer_Name*). The attribute *Customer_Name* has a lower similarity, but the presence of the micro-mapping indicates a a possible improvement that is correct in this case. Often such anomalies are related to improvements with some probability. If one has a large set of schemas and micro mapping, one can estimate the probabilities using statistical or machine learning methods.

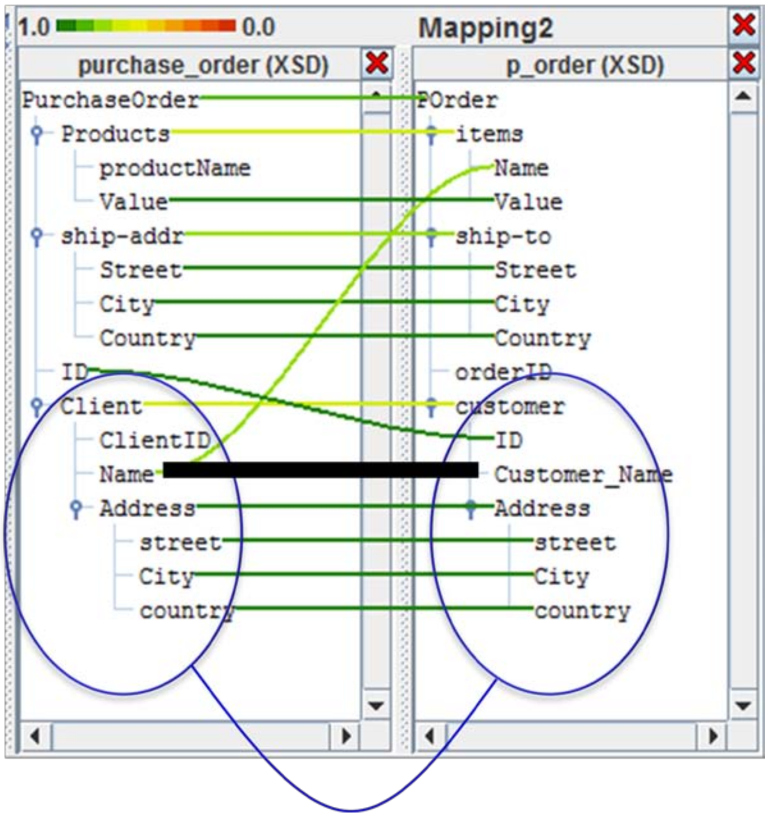


Figure 6 Inconsistent attribute correspondence

3.1.2 Missing correspondences

Micro mappings can be useful to identify missing attribute correspondences, i.e. correspondences that are not generated by the schema matching tools. Such an example is depicted in Figure 7, where the correspondence between *Telephone* and *Tel* is missing.

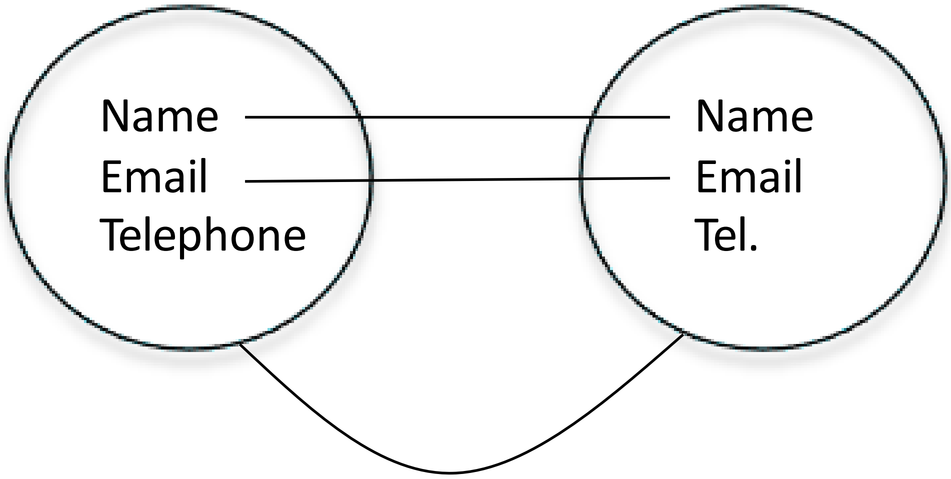


Figure 7 Missing attribute correspondence

Figure 8 shows the improvement that we can achieve by adding missing links.

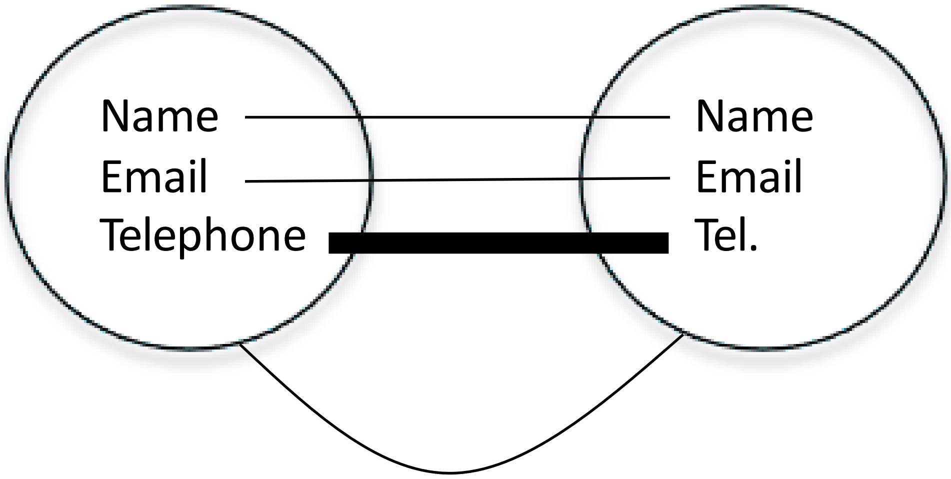


Figure 8 Attribute correspondence added

3.2 Network structure and mappings

The NisB network contains concepts, schemas (that can also be seen as concepts) and links between them that correspond to micro mappings and attribute correspondences.

3.2.1 Transitivity: Missing correspondences

Figure 9 depicts a situation where there is an attribute correspondence between the attribute “a” from the concept A and the attribute “b” from concept B. Similarly, there is a correspondence between the attribute “b” from the concept B and the attribute “c” from concept C. As attribute correspondences essentially represent equivalence relations, there should be a correspondence between A.a and C.c that is missing.

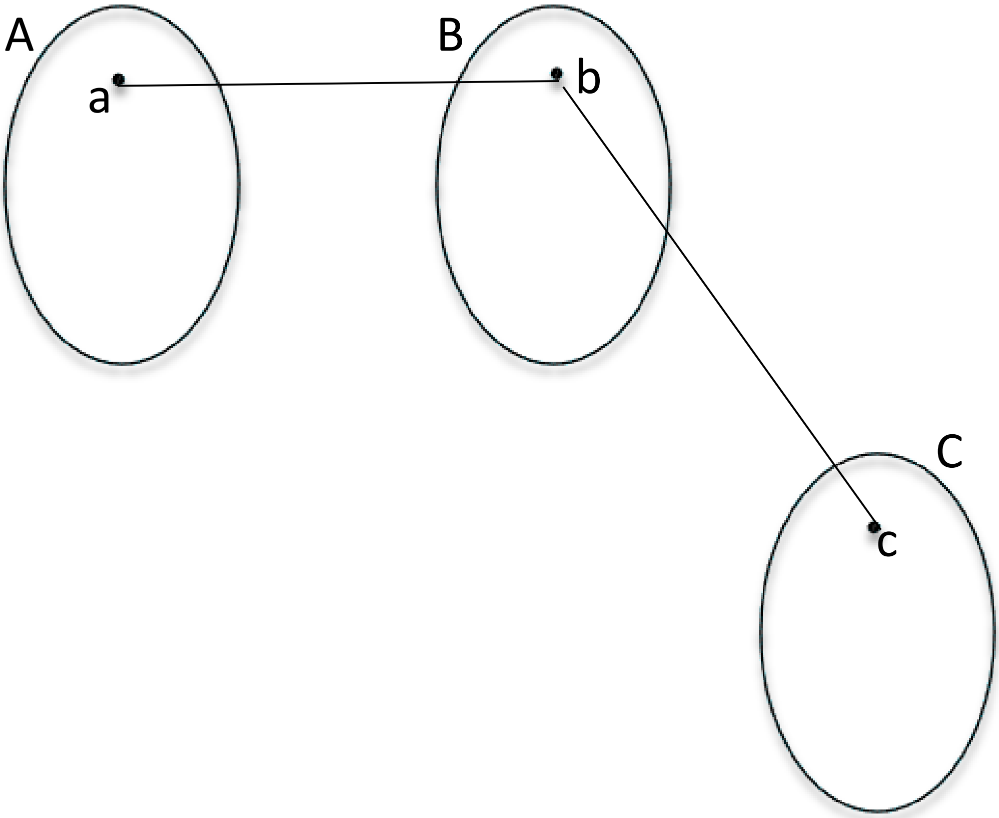


Figure 9 Missing correspondence

There could be many reasons why this correspondence is missing. For example, one has not tried to construct a micro-mapping between the concepts A and C. Or, maybe one has already constructed a micro-mapping, yet, the correspondence is missing.

One can resolve this conflict by adding the correspondence (A.a, C.c). For the confidence value of this added correspondence one can choose the minimum of the confidence values of the attribute correspondences (A.a,B.b) and (B.b,C.c).

The motivation for studying this case was that in our dataset each case with missing correspondence of this type was related to errors.

3.2.2 Transitivity: Conflicting correspondences

Figure 10 depicts a situation where transitive closure can help to identify potential errors. In this configuration there are strong connections between the attributes (A.a, B.b) and (B.b,C.c), while confidence value for the correspondence (A.a,C.c) is low. Thus adding transitive closure would result having both strong and weak connections between a pair of attributes that might indicate a potential problem.

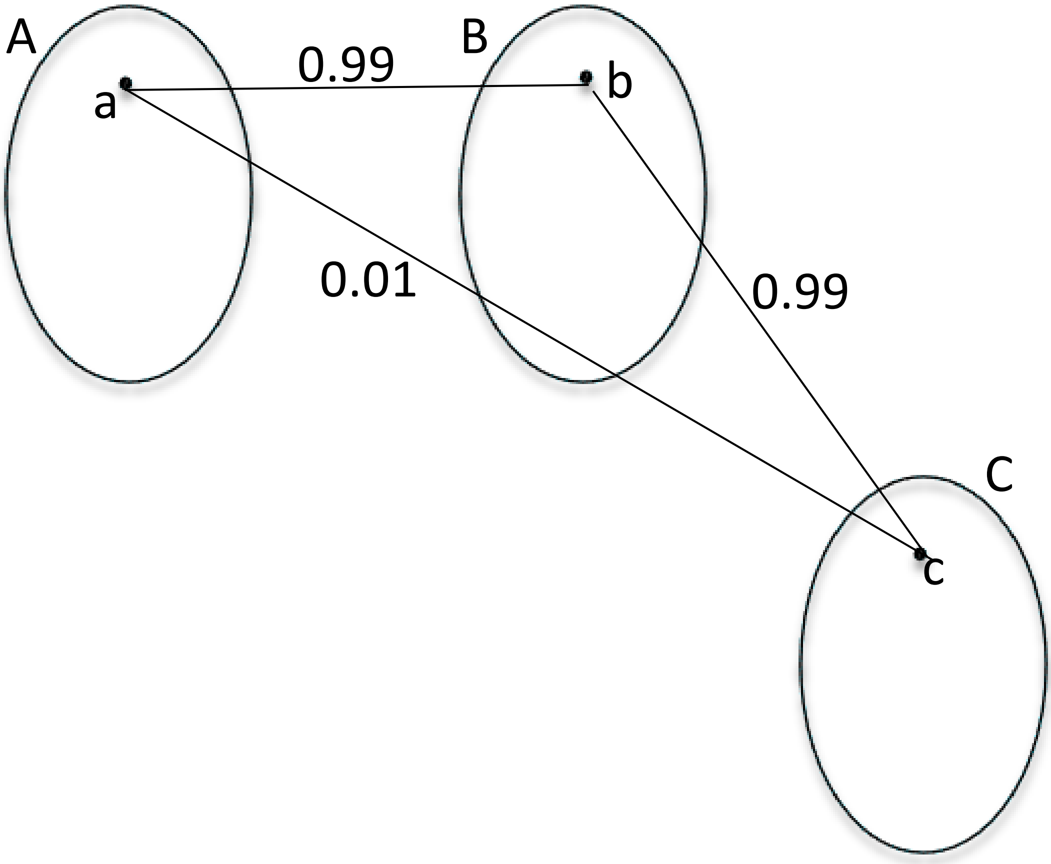


Figure 10 Conflicting transitive closure

The conflict resolution in this case can be the adjustment of confidence values, either by increasing or be decreasing the relevant values. However, there are many alternative ways as well. Changing confidence values might require adjusting the values elsewhere in the network as well. In fact, one would need a probabilistic reasoning framework to systematically propagate changes in the network. Such methods have already been studied, e.g. in [Aberer et al. 2003]. There are also other options, such as for example dropping one or other involved correspondence.

3.2.3 Inconsistent mappings

Figure 11 depicts a case where the correspondences are inconsistent. In this case, the attribute “a” corresponds to two different attributes in schema C (namely “c1” and “c2”), depending whether one considers a direct mapping from schema A to C or a mapping via a path, through schema B. In such cases, if the businesses using the corresponding schemas exchange business documents, then the data items of attribute “a” would appear as both “c1” and “c2” that can be very confusing in the practice.

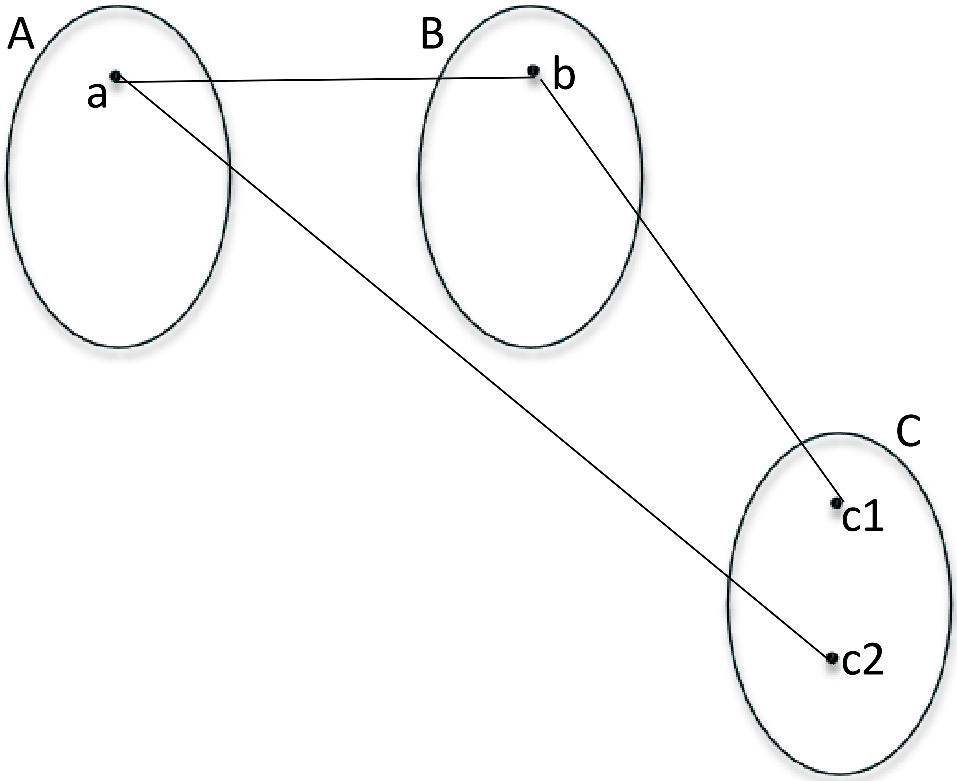


Figure 11 Inconsistent correspondences

A special variant of the above case is depicted in Figure 12. While the situation is very similar to the case of the Figure 11, the resolution strategies might be different. Here the attribute “a” from schema A is mapped to different attributes in schema C via two different paths, once via a direct mapping, once via the schema B. The specialty of the case is that all involved attributes belong to subschemas, explained by the concept “Address”. The schema C has in this case two different parts, which can be explained by the concept “Address”, that is a very common situation in business schemas (for example, they correspond to “delivery address” and “billing address”).

In such cases the schema matchers perform very poorly, that is not surprising, as it is not clear even for humans which one is the correct mapping. In the business practice such cases are resolved through human communication and bilateral agreements, which are often not explicitly recorded, but only spread as word-of-mouth.

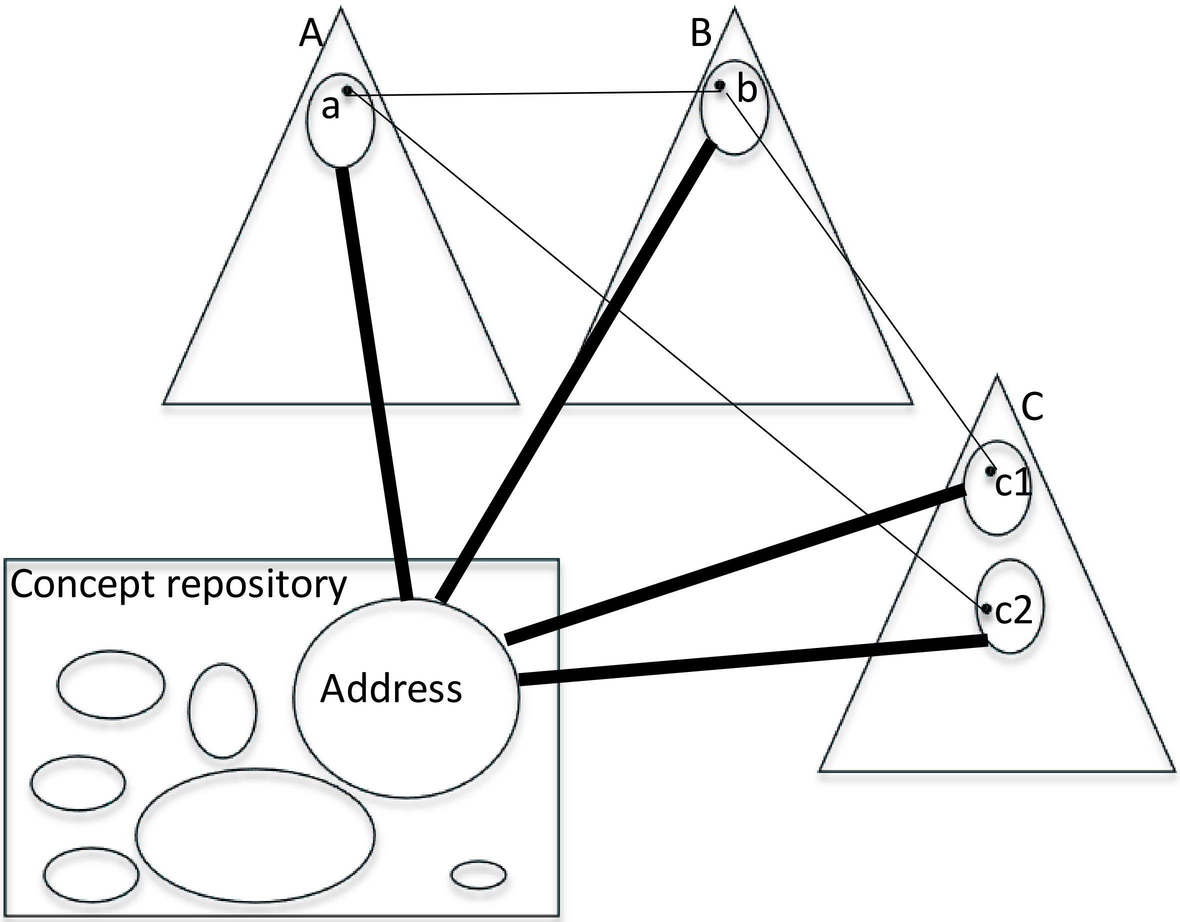


Figure 12 Inconsistent mapping: a special case

4 Improving schema covering

Our work on schema covering is described in D2.1, including precise formal definitions and experiments. As mentioned, schema covering is a technique to find explanations of parts of a database schema in terms of predefined concepts. These concepts are available in the form of a concept repository. In the NisB context, the NisB network –even if it has a number of other functionalities and purposes- can be regarded as a concept repository.

We have repeated some of the experiments presented in D2.1, further analyzed the results and we tried to identify the problems in schema covering. The motivation for this task is that as we have seen in Sections 3 and 5, schema covering indeed can be used for developing methods to improve matchings. The success of these methods however would rely heavily on the quality of the schema covering. For this reason we analyzed the common problems with the constructed schema covering.

In the following, in Section 4.1 we describe the problem cases, while in Section 4.2 relate schema covering to schema matching.

4.1 Problem cases

4.1.1 Unconnected subschema

Figure 13 depicts a situation where in the sub-schema/concept pair in a schema covering the sub-schema is not a connected graph. As the number of all possible sub-schemas is very high, as a starting phase, schema covering algorithms filter out certain sub-schemas. Thus they are not considered in the covering process. As the sub-schema in Figure 13 is not connected, the depicted cover (that is correct) cannot be found (assuming the decomposition algorithm considers connected subgraphs). We must note here that there are many sources of errors in the schema covering process: one might introduce incorrect correspondences in the covering phase, while in other cases the root of the problems is in the decomposition phase.

The opposite case is shown in Figure 14. When a subschema is large and maybe unconnected, it can introduce errors to the cover. In this case -as in Figure 14- we can decompose it into smaller parts (the Item part and the Contact part in the Figure).

In the following, for each case we also discuss possible resolutions to the problems. While we can analyze these cases, applying the appropriate resolutions automatically is rather challenging task. Nevertheless, we consider the option that after a schema covering is computed, we have the chance to improve the initially created cover. This might require human input.

Resolution strategy: While it is not possible to consider all possible sub-schemas, it is conceivable to change the sub-schemas once an initial cover is already available. Figure 13 shows a case where the considered subgraph has already been changed, and the attribute *contactNam* is included (even if the subschema is not connectd)

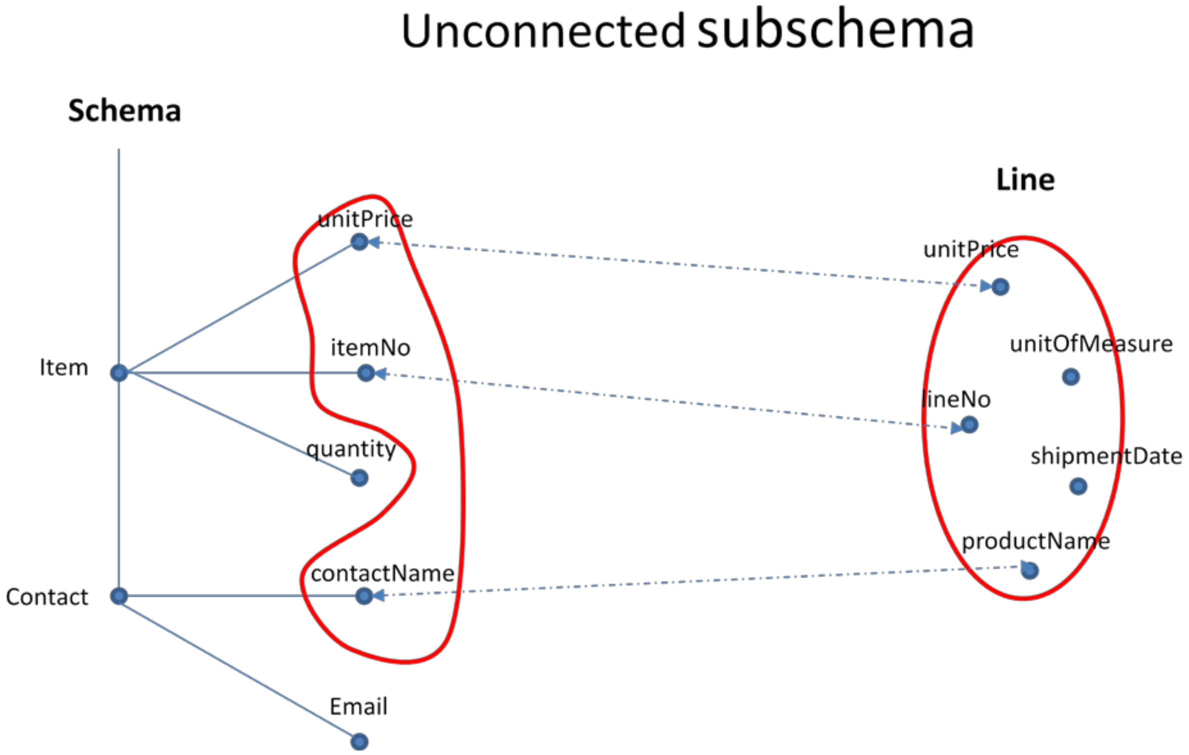


Figure 13 Unconnected subschema

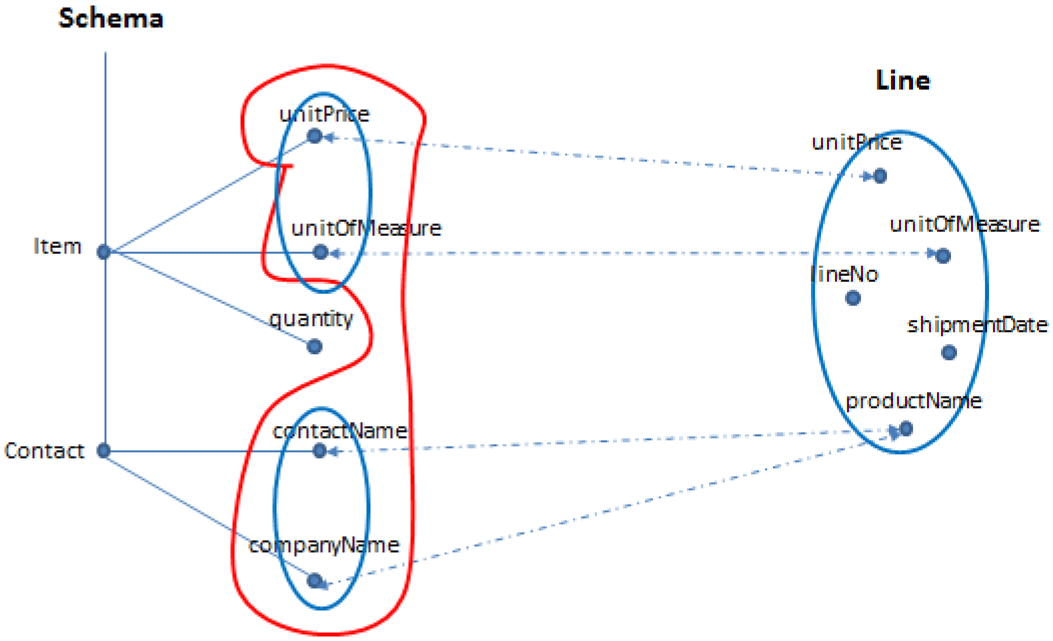


Figure 14 Breaking a larger (unconnected) subschema into smaller ones

4.1.2 One concept attribute, many schema attributes

Another common problem is depicted in Figure 15 and in Figure 16. In this situation, since the sub-schema was selected in an ad-hoc fashion, the schema covering is actually not a good “explanation” of the schema.

In this case, one should select a sub-schema of appropriate size. This could be realized through human input.

Resolution strategy: in a micromapping, we can enforce one-to-one constraint. We can have many ways to enforce this constraint such as we only keep the top rank correspondence or show the micromapping to user to get feedback. Clearly, this resolution strategy should be applied if the one-to-one constraint is applicable. If the origin of the problem is the one-to-one correspondence, one should relax this constraint.

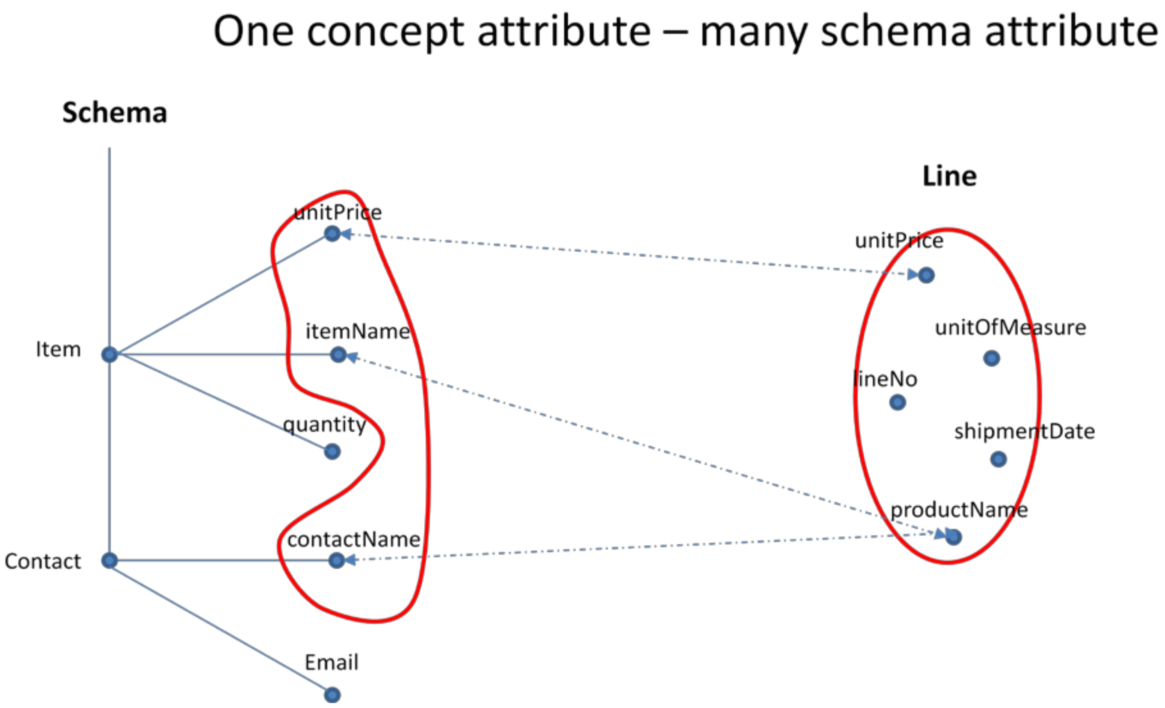


Figure 15 One concept attribute, many schema attribute

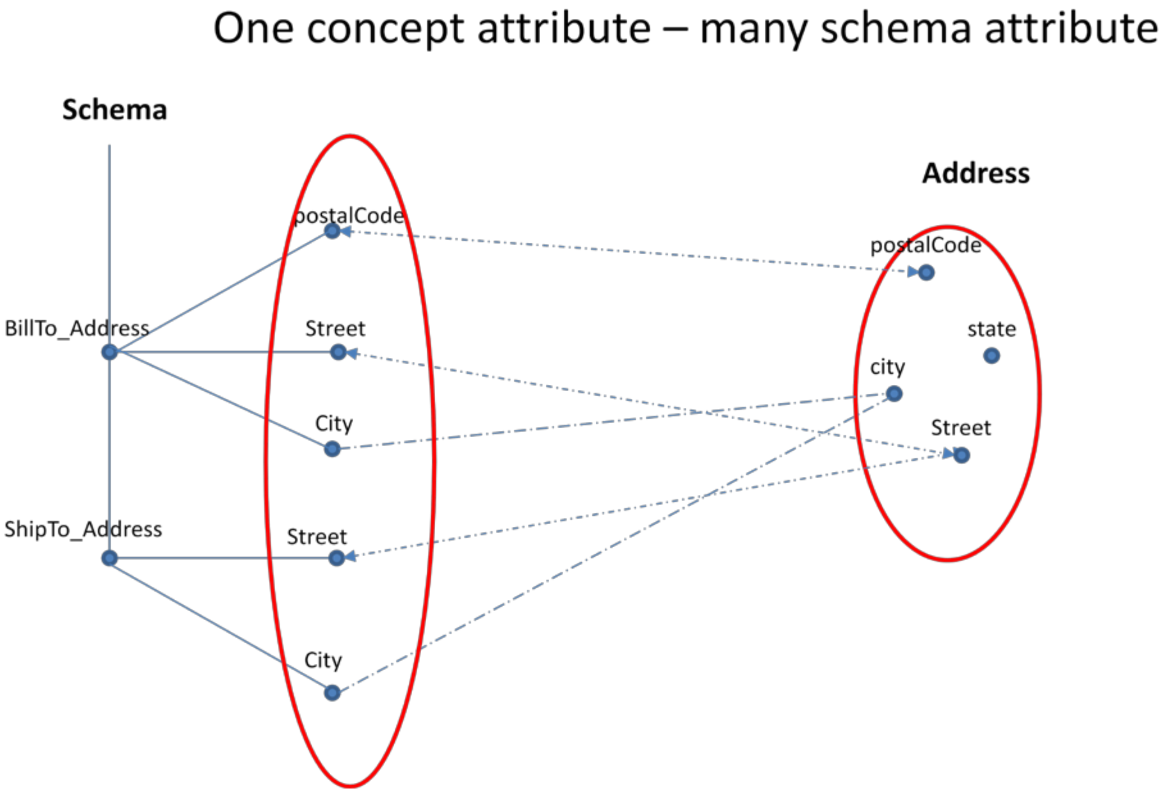


Figure 16 One concept attribute, many schema attribute (inappropriate sub-schema size)

4.1.3 One schema attribute, many concept attributes

Other common problems include the case depicted in Figure 17. Here there are several possible concepts that would cover a set of attributes in the schema. The schema covering techniques – depending on the pre-defined ambiguity constraints- might find wrong covers, in the sense that other concepts would better explain the schema or parts of the schema, if we would relax either the ambiguity constraints or change the considered subschema.

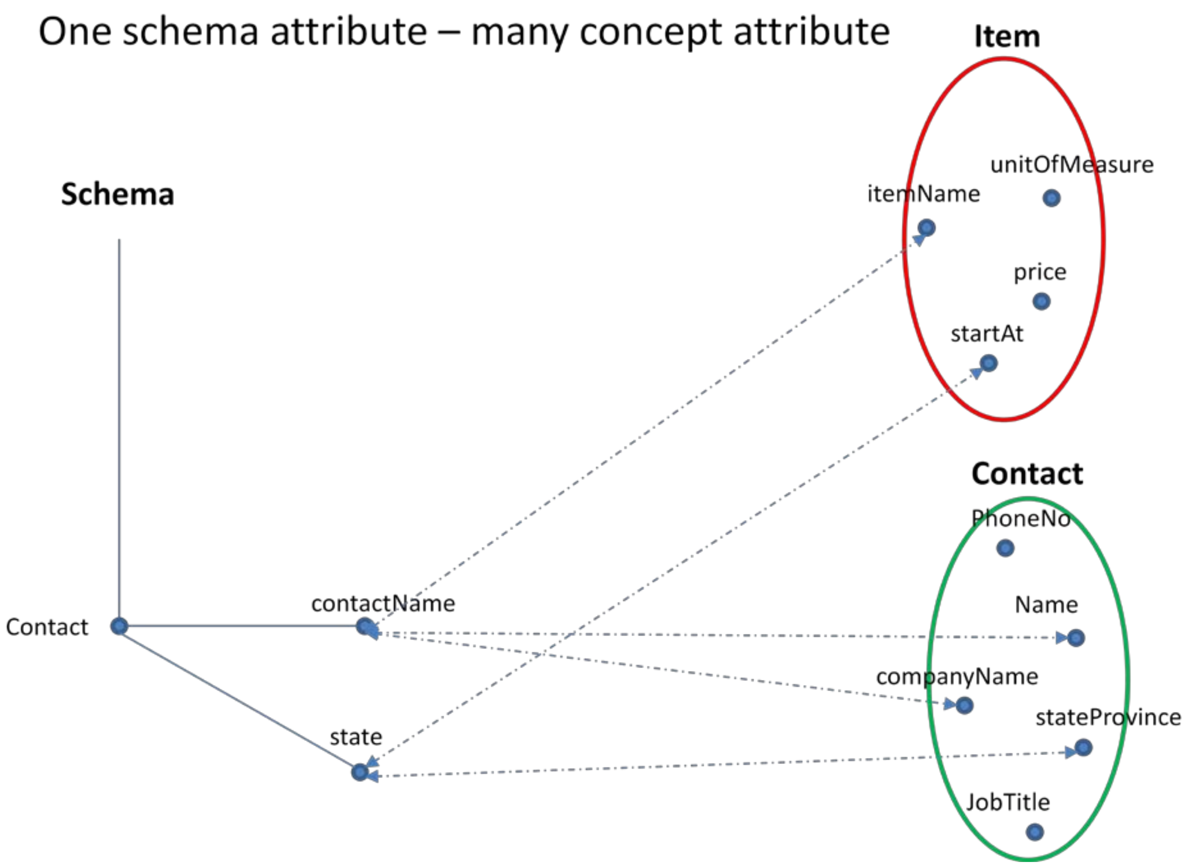


Figure 17 One schema attribute, many concept attribute

The situation on Figure 18 depicts the same situation, but here we obtained the cover with different ambiguity constraints.

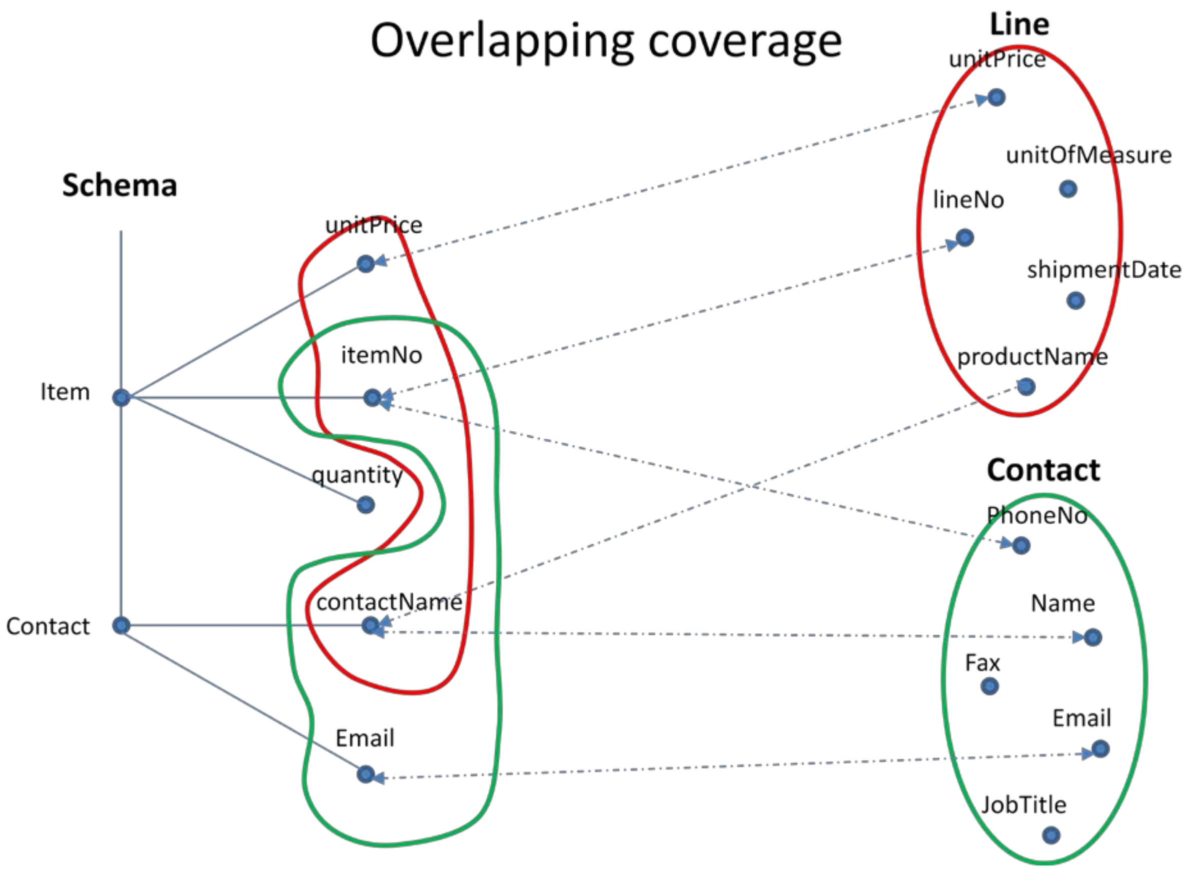


Figure 18 One schema attribute, many concept attribute

4.2 Schema covering and schema matching

In the following we discuss the interaction between schema covering and schema matching. In particular, we are interested in the following question: We are given two schemas. We generate with a schema matching tool (AMC) attribute correspondences. We also compute schema covering, with a common concept repository to both schemas.

- Can we improve the schema mappings with the help of the compute schema covering?
- Can we improve the schema covering, with the help of schema matching?

The setting is depicted in Figure 19 and in Figure 20. We consider direct schema matching of the two schemas (through AMC tool or other matching tools). At the same time, we consider a common concept repository and would like to apply schema covering to both schemas, and then analyze the induced connections through transitivity. Our goal is to explore whether and how can we exploit the two different sources of information (matching, covering) to improve one through the other.

We have also conducted experiments to analyze these questions. The experimental results are discussed in Section 5.

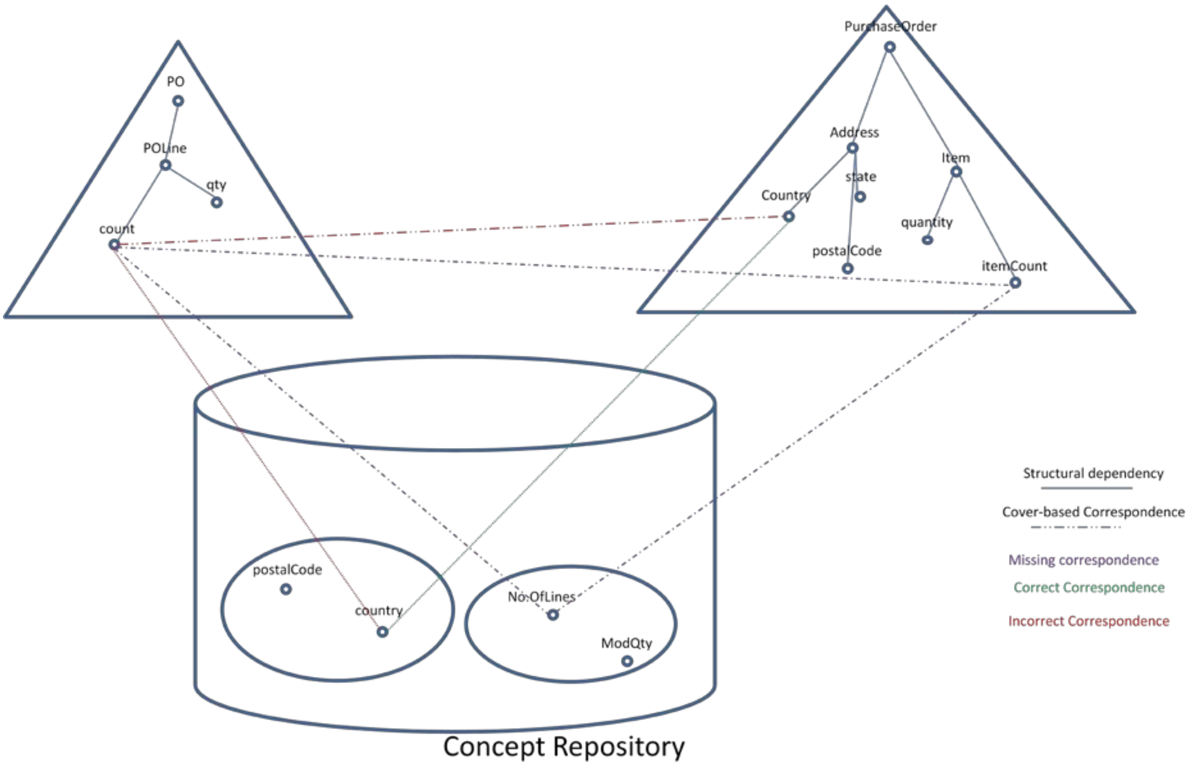


Figure 19 Schema matching and schema covering

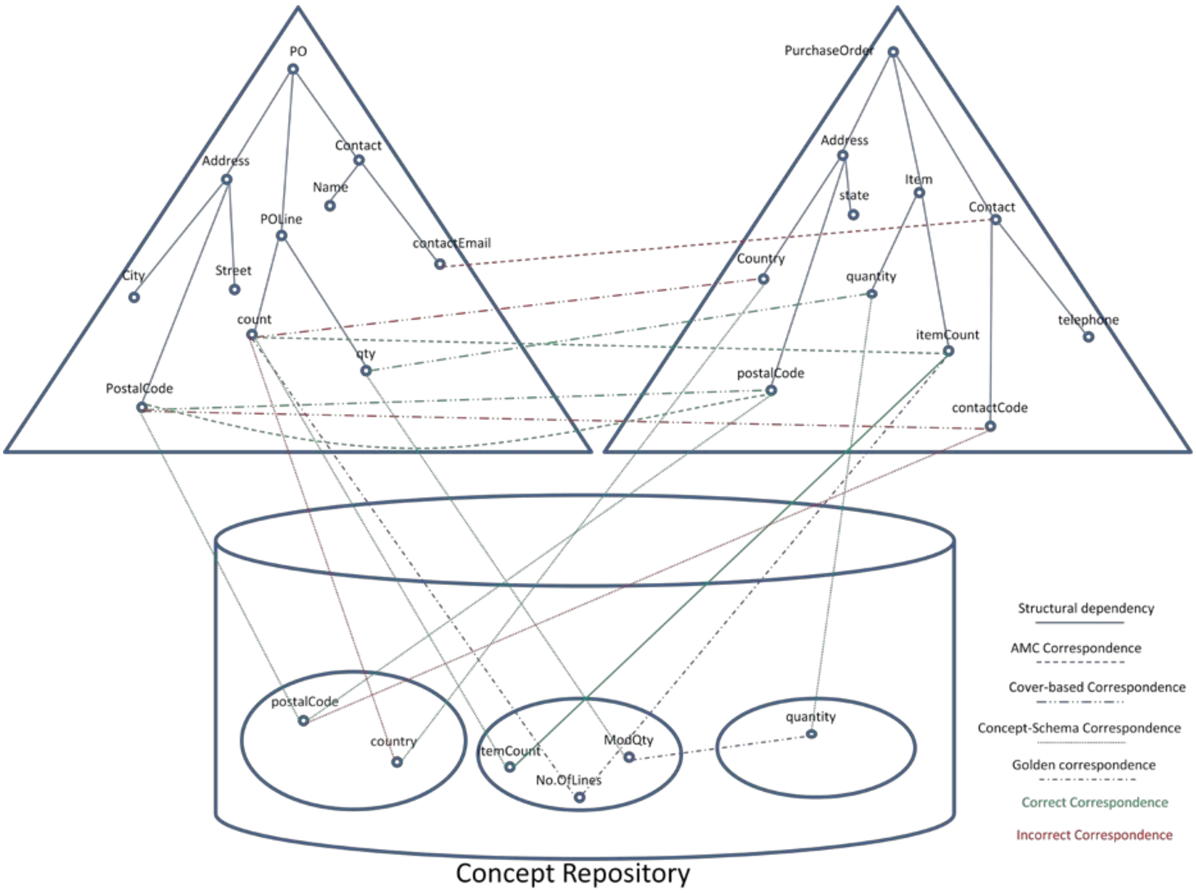


Figure 20 Schema matching and schema covering

5 Experimental results on combining schema covering and schema matching

In Section 4.2 we have discussed potential connections between schema covering and schema matching. In this section we analyze these possibilities. In the following we present our preliminary experimental results.

5.1 Setting

We conducted experiments with the following settings. We used the dataset that we have collected and constructed. For a description, see the appendix of the deliverable D3.2.

- Input
  - Two schemas to find the correspondences: Excel and CIDX
  - A concept repository generated by decomposing schemas in the same domain: Paragon, Apertum, Noris.
  - Mappings between concepts in the repository based on the ground truth
  - Covering result of Excel and CIDX using the repository
  - Correspondences between Excel and CIDX generated by the commercial matcher AMC (threshold: 0.4) (for a description of the matching tool AMC, see reference [AMC])
  - Exact correspondences between Excel and CIDX (i.e. golden mappings, or ground truth)
- Output
  - Correspondences between Excel and CIDX found using the concept repository (by applying the transitivity).

5.2 Assumption

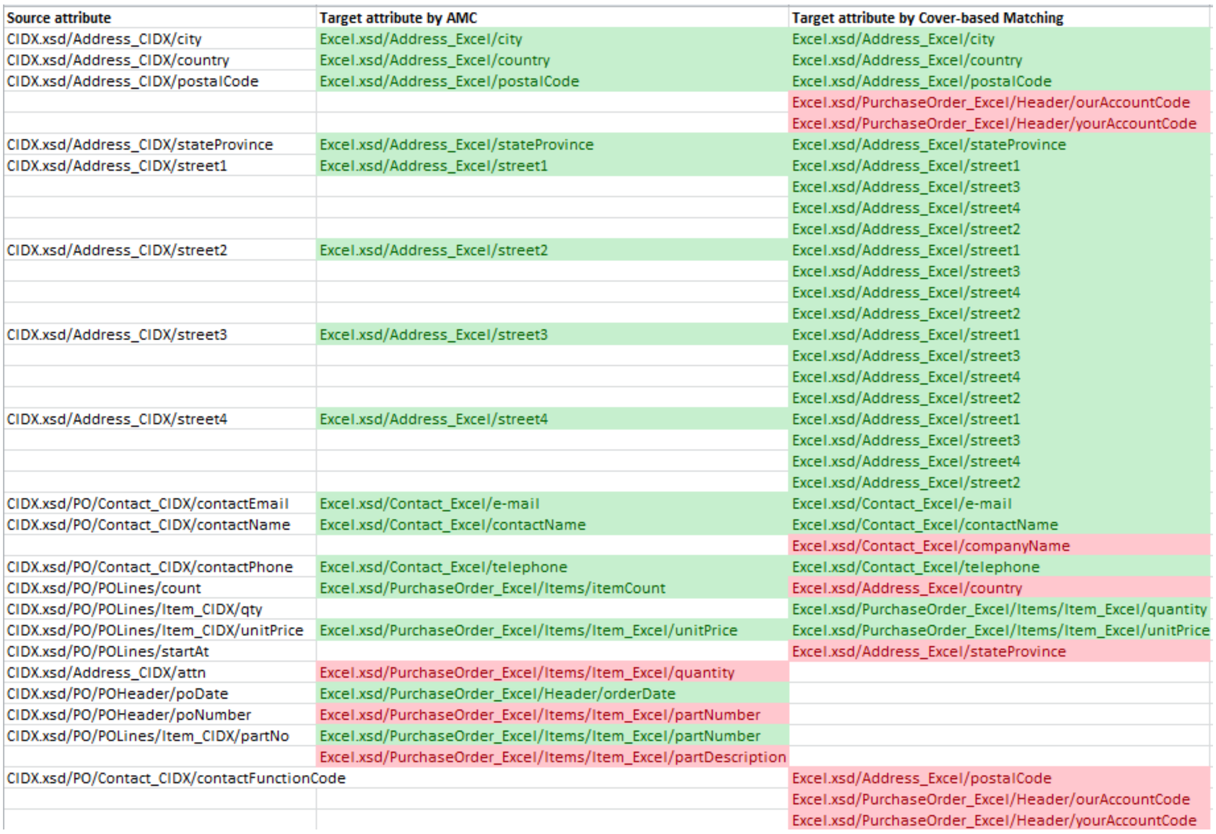
We only consider correspondences of leaf attributes in the schemas. This is reasonable as the leaf attributes can convey the meaning of the schemas. Previous experiments also point out that considering correspondences of non-leaf attributes may add noise to the result. We have also configured AMC accordingly (i.e., not the “path matcher” is used).

5.3 Evaluation procedure

First, we run schema covering to find the covers of the two schemas. (For details on schema covering, see deliverable D2.1 and the reference [gCover]. In fact, we have used the software we developed in cooperation between Technion, EPFL and SAP.) The covers also contain the correspondences from the schema attributes to the concept attributes in the repository. Moreover, we also have the exact correspondences between concept attributes. Therefore, we can find a correspondence between two schema attributes as follows: if there is a path from one schema attribute to the other schema attribute through the concept attributes, there is a correspondence between two schema attributes. We will infer all possible attribute correspondences between two schemas using this cover-based approach.

- Goal: we want to compare the correspondences found by using the concept repository (called cover-based matching) with ones generated by the matcher. Moreover, we want to find out if this matching approach can help the other.
- Metrics: precision, recall

5.4 Results



Source attribute	Target attribute by AMC	Target attribute by Cover-based Matching
CIDX.xsd/Address_CIDX/city	Excel.xsd/Address_Excel/city	Excel.xsd/Address_Excel/city
CIDX.xsd/Address_CIDX/country	Excel.xsd/Address_Excel/country	Excel.xsd/Address_Excel/country
CIDX.xsd/Address_CIDX/postalCode	Excel.xsd/Address_Excel/postalCode	Excel.xsd/Address_Excel/postalCode
		Excel.xsd/PurchaseOrder_Excel/Header/ourAccountCode
		Excel.xsd/PurchaseOrder_Excel/Header/yourAccountCode
CIDX.xsd/Address_CIDX/stateProvince	Excel.xsd/Address_Excel/stateProvince	Excel.xsd/Address_Excel/stateProvince
CIDX.xsd/Address_CIDX/street1	Excel.xsd/Address_Excel/street1	Excel.xsd/Address_Excel/street1
		Excel.xsd/Address_Excel/street3
		Excel.xsd/Address_Excel/street4
		Excel.xsd/Address_Excel/street2
CIDX.xsd/Address_CIDX/street2	Excel.xsd/Address_Excel/street2	Excel.xsd/Address_Excel/street1
		Excel.xsd/Address_Excel/street3
		Excel.xsd/Address_Excel/street4
		Excel.xsd/Address_Excel/street2
CIDX.xsd/Address_CIDX/street3	Excel.xsd/Address_Excel/street3	Excel.xsd/Address_Excel/street1
		Excel.xsd/Address_Excel/street3
		Excel.xsd/Address_Excel/street4
		Excel.xsd/Address_Excel/street2
CIDX.xsd/Address_CIDX/street4	Excel.xsd/Address_Excel/street4	Excel.xsd/Address_Excel/street1
		Excel.xsd/Address_Excel/street3
		Excel.xsd/Address_Excel/street4
		Excel.xsd/Address_Excel/street2
CIDX.xsd/PO/Contact_CIDX/contactEmail	Excel.xsd/Contact_Excel/e-mail	Excel.xsd/Contact_Excel/e-mail
CIDX.xsd/PO/Contact_CIDX/contactName	Excel.xsd/Contact_Excel/contactName	Excel.xsd/Contact_Excel/contactName
		Excel.xsd/Contact_Excel/companyName
CIDX.xsd/PO/Contact_CIDX/contactPhone	Excel.xsd/Contact_Excel/telephone	Excel.xsd/Contact_Excel/telephone
CIDX.xsd/PO/POLines/count	Excel.xsd/PurchaseOrder_Excel/Items/itemCount	Excel.xsd/Address_Excel/country
CIDX.xsd/PO/POLines/Item_CIDX/qty		Excel.xsd/PurchaseOrder_Excel/Items/Item_Excel/quantity
CIDX.xsd/PO/POLines/Item_CIDX/unitPrice	Excel.xsd/PurchaseOrder_Excel/Items/Item_Excel/unitPrice	Excel.xsd/PurchaseOrder_Excel/Items/Item_Excel/unitPrice
CIDX.xsd/PO/POLines/startAt		Excel.xsd/Address_Excel/stateProvince
CIDX.xsd/Address_CIDX/attn	Excel.xsd/PurchaseOrder_Excel/Items/Item_Excel/quantity	
CIDX.xsd/PO/POHeader/poDate	Excel.xsd/PurchaseOrder_Excel/Header/orderDate	
CIDX.xsd/PO/POHeader/poNumber	Excel.xsd/PurchaseOrder_Excel/Items/Item_Excel/partNumber	
CIDX.xsd/PO/POLines/Item_CIDX/partNo	Excel.xsd/PurchaseOrder_Excel/Items/Item_Excel/partNumber	
	Excel.xsd/PurchaseOrder_Excel/Items/Item_Excel/partDescription	
CIDX.xsd/PO/Contact_CIDX/contactFunctionCode		Excel.xsd/Address_Excel/postalCode
		Excel.xsd/PurchaseOrder_Excel/Header/ourAccountCode
		Excel.xsd/PurchaseOrder_Excel/Header/yourAccountCode

Figure 21: Comparison of correspondences found by two approaches

Figure 21 show that cover-based matching (i.e. correspondences generated through shared concept matches) generates more leaf-attribute correspondences than AMC. We can observe that *street-street* correspondences constitute the major part of correspondences between two schemas but AMC misses most of them. While this can be a special case, schema matchers frequently miss out correspondences. However, cover-based can find all these correspondences as it is greedy to get all the possible paths between two schema. This is the reason why cover-based matching generates more correspondences than AMC. Figure 21 also shows that correspondences that are found by both approaches are actually correct. Therefore, we can consider correspondences that generated by both AMC and cover-based matching are correct.

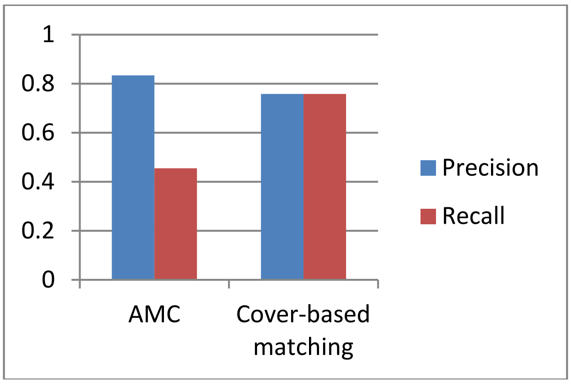


Figure 22: Comparison of precision and recall between two approaches

Figure 22 illustrates the precision, recall of two approaches. As AMC generate a small number of leaf-attribute correspondences which are only 18, its recall is very low but its precision is high (over 80%). Cover-based matching, on the other hand, generates more leaf-attribute correspondence (33 correspondences) but both the precision and recall are acceptable as they are higher than 70%. We can conclude that using this particular dataset, cover-based matching works better than AMC. (We would like to emphasize that this is a preliminary result. For more general statements, we need to conduct more extensive experiments with larger datasets. Also, we need further experiments to ensure, that this statement is not dependent on particular settings of AMC.)

Based on Figure 21, we also have an observation that most correspondences generated by only one approach are incorrect. However, there are also correct correspondences that are generated either only by AMC or only through cover-based matching. This problem together with whether one approach can help the other to improve matching result is discussed next.

5.5 Discussion

In Figure 23: Quantity-qty can only be found by cover-based matching , the correspondence between *quantity* and *qty* cannot be found by the commercial matcher as it is basically based on string matching. On the other hand, cover-based matching can find the intermediate concept attributes to connect these two schema attributes.

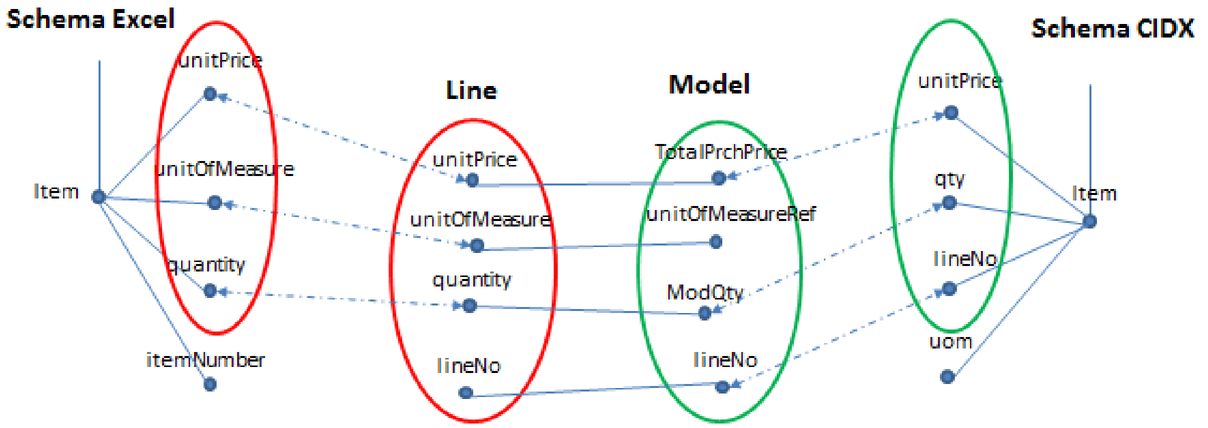


Figure 23: Quantity-qty can only be found by cover-based matching

In some cases, cover-basfed matching may emit incorrect correspondences if the matchings from the schemas to the concept repository are not correct. In Figure 21, cover-based matching considers *startAt-state* as a correct correspondence because these two attributes are matched to attribute *state* of a concept. The reason is that during the decomposition phase, we use a matcher to generate correspondences from the schema to the concepts. Therefore, the incorrect correspondence from schema attribute *startAt* to concept attribute *state* is found. This problem is hard to fix as the incorrect correspondence is actually generated by the matcher and only inherited by cover-based matching.

In addition, the cover-based matching cannot find the correspondence if there is no corresponding attributes in the repository. An illustrative example is the correspondence between *count* and *itemCount*. This correspondence is easily found by the commercial matcher. However, because we cannot find any attributes in the repository that are related to these attributes. Therefore, this correspondence is missed by cover-based matching. This issue shows the limitations of the cover-based matching. To avoid such problems, and enable the use schema covers in a more widely, we designed improvement strategies for schema covering in Section 4.

Another case we found in the experiment is that some correspondences are missed by both approaches, for example, *unitOfMeasure* and *uom*. However, as described in Section 3.1.2, we can use the micromapping between two subschemas as an evidence to increase the confidence value of the missing correspondences.

6 Improving mappings through user feedback

We discuss how to use user feedback to improve mappings created in a pay-as-you-go fashion. We demonstrate through a set of experiments, that a network of schemas can significantly contribute to improve the quality of schema mappings. Similar questions were also raised in the literature, for example the paper [Jeffrey et al.] follows such an approach, in a different setting.

This section presents our initial efforts in understanding how user feedback can improve initially generated mappings. In this work we would like to understand the relation between the order in which users give feedback and the quality improvements. In particular, we would like to understand what the most efficient way of presenting the correspondences to users is. This work is at the same time first step towards understanding the schema mapping reconciliation process.

6.1 Setting

Let assume we have a set of schemas and matching results between each pair of schemas (generated by some well-known matchers e.g., COMA++, AMC). As we assume that the generated matchings are imperfect or inconsistent, it is natural to assume that one needs to reconcile these errors. Several schema matching tools enable the users to adjust initially created mappings. This task is considerably more complex in the case of a network.

In our experiments, users verify correspondences, one by one. We conduct several experiments, in which the correspondences are ordered in different ways. We do not pre-order them; rather we order them on-the-fly, using different strategies. Using these strategies, depending on the user input and the conflicts, we re-order the remaining correspondences. Depending on the ordering strategy, we might require different number of input steps.

Technically, in our experiments we make use the golden correspondences (i.e. the ground truth). Actually, they were constructed through human input: in fact, we constructed this data for ourselves. When we study the order in which the users review the correspondences, technically we process the correspondences and compare them with the ground truth (human input) in different order.

6.1.1 Assumptions

- We assume users are expertise and their answers are always right. (While this assumption might be questionable, at this point, we make this simplifying assumption.)
- We consider only individual user feedback case. That is, there is only one user who checks and corrects all mappings. Thus, the combination of multi-users feedback is beyond the scope of this deliverable. While feedback from many users could be useful, we have no data available at the moment. We plan to come back to this question in a later phase of the project.

6.2 Suggestion and Feedback model

We run the experiments for each ranking strategy as follows.

- First, at the beginning of iteration, we ranked all correspondences based on one of the ordering strategies. After that, only the top (i.e. first ranked) one is suggested to the user for soliciting feedback and the number of feedbacks increase by one.
- Once a user reviewed a correspondence, we update the network and re-ranking the remaining correspondences. Technically, we do not work with real users; rather we rely on the ground truth (that was constructed through human input). We iterate this step until there are still correspondences.

6.3 Computational model

We have a suggestion model with M_C correct mappings and M_I incorrect. The goal is that the user should fix all M_I incorrect mappings. Here, we show how to calculate the user's effort based on the number of feedbacks in detail.

We study the number of user feedback assertions needed to achieve a clean set of correspondence. The number of necessary assertion steps depends on the order the system presents the correspondences to review. The actual effort needed by the user has a lower and upper bound. In an optimal case, the user only needs to review the incorrect correspondences, while in the worst case he needs to review everything.

- *General case*:
  - Actual User's effort AUE= $\frac{F}{M_C + M_I}$
  - F: total of user feedbacks when the perfect network acquired

- *Optimal case:* none of correct mappings MC is suggested to the user (i.e., all incorrect mappings are continuously presented from beginning to the end of the feedback cycle). Due to this, the feedback cycle stop after MI iterations that means F equal MI in this case.
  - Min User's effort Min_UE= $\frac{M_I}{M_C + M_I}$
- *Worst case:* all mappings are presented to the user. The user needs to examine all correspondences for fixing the final incorrect mapping; hence, F equals total number of correspondences in the network.
  - Max User's effort Max_UE= $\frac{M_C + M_I}{M_C + M_I}$ = 100(%)

Now we introduce a metric, called *corrected_ratio,* to measure the percent of incorrect correspondences that are corrected. Let assume that user corrected M_{ct} incorrect correspondences after i^{th} iteration of the feedback cycle. In formally, *corrected_ratio* is defined as follows:

- corrected_ratio = $\frac{M_{ct}}{M_I}$

6.4 Correspondence ordering strategy

We applied the following ordering strategies.

- *Most-Conflict*: based on the occurrence of a mapping in problematic circles in descending order. The intuition behind this strategy is that the correspondence involved in the most problematic circles and parallel paths is more likely uncertain than others. Therefore, confirming it earlier could help.
- *Less-correct*: based on the occurrence of a mapping in correct circles and parallel paths in ascending order. The rationale behind this strategy is that the more a correspondence

appears in correct circles and parallel paths, the higher the matching quality is. Hence, the first confirmed mapping is usually not included in any parallel paths.

- *Random*: the naive strategy for ordering confirmations is to treat each correspondence candidate as equally important. Thus, the next correspondence to confirm in this strategy is chosen randomly. This strategy provides a baseline to which the above strategies can be compared.

6.5 Experimental results

We describe here a set of experiments we conducted. They demonstrate the performance of our constraint-based suggestion & feedback component in terms of pre-defined metrics. We also run our set-up under variable factors to evaluate the common ground of different experimental results.

6.5.1 Evaluation procedure

We configure matching and graph parameters to evaluate our constraint-based suggestion & feedback component.

- **Goals:** In user-centric model, quality metrics may not be precision and recall. It should be amount of time to create perfect mapping. Therefore, we proposed some suggestion and feedback model based on consistency constraints. In this experiment, we examine the effectiveness of these ordering strategies.
- **Metrics:** how many feedbacks we need in order to get perfect matching (all correspondences are correct).
- **Factors:** (1) business interaction graph; (2) correspondence ordering strategy(randomly, or top-rank)

6.5.2 Evaluation Settings

- *Input*
  - COMA++ matcher with the threshold is 0.3.
  - Dataset: Business Partner with 3 schemas
  - Business interaction graph: 481 mappings, in which there are 260 correct mappings and 211 incorrect ones.
- *Output*
  - Perfect schema matching network where all incorrect correspondences are corrected.

6.5.3 Results

Figure 25 describes the results of three correspondence ordering strategies with the optimal case. Generally, the steeper the slope is, the better the ordering is. In three strategies, the *Less-Correct* one shows the outstanding performance with only 63% while two others require 100% user's effort. Regarding *corrected_ratio*, the *Less-Correct* strategy produces a network that is 80% of the perfect network with same number of feedbacks (i.e., user effort) as in the optimal case. Additionally, an interesting heuristic can be inferred is that mappings do not included in any correct circles and parallel paths have high probability of being incorrect mappings.

In contrast, the slopes of the curves for the other strategies are more moderate; it takes many more confirmations to produce a network with a high quality. The *Most-Conflict* strategy has a good beginning but after about 20% of confirmations (i.e., user's effort) it starts gradually decreasing and becomes the same as the Random strategy in the end. The Most-Conflict performs poorly because of the number of problematic circles and parallel paths rapidly fall after starting feedbacks as illustrated in **Figure 26**. Consequently, the later correspondence candidates generally have a similar number of involved problematic circles and parallel paths, hence, their uncertainty is similar too .The quality of the network increases roughly linearly as the percent of confirmations increase for the *Random* curve since it treats each correspondence candidate as equally important.

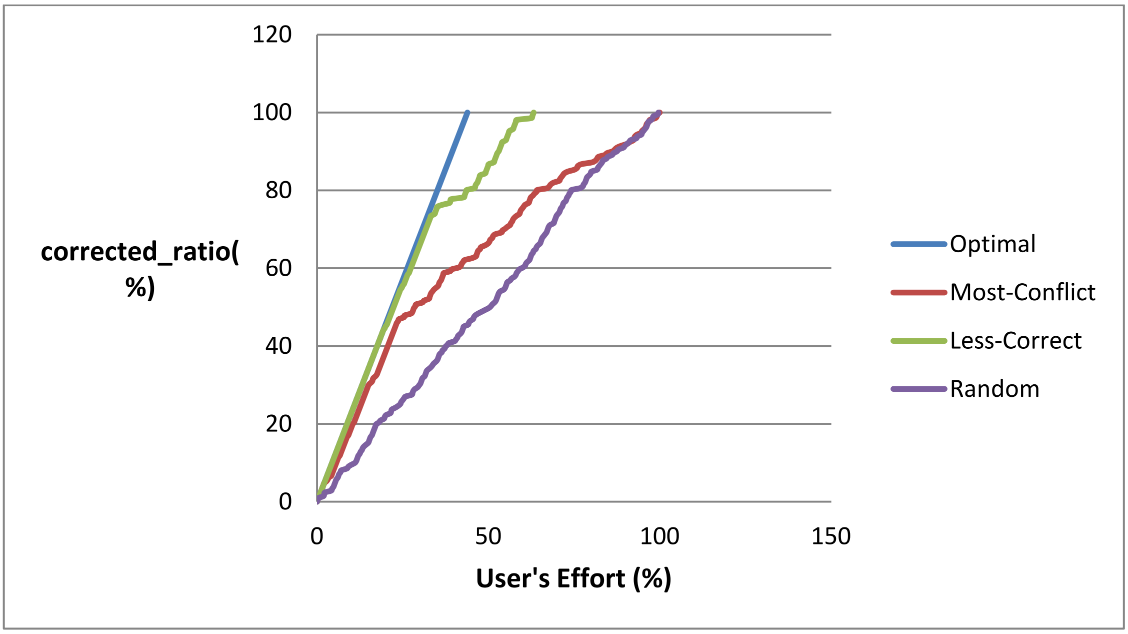


Figure 24. Comparing corrected_ratio based on user's effort between three correspondence ordering strategies constraints

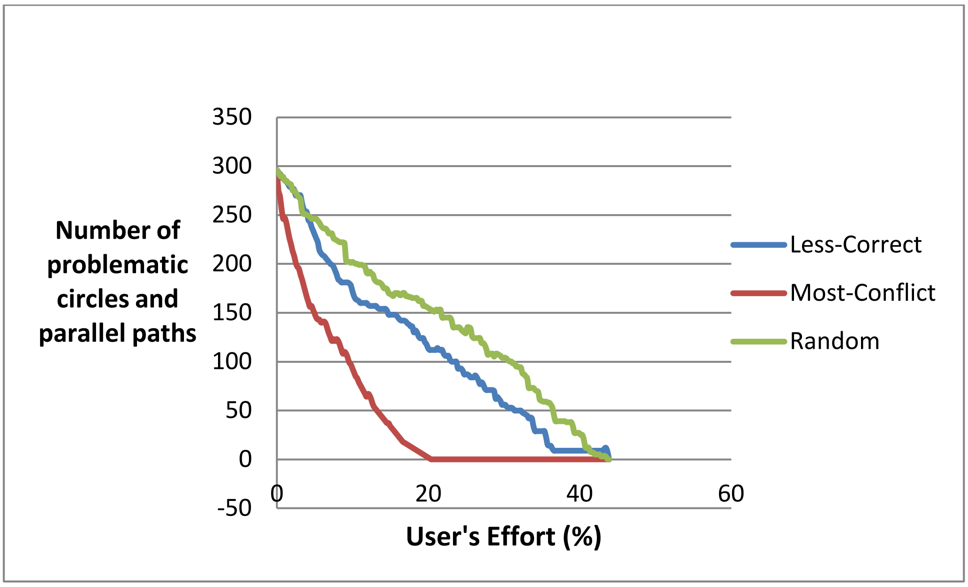


Figure 25 Comparing the number of problematic circles and parallel paths, in three different strategies

7 Conclusion and Future Work

We have discussed various methods, how to improve initially generated schema mappings, in particular with the help of micro-mappings and their combination. Unfortunately the experiments show only moderate success with improvements. There are several reasons for this. For example, we were working with very small datasets, thus the statistical effect were not present. Also, estimating probabilities (e.g. for deciding whether or not to apply an improvement technique) is hardly possible with small collections.

Nevertheless, our work helps to understand the problems and possibilities for improvements. In particular the results on human input are promising. Human input and feedback is expensive but reliable source of information. Thus, even if it is only partially available, it can be very useful to improve the quality of matchings. We are currently working on a better understanding of the effect of human input, through developing a quality analysis and estimation framework (in the task T3.2). We will report the results of these activities in D3.3.

Overall, we would like to better understand what the most effective use of human input is. This work will also contribute to WP4: certain micro-mappings have a much higher utility (as they play a central role, their change can more easily lead to inconsistencies, or they represent a piece of information that is particularly valuable).

8 Glossary

Acronym	Explanation
NisB	“Network is the Business” STREP project

9 References

[Bernstein & Rahm, 2001] A Survey of Approaches to Automatic Schema Matching. The VLDB Journal, 10(4):334-350, 2001.

[AMC] E. Peukert, J. Eberius, and E. Rahm. AMC - A framework for modeling and comparing matching systems as matching processes. In ICDE'11, pages 1304-1307, 2011.

[gCover] Karl Aberer, Avigdor Gal, Michael Katz, Eliezer Levy, Zoltan Miklos, Nguyen Quoc Viet Hung, Tomer Sagi, and Victor Shafran. A Generalized Cover Problem for Schema Matching. (manuscript), 2011.

[Jeffrey et al.] Shawn R. Jeffery, Michael J. Franklin, and Alon Y. Halevy. 2008. Pay-as-you-go user feedback for dataspace systems. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data* (SIGMOD '08)

[Aberer et al. 2003] Karl Aberer, Philippe Cudré-Mauroux, Manfred Hauswirth, Start making sense: The Chatty Web approach for global semantic agreements, Web Semantics: Science, Services and Agents on the World Wide Web, Volume 1, Issue 1, December 2003, Pages 89-114.