Summary
A Survey of Approaches to Automatic Schema Matching
Alari Lukk, Prit Rand

Schema matching is the main problem in numerous database application domains, for example data integration, E-business, data warehousing or semantic query processing. This summary focuses on a survey of approaches to automatic schema matching by Erhard Rahm and Philip A. Bernstein.

This survey presents a taxonomy which describes briefly many of the existing approaches. It distinguishes between the following types of matchers: schema-level and instance-level, element-level and structure-level, and finally language-based and constraint-based. The survey reviews some match implementations based on the given classification.

A basic operation is Match, which requires two schemas as input and generates a mapping between elements of those given schemas that correspond semantically to each other. Such matching is currently carried out manually, possibly through a graphical user interface, which is tedious, error-prone and expensive process. To provide automated support for schema matching, we need a generic implementation of Match, which could be used throughout different application areas.

Application Domains
The greatest motivation to work on schema match has been schema integration, which constructs a global view from a set of separately developed schemas. These schemas usually use different terminology and have different structure. The first step in integration process would be to identify and characterize the relationships between the given schemas.

Data warehouses are decision support databases, which are extracted from sets of data sources. Data needs to be transformed into the warehouse format and the match operation can be used to design these transformations. One approach is to find elements from the source which correspond to the warehouse elements. Another approach is to reuse existing transformations.

In E-commerce, for the systems to be able to exchange messages, developers have to convert messages between different formats used by different trading partners. One part of the conversion is translating between message schemas. Instead of manual specification of relations between formats a match operation would probably reduce the amount of effort done manually to generate a mapping between two message schemas.

In semantic query processing a user specifies the output of a query. Since the user’s specification may not be stated using the same names of elements as in the database schema, it is necessary to map the user-specified concepts to schema elements, which is also an application of the match operation.

For the purposes of given survey a schema was defined as a set of elements connected by some structure and a mapping as a set of mapping elements where certain elements are mapped between two schemas. The match operation was defined as a function which takes two schemas as input and
returns a corresponding mapping as a result. The implementation of Match should able to use auxiliary information to help find matches. Also it should match the schemas in a uniform internal representation.

**Classification of Schema Matching Approaches**

An implementation of Match uses many match algorithms, where we can divide approaches into two – the combination of individual matchers by using multiple matching criteria within a hybrid matcher and the combination of individual matchers by combining together multiple match results given by match algorithms within a composite matcher.

Individual matchers can be divided into matchings which consider instance data and the ones that consider only schema-level information (*instance vs schema*). Secondly into matchings which map individual schema elements and the ones that map combinations of elements (*element vs structure*). Thirdly into matchings which use a linguistic-based approach and the ones that use a constraint-based approach (*language vs constraint*). Then we can divide them into matchings which differ by cardinality of the overall match result. There are four cases – 1:1, 1:n, n:1, n:m. Finally there are matchers which rely also on auxiliary information, for example dictionaries, global schemas or user input.

Schema-level matchers consider schema information such as name, description, data type, relationship types and constraints. Instance-level data may clarify the meaning of the schema elements, especially if the schema information is limited or the schema holds semistructured data.

Most of the approaches can be applied for both schema-level and instance-level matchers, except for structure-level approaches, which are not used for instance-based matchers.

Element-level matching determines matching elements in the second input schema for each element of the first input schema, when structure-level matching refers to matching combinations of elements which appear together in a structure.

An element of an input schema can participate in zero, one or many mapping elements of the match result. Also within one individual mapping element it is possible that one or more elements of the first input schema match one or more elements of the second input schema. Therefore we can use the normal relationship cardinalities – 1:1, set-oriented 1:n, n:1 and n:m.

Language-based matchers use words and sentences to detect semantically similar schema elements. Name-based matching is used to match schema elements which have equal or similar names. Similarity is measured in different ways, including equality of names, equality of canonical name representations after preprocessing, equality of synonyms, hyponyms, user-specified matches and similarity of substrings, pronunciation or soundex. Using synonyms and hyponyms is possible by using thesauri or dictionaries. Often the comments in natural language which are contained by the schemas can be used to match elements.

Furthermore, the constraints which define data types, value ranges, uniqueness and other metadata can also be used as a criteria to map elements.
One further generic possibility is to use existing mappings. Such a reuse may work for some part of a schema and the main problem is to determine, which part is similar to some part of a previously matched schema.

**Combining Different Matchers**

Hybrid matchers combine several individual matching approaches directly to determine match candidates, providing better match candidates and better performance than when executing many separate matchers. Composite matchers combine the results of different separately executed matchers, which is more flexible than the hybrid matchers.

---

An automatic matcher or combination of them can reduce the amount of human interaction needed, but in the end it is still required that an end user either accepts or rejects the suggested matches.

**Prototype Schema Matchers**

The survey gives a short overview of different schema matchers which have been implemented all over the world. All given systems support multiple matching criteria, six of them use a hybrid matcher and only one (LSD – Learning Source Descriptions) – use a composite match approach, though a flexible ordering of matchers is not supported. Most systems support both structure-level and element-level matching. Only two of the systems consider instance data and all seven systems focus only on 1:1 matches. Only a system named Cupid is not developed with a specific application domain in mind but is intended for general use. Most systems support multiple schema types as input. Although the use of auxiliary information is present, the reuse of previous matches is not yet supported.

Hereby we list the seven systems introduced: SemInt (Northwestern University), which creates a mapping between individual attributes of two schemas by using several constraint-based and content-based matching criteria. It uses neural networks to determine match candidates. LSD (University of Washington) uses machine-learning techniques to match given data source against global schema. SKAT (Stanford University) uses a rule-based approach to determine matches between two ontologies. TransScm (Tel Aviv University) derives an automatic translation between schema instances by schema matching. Internal schema representation is in the form of labeled graphs. DIKE (University of Calabria) determine synonym and inclusion relationships between objects of different schemas. ARTEMIS is a schema integration tool used as a component of a database mediator MOMIS (Mediator envirOment for Multiple Information Sources. Cupid (Microsoft research) is a hybrid matcher based on element-based and structure-level matching.

The survey also introduces some of other prototypes which relate to schema matching approaches and offer required functionality like Clio, Similarity Flooding, Delta, Tess and Tree Matching.

---

The purpose of the given paper was to propose a taxonomy for existing approaches to automating schema matching process. It also gave a short overview of the existing systems which implement
these approaches in smaller or greater extent. The authors hope that it will be useful to programmers and researchers who implement and develop schema matching algorithms.