On Evaluating Schema Matching and Mapping

Zohra Bellahsene, Angela Bonifati, Fabien Duchateau, and Yannis Velegrakis

Abstract The increasing demand of matching and mapping tasks in modern integration scenarios has led to a plethora of tools for facilitating these tasks. While the plethora made these tools available to a broader audience, it led into some form of confusion regarding the exact nature, goals, core functionalities expected features and basic capabilities of these tools. Above all, it made performance measurements of these systems and their distinction, a difficult task. The need for design and development of comparison standards that will allow the evaluation of these tools is becoming apparent. These standards are particularly important to mapping and matching system users since they allow them to evaluate the relative merits of the systems and take the right business decisions. They are also important to mapping system developers, since they offer a way of comparing the system against competitors, and motivating improvements and further development. Finally, they are important to researchers since they serve as illustrations of the existing system limitations, triggering further research in the area. In this work we provide a generic overview of the existing efforts on benchmarking schema matching and mapping tasks. We offer a comprehensive description of the problem, list the basic comparison criteria and techniques and provide a description of the main functionalities and characteristics of existing systems.

Zohra Bellahsene
University of Montpellier II, France e-mail: bella@lirmm.fr

Angela Bonifati
CNR, Italy e-mail: bonifati@icar.cnr.it

Fabien Duchateau
CWI, The Netherlands e-mail: fabien@cwi.nl

Yannis Velegrakis
University of Trento, Italy e-mail: velgias@disi.unitn.eu

1 Introduction

The web has become the world's largest database. Daily, thousands of organizations and individuals are making their repositories available online. To exploit the full potential of these sources, modern information systems and web applications must be able to retrieve, integrate and exchange data. Unfortunately, the repositories and applications are developed by different people, at different times, with varying requirements in mind. Thus, the underlying data is inherently highly heterogenous. To cope with the heterogeneity and achieve interoperability, a fundamental requirement is the ability to match and map data across different formats. These two tasks are found in the literature under the names *matching* (Rahm and Bernstein, 2001) and *mapping* (Miller et al, 2000), respectively. A match is an association between individual structures in different data sources. Matches are the required components for every mapping task. The mappings are the products of the latter. A mapping, in particular, is an expression that describes how the data of some specific format is related to data of another. The relationship forms the basis for translating the data in the first format into data in the second.

Mappings can be found in almost every aspect of data management. In information integration systems (Lenzerini, 2002) mappings are used to specify the relationships between every local and the global schema. In schema integration, mappings specify how an integrated schema is constructed from the individual input schemas (Batini et al, 1986). In data exchange (Fagin et al, 2005) and P2P settings (Halevy et al, 2003; Bernstein et al, 2002), mappings are used to describe how data in one source is to be translated into data conforming to the schema of another. A similar use is found in schema evolution (Lerner, 2000) where mappings describe the relationship between the old and the new version of an evolved schema.

Mapping generation had been for a long time a manual task, performed mainly by data professionals with good understanding of the semantics of the different schemas and expertize in the transformation language. But as schemas have started to become larger and more complicated, the process has become laborious, time consuming and error-prone. On top of that, the modern mashup technologies (Wun, 2009) have given to regular Internet users the ability to build their own integration applications, systems and services. In this process, these users have to strive with the complexities of the schemas, the peculiarities of the transformation language, and the many other technical details of the data transformation specification. The need for designing and developing tools to support the mapping designer in the mapping specification task has been apparent. Thos tools are known as mapping tools, and they offer support in different styles and flavors. Certain tools raise the abstraction level by providing sophisticated graphical interfaces (Altova, 2008) or high level mapping languages (Bernstein and Melnik, 2007). Others offer advanced algorithms performing part of the reasoning the mapping designer has to make (Popa et al, 2002; Do and Rahm, 2002; Mecca et al, 2009; Madhavan et al, 2001), while some offer designer guidance (Alexe et al, 2008a). Today there exists a plethora of such systems, including the Altova Mapforce (Altova, 2008), IBM Rational Data Architect (IBM, 2006), Microsoft BizTalk Mapper which is embed-

ded in Microsoft Visual Studio (Microsoft, 2005), Stylus Studio (Stylus Studio, 2005), BEA AquaLogic (Carey, 2006), and the research prototypes Rondo (Do and Rahm, 2002), COMA++ (Aumueller et al, 2005), Harmony (Mork et al, 2008), S-Match (Giunchiglia et al, 2005), Cupid (Madhavan et al, 2001), Clio (Popa et al, 2002), Tupelo (Fletcher and Wyss, 2006), Spicy (Mecca et al, 2009) and HePToX (Bonifati et al, 2006).

Despite the availability of the many mapping tools, there has been no generally accepted benchmark developed for comparing and evaluating them. As it is the case with other benchmarks, such a development is of major importance for assessing the relative merits of the tools. This can help customers in making the right investment decisions and selecting among the many alternatives, the tools that better fit their business needs. A benchmark can also help the mapping tool developers since it offers them a common metric to compare their own achievements against those of the competitors. Such comparisons can boost competition and drive the development towards systems of higher quality. A benchmark is also offering the developers a generally accepted language for talking to customers and describing the advantages of their tools through well known features that determine performance, effectiveness and usability. Furthermore, the benchmark can highlight limitations of the mapping tools or unsupported features that may not have been realized by the developers. Finally, a benchmark is also needed in research community (Bertinoro, 2007). Apart from a common platform for comparison, a benchmark allows researchers to evaluate their achievements not only in terms of performance, but also in terms of applicability in real world situations.

In this work, we summarize and present in a systematic way existing efforts towards the characterization and evaluation of mapping tools, and the establishment of a benchmark. After a quick introduction of the architecture and main functionality of matching and mapping tools in Section 2, and we describe the challenges of building a matching/mapping system benchmark in Section 3. Section 4 presents existing efforts in collecting real world test cases with the intention of using them in evaluating the matching and mapping systems. Section 5 addresses the issue of creating synthetic test cases that are targeting the evaluation of specific features of the mapping systems. Finally, Section 6 and Section 7 present different metrics that have been proposed in the literature for measuring the efficiency and effectiveness of matching/mapping systems, respectively.

2 The Matching and Mapping Problem

Matching is the process that takes as input two schemas, referred to as the *source* and the *target*, and produces a number of matches, a.k.a., *correspondences*, between the elements of these two schemas (Rahm and Bernstein, 2001). The term schema is used with the broader sense and includes database schemas (Madhavan et al, 2001), ontologies (Giunchiglia et al, 2009), or generic models (Atzeni and Torlone, 1995). A match is defined as a triple $\langle S_s, E_t, e \rangle$, where S_s is a set of elements from the source,

E_t is an element of the target schema, and e is a matching expression that specifies a relationship between the element E_t and the elements in S_s. Note that the expression e does not specify how the elements in S_s relate to each other. Most of the time a match is as simple as an equality or a set-inclusion relationship between an element of the source and an element of the target. There are, however, cases in which the relationship can be more complex, e.g., a concatenation function, some arithmetic operation, a relationship over scalars like = or $\leq$, a conceptual model relationship such as the part-of, or some set-oriented relationships, such as overlaps or contains. Schema matching tools employ a number of different techniques in order to discover this kind of relationships. They can range from structural (Madhavan et al, 2001) and name similarities, to semantic closeness (Giunchiglia et al, 2004) and data value analysis (Doan et al, 2001, 2004). A schema matching tool accepts as input the two schemas and generates the set of matches. Since any schema matching process is based on semantics, its final output needs to be verified by a human expert. The matching process can be roughly divided into three phases: the pre-match, the match, and the post-match phase. During the first phase, the matcher performs some computations and processes the data. Typically this involves the training of the classifiers in the case of machine learning-based matchers, the configuration of the various parameters like thresholds and weight values used by the matching algorithm, and the specification of auxiliary information, such as domain synonyms and constraints (Giunchiglia et al, 2009). During the second phase, the actual discovery of the matches takes place. At the end, the matcher outputs the matches between elements of these data sources. During the post-match phase, the users may check and modify the displayed matches if needed.

Given a source and a target schema, a *mapping* is a relationship, i.e., a constraint, that must hold between their respective instances. For the mappings to be generated, a fundamental requirement are the matches between the elements of the schemas. These matches can be either generated automatically through a matching process, or can be manually provided by an expert user. In contrast to matches, that specify how instance values of individual source and target schema elements relate to each other, a mapping additionally specifies how the values within the same instance relate to each other. For example, a match may specify that the dollar price of a product in the target corresponds to the multiplication of the price of the product in the source (expressed in some foreign currency) multiplied by the exchange rate. The mapping is the one that specifies that the exchange rate with which the product price in the source is multiplied is the exchange rate of the currency in which the price of the product is expressed. The mapping does so by specifying the right join path between the price and the exchange rate attributes.

Mappings can be used in many different ways. In the case in which the target schema is a virtual, i.e., not materialized, database as happens in virtual information integration systems, in P2P applications, or in data repositories that publish an interface schema, the mappings can be used for query answering by driving the translation of queries on the target schema to queries on the source (Lenzerini, 2002). Another major application of mappings is data exchange (Fagin et al, 2003) in which given a source instance, the mappings are used to drive the materialization of a tar-

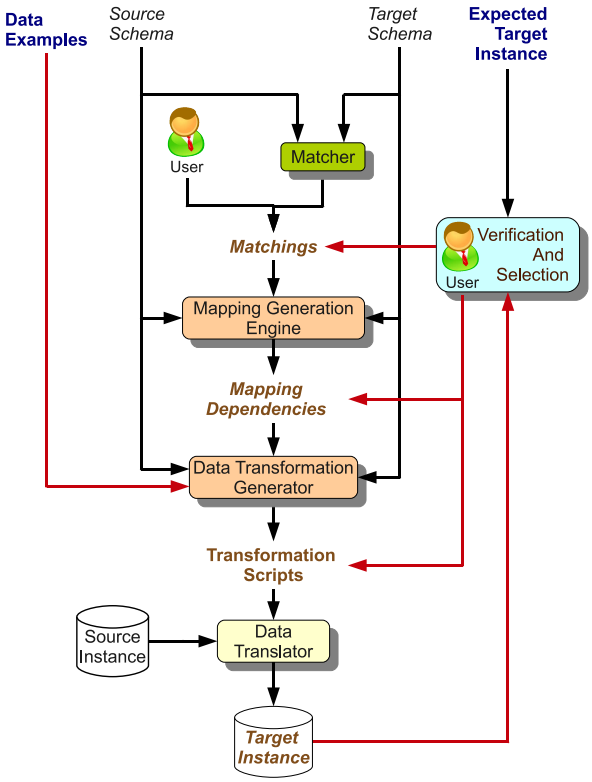


Fig. 1 Overview of the matching, mapping and data exchange tasks

get instance. Since mappings are inter-schema constraints, they may not be enough to fully specify the target instance. In other words, given a source instance and a set of mappings between a source and a target schema, there may be multiple target instances that satisfy the mappings. Finding the best target instance is known in the literature as the data exchange problem (Fagin et al, 2005). Once the right target instance is decided, then the mappings can be converted into a transformation script that translates an instance of the source schema into some target schema representation. This transformation script is typically expressed in some executable language such as XQuery, XSLT or SQL.

A mapping tool is a tool that assists the mapping designer in generating the mappings using less effort, in less time and with fewer mistakes. Mapping tools can be classified in two large categories based on what they consider as a mapping. The first category is the one that makes a clear distinction between mapping generation, i.e., the generation of the inter-schema constraints, and the data exchange, i.e., the generation of the transformation script. Tools in this category include the research prototypes Clio (Popa et al, 2002) and Spicy++ (Mecca et al, 2009). Their

main goal is the generation of the mappings in the form of constraints which can then be used either for information integration, or for data exchange. To facilitate the latter case, the tools may be equipped with a data exchange module that converts the generated mappings into some transformation script that can be executed on the source instance to produce the target. The generated mappings are typically expressed through some declarative specification in a logic formalism. The most widely used such formalism is the *tuple generating dependency*, or *tgd* for short (Abiteboul et al, 1995). The second large class of mapping tools are those that make no distinction between mapping generation and data exchange. For these tools the notion of mapping generation is actually the creation in some native language of the final transformation script that provides a full specification of the target instance in terms of a source instance. Characteristic representatives in this category are the commercial tools Altova MapForce (Altova, 2008) and Stylus Studio (Stylus Studio, 2005).

In what follows, we will use the term mapping tool to describe a tool in either category, and the term mapping to describe the output of such a tool, no matter whether it is an interschema constraint or a transformation script. In the case we want to emphasize that a mapping is not a transformation script we will use the term *mapping dependency*.

The design of existing mapping tools is based on the idea of providing the mapping designer with a graphical representation of the two schemas and a set of graphical constructs representing high level transformation abstractions. Using these graphical constructs, the mapping designer provides a specification of the desired mappings. The level of abstraction of the graphical objects may vary from direct correspondences (Popa et al, 2002; Mecca et al, 2009), i.e., matches, to graphical representations of primitive operations of the transformation script language (Altova, 2008; Stylus Studio, 2005). The high level graphical constructs provided by the mapping designer are easy to use but they are inherently ambiguous. The mapping tool will have to interpret them and make an educated guess of a transformation that the mapping designer had in mind to create (Velegrakis, 2005). An a-posteriori verification is then necessary to ensure that the generated mappings are indeed those intended by the designer. Clearly, the simpler the graphical constructs are, the easier the task is for the designer, but at the same time, the more the intelligence required by the tool to interpret these constructs and infer the desired mappings.

A number of mapping tools are equipped with a matching module which can be used by the mapping designer to suggest possible matches. One such tool is Clio (Popa et al, 2002) whose matching component is based on attribute feature analysis (Naumann et al, 2002). It generates matches in the form of *attribute correspondences*, i.e., interschema lines connecting atomic type schema elements, annotated with some value transformation functions. Other tools (Bernstein and Melnik, 2007) have the matching task not as an add-on component, but as a fully integrated and indistinguishable part of the tool. Spicy (Bonifati et al, 2006) is between these two alternatives. It has a matching module similar to the one in Clio, but is used as an integral part of the tool, allowing it to accept as input only the pair of source and target schema, if needed.

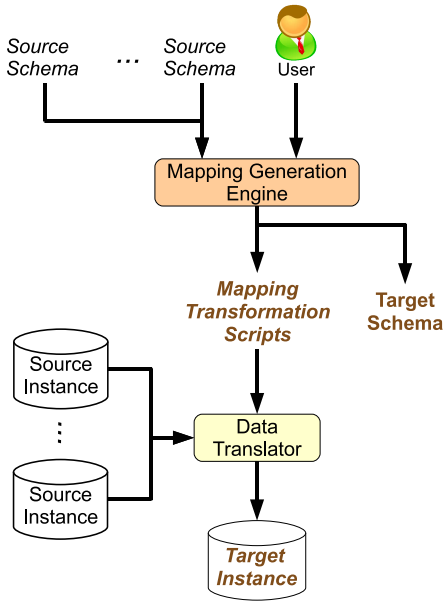


Fig. 2 Overview of the schema integration task

Since matching and mapping tools try to guess the intentions of the designer based on the provided input, it is natural to assume that their output is not always the one anticipated by the designer. As already mentioned, an a-posteriori verification is necessary. Nevertheless, there is a significant number of tools that allow the active participation of the designer in the matching/mapping generation phase in order to guide the whole process and arrive faster to the desired result. For example, once some matchings/mappings have been generated, the designer can verify their correctness. If she feels unsatisfied by the result, she can go back and modify some intermediate steps, for instance, she she can tune the matcher, select a fraction of the set of the generated matches, enhance the matches by introducing new matches not automatically generated by the matcher, tune the mapping generation process by accepting only a fraction of the generated mappings, or even edit directly the mappings. User participation is highly active in Tupelo (Fletcher and Wyss, 2006) where mapping generation is studied as a search problem driven by input data examples. Domain knowledge, that is usually an input to the matcher, is also used as input to the mapping discovery module. User feedback can be used to improve the effectiveness of the discovered semantic functions, i.e., the matches, and of the structural relationships, i.e., the mapping dependencies, that in turn can be entrusted to a data mapping module for generating the final transformation query.

Many mapping tools are used as schema integration tools. Schema integration is the process of merging multiple source schemas into one integrated schema, a.k.a., the global or mediated schema. The integrated schema serves as a uniform interface

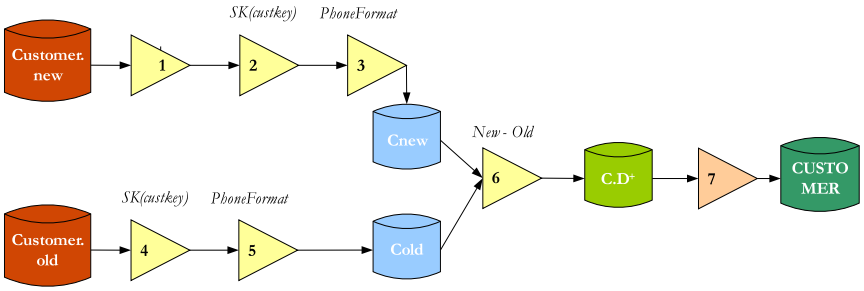


Fig. 3 An ETL data flowchart

for querying the data sources. Nowadays, construction of integrated schemas has become a laborious task mainly due to the number, size and complexity of the schemas. On the other hand, decision makers need to understand, combine and exploit in a very short time all the information that is available to them before acting (Smith et al, 2009). This reality requires the rapid construction of large prototypes and the flexible evolution of existing integrated schemas from users with limited technical expertize. Matching and mapping tools facilitate that goal. A mapping designer may be presented with a number of source schemas and an empty target. Through a graphical interface, source schema elements can be selected and "dropped" into the target. When the elements are dropped in the target, the mappings specifying how the target elements are related to those in the sources are automatically or semi-automatically generated. This functionality is graphically depicted in Figure 2. Note that schema integration involves additional tasks, however, here we concentrate only on the part related to matching and mapping.

A special case of mapping tools are the ETL systems. An ETL system is a tool designed to perform large scale extract-transform-load operations. The transformation performed by an ETL system is typically described by a graph-flowchart in which each node represents a specific primitive transformation and the edges between the nodes represent flow of data produced as a result of a primitive operator and fed as input in another. Figure 3 illustrates such a data flowchart. The triangles represent transformation operators and the cylinders data stored in some media. Although ETL systems are typically not considered mapping tools, they share many similarities with them. First, their main goal is also to facilitate the designer in describing a data transformation from one format to another. Second, they often provide a graphical interface, and they produce as a final output transformation scripts. Finally, there are mapping tools currently in the market (Stylus Studio, 2005) that operate very close to the ETL model. Their graphical interface provides a set of

primitive transformations that the designer can combine together to provide a full specification of the transformation that needs to be applied on the data. Their output looks like an ETL flowchart. ETL systems require no large intelligent capabilities since the input provided by the designer is so detailed that only a limited form of reasoning is necessary. Similar to ETL systems are mashup editors (Heinzl et al, 2009) that try to facilitate the mashup designer. The operational goals of mashup editors are similar to those of ETL systems, thus, we will not consider them as a separate category.

We will use the term matching or mapping *scenario* to refer to a particular instance of the matching or mapping problem, respectivelly. A scenario is represented by the input provided to the matching or mapping tool. More specifically, a matching scenario is a pair of source and target schema. A mapping scenario the pair of source and target schema alongside a specification of the intented mappings. A *solution* to a scenario is a set of matches, respectivelly mappings, that satisfy the specifications set by the scenario.

3 Challenges in Matching and Mapping System Evaluation

A fundamental requirement for providing universal evaluation of matching and mapping tools is the existence of benchmarks. A benchmark for a computer application or tool is based on the idea of *evaluation scenarios*, i.e., a standardized set of problems or tests serving as a basis for comparison.[1] An evaluation scenario for a matching/mapping tool is a scenario alongside the expected output of the tool, i.e., the expected solution. Unfortunately, and unlike benchmarks for relational database management tools, such as, TPC-H (Transaction Processing Performance Council, 2001), or for XML query engines, such as, XMach (Bohme and Rahm, 2001), X007 (Bressan et al, 2001), MBench (Runapongsa et al, 2002), XMark (Schmidt et al, 2002) and XBench (Yao et al, 2004), the design of a benchmark for matching/mapping tools is fundamentally different and significantly more challenging (Okawara et al, 2006), mainly due to the different nature, goals and operational principles of the tool.

One of the differences is the fact that given a source and a target schema, there is not always one "correct" set of matches or mappings. In query engines (Transaction Processing Performance Council, 2001; Bohme and Rahm, 2001), the correct answer to a given query is uniquely specified by the semantics of the query language. In matching/mapping tools, on the other hand, the expected answer depends not only on the semantics of the schemas, that by nature may be ambiguous, but also on the transformation that the mapping designer was intending to make. The situation reminisces the case of web search engines, where there are many documents returned as an answer to a given keyword query, others more and others less related to the query, but which document is actually the correct answer can only be

[1] Source: Merriam Webster dictionary

decided by the user that posed the keyword query. For that reason, many evaluations of matching or mapping tools are performed by human experts.

Another difficulty faced during the design of evaluation techniques for mapping tools is the lack of a clear specification of the input language, i.e., a standardized formalism with well defined semantics. In contrast to benchmarks for relational (Transaction Processing Performance Council, 2001) and XML systems (Bohme and Rahm, 2001) that could leverage from the respective SQL and XQuery standard query languages, it is still not clear how to describe a scenario. Formally describing the schemas is not an issue, but describing the intended transformation, i.e., the input that the designer needs to provide, is. The best way to unambiguously specify the intended transformation is through a transformation language script, or a mapping in some formalism, but there are two main issues with this option. First, there are no guarantees that the mapping tool will be able to accept the specific formalism as input, or at least that there will be an unambigius translation of the input from the formalism into the input language supported by the mapping tool. The second issue is that such an approach beats the purpose of a mapping tool which is intended to shield the mapping designer from the complexity and the peculiarities of the transformation language. It is actually for that reason that mapping tool developers have opted for simpler, higher level specification languages, such as visual objects, direct lines between schema elements, or the output of the matching process in general. Unfortunately, such specification is by nature ambiguous. Consider one of the already identified (Alexe et al, 2008c) ambiguous situations, described in Figure 4. It is a simple scenario in which the mapping designer needs to copy the *company* data from the source into *organizations* data in the target. To specify this, the designer draws the two interschema lines illustrated in Figure 4. When these are fed to a popular commercial mapping tool, the tool generates a transformation script which generates the target instance illustrated in Figure 5(a) when executed on the instance of Figure 4. A different tool, for the same input, produces a transformation script that generates the instance illustrated in Figure 5(b). A third one produces a script that generates the instance of Figure 5(c) which is most likely the one the mapping designer had in mind to create. These differences are not an error from the side of the tools, rather a consequence of the fact that in the absence of a global agreement on the semantics of the matches, or the input language in general, different tools may interpret them differently and may require different inputs for generating the same mappings. In the above example, the tool that generated the instance in Figure 5(a) could have also produced the instance of Figure 5(c), if the designer had provided one more match from the element *Company* to the element *Organization*. This match (which is between non-leaf elements) is not allowed at all in the tool that created the instance of Figure 5(c). The issue is also highly related to the level of intelligence and reasoning capabilities that the tools are offering. Some tools may require a minimum input from the user and through advanced reasoning may be able to generate the intended mappings (Bonifati et al, 2008b; Fagin et al, 2009a). Others may require from the designer to be more explicit when describing the transformation she has in mind to create (Altova, 2008; Stylus Studio, 2005).

```
<Source>
   <Company>
      <Name>IBM</Title>
      <Location>NY</Address>
   </Company>
   <Company>
      <Name>MS</Title>
      <Location>WA</Address>
   </Company>
</Source>
```

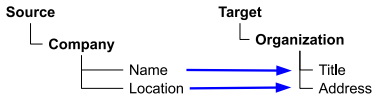


Fig. 4 A simple mapping scenario and the source schema instance

Even by considering only matches, there is a large variaty of specification options as a recent classification of mapping tools illustrates (Legler and Naumann, 2007).

The input problem goes even further. Some mapping tools allow the designer to edit the generated mappings or transformation scripts in order to correct or enhance them. In that way, the generated output is restricted only by the expressive power of the mapping language or of the transformation script. Under such circumstances, a scenario should be extended to include, apart from the two schemas and the intended mapping specification, the modifications/corrections that the designer does on the generated output. However, allowing the designer to edit the output makes unfair any comparison to mapping tools that operate under the principle that the designer can only use the high level graphical input language (Altova, 2008).

Another issue of inconsistency across different matching and mapping tools is the lack of a standardized output. Some matching tools generate only 1-1 identity function matches, i.e., simple interschema correspondences, while others generate more complex relationships. Furthermore, some mapping tools generate mappings as inter-schema dependencies only, while others produce also the transformation scripts. The problem is becoming more crucial due to the fact that there is no unique way of generating a target instance. Two different mapping tools may produce completely different transformation scripts, and yet, generate the same target instance.

Deciding the metrics with which success is measured is another challenging task. Since a general goal of a mapping tool is to reduce the required programming effort, measuring the effort spent for a matching or a mapping task using a tool can

```
<Target>
  <Organization>
    <Title>IBM</Title>
    <Title>MS</Title>
    <Address>NY</Address>
    <Address>WA</Address>
  </Organization>
</Target>
```

(a)

```
<Target>
  <Organization>
    <Title>IBM</Title>
    <Address>NY</Address>
  </Organization>
</Target>
```

(b)

```
<Target>
  <Organization>
    <Title>IBM</Title>
    <Address>NY</Address>
  </Organization>
  <Organization>
    <Title>IBM</Title>
    <Address>NY</Address>
  </Organization>
</Target>
```

(c)

Fig. 5 Three different target instances generated by different tools

serve as an indication of the success of the tool. Unfortunately, such metrics are not broadly accepted since they highly depend on the user interface. An advanced user interface will lead to good evaluation results which means that the evaluation of a mapping tool is actually a graphical interface evaluation. Furthermore, the fact that there is no global agreement on the expressive power of the interface, poses limits on the evaluation scenarios that can be run. A mapping tool with a simple interface, may require less designer effort, but may also be limited on the kind of mappings or transformations it can generate. This has led a number of researchers and practitioners into considering as an alternative metric the expressive power of the mappings that the tool can generate, while others talked about the quality of the mappings themselves (Bonifati et al, 2008b) or the quality of the integrated schema, for the case in which the mapping tool is used for schema integration. The quality of the integrated schema is important for improving query execution time, successful data exchange and accurate concept sharing. Unfortunately, there is no broadly accepted agreement on how mapping quality is measured, thus, in order to provide meaningful comparisons an evaluation method should consider a number of different metrics for that purpose.

Developing evaluation techniques for mapping tools is also limited by the non deterministic output of the scenarios. In contrast to query engines, different mapping tools may generate different results for the same input, without any of the results being necessarily wrong. In particular, for a given a high level mapping specification, there may be different interpretation alternatives, and each tool may choose one over another. The ability to effectively communicate to the mapping designer the semantics of the generated output is of major importance in order to allow the designer to effectively guide the tool towards the generation of the desired mappings. One way to do so, is to present the designer with the target instance that the generated mappings can produce. This is not always convenient, practical, or even feasible, especially for large complicated instances. Presenting the mapping to the designer seems preferable (Velegrakis, 2005), yet it is not always convenient since the designer may not be familiar with the language in which the mappings are expressed. An attractive alternative (Alexe et al, 2008a) is to provide carefully selected representative samples of the target instance or synthetic examples that effectively illustrate the transformation modeled by the generated mappings. This option is becoming particularly appealing nowadays that more and more systems are moving away from exact query semantics towards supporting keyword (Bergamaschi et al, 2010) and approximate queries, or queries that embrace uncertainty in the very heart of the system (Ioannou et al, 2010).

4 Real World Evaluation Scenarios

A close look at popular benchmarks can reveal a common design pattern. The benchmark provides a number of predefined test cases that the tool under evaluation is called to successfully execute. The tool is then evaluated based on the num-

ber of these cases that were indeed implemented successfully. The TPC-H benchmark (Transaction Processing Performance Council, 2001), for instance, consists of a set of predefined queries on a given database, with each of these queries testing a specific feature of the query language that the query engine is expected to support. For each such query, the benchmark provides the expected correct answer against which the results of the query execution on the under evaluation engine can be compared. Accordingly, a mapping tool benchmark should provide a set of evaluation scenarios, i.e., scenarios alongside the expected result.

There has been a number of efforts towards building collections of evaluation scenarios. There is an unquestionable value to these collections. The ability of a mapping method or tool to successfully execute the evaluation scenarios is a clear indication of its practical value. By successful execution we mean that the tool is able to generate the expected output as described by the evaluation scenario. Although these collections are built based on criteria such as popularity, community acceptance or by contributions of interested parties and by the user base, they often lack systematic categorization of the cases they test. For instance, they may have multiple evaluation scenarios testing the same feature of the tool, or they may provide no generalized test patterns. For that reason, this kind of collections are typically termed as *testbeds* or *standardized tests*.

A complete and generic benchmark should go beyond a simple set of test cases. It should offer a systematic organization of tests that is *consistent*, *complete* and *minimal*. Consistent, means that the existence of every test case should be justified by some specific feature upon which the tool or technique is evaluated through the test case. Complete means that for every important feature of the mapping tool under evaluation there is a test case. Minimal means that there are no redundant test cases, i.e., more than one test case for the same feature.

To evaluate a matching tool on a given evaluation scenario, the scenario is provided to the tool that produces a solution. That generated solution, which in the case of a matching tool is a set of matches, is then compared against the expected set of matches that the evaluation scenario contains. If the two sets are the same, then the tool is said to be successful for this scenario. The evaluation scenarios are typically designed to check a specific matching situation. Success or failure to a specific scenario translates into the ability or not of the matching tool under evaluation to handle the specific matching situation. This kind of evaluation is the one for which testbeds are designed for. The Ontology Alignment Evaluation Initiative (Euzenat et al, 2006), OAEI for short, is a coordinated international initiative that every year organizes a matching competition for ontologies. Ontologies can be seen as semantic schemas, thus, ontology matching is considered part of the general matching problem. The initiative provides the contesters with a set of matching test scenarios with which the contesters test their tools. Throughout the year, individuals may also submit to the initiative various scenarios they meet in practice. As a result, the collected scenarios of the initiative constitute a good representation of the reality. In some recent evaluation of a number of matching tools (Kopcke and Rahm, 2010), the number of real-world test problems that the matching tool could handle, was featuring as one of the main comparison criteria. The OAEI scenarios may be further

enhanced with datasets. In a recent effort (Giunchiglia et al, 2009) an extension was proposed that contains 4500 matches between three different Web directories and has three important features, namely, it is error-free, has a low complexity and has a high discriminative capability, a notion that will be explained later. Unfortunately, despite the fact that there is a strong need for comparing matchers using identical evaluation scenarios[2], there has been no broadly accepted agreement until today on what these evaluation scenarios should be.

The XBenchMatch (Duchateau et al, 2007) is a benchmark for matching tools. It defines a set of criteria for testing and evaluating matching tools. It may focus mostly on the assessment of the matching tools in terms of matching quality and time performance but provides a testbed involving ten datasets that can be used to quickly benchmark new matching algorithms (Duchateau, 2009). These matching scenarios have been classified according to the tasks they reflect, either at the data level, e.g. the structure or the degree of heterogeneity, or at the matching process level, e.g., the scale. Although collaborative work can help providing new datasets with their correct set of matches, the creation of such a large and complete set still remains a challenge.

It is important to add here that one of the challenges during the creation of test scenarios is deciding what the correct matches will be. As mentioned in the previous section, for a given matching scenario there may be multiple correct answers. Opting for one of them may not be fair for the others. For this reason, in cases like OAEI, the test scenarios designers perform a careful selection so that the scenarios have no multiple alternatives, or in the case that they have, the one that is considered as the correct answer to the chosen scenario is the one that is most obvious or the one that the exclusive majority of matching users would have considered as correct.

One of the first benchmarks for mapping tools is the STBenchmark (Alexe et al, 2008c). It contains a list of basic test scenarios, each consisting of a source schema, a target schema and a transformation query expressed in XQuery. The choice of describing the mapping specification in XQuery was made in order to avoid any misinterpretation of the mapping that needs to be achieved. This of course does not mean that the mappings that the mapping tool will generate will have to be necessarily in XQuery, but they have to describe an equivalent mapping. Furthermore, the selection of XQuery as a mapping specification language causes no major issues to the mapping tool evaluators, since such users are in general more experienced than regular mapping designers. They can easily understand the full details of the expected transformation, and by using the mapping tool interface they can try to materialize it. For mapping tools that accept matches as input, conversion from XQuery to matches is a straight forward task.

Each STBenchmark mapping scenario is carefully designed to test the ability of the mapping tool to create transformations of a specific kind. The evaluator is expected to understand first the desired transformation by studying the transformation script, and then try to implement it through the interface provided by the mapping tool that wants to evaluate. Some of the scenarios provided by STBenchmark are

[2] Netrics HD blog, April 2010: http://www.netrics.com/blog/a-data-matching-benchmark

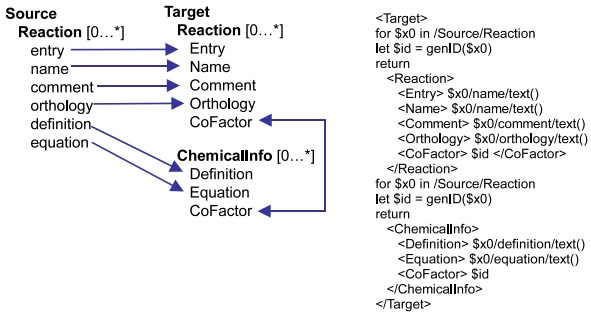


Fig. 6 A mapping scenario for vertical partition

related to copying structures, constant value generation, horizontal and vertical partitioning, key generation, nesting and un-nesting of structures, different join path selection, aggregation, value combination, and to many others. Figure 6 illustrates an example of one of these scenarios. The list of scenarios has been collected by a study of the related information integration literature and many practical applications. Definitely, one cannot build an exhaustive set of testing scenarios. There will always be cases that remain untested. This is the case even with query engine benchmarks. However, what is important for a benchmark is to cover the majority of the cases that are met in practice (Alexe et al, 2008b).

An important issue that must be brought here is that general purpose evaluation tools should contain examples from the domains the tool is intended to be used (Kopcke and Rahm, 2010). It is a known fact that certain matching or mapping tools perform well on data with certain characteristics. Thus, such tools should be evaluated using scenarios from that area. General purpose benchmarks should provide scenarios from different domains. Each STBenchmark test scenario, for instance, is accompanied by a source instance with data extracted from the DBLP bibliographic server[3], the BioWarehouse[4] collection, and other similar real sources.

The approach of using predefined evaluation scenarios is also followed by Thalia (Hammer et al, 2005), a benchmark for evaluating integration tools. Recall that in the schema integration task, the input to the tool is a set of source schemas for which the mapping designer is called to generate the integrated schema and the mappings that populate it from source data. Thalia provides a rich set of test data for integration problems exhibiting a wide variety of syntactic and semantic heterogeneities. It also provides twelve test queries each requiring the resolution of a particular type of heterogeneity.

[3] www.informatik.uni-trier.de/ ley/db/

[4] biowarehouse.ai.sri.com

5 Synthetic Evaluation Scenarios

An important issue for a benchmark is to have not only fixed evaluation scenarios but scenarios representing generic patterns. In a world where the data is becoming increasingly complicated, it is crucial to stress test the tools for data and schemas of different sizes. This means that matching and mapping benchmarks should support dynamic generation of evaluation scenarios of different sizes with which one can test how the tool under evaluation scale up.

Unfortunately, such a pluralism may be hard to find in real world applications, mainly due to privacy reasons, or because they typically originate from a single domain that restricts their pluralism and makes them unsuitable for general purpose evaluations. Thus, a benchmark should be able to create synthetic test cases in a systematic way that stress test the mapping tools and allow the evaluation of their performance under different situations.

In the case of a matching tool, generation of a synthetic test scenario involves the creation of a source and a target schema, alongside the expected matches. The construction of the two schemas, should be done in parallel so that for every part of the source schema, the part of the target schema with which it matches is known. For the case of a mapping tool, the situation is similar, but instead of the expected matches, the synthetic test scenario should have the expected transformation. The construction of the latter should also be orchestrated with the construction of the two schemas. For mapping tools in schema integration a test scenario consists of a set of source schemas, the expected integrated schema and the specification on how the expected integrated schema is related to the individual source schemas.

Generation of synthetic scenarios has in general followed two main approaches: the *top-down* and the *bottom-up* approach. The former starts with some large scenario and by removing parts of it generates other smaller scenarios. The latter constructs each scenario from scratch. Both approaches can be applied in the case of synthetic scenario generation for matching and mapping tools.

The top-down approach starts with an existing large source and target schema, and systematically removes components to generate smaller scenarios satisfying specific properties. The properties depend on the features of the matching or mapping task that needs to be evaluated. An example of an ontology matching evaluation dataset that has been built using the top-down approach is TaxME2 (Giunchiglia et al, 2009). In TaxME2, a set of original ontologies are initially constructed out of the Google, Yahoo and Looksmart web directories. In the sequel, matches across these ontologies are also defined and characterized. For every pair of ontologies, portions are cut out alongside matches using elements from these portions. The remaining parts of the two ontologies are used as the source and the target, and the remaining matches form the expected correct matches. The process is repeated multiple times, each time using a different portion that leads to the creation of a new matching evaluation scenario. The selection of the portions was done in a way that preserved five main properties: (i) the complexity of the matching operators; (ii) the incrementality, i.e., the ability to reveal weaknesses of the matching tool under evaluation; (iii) the ability to distinguish among the different matching solutions; (iv)

the quality preservation, meaning that any matching quality measure calculated on the subset of the schemas did not differ substantially from the measure calculated on the whole dataset; and (v) the correctness, meaning that any matches considered were correct.

A top-down approach has also been proposed for data exchange systems (Okawara et al, 2006) and is the model upon which the THALIA (Hammer et al, 2005) integration benchmark is based. In particular, Thalia provides a large dataset and the filters that can select portions of this dataset in terms of values and in terms of schemas.

eTuner (Lee et al, 2007) is a tool for automatically tuning matchers that utilizes the instance data in conjunction with the schema information, and can also be used to create synthetic scenarios in the top-down fashion. It starts with an initial schema, and splits it into two, each keeping the same structure but half of the instance data. The correct matches between the schemas generated by the split are known, and the idea is to apply transformations to one of the two schemas to create a new schema. The transformations are based on rules at three levels: (i) modifications on the structure of the schema, (ii) changes of the schema element names, and (iii) perturbations of the data. The matchings between schema elements are traced through the whole process so that they are known at the end and are used for evaluating the matchers. A limitation of eTuner is that the user needs to create or find a reference ontology. Furthermore, the set of modifications that can be performed on the data is limited, making the perturbated data look less similar to natural real-world data.

In the bottom-up approach of synthetic scenario generation, some small scenario is used as a seed for the construction of more complex scenarios. STBenchmark (Alexe et al, 2008b) is based on this idea in order to provide synthetic mapping test scenarios, i.e., a synthetic source schema, a target schema, an expected mapping between the source and the target schema, and an instance of the source schema. The seeds it uses are its basic scenarios that were mentioned in the previous section. Given a basic scenario, STBenchmark constructs an expanded version of it. The expanded version is an image of the original scenario but on a larger scale. The scale is determined by dimensions specified through configuration parameters representing characteristics of the schemas and the mappings. For instance, in a *copy* basic scenario the configuration parameters are the average nesting depth of the schemas and the average number of attributes of each element. In the *vertical partition* scenario (ref. Figure 6) on the other hand, the configuration parameters include additionally the length of join paths, the type of the joins and the number of attributes involved in each such join. Expanded scenarios can then be concatenated to produce even larger mapping scenarios. Figure 7(a) illustrates an expanded *un-nest* basic mapping scenario, and Figure 7(b) how a large synthetic scenario is created by concatenating smaller scenarios. STBenchmark[5] has also the ability to create synthetic mapping scenarios that involve complex transformations coming from a combination of transformations that the basic mapping scenarios describe. For the generation of the instance of the source schema, STBenchmark generates a

[5] www.stbenchmark.org

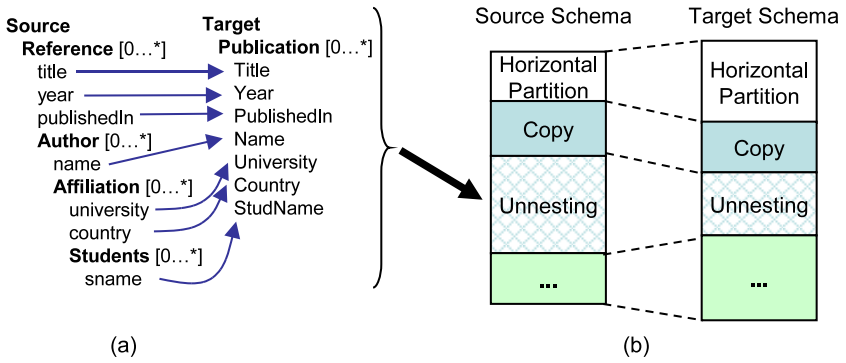


Fig. 7 Basic Scenario Expansion and Synthetic Scenario Generation

ToXGene (Barbosa et al, 2002) configuration template with which one can invoke ToXGene in order to produce the data of the source instance.

In the area of schema matching, the ISLab Instance Matching Benchmark (Ferrara et al, 2008) is also following a bottom-up approach. It uses several algorithms to create different data sets. It initially requires the creation of a reference ontology for a specific domain. Then, this ontology is populated with instances by querying websites. For example, *IMDB* enables the population of a *movie* ontology. Subsequently, a number of modifications on the data takes place, with three goals in mind: (i) to introduce variations in the data values, e.g., typographical errors, (ii) to introduce structural heterogeneity, e.g., properties represented by different structural levels, aggregations, and others, and (iii) to introduce local heterogeneity, which mainly includes semantic variations that requires ontological reasoning in order to cope with. Once the modifications have been performed, the benchmark users are provided with the initial reference ontology and the modified one, against which they evaluate matching tools.

6 Measuring Efficiency

6.1 Matching/Mapping Generation Time

Since one of the goals of mapping tools is to assist the matching/mapping designer in performing the time-concuming matching and mapping tasks faster, time plays a major role in measuring the performance of matching/mapping tools. Nevertheless, mapping tools like Spicy (Bonifati et al, 2008b), HePToX (Bonifati et al, 2006) or Clio (Popa et al, 2002), in their evaluation experiments make only a small reference to mapping generation time, and evaluation techniques proposed by Spicy (Bonifati et al, 2008a) or STBenchmark (Alexe et al, 2008c) do not elaborate extensively on

the issue. This is not an omission of their behalf. It reflects the fact that it is hard to measure time when human participation, in our specific case for the verification and guidance of the mapping tool, is part of the process. The time required by humans to understand the mappings generated by the tool and provide feedback is orders of magnitude higher than the one the tool requires for computing the mappings.

The situation is slightly different in matching tools where there is limited human intervention. Although computation time is still a central factor, it is not as important as the quality of the generated matches. A recent evaluation on a number of matching tools (Yatskevich, 2003) has extended previous evaluations (Do et al, 2003) by adding time measures for matching tasks on real-world matching scenarios. Unfortunately, these metrics have yet to be materialized in an a benchmark. In a more recent comparison (Kopcke and Rahm, 2010) of state-of-the-art matching tools, generation time has been one of the main comparison criteria and is also one of the metrics used by matching evaluation tools like XBenchMatch (Duchateau et al, 2007) and the ISLab Instance Matching Benchmark (Ferrara et al, 2008).

6.2 Data Translation Performance

It has already been mentioned that one of the popular uses of mappings is to translate data from one source to another, i.e., the data exchange task. This translation is done by materializing the target or integrated instance from the data of one or more source instances according to the mappings. Data sources typically contain a large number of records. This means that if the mappings are noumerous and describe complex transformations, then the time required to materialize the target instance may be significant. Based on this observation, it is clear that one of the factors to characterize the quality of a mapping tool is by the performance of the execution of the transformations described by the generated mappings. Metrics that can be used to measure such performance are the overall execution time and the degree of parallelization.

[Time] The most general purpose metric is the time required to perform the overall transformation time. Although this parameter is not explicitly stated in any matching or mapping evaluation effort, certain extensive experiments found in the literature (Alexe et al, 2008c) illustrate its importance. The generation of good transformation scripts is actually a way to characterize good mapping tools. Note that in order to avoid falling into the trap of evaluating the query execution engine instead of the mapping tool, when measuring the performance of the generated transformation scripts all the comparison and evaluation experiments should be performed on the same transformation engine.

There has been an increasing interest towards efficient methods for generating the right target instance given a mapping scenario, and more specifically in generating the *"core"*. The core (Fagin et al, 2003) is a minimum universal solution (Fagin et al, 2005). Core identification has been shown to be a co-NP hard problem (Fagin et al,

2005) for certain mapping dependencies. Despite these complexity results, there have been successful developments of efficient techniques that given two schemas and a set of mapping dependencies between them, in the form of tuple generating dependencies, produce a set of transformation scripts, e.g., in XSLT or SQL, whose execution efficiently generates a core target instance (Mecca et al, 2009; ten Cate et al, 2009).

Time performance is becoming particularly critical in ETL tools that typically deal with large volumes of data. Recent ETL benchmarks (Simitsis et al, 2009) consider it as one of the major factors of every ETL tool evaluation. Other similar factors that are also mentioned in ETL benchmarks are the workflow execution throughput, the average latency per tuple and the workflow execution throughput under failures. The notion of time performance in ETL tools extends beyond the end of the ETL workflow construction by considering, apart from the data translation time, the time required to answer business-level queries on the transformed data.

[Parallelization] One way to improve the data transformation time is to increase parallelization by generating mappings with minimum interdependencies. There are in general two broad categories of parallel processing: pipelining and partitioning. In pipelining different parts of the transformation are executed in parallel in a system with more than one processor and the data generated by one component are consumed immediately by another component without the need of waiting for the first component to fully complete its task. Pipelining works well for transformations that do not involve extremely large amounts of data. If this is not the case, a different parallelization mechanism called partitioning is preferable. In partitioning, the data is divided in different parts and then the transformation described by the mappings is applied on each partition independently of the others (Simitsis et al, 2009).

6.3 Human Effort

Since the goal of a matching or a mapping tool is to alleviate the designer from the laborious task of matching and mapping specification, it is natural to consider as one of the evaluation metrics of such a tool the effort required by the mapping designer.

In a schema matching task the input consists of only the two schemas. Since the task involves semantics, the designer must go through all the produced matches and verify their correctness. Consequently, the effort the designer needs to spend during a matching task can be naively quantified by the number of matches produced by the matcher and by their complexity.

A matcher may produce not only false positives, but also false negatives, which the matching designer will have to add manually to the result of the matcher, or will have to tune the tool in order to generate them. Two metrics have been proposed in the literature for quantifying this effort. One is the *overall*, which is also found under the name *accuracy* (Melnik et al, 2002) and is defined by the formula:

$$Overall = Recall \times \left(2 - \frac{1}{Precision}\right) \quad (1)$$

Recall and precision are metrics that will be presented later and evaluate the accuracy of the generated matches intuitively. The *overall* metric evaluates the amount of work an expert must provide to remove irrelevant matches (false positives), and to add those relevant that were not discovered (false negatives) (Do et al, 2003). The metric returns a value between $-\infty$ and 1. The greater the overall value is, the less effort the designer has to provide. It is a general belief (Do et al, 2003) that a precision below 50% implies that more effort is required from the designer to remove the false matches and add those missing than to manually do the matching. This is why such situations have a negative *overall* value. A limitation of the *overall* metric is that it assumes equal effort for removing an irrelevant match and for adding a missing one, which is rarely the case in the real-world.

Another metric to measure the human effort is the *human-spared resources* (HSR) (Duchateau, 2009). It counts the number of designer interactions required to correct both precision and recall, i.e., to manually obtain a 100% F-measure, a quality metric that will be discussed later. In other words, HSR takes into account not only the effort to validate or invalidate the discovered matches, but also to discover those missing. HSR is sufficiently generic, can be expressed in the range of $[0,1]$ or in time units (e.g., seconds), and does not require any input other than the one for computing precision, recall, F-measure or overall. The only limitation is that its does not take into account the fact that some matching tools may return the top-K matches instead of all of them.

In the schema mapping process, if the mapping specification is provided by the designer and is not taken from the output of an automatic matching task, the situation is different. The designer is required to provide input to the mapping tool through its interface, not only at the beginning but also throughout the mapping generation process since the designer will have to continuously verify the tool generated mappings and provide the respective modifications. Thus, the effort of the mapping designer can be measured by the number of inputs the designer provides to the tool.

This evaluation criterion is essentially an evaluation of the graphical interface of the tool. It is true, that the more intelligence a tool incorporates in interpretting the mapping designer input, the less input effort is required by the designer. However, certain interfaces may be so well-designed that even if there are many tasks the mapping designer needs to do, the human effort is kept to the minimum.

STBenchmark introduces a *simple usability (SU) model*, intended to provide a first-cut measure on the amount of effort required for a mapping scenario. It is based on a rough counting of the mouse clicks and keystrokes to quantify effort. This is important even if the time required for the mapping specification is much smaller in comparison to the time needed by the generated mappings to become transformation scripts and be executed. The click log information describing a mapping design for STBenchmark looks like this: *Right mouse click to pull up menu, left mouse click to select a schema element, typing a function into a box,* etc. Since different actions may require more effort than others (MacKenzie et al, 1991), for example, a point-

and-click is much easier than dragging or typing, weights can be assigned to each type of action in order to build a cost model for quantifying the total required effort.

One of the limitations of the above model is that it does not distinguish between clicks leading to the final mapping design and corrective actions, such as, undo or delete operations. It assumes that the mapping designer is familiar with the mapping tool and makes no mistakes. Another limitation is that the model does not capture the time the designer spends on thinking. A mapping tool that requires the designerdesignerthink for long time before designing the mapping with only few clicks, should not be considered more efficient than others that require less thinking by the designer but a few more clicks. A final limitation of this idea is that the model does not consider features such as presentation layout, visual aids, access to frequently used tasks etc.

In the area of schema integration, the Thalia benchmark (Hammer et al, 2005) can be used for objectively evaluating the capabilities of integration technology by taking into account, besides the correctness of the solution, the amount of programmatic effort (i.e., the complexity of external functions) needed to resolve any heterogeneity. For a fair comparison, any measurement of the needed effort must be done on the implementation of the twelve queries that Thalia provides. However, Thalia, does not provide any specifications on how this "effort" is to be measured.

7 Measuring Effectiveness

Measuring the effectiveness of a mapping or matching tool means measuring whether (or how much) the tool can fulfill its expectations for a given task. In the case of matching, an expert user typically knows what the correct matches are and the matching tool is expected to find them. Thus, evaluating its effectiveness boils down to a comparison between the expected set of matchings and the set of matchings that the tool generated. The situation is slightly different for the case of mapping systems. Since the expected output of a mapping system is a set of mappings that are used to generate the target (or global) instance, evaluating whether the mapping system has fulfilled its expectations can be done by checking whether the generated mappings can produce the expected target instance, or how close to the expected instance is the one that the generated mappings produce. This comparison can be done either extensionally, by comparing instances, or intensionally, by comparing the generated transformation expressions, i.e., the mappings. In this section we provide an overview of metrics that have been used in the literature for measuring such effectiveness.

7.1 Supported Scenarios

One way to evaluate a matching or mapping tool is by counting the percentage of scenarios it can successfully implement from a provided list of scenarios. A basic assumption is that there is an oracle providing the ground truth for each of these scenarios, i.e., the set of expected matches/mappings. This oracle is typically an expert user. A match/mapping generated by a tool is characterized as correct if it is part of the ground truth, or incorrect, otherwise. The successful implementation of a scenario by a tool is the generation of the expected matches/mappings.

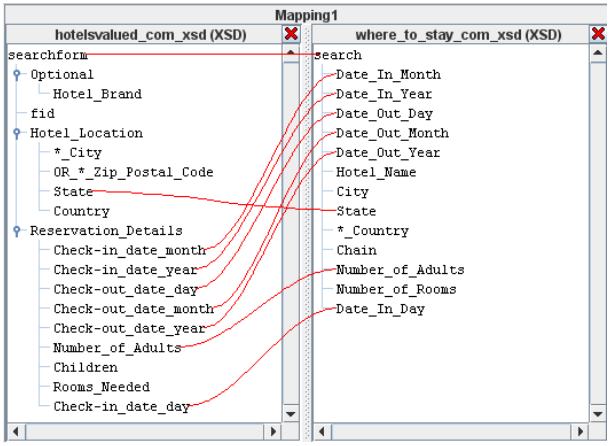
Provided with a rich set of mapping scenarios, one can test different aspects of a mapping tool. The effectiveness of the tool is the percentage of these scenarios that the tool could successfully implement. This approach is the one followed by STBenchmark (Alexe et al, 2008b). The scenarios the benchmark provides have been collected from the related scientific literature and real-world applications.

The characterization of the effectiveness of a tool based on the notion of the successful or unsuccessful implementation of scenarios, may not be the optimal approach especially in the case of systems. Very often, a mapping tool may not be able to produce exactly the expected mappings, yet it may be able to generate a pretty good approximation of them, or mappings that produce a target instance very close to the expected one. Under the above model, such a tool will be unfairly penalized as unsuccessful even though the final result is very close to the one expected. For this reason, a metric measuring proximity of the produced results to the expected is becoming an increasingly popular alternative.

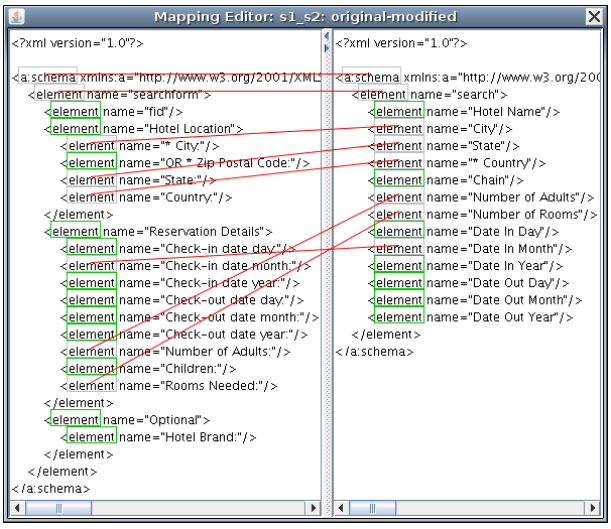
7.2 Quality of the Generated Matchings/Mappings

Four metrics that have been used extensively in the area of matching tool evaluation are the *precision*, *recall*, *f-measure* and the *fall-out* (Euzenat and Shvaiko, 2007). They are all intended to quantify the proximity of the results generated by a matching tool to those expected. They are based on the notions of *true positives*, *false positives*, *true negatives* and *false negatives*. Given two schemas S and T, let $\mathscr{M}$ represent the set of all possible matches that can exist between their respective elements. Assume that an oracle provides the list of expected matches. These matches are referred to as *relevant*, and all the other matches in $\mathscr{M}$ as *irrelevant*. The matching tool provides a list of matches that it considers true. These are the *tool relevant matches*, while the remaining matches in $\mathscr{M}$ are the *tool irrelevant matches*. A match in $\mathscr{M}$ is characterized as true positive, false positive, true negative or false negative, depending on which of the above sets it belongs. The respective definitions are illustrated in Table 1.

The precision, recall and f-measure (Van-Risbergen, 1979) are well-known from the information retrieval domain. They return a real value between 0 and 1 and have been used in many matching evaluation efforts (Duchateau et al, 2007; Do et al, 2002). Figure 8 depicts a matching example. It illustrates two schemas related to



(a) COMA++



(b) Similarity Flooding

Fig. 8 Correspondences discovered by two schema matchers

	Relevant Matches	Irrelevant Matches
Tool Relevant Matches	TP (True Positive)	FP (False Positive)
Tool Irrelevant Matches	FN (False Negative)	TN (True Negative)

Table 1 Contingency table forming the base of evaluation measures

hotel reservations and the relevant matches (illustrated by the inter-schema lines) generated by two matching tools, COMA++ (Aumueller et al, 2005) and Similarity

Flooding (Melnik et al, 2002), denoted as SF for short. COMA++ has discovered 9 matches while SF has discovered 7. Note that for SF, the matches between the root elements of the schemas are not considered.

[Precision] The *precision* calculates the proportion of relevant matches discovered by the matching tool with respect to all those discovered. Using the notation of Table 1, the precision is defined as

$$Precision = \frac{TP}{TP+FP}$$

An 100% precision means that all the matches discovered by the tool are relevant. In the particular example of Figure 8, both tools achieve a 100% precision:

$$Precision_{COMA++} = \frac{9}{9+0} = 100\% \qquad Precision_{SF} = \frac{7}{7+0} = 100\%$$

[Recall] *Recall* is another broadly used metric. It computes the proportion of matches discovered by the tool with respect to all the relevant matches. It is defined by the formula

$$Recall = \frac{TP}{TP+FN}$$

A 100% recall means that all relevant matches have been found by the tool. For the scenario of Figure 8, COMA++ has discovered 9 matches but missed 4 relevant. These missed matches are the false negatives. SF, on the other hand, discovered 7 relevant matches out of the 13. These results give the following recall values:

$$Recall_{COMA++} = \frac{9}{9+4} = 69\% \qquad Recall_{SF} = \frac{7}{7+6} = 54\%$$

[F-measure] *F-measure* is a trade-off between precision and recall. It is defined as:

$$f-measure(\beta) = \frac{(\beta^2+1) \times Precision \times Recall}{(\beta^2 \times Precision) + Recall}$$

The β parameter regulates the respective influence of precision and recall. It is often set to 1 to give the same weight to these two evaluation measures. Back to our running example, using a β equal to 1, the f-measure values obtained for COMA++ and SF, are respectively

$$f-measure_{COMA++} = \frac{2 \times 1 \times 0.69}{1+0.69} = 82\%$$

and

$$f-measure_{SF} = \frac{2 \times 1 \times 0.54}{1+0.54} = 70\%$$

[Fall-out] Another metric that is often used in the literature is the *fall-out* (Euzenat et al, 2006)(Ferrara et al, 2008). It computes the rate of incorrectly discovered matches out of the number of those non-expected. Intuitively, it measures the probability that a irrelevant match is discovered by the tool. The fall-out is defined by the formula:

$$Fallout = \frac{FP}{FP+TN}$$

In the running example of Figure 8, the number of non-expected, i.e., irrelevant, matches equals 253 (there exist a total of 266 possible matches including the 13 that are relevant). However, since neither tool discovered any irrelevant match, their fallout equals to 0%.

$$Fallout_{COMA++} = \frac{0}{0+253} = 0\% \qquad Fallout_{SF} = \frac{0}{0+253} = 0\%$$

The matching benchmark XBenchMatch (Duchateau et al, 2007) and the ontology alignment API (Euzenat, 2004) are based on the above metrics to evaluate the effectiveness of matching tools. They assume the availability of the expected set of matches through an expert user. Based on that set and the matches that the matching tool produces, the various values of the metrics are computed.

A limitation of the above metrics is that they do not take into consideration any post-match user effort, for instance, tasks that the user may need to do in order to guide the matching tool in the matching process, or any iterations the user may perform to verify partially generated results.

Measuring the quality of mappings, turns out to be more challenging than measuring the quality of the matches. The reason is that it requires comparisons among mappings which is not a straightforward task. Finding whether a generated mapping belongs to the set of expected mappings requires a comparison between this mapping and every other mapping in that set. This comparison boils down to query equivalence. Apart from the fact that query equivalence is a hard task per-se, it is also the case that a transformation described by a mapping may be also implemented through a combination of more than one different mapping. This means that it is not enough to compare with individual mappings only, but combinations of mappings should also be considered. For this reason, direct mapping comparison has typically been avoided as evaluation method of mapping tools. Researchers have instead opted for a comparison of the results of the mappings, e.g., the target instances.

Nevertheless, the precision, recall and the f-measure can be used to evaluate the large class of tools that do not differentiate among the matching and the mapping process but consider the whole task as a monolithic procedure. Spicy (Bonifati et al, 2008a) is an example of such tools, as it pipelines a matching module and a map-

ping generation module and allows the mapping designer to reiterate between the two processes to improve the quality of the generated mappings. In Spicy, the mapping tasks were designed in such a way that the source always contains a mapping that covers the entire target, meaning that no subset of the target schema remains unmapped. The set of mapping scenarios in the system are built in such a way that for a target schema, the correct set of matches that will generate a given predetermined mapping is internally identified. These matches are called the *ideal match* $\mathscr{M}_{id}$. At this point, the mapping generation algorithm can be run and a single transformation, T_{best}, i.e., the mapping that has the best scores in terms of instance similarity (cfr. next section for details), can be generated. Then the matches $\mathscr{M}_{T_{best}}$ on which this mapping is based upon are identified. In the ideal case, these matches are the same as the ideal match $\mathscr{M}_{id}$. The quality of the tool can be measured in terms of precision and recall of $\mathscr{M}_{T_{best}}$ with respect to $\mathscr{M}_{id}$. However, Spicy reports quality only in terms of precision. The reason is that in all cases the tool returns a number of matches that is equal to the size of the target, as mentioned above. As a consequence, precision and recall are both equal to the number of correct matches in $\mathscr{M}_{T_{best}}$ over the size of the target, which means that either precision or recall suffices to characterize the quality of the generated mappings.

The cases in which the source does not contain a mapping that covers the entire target are more complex and have not so far been addressed. It is believed that the most general case in which the target schema is not entirely covered by the mapping entails a new class of mapping tasks in which the target instance is partially filled with data exchanged with the source and partially filled with its own data.

The problem of characterizing mappings in a quantitative way has also been studied (Fagin et al, 2009b) through the notion of *information loss* which is introduced to measure how much a schema mapping deviates from an ideal invertible mapping. An invertible mapping is a mapping that given the generated target instance, it can be used to re-generate the original source instance. A first definition of invertibility has considered only constants in the source instance, and constants alongside labeled nulls in the target (cfr. (Fagin et al, 2010)). Labeled nulls are generated values for elements in the target that require a value but the mapping provides no specification for that value. In the inversion, these labeled nulls can propagate in the source instance, resulting into an instance that has less information that the original one. To capture in a precise way such an information loss, the notion of maximum extended recovery has been introduced for tgds with disjunction and inequalities (Fagin et al, 2009b). This new metric clearly identifies a viable approach to precisely compare schema mappings, but the full potential of this metric in benchmarking mapping tools still remains to be explored.

Another step towards the design of meaningful and high-quality schema mappings has been tackled recently (Alexe et al, 2010a) by using a *MapMerge* operator to merge multiple small mappings into a large ones. The evaluation of such an operator is done by using a novel similarity metric, that is able to capture the extent to which data associations are preserved by the transformation from a source to a target instance. The metric depends on the natural associations that exist among data values in the source instance, discovered by looking at the schema structures and by

following the schema referential integrity constraints. The idea behind the metric is that these associations must be preserved by the transformation that the mapping describes.

7.3 Quality of the Generated Target Instance

In systems that do not differentiate between the matching and the mapping task, an alternative to measuring precision, recall or f-measure would have been preferable. One such approach is to use the final expected result of the mapping process, which is the actual target instance generated by the transformation described by the mappings. This kind of evaluation is also useful in cases where one needs to avoid comparisons among mappings for reasons like those provided earlier. The expected target instance is typically provided by an expert user. Once the expected target instance is available, the success of a mapping task can be measured by comparing it to the actual target instance produced by the generated mappings. The approach constitutes an appealing verification and validation method, mainly due to its simplicity.

The comparison between the actual and the expected target instance can be done by considering an ad-hoc similarity function, such as *tree edit distance*, or by employing a general-purpose comparison technique (Bonifati et al, 2008a). Defining such a customized comparison technique is a promising direction for future developments in this area. The Spicy system ¡¡¡¡¡¡¡ .mine offers a comparison method based on circuit theory (Bonifati et al, 2008a), which is called *structural analysis*. Figure 9 shows an example of a circuit for a tree, that is easily constructed starting from building blocks corresponding to atomic attributes; more specifically, for each intermediate node n in a tree representation of the schema t, a resistance value, $r(n)$ can be defined. Such value cannot be based on instances, since intermediate nodes do not have a sample of instances, but rather on the topology of the tree. More specifically, $r(n) = k \times level(n)$, where k is a constant multiplicative factor, and $level(n)$ is the *level* of n in t, defined as follows: (i) leaves have level 0; (ii) an intermediate node with children $n_0, n_1, \ldots n_k$ has level $max(level(n_0), level(n_1), \ldots level(n_k)) + 1$. ======= offers a comparison method based on circuit theory (Bonifati et al, 2008a), called *structural analysis*. Figure 9 shows an example of a circuit generated by the tree representation of a schema, as shown on the left-hand side. The circuit is based on building blocks corresponding to atomic attributes. More specifically, for each intermediate node n in the schema tree, a resistance value $r(n)$ is defined. Such a value cannot be based on instances, since intermediate nodes of the tree represent higher structures, but it is rather based on the topology of the tree. In particular, $r(n) = k \times level(n)$, where k is a constant multiplicative factor, and $level(n)$ is the *level* of n in the tree, defined as follows: (i) leaves have level 0; (ii) an intermediate node with children $n_0, n_1, \ldots n_k$ has level $max(level(n_0), level(n_1), \ldots level(n_k)) + 1$; ¿¿¿¿¿¿¿ .r3386

¡¡¡¡¡¡¡ .mine The complete circuit is defined by means of a *circuit mapping function*, $circ(t)$ over a tree t. For a leaf node A, $circ(A)$ is defined by mapping a sampled attribute to a circuit. Intuitively, $circ(A)$ is assembled by assigning a set of features to a number of resistor and voltage generators. For a tree t rooted at node n with children $n_0, n_1, \ldots n_k$, $circ(t)$ is the circuit obtained by connecting in parallel $circ(n_0), circ(n_1), \ldots circ(n_k)$ between ground and an intermediate circuit node n_{top}, and then adding a resistor of value $r(n)$ from node n_{top} to output. Examples of such transformation are given in Figure 9. Note that the circuit mapping function makes the resulting circuits isomorphic to the original trees. ======= The complete circuit is defined by means of a *circuit mapping function*, $circ(t)$ over a tree t. For a leaf node A, $circ(A)$ is defined by mapping a sampled attribute to a circuit. Intuitivelly, $circ(A)$ is assembled by assigning a set of features to a number of resistors and voltage generators. For a tree t rooted at node n with children $n_0, n_1, \ldots n_k$, $circ(t)$ is the circuit obtained by connecting in parallel $circ(n_0), circ(n_1), \ldots circ(n_k)$ between ground and an intermediate circuit node n_{top}, and then adding a resistor of value $r(n)$ from node n_{top} to the output. Examples of such transformation are illustrated in Figure 9. Note that the circuit mapping function makes the resulting circuits isomorphic to the original trees. ¿¿¿¿¿¿¿ .r3386

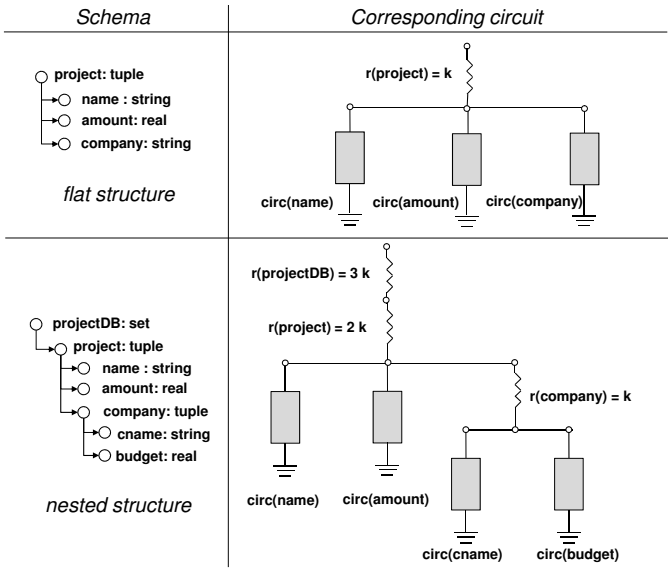


Fig. 9 Examples of circuits for flat and nested structures.

In Spicy, similarly to the *opaque* schema matching (Kang and Naughton, 2003), labels are ignored by the circuit mapping function, and values are basically treated as uninterpreted strings. Furthermore, values correspond to alphanumeric data in the underlying Spicy data model. The circuit features discussed above reflect this

choice. However, the circuit model is sufficiently flexible to allow the treatment of special data, like large texts or multimedia, as discussed in other orthogonal usage of circuits (Palmer and Faloutsos, 2003).

Given two trees t_1 and t_2, a measure of their similarity can be computed by mapping t_1 and t_2 to the corresponding circuits, $circ(t_1), circ(t_2)$, as depicted in Figure 9, solving the two circuits to determine their currents and voltages, and choosing a number of descriptive features of the corresponding circuits, $f_0, f_1, \ldots f_i$. A notion of *comparator* for each feature f_i as a module that computes the index of similarity Δ_i between the two structures with respect to feature f_i, is defined in Spicy as follows $\Delta_i = abs(f_i(circ(t_1)) - f_i(circ(t_2)))/f_i(circ(t_1))$. Finally, the overall similarity of the two trees is computed based on the values of $\Delta_0, \Delta_1, \ldots \Delta_i$ (Bonifati et al, 2008a).

The quality of the target instance is also an important factor in the case of ETL systems. For these systems, the quality is typically determined by the *data freshness*, the *resiliency to occasional failures* and the *easy of maintenance* (Simitsis et al, 2009). Data freshness means that the effect of any modification in the source instance is also implemented in the target. Resiliency to failures measures whether different transformation routes or recovery procedures can guarantee that in the case that a part of the transformation fails, the data that was to be generated can be generated either through different routes or by repetition of the failed procedure. Finally, the maintainability is affected, among others, by the simplicity of the transformation. A simple ETL transformation is more maintainable, whereas in a complex transformation it is more difficult to keep track of the primitive transformations that take place. Occasionally, the *compliance to business rules* is also one of the considered factors for measuring the quality of an ETL system.

7.4 Data Examples

Generating the expected target instance for evaluating a mapping system may not always be the most desired method. The size of the target schema may be prohibitively large and its generation at mapping design time may not be feasible. Even if its generation is possible, due to its size, even an expert mapping designer may find hard to understand the full semantics of the generated transformation, since it is practically impossible to always obtain a full view of the target data. The generated mappings between a source and the target schema may also be numerous, ambiguous and complicated to a degree that the designer is not able to understand what and how some target data was created from data in the source. To cope with these issues and help the designer in quickly and fully understanding the semantics of the mapping system generated transformations and validate them, carefully selected representative samples of the target instance can be used. Samples of the expected target instance can be used to drive the mapping process, while samples of the generated target instance can be used to communicate to the designer the semantics of the mappings the system has generated.

The importance of data examples in mapping generation has long ago been recognized (Yan et al, 2001). In the specific work, each mapping is considered a transformation query and is interpreted as an indirectly connected graph $G = (N,E)$, where the set of nodes N is a subset of the relations of the source schema and the set of edges E represents conjunctions of join predicates on attributes of the source relations. Typically, joins are inner joins but they can also be considered as outer joins or combinations of inner and outer joins. Given a query graph G, the *full* and the *possible* data associations can be computed. A data association is a relation that contains the maximum number of attributes whose data are semantically related through structural or constraint, e.g., foreign key, constructs. A full data association of G is computed by an inner join query over G and it involves all nodes in G. Given an induced, connected subgraphs of G, a data association can be constructed in the same way, but since it is based on a subgraph of G, the data association is referred to as a possible association. Full and possible data associations can be leveraged to understand what information needs to be included into a mapping.

From a different perspective, one could think of a wizard or a debugging tool that allows to better understand the semantics of the mappings by illustrating the flow of tuples from source to target in a schema mapping task. The notion of *routes* (Chiticariu and Tan, 2006) captures this idea and is useful in the mapping debugging process to understand the behavior of mappings. Routes can be created between original source and target instances or between illustrative data examples. Ultimately, routes can be used in conjunction with data examples to help the user dig in the semantics of a mapping.

To understand what a data example represents, assume a mapping generation situation with a source schema S, a target schema T and a set of mappings Σ. It is said that a data example (I,J) is satisfied by the set of mappings Σ, denoted as $(I,J) \models \Sigma$, if I is a fraction of an instance of S, J is a fraction of an instance of T, and there is a mapping $m \in \Sigma$ such that $m(I) = J$. Such a data example is called a *positive* data example. If I is a fraction of an instance of S, J is a fraction of an instance of T, but $(I,J) \not\models \Sigma$, then the data example is called *negative*. Positive examples are used to illustrated intended transformed data in the instance while negative examples can be used to describe undesired mapping transformations.

In the special case that the data J of a data example (I,J) is a universal solution (cfr. (Bonifati et al, 2010)), the example is called a *universal data example*. Universal data examples are of major importance due to their generality. A recent study (Alexe et al, 2010b) has highlighted that if the only kind of mappings considered are source-to-target tgds, a mapping can be characterized by a finite set of positive and negative data examples if and only if the source and the target schema contain only unary relation symbols. Nevertheless, the study has also shown that the universal examples may characterize the entire class of local-as-view (Lenzerini, 2002) source-to-target tgds.

In short, data examples have already found their way into mapping systems as a way of helping the designer understand and refine the generated mappings (Alexe et al, 2008a) and in certain cases select a subset of those mappings that the mapping system produces (Yan et al, 2001). They can also become an asset in mapping sys-

tem evaluation as indicated by some first efforts towards this direction (Alexe et al, 2010b). In particular, the mapping system Clio is employing debugging tools like Routes (Chiticariu and Tan, 2006) to build a mapping designer evaluation framework that is based on data examples. There are still many challenging research issues around that topic, for instance, a deeper study of the use of positive, negative and universal examples.

7.5 Quality of the Generated Target Schema

When the mapping system is used to create an integrated (or target) schema, a technique to evaluate the quality of the system is to measure the quality of the generated integrated schema. This can be done mainly by measuring its relationship to the schema that the designer had in mind to create, i.e., the intended integrated schema. The relationship can be measured in terms like the amount of information in the source schema that is also described in the integrated schema, the difference in the schema structures, etc. Three metrics have been recently proposed: the *completeness*, *minimality* and *structurality*.

Completeness Let Si_{tool} represent the target schema generated by the mapping tool and Si_{int} the intended target schema that models the integration. The notation $|S|$ is used to refer to the number of elements in a schema S. The *completeness* (Batista and Salgado, 2007) is a metric in the range of 0 to 1, that intuitively measures how many of the concepts that can be modeled by the source schema(s) can also be modeled by the target schema, i.e., the integration. More formally,

$$Completeness = \frac{|S_{tool} \cap S_{int}|}{|S_{int}|}$$

Minimality The *minimality* (Batista and Salgado, 2007) is another metric also in the range of 0 to 1, that indicates the redundancy that may appear in the integrated schema. The higher minimality, the lower the redundancy. Minimality is defined by the following expression, which basically calculates the percentage of extra elements in the integrated schema produced by the mapping tool with respect to the intended instance. In particular:

$$Minimality = 1 - \frac{|S_{tool}| - |S_{tool} \cap S_{int}|}{|S_{int}|}$$

Structurality The *structurality* has been introduced (Duchateau, 2009) to intuitively measure the "*qualities of the structure an object possesses*"[6]. In the case of schemas, this notion is translated to the set of ancestors of a schema structure. In other words, the structurality measures whether the elements of the generated and the intended schema contain the same set of ancestors. To compute structurality, the schemas are viewed as trees. Let S_{int} and S_{gen} denote the intended and the generated target schema, respectively. Assume also that in the tree representation of a schema S, $P_S(e)$ is the set of elements in the path from the root to the element e, exclusively. The structurality of an element e is defined as:

$$Structurality(e) = \max\left(0, \frac{\alpha|P_{S_{int}}(e) \cap P_{S_{gen}}(e)| - (|P_{S_{gen}}(e)| - |P_{S_{int}}(e) \cap P_{S_{gen}}(e)|)}{\alpha|P_{S_{int}}(e)|}\right)$$

Intuitivelly, the formula checks that an element shares most ancestors both in the generated and the intended integrated schemas. Besides, it takes into account the insertion of incorrect ancestors in the generated integrated schema. Note that the structurality of an element e of the intended schema that does not appear in the schema generated by the tool is zero. The parameter α is a constant factor that allows higher importance to be given to ancestors that have been created in the generated schema, as opposed to those that have not. Since the number of ancestors P_{gen} may be large, an element structurality may become negative, which explains the existence of the max function in the above formula. A negative value would be difficult to interpret by end-users, as this is the case for the overall measure when dealing with matching quality.

The structurality of a schema S_{gen} generated by the mapping tool is the average of the structuralities of the individual elements in the intended schema, i.e.,

$$Structurality\ of\ S_{gen} = \frac{\sum_{e \in S_{int}} Structurality(e)}{|S_{int}|}$$

The completeness, minimality and structurality metrics can be combined into a weighted sum to provide an overall metric for the proximity of the generated schema and the intended, i.e.,

$$Proximity = w_1 * Completeness + w_2 * Minimality + w_3 * Structurability$$

with $w_1 + w_2 + w_3 = 1$.

To illustrate the above metrics, consider the abstract schema shown on the left-hand side of Figure 10, and assume that it is the schema generated by the mapping tool. The schema that was intended to be created is the one on the right-hand side of the same figure. The number of common schema elements between these two schemas are 6, thus, $Completeness = \frac{6}{7}$ and $Minimality = 1 - \frac{8-6}{7} = \frac{5}{7}$. Assuming an α factor with value 2, the structuralities of the elements of the intended schema are

[6] http://en.wiktionary.org/wiki/structurality

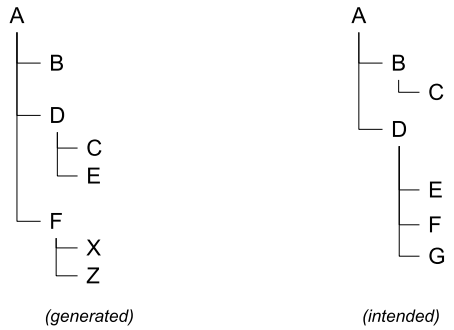


Fig. 10 An abstract example of a schema generated by a mapping tool (left) and the intended

Element	P_{int}	P_{gen}	**Element Structurality**
B	A	A	$\max(0, \frac{2\times1-(1-1)}{2\times1}) = 1$
D	A	A	$\max(0, \frac{2\times1-(1-1)}{2\times1}) = 1$
E	A,D	A,D	$\max(0, \frac{2\times2-(2-2)}{2\times2}) = 1$
G	A,D	$\emptyset$	$\max(0, \frac{2\times0-(0-0)}{2\times2}) = 0$
C	A,B	A,D	$\max(0, \frac{2\times1-(2-1)}{2\times2}) = \frac{1}{4}$
F	A,D	A	$\max(0, \frac{2\times1-(1-1)}{2\times2}) = \frac{1}{2}$

Table 2 Element Structuralities for the intended schema of Figure 10

illustrated in Table 2. According to these values, the structurality of the generated schema with respect to the intended schema is $\frac{1+1+1+0+\frac{1}{4}+\frac{1}{2}}{6} = 0.625$. Giving equal weight to completeness, minimality and structurality, the overall proximity of the generated schema to the intended is: $\frac{0.86+0.71+0.625}{3} = 0.73$

There has been an interesting set of experimental results (Duchateau, 2009) on computing the above metrics using a number of different datasets with the two popular matching systems: COMA++ (Aumueller et al, 2005) and Similarity Flooding (Melnik et al, 2002). The former system builds integrated schemas using an ASCII-tree format (then converted into XSD using a script (Duchateau et al, 2007)) while the latter system directly generates an XSD integrated schema. The matches discovered by the tools before building the integrated schema have not been checked. The experiments include a dataset extracted from the XCBL[7] and OAGI[8] collections, a dataset on university courses provided by the Thalia benchmark (Hammer et al, 2005), a Biology dataset from Uniprot[9] and GeneCards[10], a currency and

[7] www.xcbl.org

[8] www.oagi.org

[9] http://www.ebi.uniprot.org/support/docs/uniprot.xsd

[10] http://www.geneontology.org/GO.downloads.ontology.shtml

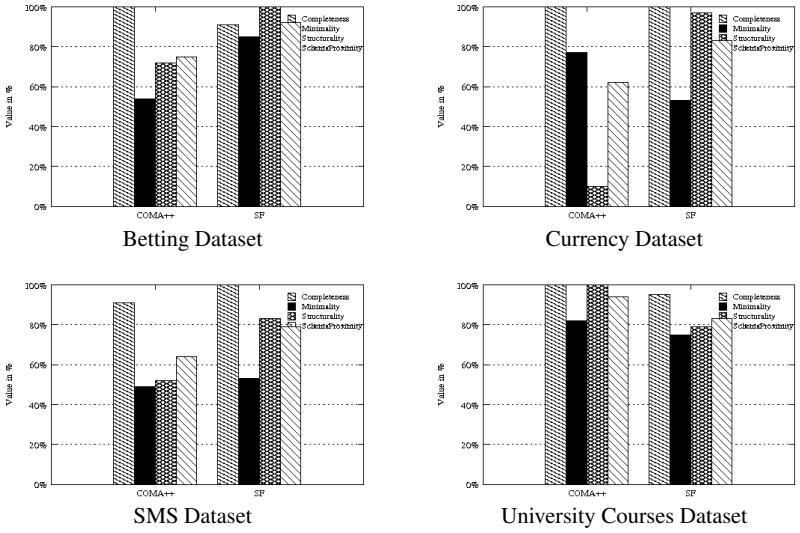


Fig. 11 Experimental Results for the Evaluation of the Target Schema Quality

sms dataset[11] , and a university department dataset (Duchateau et al, 2008). These datasets present various features that reflect real-world scenarios. For instance, the biology dataset contains a specific vocabulary which is not usually found in common dictionaries. The dataset about university courses describes a case in which many schemas have to be integrated. A part of the experimental results obtained from that effort are illustrated in Figure 11. It has been noticed that the tools can obtain a high completeness in most cases, mainly because the tools promote precision during the matching phase. On the contrary, the minimality is more difficult to achieve, since it depends on the recall. Finally, structurality is mostly preserved because the tools try to keep the same structure that they find in the source schemas.

8 Conclusion

We have presented a retrospective on key contributions in the area of evaluating matching and mapping tools. Schema matching and mapping is a relatively new area that has received considerable attention in the last few years. Since these notions may not have yet matured in the minds of researchers and of the commercial developers and users, and in order to avoid confusions, we have first attempted to provide a complete description of the architectural components, tasks, and goals of matching and mapping tools. Then we motivated the importance of evaluation methods and benchmarks for researchers, developers, businesses and users.

[11] www.seekda.com

Schema matching is a topic that has been extensively studied. There is already a long list of research prototypes and tools. Since the matching task involves semantics, evaluating the correctness of the output of a matching tool is a task requiring human intervention. The major issue in all these matching cases is deciding what is the correct answer, i.e., the intended matches. This is a challenging task since, due to the semantic heterogeneity, different perspectives may give different answers. Evaluation techniques for matching tasks have focused on the development of metrics that will allow a common evaluation base and effective communication of the evaluation results. We have provided a description of these metrics and have highlighted features and limitations.

Schema mapping seems to be a problem for which there is still some confusion as to what constitutes a mapping tool, what is its input, in what form, and what is its output. Different research prototypes and commercial tools have followed different approaches, something that makes their direct comparison and evaluation difficult. We have attempted to provide a definition of what a mapping tool is and the parameters one should consider when evaluating such tools. We have highlighted the lack of evaluation standards and have provided a complete picture of what an evaluation standard (or benchmark) should contain, alongside existing efforts towards the creation of such a standard.

Mapping tools have been mainly designed for data exchange. Nevertheless, they have been extensively used in integration systems for constructing an integrated global schema. Based on this dimension, we have also provided a number of metrics for measuring the success of the schema integration task performed by mapping tools.

Acknowledgments: We are grateful to B. Alexe, L. Chiticariu, A. Kementsietsidis, E. Rahm and P. Shvaiko for their valuable comments and suggestions.

References

Abiteboul S, Hull R, Vianu V (1995) Foundations of Databases. Addison-Wesley

Alexe B, Chiticariu L, Miller RJ, Tan WC (2008a) Muse: Mapping Understanding and deSign by Example. In: ICDE, pp 10–19

Alexe B, Tan WC, Velegrakis Y (2008b) Comparing and evaluating mapping systems with STBenchmark. Proceedings of VLDB 1(2):1468–1471

Alexe B, Tan WC, Velegrakis Y (2008c) STBenchmark: towards a benchmark for mapping systems. Proceedings of VLDB 1(1):230–244

Alexe B, Hernandez M, Popa L, Tan WC (2010a) MapMerge: Correlating Independent Schema Mappings. Proceedings of VLDB 3(1)

Alexe B, Kolaitis PG, Tan W (2010b) Characterizing Schema Mappings via Data Examples. In: PODS

Altova (2008) MapForce. Http://www.altova.com

Atzeni P, Torlone R (1995) Schema Translation between Heterogeneous Data Models in a Lattice Framework. In: IFIP, pp 345–364

Aumueller D, Do HH, Massmann S, Rahm E (2005) Schema and ontology matching with COMA++. In: SIGMOD, pp 906–908

Barbosa D, Mendelzon AO, Keenleyside J, Lyons KA (2002) ToXgene: a template-based data generator for XML. In: SIGMOD, p 616

Batini C, Lenzerini M, Navathe SB (1986) A Comparative Analysis of Methodologies for Database Schema Integration. ACM Comp Surveys 18(4):323–364

Batista M, Salgado A (2007) Information Quality Measurement in Data Integration Schemas. In: Workshop on Quality in Databases, pp 61–72

Bergamaschi S, Domnori E, Guerra F, Orsini M, Lado RT, Velegrakis Y (2010) Keymantic: Semantic Keyword based Searching in Data Integration Systems. Proceedings of VLDB 3(2)

Bernstein PA, Melnik S (2007) Model management 2.0: manipulating richer mappings. In: SIGMOD, pp 1–12

Bernstein PA, Giunchiglia F, Kementsietsidis A, Mylopoulos J, Serafini L, Zaihrayeu I (2002) Data Management for Peer-to-Peer Computing : A Vision. In: WebDB, pp 89–94

Bertinoro (ed) (2007) Bertinoro Workshop on Information Integration, www.dis.uniroma1.it/˜lenzerin/INFINT2007

Bohme T, Rahm E (2001) XMach-1: A Benchmark for XML Data Management. In: BTW, pp 264–273

Bonifati A, Chang EQ, Ho T, Lakshmanan LV, Pottinger R (2006) HePToX: Marrying XML and Heterogeneity in Your P2P Databases. In: VLDB, pp 1267–1270

Bonifati A, Mecca G, Pappalardo A, Raunich S, Summa G (2008a) Schema Mapping Verification: The Spicy Way. In: EDBT, pp 85 – 96

Bonifati A, Mecca G, Pappalardo A, Raunich S, Summa G (2008b) The Spicy system: towards a notion of mapping quality. In: SIGMOD, pp 1289–1294

Bonifati A, Mecca G, Papotti P, Velegrakis Y (2010) Advances in Schema Matching and Mapping by Z. Bellahsene, A. Bonifati, E. Rahm, Data-Centric Systems and Applications 5258, Springer, chap Discovery and Correctness of Schema Mapping Transformations, pp XX–YY

Bressan S, Dobbie G, Lacroix Z, Lee M, Li YG, Nambiar U, Wadhwa B (2001) X007: Applying 007 Benchmark to XML Query Processing Tool. In: CIKM, pp 167–174

Carey MJ (2006) Data delivery in a service-oriented world: the BEA AquaLogic data services platform. In: SIGMOD, pp 695–705

ten Cate B, Chiticariu L, Kolaitis P, Tan WC (2009) Laconic Schema Mappings: Computing Core Universal Solutions by Means of SQL Queries. Proceedings of VLDB 2(1):1006–1017

Chiticariu L, Tan WC (2006) Debugging Schema Mappings with Routes. In: VLDB, pp 79–90

Do HH, Rahm E (2002) COMA - A System for Flexible Combination of Schema Matching Approaches. In: VLDB, pp 610–621

Do HH, Melnik S, Rahm E (2002) Comparison of Schema Matching Evaluations. In: Web, Web-Services, and Database Systems, pp 221–237

Do HH, Melnik S, Rahm E (2003) Comparison of Schema Matching Evaluations. In: Revised Papers from the NODe 2002 Web and Database-Related Workshops on Web, Web-Services, and Database Systems, Springer-Verlag, London, UK, pp 221–237

Doan A, Domingos P, Halevy AY (2001) Reconciling schemas of disparate data sources: A machine-learning approach. In: SIGMOD, pp 509–520

Doan A, Madhavan J, Domingos P, Halevy AY (2004) Ontology Matching: A Machine Learning Approach. In: Handbook on Ontologies, pp 385–404

Duchateau F (2009) Towards a Generic Approach for Schema Matcher Selection: Leveraging User Pre- and Post-match Effort for Improving Quality and Time Performance. PhD thesis, Universite Montpellier II - Sciences et Techniques du Languedoc

Duchateau F, Bellahsene Z, Hunt E (2007) XBenchMatch: a Benchmark for XML Schema Matching Tools. In: VLDB, pp 1318–1321

Duchateau F, Bellahsene Z, Roche M (2008) Improving quality and performance of schema matching in large scale. Ingenierie des Systemes d'Information 13(5):59–82

Euzenat J (2004) An API for Ontology Alignment. In: ISWC, pp 698–712

Euzenat J, Shvaiko P (2007) Ontology matching. Springer-Verlag, Heidelberg (DE)

Euzenat J, Mochol M, Shvaiko P, Stuckenschmidt H, Svab O, Svatek V, van Hage WR, Yatskevich M (2006) Results of the Ontology Alignment Evaluation Initiative. In: OM

Fagin R, Kolaitis PG, Popa L (2003) Data exchange: getting to the core. In: PODS, pp 90–101

Fagin R, Kolaitis PG, Miller RJ, Popa L (2005) Data exchange: semantics and query answering. Theoretical Computer Science 336(1):89–124

Fagin R, Haas LM, Hernandez M, Miller RJ, Popa L, Velegrakis Y (2009a) Conceptual Modeling: Foundations and Applications by A. Borgida, V. Chaudhri, P. Giorgini, E. Yu, Springer, chap Clio: Schema Mapping Creation and Data Exchange, pp 198–236

Fagin R, Kolaitis PG, Popa L, Tan WC (2009b) Reverse data exchange: coping with nulls. In: PODS, pp 23–32

Fagin R, Kolaitis P, Popa L, Tan W (2010) Advances in Schema Matching and Mapping by Z. Bellahsene, A. Bonifati,E. Rahm, Data-Centric Systems and Applications 5258, Springer, chap Schema Mapping Evolution through Composition and Inversion, pp XX–YY

Ferrara A, Lorusso D, Montanelli S, Varese G (2008) Towards a Benchmark for Instance Matching. In: OM

Fletcher GHL, Wyss CM (2006) Data Mapping as Search. In: EDBT, pp 95–111

Giunchiglia F, Shvaiko P, Yatskevich M (2004) S-Match: an Algorithm and an Implementation of Semantic Matching. In: ESWS, pp 61–75

Giunchiglia F, Shvaiko P, Yatskevich M (2005) S-Match: an algorithm and an implementation of semantic matching. In: Semantic Interoperability and Integration

Giunchiglia F, Yatskevich M, Avesani P, Shvaiko P (2009) A large dataset for the evaluation of ontology matching. Knowledge Eng Review 24(2):137–157

Halevy AY, Ives ZG, Suciu D, Tatarinov I (2003) Schema Mediation in Peer Data Management Systems. In: ICDE, p 505

Hammer J, Stonebraker M, Topsakal O (2005) THALIA: Test Harness for the Assessment of Legacy Information Integration Approaches. In: ICDE, pp 485–486

Heinzl S, Seiler D, Unterberger M, Nonenmacher A, Freisleben B (2009) MIRO: a mashup editor leveraging web, Grid and Cloud services. In: iiWAS, pp 17–24

IBM (2006) Rational Data Architect. Www.ibm.com/software/data/integration/rda

Ioannou E, Nejdl W, Niederee C, Velegrakis Y (2010) OntheFly Entity-Aware Query Processing in the Presence of Linkage. Proceedings of VLDB 3(1)

Kang J, Naughton JF (2003) On Schema Matching with Opaque Column Names and Data Values. In: SIGMOD, pp 205–216

Kopcke H, Rahm E (2010) Frameworks for entity matching: A comparison. DKE 69(2):197–210

Lee Y, Sayyadian M, Doan A, Rosenthal A (2007) eTuner: tuning schema matching software using synthetic scenarios. VLDB Journal 16(1):97–122

Legler F, Naumann F (2007) A Classification of Schema Mappings and Analysis of Mapping Tools. In: BTW, pp 449–464

Lenzerini M (2002) Data Integration: A Theoretical Perspective. In: PODS, pp 233–246

Lerner BS (2000) A Model for Compound Type Changes Encountered in Schema Evolution. TPCTC 25(1):83–127

MacKenzie IS, Sellen A, Buxton W (1991) A comparison of input devices in elemental pointing and dragging tasks. In: CHI, pp 161–166

Madhavan J, Bernstein PA, Rahm E (2001) Generic Schema Matching with Cupid. In: VLDB, pp 49–58

Mecca G, Papotti P, Raunich S (2009) Core schema mappings. In: SIGMOD, pp 655–668

Melnik S, Garcia-Molina H, Rahm E (2002) Similarity Flooding: A Versatile Graph Matching Algorithm and Its Application to Schema Matching. In: ICDE, pp 117–128

Microsoft (2005) Visual Studio. Msdn2.microsoft.com/en-us/ie/bb188238.aspx

Miller RJ, Haas LM, Hernandez MA (2000) Schema Mapping as Query Discovery. In: VLDB, pp 77–88

Mork P, Seligman L, Rosenthal A, Korb J, Wolf C (2008) The Harmony Integration Workbench. JODS 11:65–93

Naumann F, Ho CT, Tian X, Haas LM, Megiddo N (2002) Attribute Classification Using Feature Analysis. In: ICDE, p 271

Okawara T, Morishima A, Sugimoto S (2006) An Approach to the Benchmark Development for Data Exchange Tools. In: Databases and Applications, pp 19–25

Palmer C, Faloutsos C (2003) Electricity Based External Similarity of Categorical Attributes. In: Proc. of PAKDD

Popa L, Velegrakis Y, Miller RJ, Hernandez MA, Fagin R (2002) Translating Web Data. In: VLDB, pp 598–609

Rahm E, Bernstein PA (2001) A survey of approaches to automatic schema matching. VLDB Journal 10(4):334–350

Runapongsa K, Patel JM, Jagadish HV, Al-Khalifa S (2002) The Michigan Benchmark: A Microbenchmark for XML Query Processing Systems. In: EEXTT, pp 160–161

Schmidt AR, Waas F, Kersten ML, Carey MJ, Manolescu I, Busse R (2002) XMark: A Benchmark for XML Data Management. In: VLDB, pp 974–985

Simitsis A, Vassiliadis P, Dayal U, Karagiannis A, Tziovara V (2009) Benchmarking ETL Workflows. In: TPCTC, pp 199–220

Smith K, Morse M, Mork P, Li M, Rosenthal A, Allen D, Seligman L (2009) The Role of Schema Matching in Large Enterprises. In: CIDR

Stylus Studio (2005) XML Enterprise Suite. Www.stylusstudio.com

Transaction Processing Performance Council (2001) TPC-H Benchmark. Tpc.org

Van-Risbergen C (1979) Information Retrieval. 2nd edition, London, Butterworths

Velegrakis Y (2005) Managing Schema Mappings in Highly Heterogeneous Environments. PhD thesis, University of Toronro

Wun A (2009) Mashups. In: Encyclopedia of Database Systems, Springer, pp 1696–1697

Yan L, Miller RJ, Haas LM, Fagin R (2001) Data-Driven Understanding and Refinement of Schema Mappings. In: SIGMOD, pp 485–496

Yao B, Ozsu T, Khandelwal N (2004) XBench benchmark and performance testing of XML DBMSs. In: ICDE, pp 621–633

Yatskevich M (2003) Preliminary evaluation of schema matching systems. Tech. Rep. DIT-03-028, University of Trento