THE NETWORK IS THE BUSINESS

D3.2 – Emergent semantics in NisB network v1.0

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

Document History

Version	Date	Changes	From	Review
V1.0	August 2, 2011	First draft	Zoltan Miklos	
V1.1	August 16,2011	Content added	Zoltan Miklos	
V1.2	September 16,2011	More content added	Zoltan Miklos, Ngyuyen Quoc Viet Hung	
V1.3	October 10, 2011	Finalizing for peer review	Zoltan Miklos, Ngyuyen Quoc Viet Hung, Nguyen Thanh Tam, Duong Chi Thang, Do Son Thanh	
V1.4	October 25, 2011	Incorporating peer review comments	Zoltan Miklos	Eliezer Levy, Matthias Weidlich

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

This deliverable describes our studies on bottom-up establishment of semantic relations, in the context of business networks. Emergent behaviour of semantic relationships was studied in many contexts, in particular in Peer-to-Peer (P2P) networks. While in business settings the interoperability establishment is often happening through adopting standards, in today's agile business environments businesses need to interact with previously unknown companies on a frequent basis. As a result, they might not have the time for adopting the standards since it could take longer time and it could also cost more, especially if it is needed only temporary and no further communication is needed, so the time and financial costs might not be justified. Moreover, as many business standards coexist, which are adopted in some specific sub-domains, in a heterogeneous and global space, where the businesses interact, there is a high chance that partners use different standards. Thus, the semantic interoperability problem needs to be addressed.

Building up on the related works on emergent semantics, detailed in deliverable D1.1, we have developed emergent semantic techniques, particularly tailored for business networks and the NisB setting. This deliverable describes our achievements and ongoing work on this topic.

The deliverable is organized as follows. In Section 2 we discuss the problems of schema matching and the particularities of emergent semantics for business networks. In Section 3 we discuss the consistent mapping networks model and algorithms. These techniques are in fact emergent semantic techniques in a setting somewhat simpler than the NisB model. This simpler setting is however very close to the NIsB model and it enabled us to test and demonstrate the effectiveness of our methods. Moreover they form the basis for our more advanced algorithms, in the full NisB context that is described in Section 4. We are currently working on the evaluation of these more advanced methods. Finally, in Section 5 we conclude the status of our efforts and outline the next steps for our work.

2 Schema matching and uncertainties

In NisB project we are concerned with interoperability establishment inB2B networks, in particular for SMEs. On the technical level this requires interoperability establishment between database schemas, business standards, or other descriptions of business related documents. Our work is also closely related to schema matching in large enterprises [Smith et al. 2009]. While we are not only focusing on database schemas, our work heavily relies on schema matching techniques. In this section, we make some important observations about schema matching. While in this deliverable we are referring to schemas, they could equally mean other business interfaces as well.

Schema matching is the process of establishing correspondences between independently designed database schemas. The attribute correspondences can then be used to express relations between the data, or business documents, that are described by the schemas. Thus an attribute correspondence reflects in indirect ways the more complex informal relations and business goals: for example one might want to establish a connection between *customer* and *client* present in two schemas. Clearly, the semantic of these attributes - if we look the attributes as words in natural language- have a different meaning. In a business setting however the business goals and the given context might indicate that these two attributes should be matched. While the context and business goals are hidden or implicit, one still would like to establish connections, if possible automatically – to save human effort and thus costs.

It is clear from this setting, that schema matching is an inherently uncertain process. Sometimes even human observers cannot easily determine the attribute correspondences based on the schema alone. Schema matching tools intended to help in this process: they take two schemas as input and produce attribute correspondences using heuristic methods. Clearly, they evaluate the similarities of the attribute names (as strings), exploit the structures of the schemas, etc. For a survey, see [Bernstein Rahm,2001]. The schema matchers often work in two phases:

1. First they create a set of candidate correspondences (together with some confidence values)
2. Then they chose a set of such correspondences, such that they satisfy the expected constraints (for example, one-to-one correspondences) and maximize the total utility (i.e. they try to choose high confidence values).

Figure 1 depicts one such situation, where a schema matcher produces various attribute correspondences, with different confidence values.

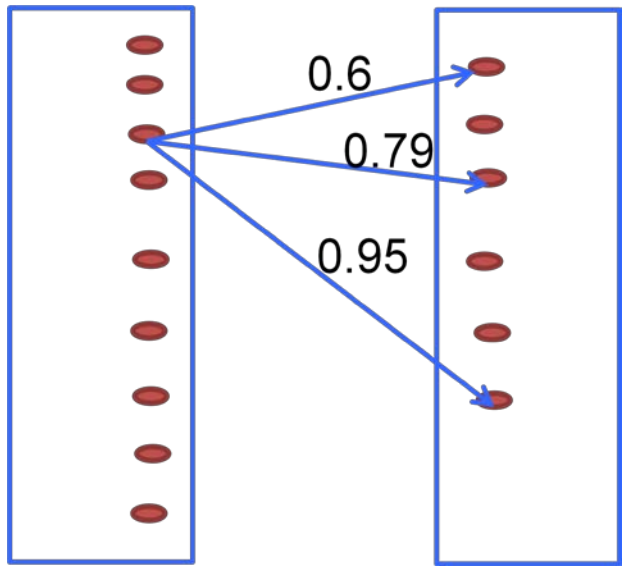

Figure 1 Uncertainty of attribute correspondences

3 Consistent Mapping Networks

In this Section we describe a set of algorithms that we developed. These concepts and algorithms we often refer -especially on the NisB wiki page https://research-projects.sap.com/nisb/wiki/index.php/CMN2.0 - as Consistent Mapping Networks, v1.0, or CMN v1.0. We developed these techniques very early in the project, in parallel with the common NisB terminology. The setting we assume for these algorithms is slightly different from the setting we eventually agreed. Nevertheless, we think that this setting is very important and valid for many reasons. Firstly, it is a very frequent and common setting in real business environments. Secondly, it is a rather simple model, thus it enabled us to experiment and develop useful techniques that we later extended for more complex settings. This extended setting is discussed in Section 4 that is sometimes also referred as CMN v2.

This section is organized as follows. Section 3.1 gives an overview of the CMN approach. In Section 3.2 we give details about the constraints we used, while Section 3.3 explains our constraint repair heuristics. Section 3.4 gives an overview of our testbed that we used for testing the repair heuristics. We further developed this testbed, and realized a schema reconciliation tool that we describe in the Appendix (Section 8). Finally, Section 3.5 gives details about the experimental results we obtained.

3.1 From schema matching to consistency

Schema matchers are mainly used to find corresponding attributes between two data-base schemas. They are often referred as source and target schemas. As the matching process is inherently uncertain, the matching outcomes are often erroneous. Despite the large body of academic research and commercial tool development, one cannot eliminate these problems completely. One should rather assume that the matching outcome has some errors. If establishing the attribute correspondences is a business-critical task, one should rather involve human effort to eliminate the problems in the matching outcome.

If we consider a network of databases, rather than a pair, it is clear that such errors can easily lead to inconsistencies. Consider the attribute correspondences in Figure 2. The matcher between the schemas A and B assigned a correspondence (name,aName). This might be the correct correspondence (or one of the correct correspondences), as well as (name,bName). However, if we consider more schemas, not only two, the correspondences might be conflicting. Such conflict is depicted in Figure 2: on two paths, namely A-B and A-C-B the attribute “name” is mapped differently. This situation is certainly not desirable, if one uses these correspondences for exchanging real data, e.g. business documents, one would have unexpected effects. Such situations should be avoided in business settings.

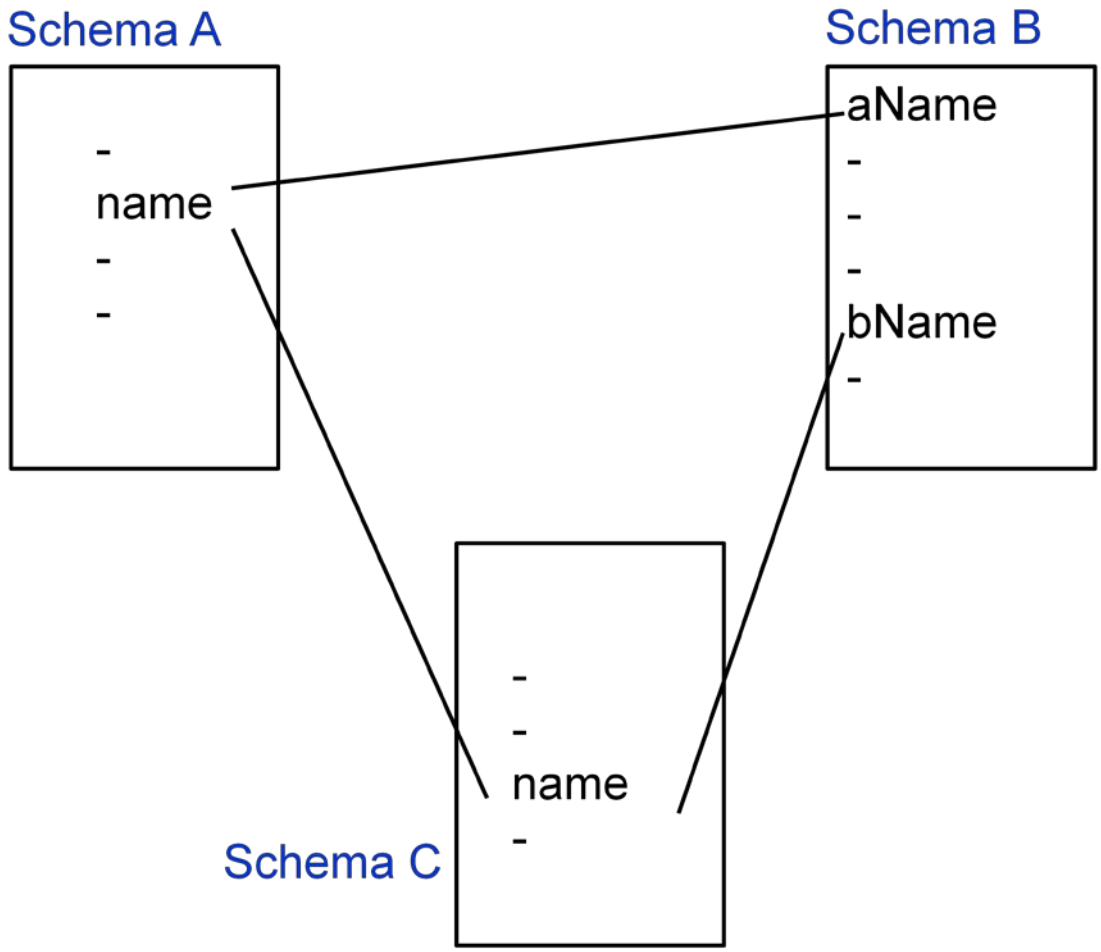

Figure 2 Problematic circle of attribute correspondences

This is exactly what we followed. A mathematical framework widely used in computer science for this type of consistency problems is the so called Constraint Satisfaction Problem (CSP). If we would like to create mappings for a network of databases, we would like to enforce certain constraints. Our setting is however not only a pure CSP, it also involves some optimization, as we will explain later. Our overall strategy is explained in Figure 3.

- The interaction between schemas requires certain mappings to establish interoperability.
- We generate the attribute correspondences with the help of schema matchers, such as AMC [AMC].
- We define a set of constraints. These constraints reflect the business expectations, such as avoiding problematic cycles of correspondences as on Figure 2. (We will discuss the set of constraints in Section 3.2 . See also Deliverable D3.1)
- We solve the constraint satisfaction or optimization problem. We can apply well-known techniques. Repair-based heuristics are especially suited to our setting. (See below).

In the following we discuss the details of the setting, namely the constraints we used, the various heuristic repair techniques, the optimization functions. We conclude this section with the experimental results.

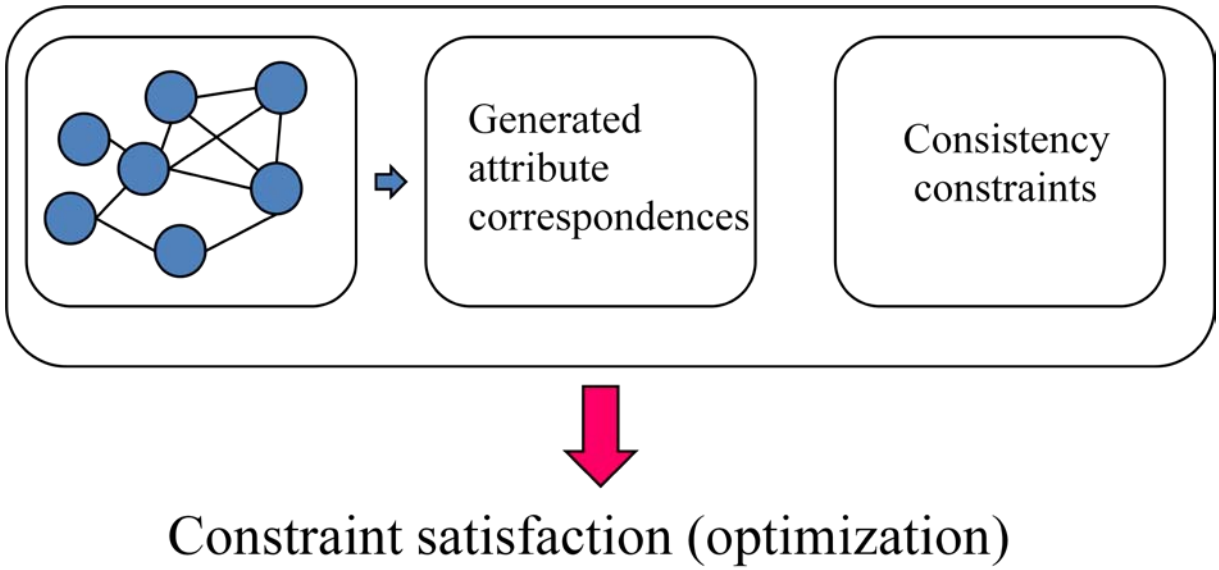

Figure 3 From mapping networks to constraint satisfaction (optimization)

3.2 Consistency constraints

For the CMN experiments we used the following consistency constraints.

3.2.1 One-to-one mappings

We consider one-to-one attribute mappings. This was on the one hand a simplifying assumption in the early phases of the project, on the other hand, at least in our dataset; the majority of correspondences are also of this type. However, it is clear in the phase of deliverable writing that this constraint is too strict, even if the majority of correspondences is of this type, there are very often exceptions. Even if the number of these exceptions is low, there is a high chance, that the correct correspondences are of type 1-to-n or n-to-m.

3.2.2 Parallel path constraint

The Parallel Path constraint is demonstrated in Figure 2. If we have a circle of attribute correspondences, i.e. if we follow up attribute correspondences, and in a path we arrive to the same schema we started at, we should arrive at the same attribute. On such path we should not arrive at a different attribute in the same schema. We call this condition the parallel path constraint. If such a circle violates the constraint, we call it “problematic circle”.

In fact we have studied the available dataset and correspondences generated by schema matchers, the violations of the constraint were problematic cases. Indeed if we consider that businesses will exchange documents corresponding to the schemas, such problematic circles can easily lead to side-effects.
However one needs to be very careful with the cases, where the attribute correspondences are not one-to-one. We handle these cases separately.

3.2.3 Smoothness

The smoothness constraint is a variant of the triangle inequality. During the exploration of the available schemas we realized that the situations depicted in Figure 4 were very often problematic. If a chain of “strong” attribute correspondences connect two attributes (0.99 and 0.95 on the figure)

and the two attributed are also connected by a direct, but "weak" correspondences (0.2), then we say that the correspondences are not smooth. This condition is not derived theoretically, rather through explorations. We have defined minor variants of this condition, as it is not easy not easy to capture the notions of strong and weak. For this reason, we defined minor variants of this condition (for example, we consider the minimum of the confidence values on a path between two attributes, and we require that it should be not bigger than the confidence on the direct link + a threshold.)

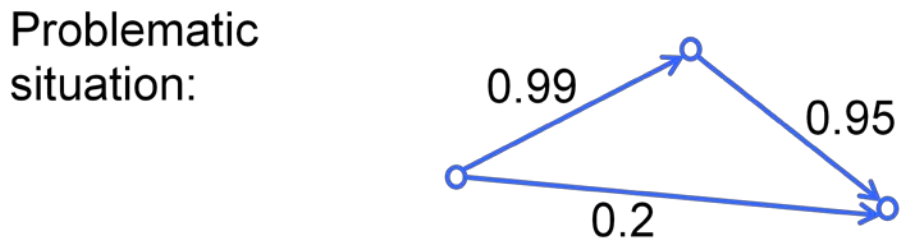

Figure 4 Smoothness constraint

3.3 Repair heuristics

Our algorithms make use of repair heuristics. The repair process starts with the attribute correspondences generated by the matchers. The repair algorithms try on the one hand to do as little changes to this set as possible. On the other hand, the minimal number of changes still leaves several options open, thus we have defined heuristic optimization functions that we try to optimize.

The rationale behind this strategy is that the matchers produce a set of correspondences which they think is the most probably correct, so we would like to remain as close to this set as possible, just without the constraint violations. For the optimization, we would like to maximize the number of correct circles in the network. This is promising as multiple links and paths between attributes could indicate a stronger connection.

Moreover, we apply the following heuristic strategies as well:

- We compute the entropy of the confidence values. For example, if a correspondence has much higher confidence value than any other, than we would realize this from the entropy value. In this case we change the given correspondence with a much lower probability.
- We compute the difference of confidence values between the first and second ranked candidate. If the two values are very close, we would repair the correspondence with much higher probability as if the first would be significantly higher than the second.

Moreover we also consider transitive closure in the repair phase (for attributes). We plan to explore the effects of transitive closure more precisely in the future.

3.4 Testbed

To support the development of the repair algorithms, we developed a testbed. We indeed used the testbed in the algorithm development. We started the development of this tool at a very early stage in the project. While our goal is to use the common project testbed, developed by SAP, for experimentation, the EPFL testbed is still maintained. Technically, we try to use a common model and common interfaces in the development, such that the algorithms and modules shall be easily reused.

The testbed takes the following input:

- Business schemas
- Interaction graph (as a file)
- Generated mappings (the current version does not automatically generate the mappings). We use here the mapping format generated by AMC.
- Parameters

The main goal of the testbed is to visualize the generated mappings and support the algorithm development. The visual appearance of a mapping network with 3 schemas is depicted in Figure 5. The testbed can compute and visualize circles and parallel paths in the mappings, in fact, there are several ways how to select, which one to display. It can display mapping alternatives for a given attribute correspondence as well, together with the confidence value, provided by AMC. Visualizing circles turned to be a very useful feature, as it is hard to overview the circles otherwise.
In the latest version, the users can also select the alternative they suggest to be the correct correspondence.

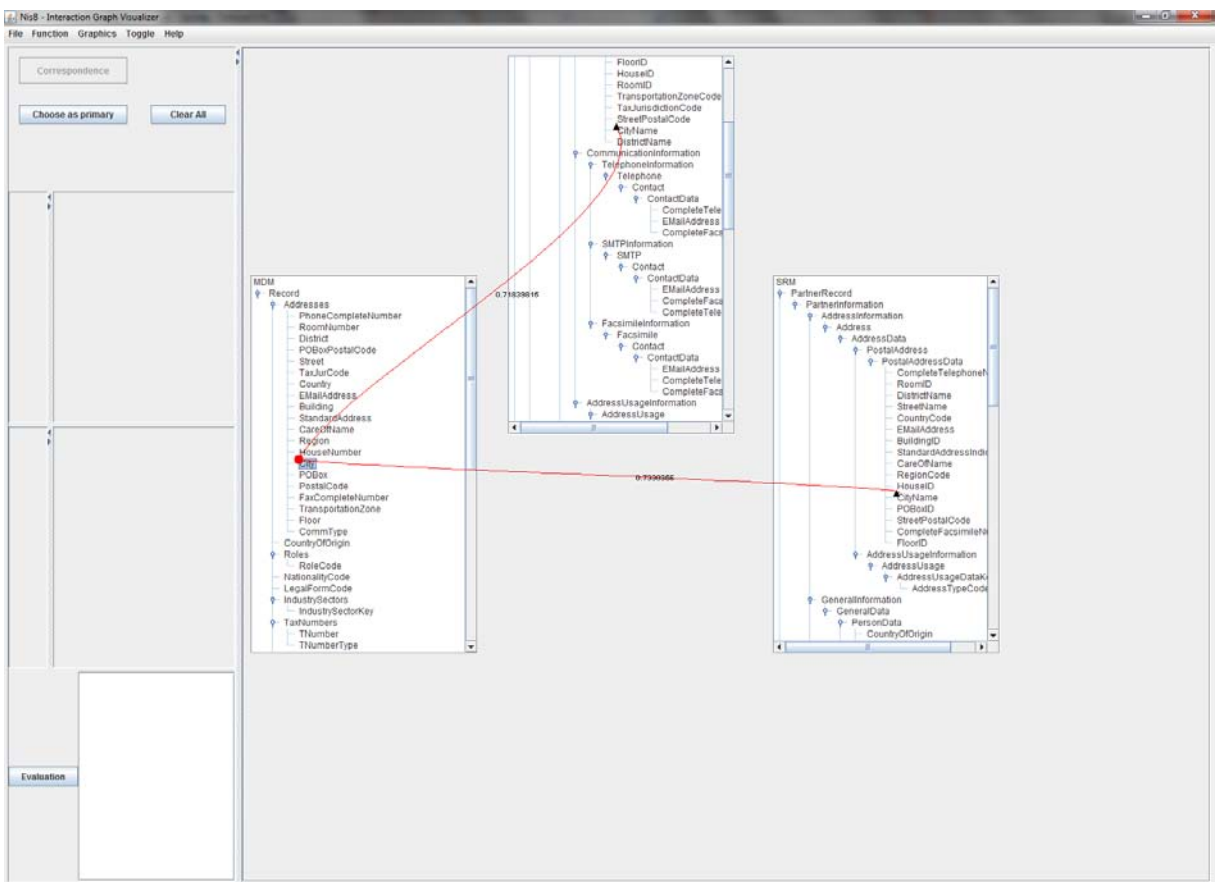

Figure 5 Testbed

The functionalities of testbed make it useful also beyond analyzing the repair algorithms. It can be regarded as a schema mapping network analysis and repair tool. We wrote a demo paper, that we submitted to a database conference. We include the paper in the appendix of this deliverable.

3.5 Experimental evaluation

In this section we summarize the experimental results w.r.t. detecting constraint violations and repairing these violations. They demonstrate the performance of our algorithms in terms of pre-

defined metrics (e.g. precision, recall, count, percentage etc.). We also run our algorithms under variable factors (e.g. matcher, threshold, business interaction etc.) to evaluate the common ground of these experimental results.

3.5.1 Experimental Datasets

To set-up experiments, we choose two datasets of XSD schemas in business domain, namely Business Partner and Purchase Order. Business Partner dataset has 3 schemas, including CRM (103 elements), MDM (79 elements), SRM (105 elements). Purchase Order dataset has 5 schemas, including Apertum (140 elements), CIDX (40 elements), Excel (54 elements), Noris (65 elements), Paragon (77 elements). Table 1 gives an overview of the datasets. The term "Golden mappings" refers to the available ground truth.

Table 1 Dataset

Domain	**Description**	
	Number of Schemas	***Availability of Golden mappings***
Business Partner	3 schemas	Between all pairs of schemas
Purchase Order	5 schemas	Between all pairs of schemas

For more details about the schemas, see the Appendix. Besides, we choose 3 well-known matchers which support XSD schemas, namely COMA++, Auto Mapping Core (AMC), and OntoBuilder.

Table 2 Matching Tools

Matching Tools	**Description**	
	Features	***Supported format***
COMA++	GUI, API, commercial	XSD, XDR, TXT, RDF
AMC	GUI, API, commercial	XSD
OntoBuilder	GUI, API, research	RDF, XSD (partial)

For our experiments, we not only needed business schemas, but information about their interaction. As this was not available, we generated random business interaction graphs.

A business interaction graph is a network of schemas where each edge (i.e. generated mapping) between two schemas represents the interaction of them. Unfortunately we do not have such graphs available. We use randomized techniques to generate such graphs for our experiments. To build this graph randomly, we choose an interaction degree (varying from 0 to 1) as a probability of the existence of an edge between any two nodes. With interaction degree = 1.0, we obtain a complete graph.

As unfortunately our datasets are rather small, we worked with rather dense and complete graphs, to observe the effects of the networks. For a larger set of schemas, if the interaction graph is not known, we need to develop more sophisticated ways to generate them, such that they model realistic business settings.

3.5.2 Detecting constraint violations

3.5.2.1 Evaluation procedure

We study various settings to evaluate our constraint-based detector under important factors such as (1) chosen matchers, (2) threshold of selecting generated correspondences and (3) the business interaction graph.

- **Goals:** we would like to study how much the generated attribute correspondences do satisfy the consistency constraints.
- **Metrics:** The percentage of incorrect correspondences that are detected. An incorrect correspondence is detected if it appears in a problematic circle (i.e. the parallel path constraint is violated). If there are no incorrect correspondences (precision = 1.0) in the generated mappings, the percentage is undefined.
- **Factors:** (1) matchers and (2) threshold for each matcher; (3) business interaction graph (interaction degree of schemas in the network).

3.5.2.2 Results

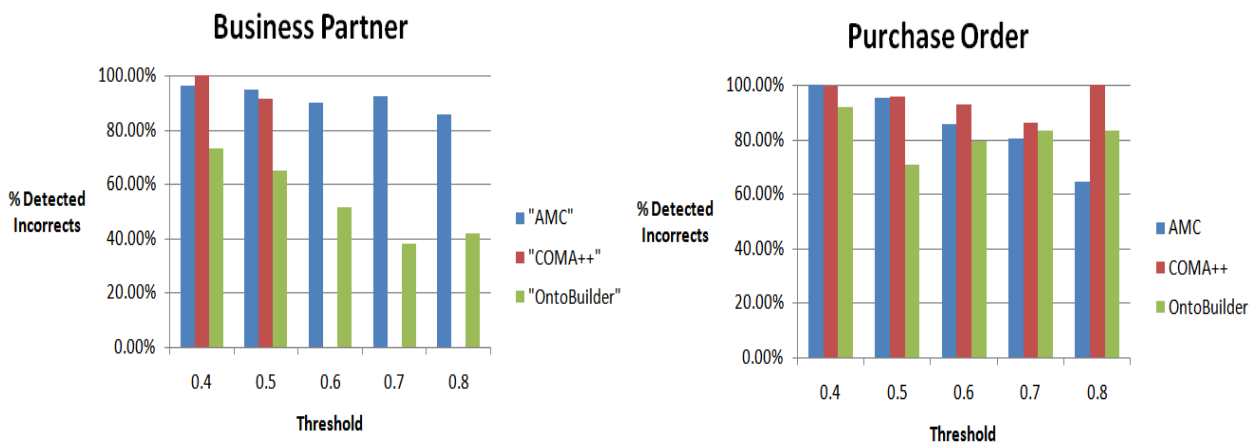

Figure 6 The percentage of incorrect correspondences detected (given a business interaction graph)

3.5.2.2.1 Matcher and threshold

Figure 6 illustrates the performance of detecting constraint violations (through detecting problematic circles) in terms of the percentage of detected incorrect correspondences. In the experiment we used a complete business interaction graph that corresponds to an interaction degree 1.0. Given interaction degree = 1.0, most of the incorrect correspondences are detected with low matching threshold (< 0.6). In the region 0.6-0.8, we could not detect any incorrect correspondences through this method for the correspondences generated by COMA++. (In some cases there were some incorrect ones, but we could not detect, while for higher threshold values there were no such correspondences.)

More precisely, **Table 3** lists the statistics of the total incorrect correspondences in whole graph and the number of detected correspondences. Both datasets show that the number of detected incorrect correspondences is high, when matching threshold is small. This means that the more interaction (i.e. attribute correspondences) between schemas, the more incorrect mappings could be detected.

Table 3 Total incorrect & detected correspondences, Threshold from 0.4 to 0.8 and Interaction Degree = 1.0

Dataset	Threshold	AMC		COMA++		OntoBuilder	
		# Incorrect	*#Detected*	*# Incorrect*	*#Detected*	*# Incorrect*	*#Detected*
Business Partner	0.4	52	50	41	41	71	52
	0.5	37	35	23	21	43	28
	0.6	20	18	4	0	31	16
	0.7	13	12	0	0	21	8
	0.8	7	6	0	0	19	8
Purchase Order	0.4	195	195	188	187	224	206
	0.5	165	157	147	141	160	113
	0.6	92	79	109	101	156	124
	0.7	41	33	79	68	147	122
	0.8	14	9	50	50	147	122

3.5.2.2.2 Matcher and business interaction degree

Because of the above observation, we have increased the matching threshold and change the interaction degree from 0.5 to 1.0. (We recall that the matching threshold is a parameter to the matching tools, while the business interaction degree is a parameter of the randomly generated business interaction graph. Higher interaction degree relates to higher number of business interactions, i.e. more edges in the business interaction graph.) **Figure 7** depicts the performance of detecting constraint violations in terms of the percentage of detected incorrect correspondences. With matching threshold = 0.4, both dataset show that most of incorrect correspondences are detected along with the increase of interaction degree. This also implies that the more interaction (i.e. attribute correspondences) between schemas, the more incorrect mappings could be detected.

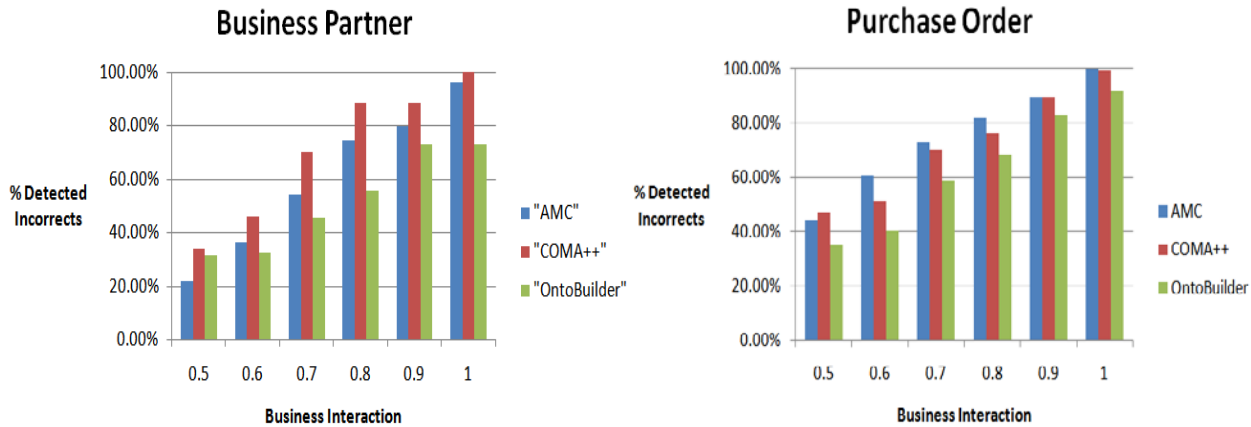

Figure 7 Interaction Degree varies from 0.5 to 1.0 and Matching Threshold = 0.4

In detail, **Table 4** lists the statistics of incorrect and detected correspondences with Threshold = 0.4 and Interaction Degree increases from 0.5 to 1.0. It is clearly seen that more incorrect correspondences will be detected when the interaction degree increases. However, there are still some "tough" undetectable correspondences since they do not appear in any circles or parallel paths.

The number of detected incorrect correspondences is presented as an average over 10 runs of our algorithms. (For this reason they are not always integers.) As we vary the interaction degree, the underlying business interaction graph has different number of edges during these 10 runs.

Table 4 Total incorrect & detected correspondences, Interaction Degree from 0.5 to 1.0 and Threshold = 0.4

Dataset	Interaction Degree	AMC		COMA++		OntoBuilder	
		# Incorrect	*#Detected*	*# Incorrect*	*#Detected*	*# Incorrect*	*#Detected*
Business Partner	0.5	52	11.3	41	13.9	71	22.5
	0.6	52	19	41	18.8	71	23
	0.7	52	28.2	41	28.7	71	32.5
	0.8	52	38.8	41	36.3	71	39.7
	0.9	52	41.5	41	36.3	71	51.9
	1.0	52	50	41	41	71	52
Purchase Order	0.5	195	86.9	188	88.9	224	78
	0.6	195	118.2	188	96.1	224	90.3
	0.7	195	142.4	188	132.2	224	131.8
	0.8	195	159.7	188	143.1	224	153.3
	0.9	195	174.7	188	168.8	224	186.8
	1.0	195	195	188	187	224	206

3.5.3 Consistency constraints and corrector

In the following we present the results on our repair procedures and demonstrate effect of various parameters to the results, in terms of precision and recall.

3.5.3.1 Evaluation procedure

We set-up various configuration to evaluate our constraint-based corrector under important factors such as (1) threshold of selecting generated correspondences, (2) chosen matchers, (3) business interaction graph.

- **Goals:** we study the effect of our repair algorithms. Given a network of schemas, our corrector will select the best instance to maximize the utility function *F(I)* (i.e. the total number of parallel paths).
- **Metrics:** (1) Precision and recall. The performance of corrector is evaluated in terms of precision and recall. By comparing performance before and after applying the repair algorithm, we can verify the correctness of our deleting and adding strategy.
- **Factors:** (1) matchers and threshold for each matcher; (2) business interaction graph. These factors are already explained in above section.

3.5.3.2 Results

Figure 8 Matching threshold varies from 0.4 to 0.8 and Interaction Degree = 1.0

Figure 8 depicts the performance of constraint-based detector in terms of precision and recall. Given Interaction Degree = 1.0, we raise the matching threshold from 0.4 to 0.8. In Business Partner dataset, the repair algorithm improves precision and recall slightly when the matching threshold is low (< 0.6). In Purchase Order dataset, the precision is improved by corrector but the recall is reduced nevertheless. The decrease of recall is caused by the repair step: the algorithm deletes correspondences that violate the defined constraints.

Figure 9 Interaction Degree varies from 0.5 to 1.0 and Matching Threshold = 0.4

In the second setting, we let the interaction degree increase from 0.5 to 1.0 while matching threshold is fixed to 0.4. **Figure 9** illustrates that the performance of whole network is improved along with the increase of interaction degree. This suggests that the more schemas participate in the consistent mapping network, the better chances of improving mapping quality in terms of precision and recall.

4 Consistent Mapping Networks with Business Concepts

The CMN model and algorithms (see Section 3) has promising results in terms of possible improvements. Our goal and ongoing effort is to extend the model with the business objects and make more use of the NisB network. Moreover, the evolution of the NisB network should be also driven, such it can support the interoperability establishment and contribute to matching improvements.

4.1 Extending the CMN with business concepts

We have conducted a set of experiments that combine the CMN model with the schema covering approach and the NisB network. We reported this work in D3.1. The experiments are still ongoing; unfortunately the results on our currently available datasets show moderate results.

We are in the progress of defining the constraints for this case. In the latest version we make use of variants of constraints we described in Section 3.2. In particular we would like to relax the one-to-one correspondence constraint.

We are in the process of implementing the adapted algorithms in close cooperation between SAP and EPFL, based on the common NisB code-base.

4.2 Evolution mechanisms

We are currently in the process of identifying the evolution mechanisms suitable for the NisB network. We would like to demonstrate the emergent, self-organizing and self-improving mechanisms, in the context of business networks and schema matching. For the moment, we cannot demonstrate the emergence in a larger network, thus we will simulate the situation. However, we would like to keep the simulation as close to real situations and of course to use cases as possible. Thus the most promising direction is to simulate the behaviour of users and user input and feedback.

4.3 Experimental results

We would like to analyze the evolution mechanisms, in the presence of business concepts. As a first step we need to better understand the concept repository generation process, in particular because we need such a repository for CMN2.0. One of the mechanisms we are studying in NisB is the decomposition of schemas that is one possible way for obtaining a concept repository. In the following we study experimentally the repository construction process.

4.3.1 Setting

We have a set of schemas in the same domain. We want to create a concept repository from these schemas by breaking the schemas into concepts. The mappings between concepts are created from the exact mappings between schemas. In the next section, we describe how to create the repository in detail.

4.3.2 Concept repository building process

First, we break down the schema into concepts based on some predefined strategies. Here we list some strategies and their description:

- Leaf-only concepts: from a set of leaf-attributes that have the same parent, we create a concept. This concept contains all these leaf attributes. The concept name is the parent attribute's name.

- Two-level concepts: for each attribute that have a set of sub-attributes, we create a concept from this set. The concept contains all the sub-attributes. The concept name is the attribute's name.

After breaking the schemas into concepts, we connect these concepts using the available golden correspondences (i.e. correspondences present in the ground truth). For each exact correspondence, we find the concept that contains the source attribute and the concept that contains the target attribute then we create a correspondence between these two attributes. We apply this process to generate all correspondences between the concept attributes. Finally, we have a concept repository that contains concepts and correct mappings between concepts.
Moreover, because concepts have mappings, we can group the concepts that are connected. Using this approach, we can group the concepts in the repository into clusters.

4.3.3 Metrics

Let n_C be the number of attributes in a concept C. Let d_C be the number of distinct attributes in a concept C that participates in a correspondence. The similarity value between concept A and concept B is calculated as follows:

- $sim(A,B) = \frac{d_A}{n_A} * \frac{d_B}{n_B}$

We denote $x = \frac{d_A}{n_A}$ and $y = \frac{d_B}{n_B}$.
If the similarity value of two concepts is 1, it means that every attribute in one concept has a corresponding attribute in another concept and vice versa. In other words, these concepts are basically the same.
However, if either x or y is 1, this means one concept is contained in another concept. The reason is that while one concept has all its attributes participate in correspondences, the other concept has only a few attributes doing so. In this case, one concept can be considered the parent concept of the other and these two concepts have a containment relationship. Moreover, if x and y are both low, two concepts are highly dissimilar as there are few connections between them.

4.3.4 Evaluation procedure

- Input
  - A set of schemas in the same domain: Apertum, Noris, Paragon
  - Leaf-only concept strategy
  - Exact correspondences between them
- Output
  - A concept repository of concepts and mappings between concepts
- Goal: analyse the characteristics of the concept repository
- Metrics: similarity value between concepts.

4.3.5 Results

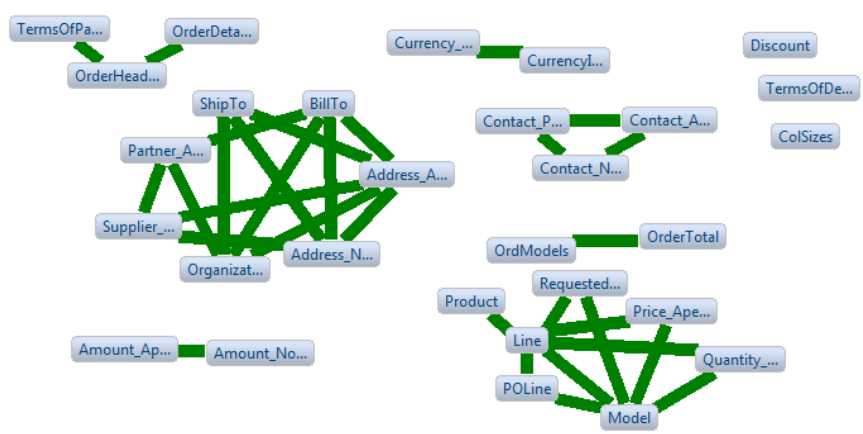

Figure 10: The concept repository and the mappings between concepts

Figure 10 illustrates the concept repository after we break down the schemas into concepts and connect these concepts using the ground truth. It is clear that the concepts form many groups. The concepts in each group belong a certain domain. For example, Contact_Apertum, Contact_Noris, Contact_Paragon concepts described the concept contact. However, the links between concepts may be different as concepts similarity values are different. We will analyze these groups in more detail in Figure 12.

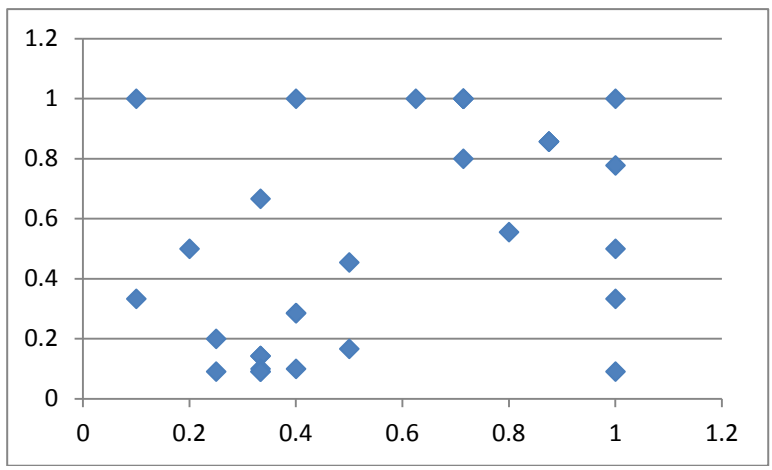

Figure 11: Plotting points of x and y of the concepts in the repository

In Figure 11, if two concepts are connected, we draw a point based on x and y defined in Section 4.3.3. We can observe that the points scatter among the space. However, there are various points lie on the x=1 and y=1 line which indicates there are numerous containment relationship in the repository. Another observation is that a lot of points are contained in the circle that has the center at (0,0) and the radius is 0.4. This observation also points out that some concepts are connected by only a few correspondences. The mappings between these concepts should be removed as these concepts are dissimilar. Analyzing the raw data, we also get that there is only one point that its x and y are both 1. This means that there is only a pair of concepts that are identical in this repository.

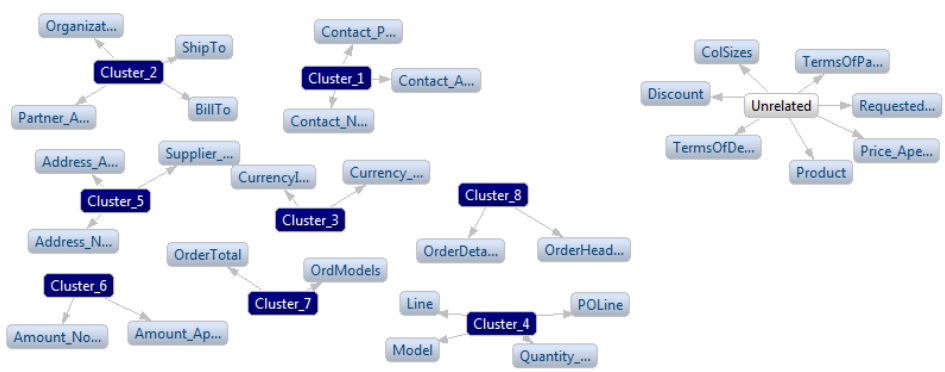

Figure 12: Clustering concepts with the similarity value threshold is 0.3

In Figure 12, we cluster the concept using DBSCAN clustering algorithm. Two concepts are considered in the same cluster if their similarity value is higher than 0.3. Comparing this figure with Figure 10, one can observe that many concepts that are connected in Figure 10 but don't belong to the same cluster in this figure such as Organization and Address_Apertum. The reason is obvious that their similar value is lower than 0.3 as they are not similar.

5 Conclusion and future work

We have developed an algorithmic framework for improving the quality of mappings created in a pay-as-you-go fashion, for business networks. In this setting it is essential to ensure consistency; this is why we focused on a constraint satisfaction and optimization-based framework. We demonstrated on the available dataset that our techniques indeed can improve the quality of mappings. While this setting is somewhat simpler than the NisB use cases, the results are promising. However, our datasets are rather small, thus one needs to be careful with the conclusions. We are looking forward to repeat the test on the use case data.

We are also working on a more comprehensive framework, building upon our results. In this case we also consider business concepts (of two types, induced concepts and concepts from repositories derived from business standards). We apply similar algorithms, but we would like define more complex constraints. We also relate our work to WP2 on schema covering that can help us to define better constraints. On the experimental side, we are in the process of demonstrating the usefulness of the more comprehensive setting. We also make use of user feedback that should get even more attention in our future work.

6 Glossary

Acronym	Explanation
NisB	Network is the Business, EU project
AMC	Auto Mapping Core, see [AMC]

7 References

[Smith et al. 2009] Smith, K., Morse, M., Mork, P., Li, M.H., Rosenthal, A., Allen, D., Seligman, L., and Wolf, C. The Role of Schema Matching in Large Enterprises. In Proceedings of CoRR. 2009.

[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.

8 Appendix

8.1 Dataset Description

We describe here the datasets we are using in our experiments. Regarding the provenance of the Purchase Order dataset: we have collected the schemas and we have constructed the golden mappings manually.

Until now, we have prepared schemas from various domains. The datasets were chosen by their common usage and real-world characteristics. We have two datasets of XSD schemas in business domain, namely Business Partner and Purchase Order. Business Partner dataset has 3 schemas, namely CRM, MDM, and SRM. Purchase Order dataset has 10 schemas, including Apertum, CIDX, Excel, Noris, Paragon, CRM, RosettaNet, xCBL, OpenTrans, and SAP.

Table 5 Business-Domain Dataset

Domain	**Description**		
	Schemas and Size	***Format***	***Golden mappings***
Business Partner	**3 schemas:** CRM (103 elements) MDM (79 elements) SRM (105 elements)	XSD	• all pairs of schemas
Purchase Order	**10 schemas:** Apertum (140 elements) CIDX (40 elements) Excel (54 elements) Noris (65 elements) Paragon (77 elements) CRM (50 elements) RosettaNet (141 elements) xCBL (191 elements) OpenTrans SAP	XSD	• all pairs of (CIDX, Excel, Noris, Paragon, Apertum, OpenTrans) • (OpenTrans, xCBL) • all pairs of (xCBL, SAP, CRM, RosettaNet)

8.1.1 Business Partner dataset

Business Partner dataset is a part of SAP Service Interface dataset. There are 3 schemas, namely MDM, CRM and SRM. Each schema has around 70 elements and 3 levels of hierarchy. This dataset gives a good start to run the mapping propagation algorithms in testbed framework with simple set-up and observation. Its schemas and golden mappings are converted directly by-hand from "MDM_5.5_&_7.1_-_Field_Mapping_and_Check_Tables_Business_Partner.xls" to XSD format.

8.1.2 Purchase Order dataset

In this section, we describe the origin of Purchase Order dataset and how did we create the XSD files along with golden mappings. We converted them from their origins in terms of copy/paste manner. As a result, the process of creating XSD schemas is mainly separated from the process of creating golden mappings.

Schemas:

- CIDX.xsd: converted from CIDX.xdr (belongs to dataset *"Sources/PO_xdr/"* of COMA++)
- Excel.xsd: converted from Excel.xdr (belongs to dataset *"Sources/PO_xdr/"* of COMA++)
- Apertum.xsd: converted from Apertum.xdr (belongs to dataset *"Sources/PO_xdr/"* of COMA++)
- Paragon.xsd: converted from Paragon.xdr (belongs to dataset "*Sources/PO_xdr/"* of COMA++)
- Noris.xsd: converted from Noris.xdr (belongs to dataset *"Sources/PO_xdr/"* of COMA++)
- OpenTrans.xsd: its origin is OpenTrans_ORDER.xsd (belongs to dataset *"Sources/OpenTrans/"* of COMA++). By the way, you should see openTRANS business document standards in http://www.opentrans.de/
- xCBL.xsd: its origin is *"schemas/xcbl/v4_0/ordermanagement/v1_0/ordermanagement.xsd"* (xCBL dataset http://www.xcbl.org/). For testing purpose, we simplified it into only one XSD by cutting-down attributes from 5th-level to avoid <xs:include> multiple xsd files. Please refer to original schema for complex schema structure.
- RosettaNet.xsd: its origin is "*RosettaNet\PIP3A4 Purchase Order\PIP3A4_V11.14.00_RequestPurchaseOrder\XML\Interchange\PurchaseOrderRequest_02_05.xsd"* (RosettaNet dataset http://www.rosettanet.org/). For testing purpose, we simplified it into only one XSD by cutting-down attributes from 5th-level to avoid <xs:include> multiple xsd files. Besides, we also insert more attributes from *"PurchaseOrder_ElementStructure_2004-07-12.xls"*. Please refer to original schema for complex schema structure.
- SAP.xsd: copy/paste converted from *"PurchaseOrder_ElementStructure_2004-07-12.xls"*
- CRM.xsd: copy/paste converted from *"PurchaseOrder_ElementStructure_2004-07-12.xls"*

Golden mappings:

- 30 directed mappings (CIDX, Excel, Noris, Paragon, Apertum, OpenTrans): copy/paste converted from mappings-PO.txt (belongs to dataset *"Sources/Mappings/"* of COMA++)
- 2 directed mappings (OpenTrans, xCBL): copy/paste converted from mappings-OP-Xcbl.txt (belongs to dataset *"Sources/Mappings/"* of COMA++)
- 12 directed mappings (xCBL, SAP, CRM, RosettaNet): copy/paste converted from *"PurchaseOrder_ElementStructure_2004-07-12.xls"*

8.2 Demo paper

We attach here the demo paper that we submitted for publication.

SMART: A tool for analyzing and reconciling attribute matchings for a network of database schemas

[Demo paper]

Nguyen Quoc Viet Hung
EPFL IC LSIR
Lausanne, Switzerland
quocviethung.nguyen@epfl.ch

Zoltán Miklós
EPFL IC LSIR
Lausanne, Switzerland
zoltan.miklos@epfl.ch

Nguyen Thanh Tam
EPFL IC LSIR
Lausanne, Switzerland
tam.nguyenthanh@epfl.ch

Duong Chi Thang
EPFL IC LSIR
Lausanne, Switzerland
chi.duong@epfl.ch

Do Son Thanh
EPFL IC LSIR
Lausanne, Switzerland
sonthanh.do@epfl.ch

Karl Aberer
EPFL IC LSIR
Lausanne, Switzerland
karl.aberer@epfl.ch

ABSTRACT

Schema matching is the process of establishing connections between the attributes of independently designed database schemas. This is task is at the core of many data integration problems, and it is highly relevant in everyday practice. To support the manual matching process, a number of academic and commercial schema matching tools have been developed which apply various heuristic techniques to obtain attribute correspondences between two schemas. We consider a practically relevant scenario, where the task is not only two match two schemas, but establish connections in a network of schemas. We still rely on pairwise matching tools, but we need to consider network-level constraints as well.

While the matching tools have acceptable performance in pairwise matching, their output is often incomplete or contains errors, which can further escalate in the network setting. Many matching tools enable to adjust the initially created attribute correspondences. This task is considerably more complex in the network because of the correspondences might be dependent on each other. In this demo we present the Schema Matching Analyzer and Reconciliation Tool (SMART). SMART can detect and visualize network-level inconsistencies. Moreover, it can help to analyze and reconcile the problems and guides the human users through the reconciliation process in an optimized manner. It also offers automatic conflict-resolution techniques.

Categories and Subject Descriptors

H.2.5 [**Heterogeneous Databases**]: Data translation

General Terms

Database

Keywords

Schema matching, consistency, reconciliation

1. INTRODUCTION

Schema matching is the process of establishing connections between the attributes of independently designed database schemas. The outcome of the schema matching process is a set of attribute correspondences, representing the connections between the schema elements. The attribute correspondences can then be used to express more complex mappings, i.e. relations between the data that are described by the schemas. Schema matching tools intended to help in this process: they take two schemas as input and produce attribute correspondences using heuristic methods. There is a large body of research on such algorithms, for surveys, see [8], [3], [5], [2], [9], [10], [6]. Several commercial and academic matching tools have been developed, such as COMA++ [4], [1] or AMC [7].

The schema matchers often work in two phases:

1. First they create a set of candidate correspondences (together with some confidence values)
2. Then they choose a set of such correspondences, such that they satisfy the expected constraints (for example, one-to-one correspondences) and maximize the total utility (i.e. they try to choose correspondences with high confidence values).

Often the available information for schema matching is incomplete or implicit; there is a lot of uncertainty involved in the schema matching process. As a result, even if the matching tools have an acceptable performance in practice, there is often human input needed to finalize the correspondences. The role of matching tools is mainly to save human effort. Indeed, schema matching tools are widely used in practice and they help the work of their users in data integration.

SMART is not a new matcher, rather it can analyze matching outcomes of existing matching tools, in particular when they are used in P2P data integration. This is important

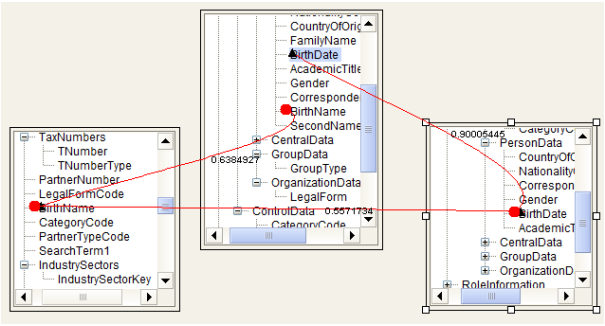

Figure 1: Problematic circle of attribute correspondences.

in various situations where we need to construct matchings in a network, rather than only between two schemas. For example, as businesses need to integrate data from many independently constructed databases, they do not create a global consolidated schema, rather they exchange data in a P2P fashion. Similarly, in a business network in domains where still no business standards exists (or there are multiple co-existing standards) the data exchange (and the schema matching) is happening through pairwise communication, rather than through global coordination. In these settings, as the schema matchers are applied in a pairwise, pay-as-you-go fashion, the matchers do not consider global consistency condition, that easily leads to problems (such as e.g. the same source data items appear unexpectedly several times in the target dataset). Moreover, it can be rather challenging for humans to resolve the conflicts in the generated correspondences, as they cannot be looked independently.

Our tool addresses exactly these points. We demonstrate

- how to analyze matching networks and how to detect conflicts, and
- how to reconcile these conflicts (semi)-automatically and how to support the users in the reconciliation task.

The paper is organized as follows. Section 2 gives an overview of the functionality of the tool. Sections 3 and 4 elaborate on the network analysis and reconciliation techniques, supported by the tool. Section 5 outlines the demo scenarios, finally we conclude the paper in Section 6.

2. OVERVIEW OF THE SYSTEM

Our tool supports data integration in a P2P setting. In our setting one would like to exchange data between databases, organized in a P2P network, i.e. the exchange of data happens in a pairwise fashion. Even though one could argue that this is inefficient or this setting easily leads to problems, this way of organizing databases is rather common in business practice. Indeed, the lack of overview of communication or data flows often leads to unexpected problems [11].

The tool helps to visualize the flow of information between peers (databases). In fact, the user can define a so called interaction graph that represents the intended data flow in the network. As we do not assume an global schema, whenever a communication is needed, one needs to match the attributes and construct the correspondences. Once the database schemas and the interaction graph are specified, the tool can generate pairwise schema matchings. It uses existing (3rd party) matching tools to generate these matchings. The current version includes plugins for COMA++ [4] for AMC [7] and the schemas are represented by XML Schema (XSD). As we develop new plugins for further matching tools, we can relax this condition on the representation and accept other formalisms for representing the schemas (as long as the matchers are able to handle them).

Once we have generated the mappings (possible by multiple matchers), we can analyze with the tool the generated mapping, visualize the potential problems and study various properties of the network. The tool includes algorithm to automatically resolve conflicts according some predefined settings. Moreover the tool supports the user to manually give feedback and adjust attribute correspondences. Here we not only allow the user to change individual attribute correspondences, but support him with working dependencies or change several correspondences at the same time. We do not only offer the option of reconciling individual conflicts, rather we optimize the order in which the user can enter his input, such that he needs to give the possibly minimal input.

3. ANALYSIS OF MATCHING NETWORKS

Consider the network depicted in Figure 1. On the picture we depict attribute correspondences generated by a matcher. In fact, we only depict here a subset of them, to avoid having a very overcrowded figure. Indeed, the tool also supports this functionality for analyzing the matching network, the user often would like to see only a given subset of generated mappings. The situation depicted in the figure is particularly interesting for an another reason as well, namely it represents an incorrect circle of attribute correspondences. While the schema matcher correctly finds the correspondences between *BirthName* present in two different schemas, as well as the correspondences between the attributes *BirthDate*, it has also an incorrect correspondence *BirthName–BirthDate*. As often string similarity functions form the basis of the matching algorithm, such cases cannot be fully avoided. In the given setting this not only causes direct problems, but through transitivity[1], if one exchanges

[1]We assume that the attribute correspondences are equivalence relations. Note, that the correspondences can be used to define more complex schema mappings, which of course

tuples between the databases, one could have surprises as the values of *BirthDate* appear among the *BirthName* values.

We call a circle of attribute correspondences incorrect, if it connects two different attributes of the same schema, see Figure 1. Otherwise, if the circle of attribute correspondences connects only a single attribute from a given schema, we call it correct circle. [2] Our tool can display the set of incorrect circles that are present in a matching network. As the incorrect circles lead to inconsistences, it is important to study them, thus this visualization option turned out to be very useful when we analyzed the networks.

The tool offers the following functionality to analyze the matching networks.

- Display the alternative attribute correspondences, together with their confidence value, generated by a matcher (see Figure 2). Often, the matchers consider attribute correspondence candidates with confidence value above a predefined threshold. The alternative correspondences might have lower confidence value -assigned by the matcher- however these alternatives could still be valuable, in particular when we would like to avoid network-level problems.

- The user can search attribute correspondence, by name (in the source or target schema).

- If we use multiple matchers, the tool can compute the number of matchers that generated a given attribute correspondence. If one uses several matchers and for example, all of them generate the correspondence this is a much higher evidence, than say, 1 out of 10 matchers think the correspondence is valid.

- The tool can compute the entropy value of the generated alternative correspondences. This is also a useful tool, in some cases attribute names are rather close, thus the heuristic algorithms of the matcher might fail more easily, as opposed to the case, where one attribute has a much higher confidence value than the others.

- The tool can visualize correct and incorrect circles of attribute correspondences. For a given attribute correspondence we can compute the number of correct or incorrect circles that goes through this correspondence and also visualize these circles.

- The tool can detect missing links in the transitive closure of the attribute correspondence graph. Also, we defined some natural conditions the attribute correspondences should satisfy, such as a specific version of triangle inequality for the confidence values. The tool can detect, when these conditions are violated.

do not need to be equivalences.

[2]In the current version we focus on one-to-one attribute correspondences. In later versions we plan to consider also more complex mappings, then we also need to relax the notion of "incorrect circle".

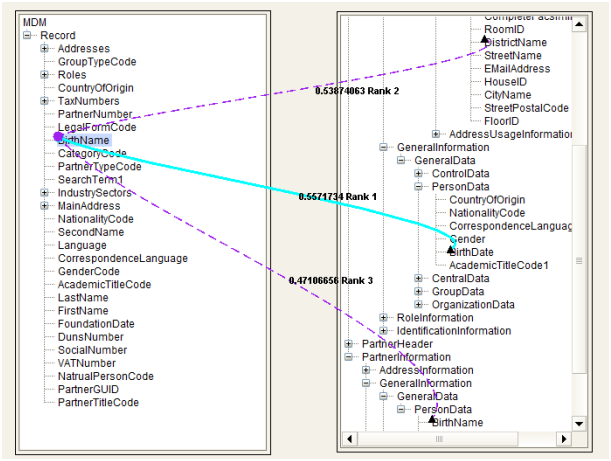

Figure 2: Matching candidates.

In the current version, the user cannot change the (otherwise explicit) network-level constraints. In the future versions we would like to allow the user to modify these constraints and enable to define application- or domain-specific constraints.

4. RECONCILIATION METHODS

If the initially generated schema matchings contain errors, or conflicts, one would like to change the correspondences to eliminate them. We have developed automatic conflict resolution methods as well as intelligent support for human input.

4.1 Automatic conflict resolution

Our automatic algorithms make use of predefined constraints. Whenever these constraints are violated by the set of generated correspondences, our algorithms try to repair these conflicts. We make use here constraint repair techniques. We start from the set of generated correspondences and change correspondences (i.e. we choose from the available alternative correspondences, that are not ranked first by the matcher). We proceed in a way that we minimize the changes and at the same time we optimize some desirable properties of the network, such as connectedness.

In the current version, the constraints are predefined, we use the no-incorrect-cycle and one-to-one attribute correspondences. In future versions we plan to allow the users to define their own specific constraints and relax the currently used ones.

4.2 Reconciliation based on user feedback

The automatic reconciliation techniques might be a good option in many cases, but clearly it also involves heuristic methods, thus the outcome might be partially incorrect. SMART also supports direct user input to correct mapping problems. In this way the user can manually influence the reconciliation process. We assume that the user is able to give correct feedback on correspondences, but he has difficulty to overview the unintended effects of local changes he suggests.

The user has the option to change a given attribute cor-

respondence. He can either choose among the alternative correspondences, or define a new one. Changing individual correspondences might be challenging to the user, as there are dependencies: if a user changes a given correspondence, he might introduce further errors. To avoid this we offer the possibility to give feedback not on single attribute correspondences, but on entire circles. SMART also has the option that the user is presented circles that he can validate (if he thinks is correct) or change. Moreover, whenever a user corrects a correspondence, and he unintendedly introduces further conflicts, this will be displayed. In this cases he can undo the changes he made or consider further modifications. We also keep a log of changes, such that the changes can be revoked, if needed.

We give particular attention in which order we present the correspondences to the user to validate (or change if incorrect). We apply different ranking heuristic, such that he can start with the most critical correspondences. For example, we compute the number of conflicts in which a give correspondence is involved. Then, we can present them in decreasing order to the user to ask feedback. After each feedback, we update this number, as the correspondences might be dependent on each other. In this way, we can also optimize the number of user input assertions needed for eliminating all problems.

5. DEMO SCENARIOS

We would like to demonstrate how SMART can support the user in the schema matching and reconciliation process. In particular,

- we demonstrate generate schema matchings for a given set of schemas, with a given interaction (workflow) graph,
- we show the possibilities for analyzing the attribute correspondence in the network,
- we demonstrate the reconciliation process (through algorithms and through human input).

6. CONCLUSION

We have developed a tool, SMART, that enables to analyze schema matching networks. Such networks arise when businesses integrate data in P2P fashion, without defining a global mediated schema. In particular, if the data integration happens in pay-as-you-go manner. SMART not only helps to analyze the network level conflicts, that are otherwise hard to detect and indeed often occur in practice, but also guides the users to resolve the conflicts.

Acknowledgments

This work was partially supported by the NisB project (FP7-ICT-256955).

7. REFERENCES

[1] D. Aumueller, H.-H. Do, S. Massmann, and E. Rahm. Schema and ontology matching with COMA++. In *VLDB'02*, pages 906–908, 2005.

[2] Z. Bellahsene, A. Bonifati, and E. Rahm. *Schema Matching and Mapping*. Springer, 2011.

[3] P. A. Bernstein, J. Madhavan, and E. Rahm. Generic Schema Matching, Ten Years Later. In *PVLDB'11*, 2011.

[4] H.-H. Do and E. Rahm. COMA: a system for flexible combination of schema matching approaches. In *VLDB'02*, pages 610–621, 2002.

[5] A. Gal. Why is schema matching tough and what can we do about it? *SIGMOD Rec.*, 35:2–5, 2006.

[6] A. Gal. *Uncertain Schema Matching*. Synthesis Lectures on Data Management. Morgan & Calypool Publishers, 2011.

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

[8] E. Rahm and P. A. Bernstein. A Survey of Approaches to Automatic Schema Matching. *The VLDB Journal*, 10(4):334–350, 2001.

[9] P. Shvaiko and J. Euzenat. A Survey of Schema-Based Matching Approaches. *Journal on Data Semantics IV*, 3730:146–171, 2005. Springer, LNCS.

[10] P. Shvaiko and J. Euzenat. *Ontology matching*. Springer, 2007.

[11] K. P. Smith, M. Morse, P. Mork, M. H. Li, A. Rosenthal, D. Allen, and L. Seligman. The Role of Schema Matching in Large Enterprises. In *Fourth Biennial Conference on Innovative Data Systems Research (CIDR'09)*, 2009.