Maximilian Eibl, Martin Gaedke. (Hrsg.): INFORMATIK 2017,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 1125

Explicating knowledge on data models through domain specific languages

Tobias Gradl[1], Andreas Henrich[2]

Abstract: Digital artifacts and metadata of the arts and humanities exhibit a wide spectrum of formats, structures and contexts, and hence a high level of heterogeneity. With particular respect of the characteristics of the application domain, we propose a concept on the basis of formal languages, which allows the separation of technical and contextual aspects of data modeling. Based on a developed framework, domain experts explicate knowledge about data in terms of domain specific languages and derived transformation functions. Independent of actual technical aspects of data transformation and integration (e.g. formats, access protocols, schema languages), experts of particular disciplines, collections or research questions can describe and define data models and relations in an extensive, declarative fashion—utilizing custom data models or standards as applicable. As implicit knowledge is continuously explicated within the data models, interpretations external to the generative context of data are facilitated—thereby promoting interoperability.

Keywords: Digital humanities; descriptive data modeling; language applications; DARIAH

1 Introduction

Similar to the traditional forms of museums, archives or libraries, digital collections of the arts and humanities provide access to diverse contexts and forms of research objects and metadata. On the basis of tolerant licenses and sufficient technical infrastructures, digital resources can be provided to a greater, distributed public.

Despite current trends towards the development of standards and best practices for digitizing, describing and annotating research data of the arts and humanities, recent studies show a hesitant or practically non-existent adaption of standards other than that of simple Dublin Core (DC) [Po05, Vi13]. With an increasing impact of research infrastructures such as Europeana, DARIAH and CLARIN, existing standards might be favored over the definition of custom data models in the future. However, a continuing use and publication of data that conforms to custom or legacy data models should be expected as:

- funding and knowledge required to recreate data and metadata might not be available

[1] University of Bamberg, Media Informatics Group, An der Weberei, 96047 Bamberg, Germany tobias.gradl@uni-bamberg.de
[2] University of Bamberg, Media Informatics Group, An der Weberei, 96047 Bamberg, Germany andreas.henrich@uni-bamberg.de

doi:10.18420/in2017_114

- information loss might be feared when transforming existing data and
- standards that exactly match the demands might not exist.

In this paper we present an approach for modeling and relating data, which enables experts within the digital humanities to declaratively describe data—irrespective of the degree of structuredness and standardization. The novelty of the concept mainly consists in the language theoretical foundation [GH16a] of this modeling task, which results in data description facilities that are (1) expressive enough to incorporate the complex models required for scholarly research and (2) reduce technical overhead—allowing domain experts to focus on the semantic aspects of data modeling.

This paper is structured as follows: In section 2 we will introduce two examples that illustrate the types of data our approach is focusing on. Section 3 will present a formal foundation of data models into which *labeling functions* are incorporated. After detailing the composition of such functions and providing an overview of the behavior of the implemented framework, we conclude the paper with section 4 and a brief reflection of a research-oriented application that has been implemented on the basis of our framework.

2 Context

Traditional data modeling and integration backgrounds (see e.g. [Le02], [SL90], [BLN86]) often involve an extensive data analysis, which results in the specification of a global view—combining all information on local models that is considered relevant to a defined application context. The primary problem with such traditional approaches in domains of the Digital Humanities consists in the context-specificity of individual subdomains. Despite working on the same collection, art historians might come to different, possibly conflicting perspectives than archaeologists or epigraphs. Since research data of the Digital Humanities include uncountable nuances of structured, semi-structured and unstructured variants, the consolidation of one particular global model thus seems unfeasible—especially when attempting to satisfy the desirable objectives of completeness, correctness, minimality and understandability of integrative schemata [BLN86].

The primary goal behind our approach is to provide such capabilities that allow the description of data within its context by domain experts of the arts and humanities. By explicating contextual knowledge, they are enabled to extend and enrich original data. Due to the diversity of the domain, its research questions and data, we developed a concept that abstracts semantic aspects of data modeling from technical problems [GH14]—allowing experts within a specific discipline or collection to focus on those aspects of data, which require their domain expertise. Technical problems of data access, decoding, processing or integration are solved in a generic, reusable fashion.

For the sake of understandability, two rather generic examples have been selected for this paper. Although no specific disciplinary knowledge is required, they show the typical stretch

```
<oai_dc:dc>
  <dc:creator>Grobe, Hannes</dc:creator>
  <dc:date>1996-02-29</dc:date>
  <dc:format>text/tab-separated-values, 1148 data points</dc:format>
  <dc:language>en</dc:language>
  <dc:coverage>LATITUDE: 68.556667 * LONGITUDE: -21.210000 * DATE/TIME START:
      1994-09-19T14:56:00 * DATE/TIME END: 1994-09-19T14:56:00 * 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>
```

List. 1: Pangaea DC example

of data with respect to syntactical and structural complexity. Assuming e.g. codes instead of natural labels for keys under `dc:coverage` in the first example or a metalanguage other than Wikitext[3] in the second example: knowledge-dependent variants are not difficult to imagine.

Structured data As an example of data that is commonly provided by digital collections we consider simple DC metadata (see listing 1) as provided by the PANGAEA (Data Publisher for Earth & Environmental Science)[4] database. With respect to *atomicity*, the structural decomposition of each element is almost intuitive: the `creator` element consists of multiple name components, the `format` encapsulates some substructure, `coverage` contains key/value-organized data elements and `subject` actually contains a list of individual subjects. For each of these fields, simple lexical and syntactical rules can be defined, which generate the language of its data. For instance, in the `coverage` element, the colon and asterisk symbols are critical for tokenizing the otherwise unstructured text—differentiating the list of key/value pairs as well as the keys and values themselves.

Unstructured data Listing 2 shows the first few lines of the German Wikipedia article on Gustave le Bon. The document is composed of a mixture of structured and unstructured elements as the document itself conforms to the Extensible Markup Language (XML) and the MediaWiki export schema and the content of the *text* element conforms to Wikitext. As such, the unstructured text of the article is complemented with structural rules that define elements such as headings and links. Like the `coverage` element in listing 1, Wikitext follows grammatical constraints that can be utilized to define and describe conforming data.

[3] `https://en.wikipedia.org/wiki/Help:Wiki_markup`

[4] `https://pangaea.de/`

```
<page xmlns="http://www.medi...export-0.9/">
  <title>Gustave Le Bon</title>
  <ns>0</ns>
  <id>104619</id>
  <revision>
    <id>135522322</id>
    <parentid>135491542</parentid>
    <timestamp>2014-11-04T17:40:26Z</timestamp>
    <contributor>
      <ip>146.52.78.48</ip>
    </contributor>
    <comment>/* Der Rassebegriff bei Le Bon */</comment>
    <text xml:space="preserve">[[Datei:Gustave Le Bon.jpg|thumb|Gustave Le Bon im [[
        fin de siècle]]]] '''Gustave Le Bon''' (* [[7. Mai]] [[1841]] in [[Nogent-le
        -Rotrou]]; + [[15. Dezember]] [[1931]] in [[Paris]]) gilt als Begründer der
        [[Massenpsychologie]]. Seine Wirkung auf die Nachwelt, wissenschaftlich auf
        [[Sigmund Freud]] und [[Max Weber]], politisch insbesondere auf den [[
        Nationalsozialismus]] und seine ...
```

List. 2: Wikipedia example

3 Data modeling

Before diving further into the examples introduced in the previous section and the definition of grammatical constraints, this section is intended to provide a more formal, language-oriented perspective on data. Please note that although two XML-based examples have been chosen above, data in the digital humanities is not strictly structured in terms of the XML. Among further formats, plain text, Comma Separated Values (CSV) or JavaScript Object Notation (JSON) files are commonly utilized in the digital humanities—not in publicly accessible digital collections, but the locally stored research data of scholars and research projects.

3.1 Perspectives on data

Based on the foundation in [Zh08] and [Mu05], semi-structured data models can be interpreted in terms of finite structures $\langle N, T, R, P \rangle$—regular-tree grammars with the finite sets of nonterminals (N) and terminals (T), the root symbol ($R \in N$) and the set of production rules (P). This definition allows the production of a semi-structured document according to its e.g. XML or JSON schema formulated constraints. As such, we consider the interpretation of a schema as $\langle N, T, R, P \rangle$ the *parsing-oriented perspective* on semi-structured data. The definition allows the specification of 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.

With regard to the introduced examples, this definition allows the production of terminal symbols of the XML documents from nonterminals and hence the formal definition of

an XML schema. Although a strictly parsing-oriented perspective is necessary for the syntactically correct interpretation of data, the presented definition does not allow the extension of document structure based on the content of the elements.

To allow the representation of substructures or alternative elements within the formal definition of a schema, we extend the parsing-oriented view to a 6-tuple $S = \langle N, T, R, P, L, F \rangle$, where N, T, R and P form components of the original definition. The components of L and F provide the extension 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 $|I|$.

Figure 1 shows the editor component of the modeling interface of our framework[5]. In this editor, the blue nodes represent the nonterminals. The yellow nodes are labeling functions, which are formed of a grammatical and a transformation component. The purple nodes finally represent produced labels. Through the hierarchy, parenting nodes of labeling functions also define the set of input values.

3.2 Labeling functions

Within our framework, labeling functions are composed of the two constructs *grammars* and *functions*, which represent two distinguishable modeling tasks. Grammars are used to define the grammatical constraints that generate a language—i.e. the Domain Specific Language (DSL) that an element conforms to. Functions build on resulting syntax trees to transform data into subsequent labels.

The example in figure 1 shows an intermediary result of modeling the presented Wikipedia example: the *grammar* `g: SplitContent` for the separation of encapsulated structured information from unstructured text is inserted below the `Text` element. The *transformation function* `f: SplitContent` applies commands to produce key/value pairs of structured data and the remaining article (`Fulltext`). An additional grammar `g: WikipediaSections` then decomposes Wikitext from encapsulated textual content. The transformation function `f: WikipediaSections` is then applied on the produced parse tree and generates the element hierarchy modeled under `Section`.

[5] `http://schereg.de.dariah.eu`

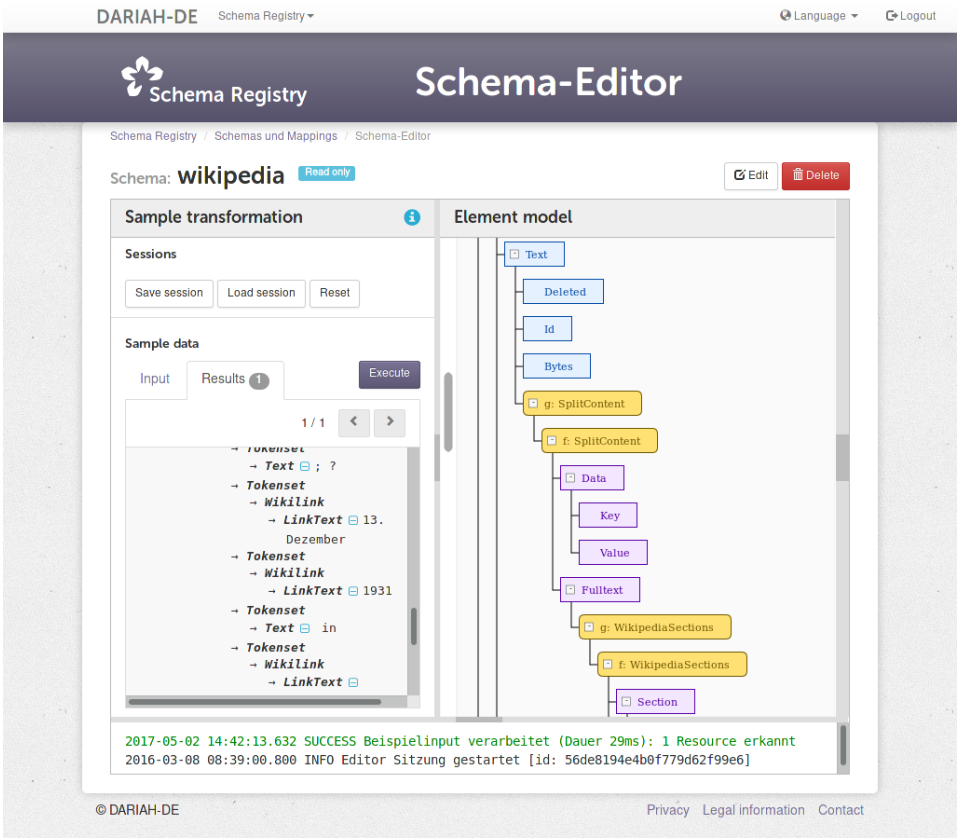

Fig. 1: Part of data model and transformation of the wikipedia example

Description grammars

The specification of data with respect to syntactical and semantic constraints is accomplished by means of an individual, element-specific DSL [PF11]. Listing 3 shows an excerpt of the *parser grammar* for `g: WikipediaSections`. The complete *parser grammar* has 74 lines (including comments etc.) and defines rules (e.g. `page`, `h1block`, `text`) for the syntactic analysis of a provided input. It is complemented by a *lexer grammar* of 109 lines, which specifies the rules (e.g. `TEXT`, `EQ`, `H1_OPEN`) for a preliminary lexical analysis—i.e. the tokenization of input. Please consider the lexer rule `h1`: it defines that the `title` of a level 1 heading is embraced by the lexer rule `H1_OPEN` (defined as `'\n='`) and `EQ` (`'='`)—the latter being also used in parser rules other than `h1` such as `text` in the presented excerpt.

```
page      : container+;
container : (block | preface) block*;
preface   : content;
block     : h1block | ... | h6block;

h1block   : h1  (h2block | ... | h6block | content)*;
...
h6block   : h6  (content)*;

h1        : H1_OPEN title EQ;
...
h6        : H6_OPEN title H6_CLOSE;

content   : (tokenset | categoryContainer)+;
title     : tokenset+;
tokenset  : text
          | link;
...
text      : TEXT | EXCL | EQ;

intLink     : intLinkOpen intLinkCont INT_LINK_CLOSE;
intLinkCont : intLinkComp? linkValue;
intLinkComp : intLinkRes INT_LINK_SEP
              (linkContent INT_LINK_SEP)*;
intLinkRes  : linkContent;
...
```

List. 3: Element grammar g: WikipediaSections

Applying the grammars at the modeled element conveniently results in the generation of Java code and classes that represent a lexer, a parser, tree traversal helpers and auxiliary classes per defined rule. Upon requiring the grammar during the runtime of our developed system, these classes are dynamically loaded and hence—without any intervention by a programmer—the declarative definition of data in terms of DSLs results in the execution of native Java code that processes the defined data. As an end-result of the descriptive phase, a parse-tree is generated and reflects input data with respect to the specified grammar. Figure 2 shows the resulting parse tree of the first tokens of the textual content in listing 2. In subsequent steps of the sample Wikipedia data model, file links have been modelled as

irrelevant for the task at hand, which is why the rule `skip` terminates the parse tree at that particular subtree.

Transformation functions

Comparable to its native context in compiler engineering, a parse tree reflects only an intermediate representation for the derivation of an enriched document and does not conclude our data enrichment process. Transformation functions form a subsequent step and are applied to parse trees.

Listing 4 reflects the transformation function in `f: WikipediaSections`. Opposed to the grammar, this particular example function is not generic, but influenced by a specific application context in which only biographical sections are considered relevant. Whereas the first statement assigns any preface of an article to appropriate labels, the second statement contains a (simplified) filtering statement that is only satisfied, if the heading of a top-level (h2 in Wikipedia) heading contains the word 'life'.

Multiple transformation languages have been designed and standardized such as Query/View/-Transformation (QVT), which is defined by the Object Management Group (OMG) [OM15]. Due to the refinement of input data by means of a grammatical specification, the complexity of such languages has not shown to be required in our case yet. Instead, we chose to define a simple transformation language that allows the specification of value and object assignments and command execution.

However, as further indicated in figure 3, the transformation language is yet another DSL,

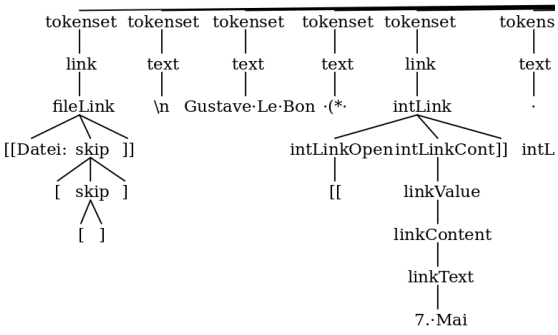

Fig. 2: Parse tree of the wikipedia sample

```
Section = @page.container.preface {
  Tokenset = @content.tokenset {
    Text = @[text, extLink.extLinkCont.linkValue];
    Wikilink = @intLink {
      LinkText = @linkValue;
      Link = @intLinkRes;
    };
  };
};
Section = @page.container.block.h2block[CORE::CONTAINS(@h2.title, "true", "life")] {
  Heading = @h2.title;
  Tokenset = @content.tokenset {
    Text = @text;
    Wikilink = @link {
      LinkText = @linkValue;
      Link = @intLinkRes;
    };
  };
};
AvailableHeading = @page.container.block.h2block.h2.title;
```

List. 4: Wikipedia function

which is technically processed comparable to the data languages. As such, other—possible additional—transformation languages could be integrated as deemed necessary.

3.3 Rule processing overview

Figure 3 shows the overall concept of the transformation rule framework, which forms a combination of the application of DSLs for the description of data and a transformation language for the formulation of transformation rules. The building blocks within the framework are summarized 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—each accomplishing the tasks of processing input, constructing intermediate representations and traversing and rewriting these according to the needs of the concluding back end (*semantic analysis* or *optimization*). These *needs* are the named nodes and subtrees of parsed input and are known as soon as the transformation function has been parsed and analyzed.

At runtime of the rule framework, transformation functions are assumed to be reused for multiple instances of a defined element model. As such, only the content of individual documents needs to be interpreted, whereas the executable representation of the transformation function is cached in its intermediary state.

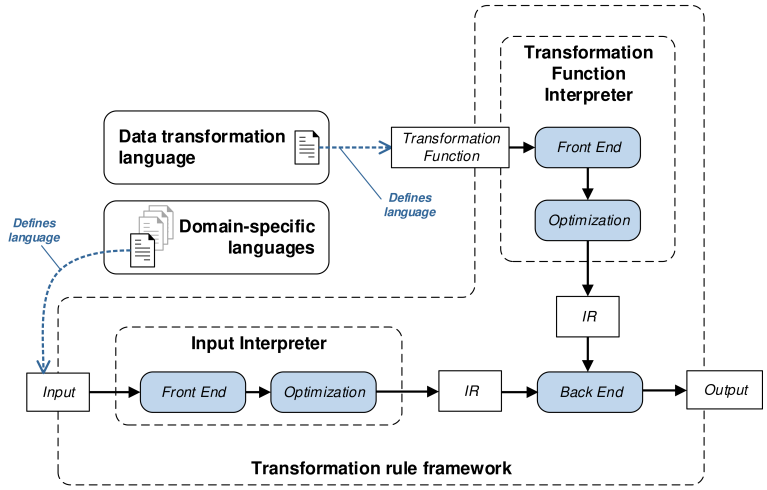

Fig. 3: Transformation rule framework

4 Conclusion

Although the example here touches all introduced topics, the results present only an intermediary step towards data that has been specifically modeled for a domain and research question. As subsequent steps, we have e.g. implemented functionality as usable commands for transformation functions to detect named entities by analyzing internal Wiki links or execute state-of-the-art methods of natural language processing (NLP)—to which the named entities that were identified by analyzing internal Wiki links can then be provided as known entities. With the help of the modeling framework we were able to compile an integrated data source for the prototypical implementation of a biographical analysis tool, which has been motivated by a feasibility study at the Leibniz Institute of European History in Mainz, Germany [GH16b]. Despite the functional extensions to support NLP or Wiki link analysis functionality, no code was required to be written in order to access, process and transform data. These aspects have been solved by modeling data (along with the application context) in a declarative fashion, which not only resulted in a quicker prototypical implementation of the tool, but improved transparency and iterative adaptability for the domain experts.

Although this paper has focused on the task of enriching data models intrinsically by explicating domain knowledge, executed tasks of data description and transformation can be applied when mapping data models to another, as shown exemplarily in figure 4. Mapping concepts between source and target elements[6] are visually represented by the yellow nodes and are comprised of a grammar per input element and one transformation function per mapping—reusing the same functionalities and principles as outlined above.

[6] 1:N, N:1 and N:M mappings are possible

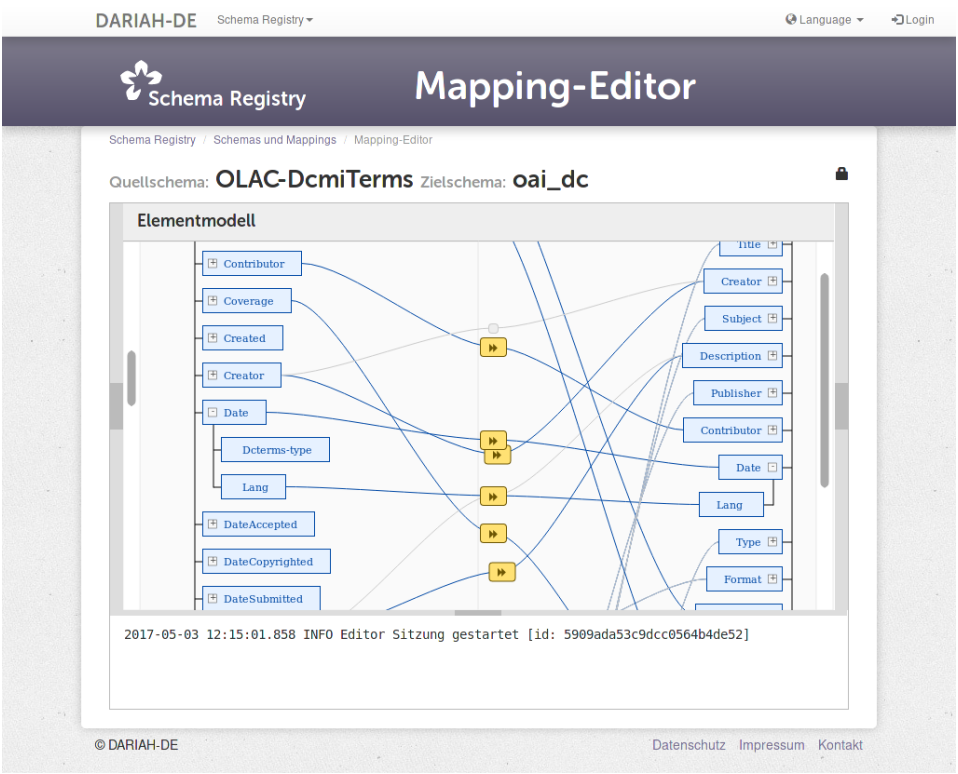

Fig. 4: Part of data model and transformation of the wikipedia example

References

[BLN86] Batini, C.; Lenzerini, M.; Navathe, S. B.: A comparative analysis of methodologies for database schema integration. ACM Computing Surveys, 18(4):323–364, 1986.

[GH14] Gradl, Tobias; Henrich, Andreas: A novel approach for a reusable federation of research data within the arts and humanities. In: Digital Humanities 2014 - Book of Abstracts. Lausanne, Switzerland, pp. 382–384, 2014.

[GH16a] Gradl, Tobias; Henrich, Andreas: Data Integration for the Arts and Humanities: A Language Theoretical Concept. In (Fuhr, Norbert; Kovács, László; Risse, Thomas; Nejdl, Wolfgang, eds): Research and Advanced Technology for Digital Libraries: 20th International Conference on Theory and Practice of Digital Libraries, TPDL 2016, Hannover, Germany, September 5–9, 2016, Proceedings, pp. 281–293. Springer International Publishing, Cham, 2016.

[GH16b] Gradl, Tobias; Henrich, Andreas: Nutzung und Kombination von Daten aus strukturierten und unstrukturierten Quellen zur Identifikation transnationaler Lebensläufe. In (Burr, Elisabeth, ed.): DHd 2016. nisaba Verlag, pp. 135–138, 2016.

[Le02] Lenzerini, Maurizio: Data Integration: A Theoretical Perspective. In (Abiteboul, S., ed.): Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. ACM, New York, NY, p. 233, 2002.

[Mu05] Murata, Makoto; Lee, Dongwon; Mani, Murali; Kawaguchi, Kohsuke: Taxonomy of XML schema languages using formal language theory. ACM Transactions on Internet Technology, 5(4):660–704, 2005.

[OM15] OMG: , Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification, 2015.

[PF11] Parr, Terence; Fisher, Kathleen: LL(*): The Foundation of the ANTLR Parser Generator. In: Proceedings of the 32Nd ACM SIGPLAN Conference on Programming Language Design and Implementation. PLDI '11, ACM, New York, NY, USA, pp. 425–436, 2011.

[Po05] Polfreman, Malcolm: , Commonly-used metadata formats in the Arts and Humanities, 2005.

[SL90] Sheth, Amit P.; Larson, James A.: Federated database systems for managing distributed, heterogeneous, and autonomous databases. ACM Computing Surveys, 22(3):183–236, 1990.

[Vi13] Vierkant, Paul: , Leuchttürme der deutschen Repositorienlandschaft, 2013.

[Zh08] Zhang, Zhi; Shi, Pengfei; Che, Haoyang; Gu, Jun: An Algebraic Framework for Schema Matching. Informatica, 19(3):421–446, 2008.