Data Integration for the Arts and Humanities: A Language Theoretical Concept

Tobias Gradl(✉) and Andreas Henrich

Media Informatics Group, University of Bamberg, Bamberg, Germany
{tobias.gradl,andreas.henrich}@uni-bamberg.de

Abstract. In the context of the arts and humanities, heterogeneity largely corresponds to the variety of disciplines, their research questions and communities. Resulting from the diversity of the application domain, the analysis of overall requirements and the subsequent derivation of appropriate unifying schemata is prevented by the complexity and size of the domain. The approach presented in this paper is based on the hypothesis that data integration problems in the arts and humanities can be solved on the theoretical foundation of formal languages. In applying a theoretically substantiated framework, integrative solutions on the formal basis of language specifications can be tailored to specific and individual research needs—abstracting from reoccurring technical difficulties and leading the focus of domain experts on semantic aspects.

1 Introduction

The research data landscape of the arts and humanities is characterized by a high degree of distribution, heterogeneity and autonomy, which often originate from the logical and geographical distribution of institutions and their data sources. Without the existence of authorities that focus on the consolidation and coordination of data structures and processing practices, the aspect of autonomy further promotes the development of heterogeneity on various abstraction levels [14].

With respect to traditional application domains of data integration, the term *heterogeneity* is often associated with its connotation as fundamental *data integration problem* [7]. In the particular context of the arts and humanities, however, heterogeneity—particularly on structural and semantic levels—reflects the diversity of the disciplines, their research questions and communities. With particular focus on metadata, heterogeneity typically exists in terms of applied schemata and the context- and discipline-specific knowledge that is required to correctly understand the schemata and contained data. Situated within a particular academic surrounding, the digitization, description and usage of research objects in digital collections is embedded within context-specific semantics, which we understand as *generative context*. In contrast to this original context, data can be interpreted and utilized with respect to research questions that are situated in other disciplines—representing the *interpretive or application context* of data. In this respect, our goal is to facilitate data integration for the diverse domains and

research questions of the arts and humanities. Traditional approaches, which often define a global, integrative view [8] lead to a unifying harmonization of data. For the overall case of the arts and humanities, this harmonization would, however, result in the loss of structural elements, which are not considered relevant for most disciplinary contexts in order to compose a view of reasonable complexity. Opposed to such harmonizing efforts, we assume the evolution of so called *semantic clusters* [5]. Such clusters are composed of data sources that are associated through a common application context such as a funded research project, which then explicates the relations between the sources. A cluster is modeled by domain experts in accordance with the specific information needs of researchers and can be reused for similar research or interconnected with other clusters.

In this paper we present an integration concept [4], which satisfies requirements that we consider critical for supporting the evolution of semantic clusters and hence the research-driven definition, transformation and integration of data: A fundamental requirement thereby consists in the *adaptivity* of the concept with respect to the structuredness of relevant data sources and the usage scenarios in the domains. To achieve such adaptivity, a high degree of *expressiveness* of data descriptions and transformations is required—allowing a detailed explication of the generative context and the conversions into application contexts. Despite this expressiveness, the concept *encapsulates technical aspects* of data integration, which can be solved generically—allowing domain experts to focus on the semantic aspects of integration. The remainder of this paper is structured as follows: After an overview of the context in Sect. 2 we will introduce the language theoretical foundation and the derived modeling concept in Sect. 3. In Sect. 4 we present the architecture of the implemented grammatical transformation framework as well as the web-based modeling tool. After presenting exemplary evaluation results, we conclude the paper with Sect. 5.

2 Context

In order to mediate between locally contextualized schemata, data integration involves an extensive requirements analysis to gain insight of the concepts that are contained in the local perspectives and need to be incorporated within a global schema [8]. The appropriateness of such a structure is traditionally expressed in terms of four requirements [2]:

- *Completeness*: The schema includes all concepts that are contained in the local schemata and that are relevant to the considered application domain.
- *Correctness*: Any local concept that is represented by the global schema must be reflected with equivalent semantics.
- *Minimality*: Concepts with identical semantics are represented only once.

– *Understandability*: Labels of elements are chosen in a fashion that prevents ambiguities and enhances overall understandability.

Approaches to data integration often follow the theoretical foundation expressed in [8] by validly employing the concept of a global view to provide integrative services (see e.g. ISIDORE [13], OAIster [6] and Europeana [12] in this context). Resulting from the consideration of the arts and humanities as application domain of data integration, the methodological execution of an extensive requirements analysis and the subsequent derivation of appropriate unifying schemata is prevented by the complexity and size of the domain: If the design of a global schema aims a the requirement of completeness, (1) a large amount of concepts have to be included within the schema that are irrelevant in specific use cases and (2) the understandability and correctness are potentially impacted by the complexity of the schema.

The approach presented in this paper is based on the hypothesis that the data integration problem in the particular context of the arts and humanities can be interpreted and solved individually by experts within the arts and humanities on the theoretical foundation of formal languages. The recognition of provided input, its transformation into an intermediate representation and the generation of desired output are all typical tasks of language applications. In applying a theoretically substantiated framework, integrative solutions on the formal basis of language specifications can be tailored to specific and individual research needs.

Traditional language applications are formed by compilers—i.e. computer programs, which parse source code specified in terms of a computer language and translate instructions to machine executable code. More generally, compilers can be understood as "computer programs that translate a program written in one language into a program written in another language" [3]. Compilation is performed against human readable source code and results in immediately executable binary code or an intermediate representation. In the latter case, a runtime environment is needed to interpret such representations at the execution time of the program. Despite the focus on the complex tasks of source code compilation or interpretation, the concept of *language application* can be interpreted from a wider perspective as "any program that processes, analyzes, or translates an input file" [10]. Although applications or components e.g. for processing configuration files or importing data from external files are typically not defined as language applications, they are based on the (often implicit) specification of a language, providing rules, which input should be considered as valid and how it should be further processed. Irrespective of being explicitly or implicitly defined, language applications implement the syntactical and semantic rules of a language by performing tasks according to sentences of the specified language.

3 Language Theoretical Foundation

Our concept is based on the formal description of schemata as regular tree grammars (RTG) [4,9,15], which results in the interpretation of a schema as a finite structure $\langle N, T, R, P\rangle$—an RTG with the finite sets of nonterminals (N) and terminals (T), the root symbol ($R \in N$) and the set of production rules (P). Due to the generalization from strings to trees, regular tree grammars allow production rules of the form $n \rightarrow te_c$, where

- $n \in N$,
- $t \in T$ and
- $e_c \subset N$ reflects the content model that is defined over the set of non-terminals.

Based on actual schemata and documents of the arts and humanities, the above definition for schemata is considered to represent the *parsing-oriented view*, which allows an initial validation and processing of external data, but does not necessarily reflect the full extent of the semantic structure and content that is encoded within a document. If the content of individual elements can be further described, decomposed or transformed into semantically enriched forms, a perspective different from the strictly parsing-oriented view is required.

To facilitate the representation of such substructures or alternative elements within a schema, we define a *semantic extension*—resulting in the definition of a schema as 6-tuple $S = \langle N, T, R, P, E, F\rangle$, where N, T, R and P form grammatical components and as such the parsing-oriented view as introduced above. The components of L and F provide the semantic extension of the original structure of the schema, where:

- L forms a set of labels and
- F is a set of labeling functions $x \rightarrow le_l$, where:
  - $x \in (N \cup L)$,
  - $l \subseteq L$ and
  - $e_l := \{I, op\}$ defining a function over a set of input values $I \subseteq N$ and an operation of the arity $|I|$.

For the creation of unifying views over heterogeneous semi-structured data, mappings between the relevant source schemata and the selected target schema need to be evaluated and applied. Data that is specified in terms of local schemata is transformed into the corresponding representation of the integrative view. Mappings can be defined as functions for relating source and target objects [4,14], which indicates that simple associations between individual elements are insufficient to represent related objects. We introduce the notion of *concept mappings* $cm = \langle E_{S_S}, E_{S_T}, f\rangle$ as correlation between a set of source elements E_{S_S} and a set of target elements E_{S_T} [4], where:

- $E_{S_S} = \{e_{S_S,i} \ldots e_{S_S,j}\} \mid e_{S_S} \in (N_{S_S} \cup L_{S_S})$
- $E_{S_T} = \{e_{S_T,k} \ldots e_{S_T,l}\} \mid e_{S_T} \in (N_{S_T} \cup L_{S_T})$
- $f\colon E_{S_S} \rightarrow E_{S_T}$

A schema mapping $M_{S_S \rightsquigarrow S_T}$ is then defined as a set of concept mappings $cm_1 \ldots cm_n$ defined over a source and target schema S_S and S_T. Aside from the labeling functions, the mapping functions f of concept mappings present the second use-case of our language specifications based framework.

Defining and Deriving Structure. Returning to the distinction between the generative and interpretive context of data, the approach presented in this paper identifies two separate tasks of data modeling:

- The *description task*, in which domain experts are enabled to describe data in terms of the language specifications—i.e. to express the information, which is required to describe data (the syntax and semantics) in terms of a technology-agnostic notation: context free grammars (CFG).
- The *transformation task* then allows researchers with a knowledge of particular research questions to express data transformation statements, which can be based on the enriched semantic representation that resulted from the description task.

The following figure shows the overall concept of the transformation rule framework, which forms a combination of the application of domain-specific languages (DSL) as the descriptive task and a transformation language for the formulation of transformation rules as the transformation task.

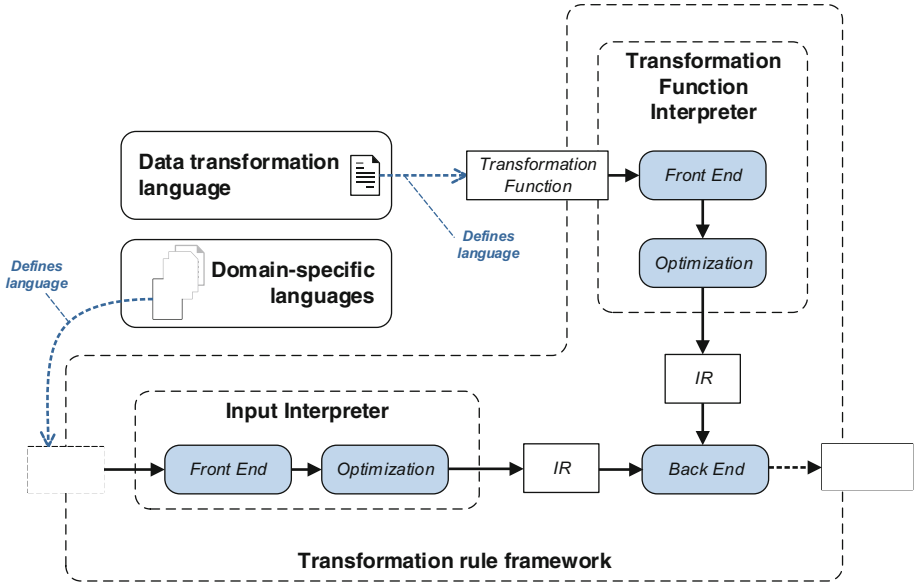

Fig. 1. Overview of the transformation rule framework

The building blocks within the framework are consolidated in two interpreters, one for the validation and processing of input against a DSL and one for the validation and processing of a transformation function against the transformation language. Both are conceptualized as autonomous language application components accomplishing the tasks of input processing, the construction of intermediate representations, their traversal, optimization and transformation [10].

In contrast to the typical structure of compilers, the back end of the rule framework needs to combine two parse trees, one with encoded transformation instructions and the other with a decomposed and structured version of the input. The back end finally concludes individual transformation tasks by:

- *interpreting input* executing the encoded transformation instructions and
- *generating the output* as output tree that forms a representation of the input in a target structure (mapping function) or that is included under the input element in the same structure (labeling function).

4 Data Description and Integration

In order to detail the principles behind our concept and the actual behavior of the developed framework, we present a exemplary metadata record that conforms to simple Dublin Core (DC) [1]. After modeling and extending the structure of the presented record in Sect. 4.1, we will model the correlation with another schema in Sect. 4.2 illustrate the idea of data integration in terms of our concept. Please note that the very simple example data presented in this paper is deliberately chosen in order to be able to focus on the theoretical background. Actual case studies are performed when writing this paper and will be published as applicable.

4.1 Modeling Schemata

The record in Listing 1 is available through the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH)[1] interface of Pangaea, a data publisher for earth and environmental science[2] and serves us to illustrate

- problems that can arise, when simple, interoperability-oriented standards are chosen to represent complex data or metadata, and
- how the explication of language specifications facilitates the tasks of data description and transformation.

[1] http://www.openarchives.org/pmh/.
[2] http://ws.pangaea.de/oai/?verb=GetRecord&metadataPrefix=oai_dc&identifier=oai:pangaea.de:doi:10.1594/PANGAEA.50542.

```
<oai_dc:dc xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:xsi="http
    ://www.w3.org/2001/XMLSchema-instance" xmlns:oai_dc="http://www.
    openarchives.org/OAI/2.0/oai_dc/"      xsi:schemaLocation="http://
    www.openarchives.org/OAI/2.0/oai_dc/ http://www.openarchives.org/
    OAI/2.0/oai_dc.xsd">
    <dc:title>Ice rafted debris (&gt; 2 mm gravel) distribution in
        sediment core PS2646-5</dc:title>
    <dc:creator>Grobe, Hannes</dc:creator>
    <dc:source>Alfred Wegener Institute, Helmholtz Center for Polar
        and Marine Research, Bremerhaven</dc:source>
    <dc:publisher>PANGAEA</dc:publisher>
    <dc:date>1996-02-29</dc:date>
    <dc:type>Dataset</dc:type>
    <dc:format>text/tab-separated-values, 1148 data points</dc:format>
    <dc:identifier>http://doi.pangaea.de/10.1594/PANGAEA.50542
        </dc:identifier>
    <dc:identifier>doi:10.1594/PANGAEA.50542</dc:identifier>
    <dc:language>en</dc:language>
    <dc:rights>CC-BY: Creative Commons Attribution 3.0 Unported</dc:
        rights>
    <dc:rights>Access constraints: unrestricted</dc:rights>
    <dc:coverage>LATITUDE: 68.556667 * LONGITUDE: -21.210000 * DATE/
        TIME START: 1994-09-19T14:56:00 * DATE/TIME END: 1994-09-19
        T14:56:00 * MINIMUM DEPTH, sediment/rock: 0.0 m * MAXIMUM
        DEPTH, sediment/rock: 11.5 m</dc:coverage>
    <dc:subject>ARK-X/2; AWI_Paleo; Denmark Strait; Gravity corer (
        Kiel type); Ice rafted debris; IRD-Counting (Grobe, 1987);
        Paleoenvironmental Reconstructions from Marine Sediments @
        AWI; Polarstern; PS2646-5; PS31; PS31/162</dc:subject>
</oai_dc:dc>
```

Listing 1. Pangaea DC example

The document contains rather obvious weaknesses with respect to further data processing or integration. In the following, we focus on three particular issues that we resolve though modeling. Although an analysis of other records is required to specify data—for the purpose of simplicity, we assume the presented example as representative for the entire collection:

- The `dc:creator` element contains the full name of a person, which we separate into a last and first name.
- The `dc:coverage` element is composed of a substructure, whose explication results in a decomposition of the encapsulated data.
- The `dc:subject` element actually contains a list of subjects, which is split into individual elements.

As an initial step, both the structure required to parse data (the Extensible Markup Language (XML) schema) and the extensions to that schema are modeled. In order to provide a convenient interface for experts within the arts and humanities, we provide a web-based schema editor for this task as shown in Fig. 2. Within the displayed element model, blue nodes represent original elements of the XML schema, yellow nodes symbolize grammars (`g: ...`) and functions (`f: ...`), which produce the purple colored labels of the schema. On the left side of the screen, editor sessions can be managed and part of the result

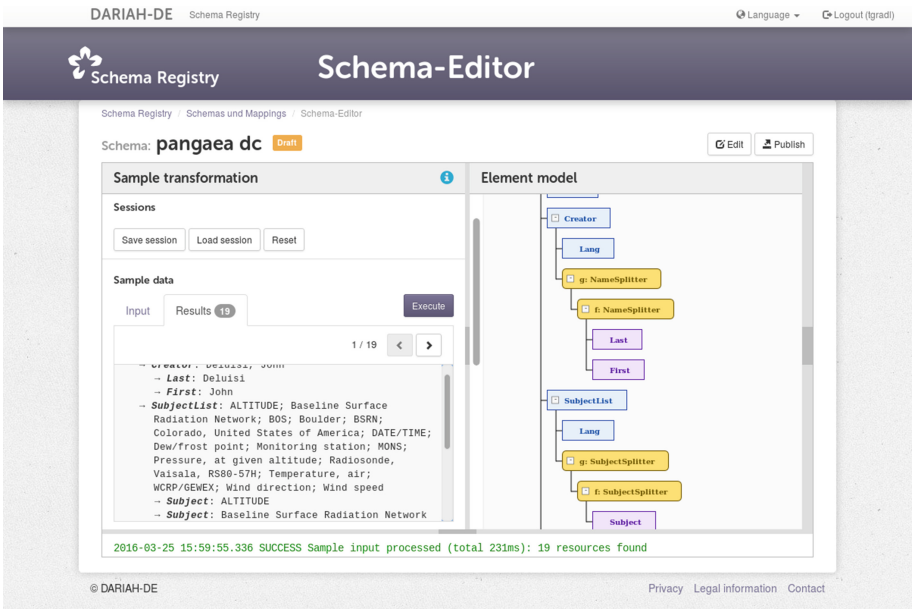

Fig. 2. User interface of the schema editor

of our example model is shown: the creator name has been decomposed as well as the subject list.

In order to fill subordinate labels, CFGs are formalized. In the cases of the `Creator` and `SubjectList`[3] elements, these grammars define a very simplistic language as shown in Listings 2 and 3. Both grammars define one rule for lexical analysis, which results in tokens separated by the specified character (',' and ';'). The original elements correspond to the `name` and `subjectlist` entry parser rules, which each produce the specified subsequent rules.

```
name: lname ', ' fname;

lname: STRING;
fname: STRING;

STRING: ~(',')+;
```

Listing 2. `Creator` grammar

```
subjectlist: subject ('; ' subject)*;

subject: STRING;

STRING: ~(';')+;
```

Listing 3. `Subject` grammar

One possible specification of the content within the *Coverage* element results in the grammar shown in Listing 4. Multiple lexer rules are defining types of

[3] The nonterminal element has been renamed to represent its content, the associated terminal references `dc:subject`—keeping the parsing-oriented structure intact.

tokens: IDs[4], DATE and SEPARATOR. The components of DATE are specified as lexical fragments, which is not strictly necessary in this case, but simplifies the parser grammar. The WS lexer rule specifies any tokens that are composed solely of whitespaces not to be passed to the parser. The parser rules then detail the logical composition of the content in Coverage. For further reference on such context free grammars we refer to guides on the Extended Backus-Naur Form (EBNF) or the ANTLR framework that we use to derive executable lexer and parser Java code from the grammatical specifications [10,11].

```
subelem     :   (longitude | latitude | start | end | minDepth | maxDepth |
                            otherElem) SEPARATORé;

longitude   :   'LONGITUDE' ': ' value;
latitude    :   'LATITUDE' ': ' value;
start       :   'DATE/TIME START' ': ' value;
end         :   'DATE/TIME END' ': ' value;
minDepth    :   'MINIMUM DEPTH, sediment/rock' ': ' value;
maxDepth    :   'MAXIMUM DEPTH, sediment/rock' ': ' value;

otherElem   :   key ': ' value;

key         :   ID;
value       :   DATE
            |   ID;

ID          :   ~(' '|':'|'*') ~(':'|'*')+ ~(' '|':'|'*');
DATE        :   YEAR '-' MONTH '-' DAY 'T' HOUR ':' MIN ':' SEC;
SEPARATOR   :   ' 'é '*' ' 'é;

fragment    YEAR    :   [1-2][0-9][0-9][0-9];
fragment    MONTH   :   [0-1][0-9];
fragment    DAY     :   [0-3][0-9];
fragment    HOUR    :   [0-2][0-9];
fragment    MIN     :   [0-6][0-9];
fragment    SEC     :   [0-6][0-9];

WS  :   [ \t\r\n]+ -> skip ;
```

Listing 4. Grammar for the Coverage element

Part of the parse tree that results from the interpretation of the dc:coverage content in the example document is presented in Fig. 3—with intermediary nodes representing the applied *parser rules* and leaf nodes the *lexer rules*.

As discussed, grammars defines language constraints to which the content of a particular element (*nonterminal* or *label* with respect to the definition above) has to conform. The parser and lexer classes that ANTLR generates from provided grammars do not fail on any erroneous input, but for cases, where ambiguities are raised. Our modelling interface shows any data interpretation error or warning directly within produced parse trees.

In order to finally transform content and assign output to labels, transformation functions are specified, which relate to parser rules. In Listing 5 nodes of

[4] As pendant to STRING in the Creator and SubjectList grammars.

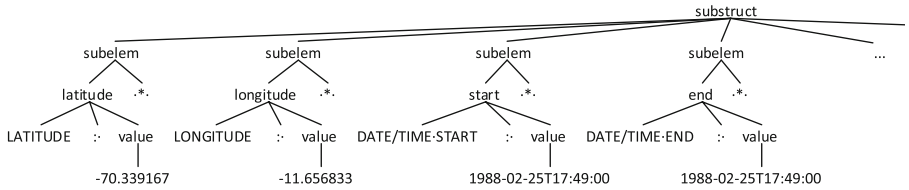

Fig. 3. Example parse tree of the `Coverage` grammar

the parse tree (`@latitude.value` and `@longitude.value`) are selected and (in lines 1 and 2) directly assigned to produced labels or used in commands (line 3). Such commands provide extension points of the framework and currently—based on actual use cases—include logical, string and arithmetic commands in the *Core* commands package. Othfronter packages wrap access to natural language processing (NLP) libraries, Wiki markup related functionality and geotemporal code. The specific command in line 3 of the listing concatenates the value of longitude, latitude and 0 to provide a label, which—continuing the example—is needed for mapping and thus data integration.

```
Lat = @latitude.value;
Long = @longitude.value;
LatLng = CORE::CONCAT(@longitude.value, "," , @latitude.value, ",0");
```

Listing 5. Transformation for the `Coverage` element

4.2 Modeling Mappings

The research around this paper is associated with the Digital Research Infrastructure for the Arts and Humanities (DARIAH-DE). A component that has been developed in DARIAH-DE consists in the Geo-Browser[5], a tool that visualizes spatial and/or temporal elements of data and allows the upload and analysis of data specified in terms of the Keyhole Markup Language (KML)[6]. Listing 6 shows an exemplary KML document.

```
<kml xmlns="http://www.opengis.net/kml/2.2">
    <Placemark>
        <name>Bamberg</name>
        <description>University of Bamberg</description>
        <Point>
            <coordinates>10.869461,49.902974,0</coordinates>
        </Point>
    </Placemark>
</kml>
```

Listing 6. KML placemark example

[5] http://geobrowser.de.dariah.eu/.
[6] https://developers.google.com/kml/.

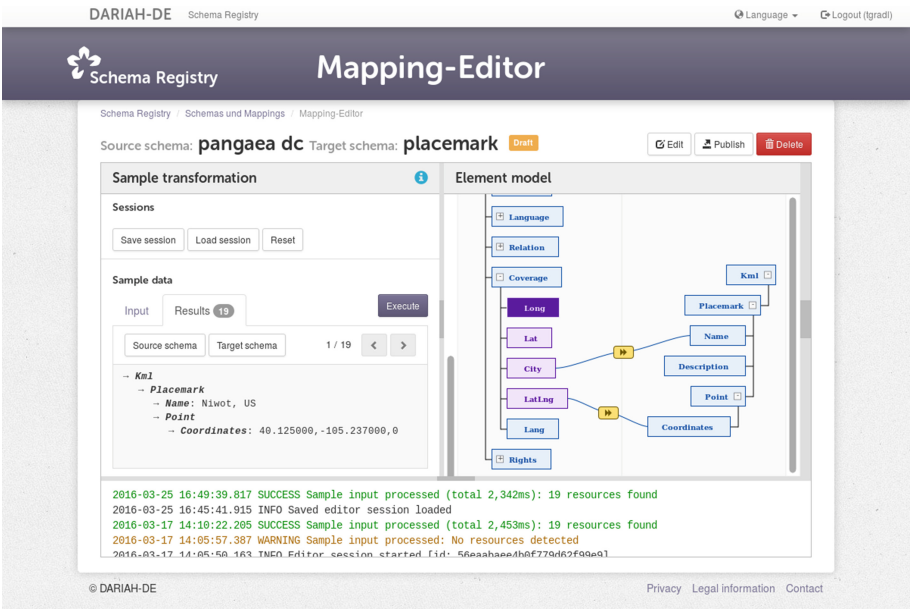

Fig. 4. User interface of the mapping editor

Based on the extracted latitude and longitude from the Pangaea record above—and especially the prepared `LatLng` label—the mapping between the two schemata can be modeled as shown in Fig. 4. In the screenshot, the element model for the assigned schemata is shown on the right side. On the left the user again finds access to the session management as well as sample transformation controls. Data that conforms to the modeled Pangaea schema has been uploaded and executed (1) against its schema definition to produce enriched content and (2) against the mapping specification—the result of the latter being shown in Fig. 4.

To fill the title of the KML document, we chose to utilize an example of the *GeoTemp* extension of our framework, which—in this particular case—is called to determine the nearest populated place based on the provided coordinates and GeoNames[7] data.

This reverse lookup deliberately uses an external, web-based interface and introduces latency. For this reason, the command is not executed in terms of the element mapping, but instead resulted in the definition of another element of the Pangaea structure. This results in the execution of the functionality when processing and indexing data, not when ad-hoc transforming data into another structure. Listing 7 shows the extended transformation function of the `Coverage` element.

[7] http://www.geonames.org/.

```
Lat = @latitude.value; Long = @longitude.value; LatLng =
CORE::CONCAT("[", @longitude.value, "," , @latitude.value, "]");
City = GEO::SIMPLEREVERSE(@latitude.value, @longitude.value);
```

Listing 7. Transformation for the `Coverage` element

5 Conclusion

Based on grammatical descriptions and transformation functions, this paper presented a concept and framework that facilitates the domain-specific description of the language of data and the subsequent formulation of transformation functions. Based on the generic implementation of reoccurring technical problems such as data decoding and XML parsing, we expect domain experts to be able to focus on integrative tasks, which require their expertise. Applications that are currently being built and tested on the base of our framework promise applicability of the concept to real-world data problems of the arts and humanities. We are eager to find more use cases to test and extend the limits of our approach. The presented front end to our framework is currently being developed and available at http://schereg.de.dariah.eu.

References

1. Dublin Core Metadata Element Set, Version 1.1 (2012). http://dublincore.org/documents/dces/
2. Batini, C., Lenzerini, M., Navathe, S.B.: A comparative analysis of methodologies for database schema integration. ACM Comput. Surv. **18**(4), 323–364 (1986)
3. Cooper, K.D., Torczon, L.: Engineering a Compiler, 2nd edn. Elsevier Morgan Kaufmann, Amsterdam (2012)
4. Gradl, T.: Concept and implementation of a rule framework to dynamically transform data and queries for heterogeneous collections. Master thesis, University of Bamberg, Bamberg (02-07-2014)
5. Gradl, T., Henrich, A.: A novel approach for a reusable federation of research data within the arts and humanities. In: Digital Humanities 2014 - Book of Abstracts, pp. 382–384, Lausanne, Switzerland (2014). http://dharchive.org/paper/DH2014/Paper-779.xml
6. Hagedorn, K.: OAister: a no dead ends OAI service provider. Libr. Hi Tech **21**(2), 170–181 (2003)
7. Hull, R.: Managing semantic heterogeneity in databases. In: Mendelzon, A., Özsoyoglu, Z.M. (eds.) Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium, PODS 1997, pp. 51–61 (1997)
8. Lenzerini, M.: Data integration: a theoretical perspective. In: Abiteboul, S. (ed.) Proceedings of the Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, p. 233. ACM, New York (2002)
9. Murata, M., Lee, D., Mani, M., Kawaguchi, K.: Taxonomy of XML schema languages using formal language theory. ACM Trans. Internet Technol. **5**(4), 660–704 (2005)

10. Parr, T.: Language implementation patterns: create your own domain-specific and general programming languages. The Pragmatic Programmers, The Pragmatic Bookshelf, Raleigh, NC, P3.0 Printing, Version: 2011-7-13 edn. (2011)
11. Parr, T.: The Definitive ANTLR 4 Reference. The Pragmatic Programmers, 2nd edn. Pragmatic Bookshelf, Dallas and Raleigh (2012)
12. Peroni, S., Tomasi, F., Vitali, F.: Reflecting on the Europeana data model. In: Agosti, M., Esposito, F., Ferilli, S., Ferro, N. (eds.) IRCDL 2012. CCIS, vol. 354, pp. 228–240. Springer, Heidelberg (2013)
13. Pouyllau, S.: ISIDORE: acces to open data of arts & humanities (2011). http://de.slideshare.net/stephanepouyllau/prsentation-gnraleisidore-generalenv
14. Sheth, A.P., Larson, J.A.: Federated database systems for managing distributed, heterogeneous, and autonomous databases. ACM Comput. Surv. **22**(3), 183–236 (1990)
15. Zhang, Z., Shi, P., Che, H., Gu, J.: An algebraic framework for schema matching. Informatica **19**(3), 421–446 (2008). http://dl.acm.org/citation.cfm?id=1454341.1454348