Band 22 – Heft 2 – Jahrgang 2007 – ISSN 0175-1336

Zeitschrift für Computerlinguistik und Sprachtechnologie
GLDV-Journal for Computational Linguistics and Language Technology

LDV Forum

Foundations of Ontologies in Text Technology

Herausgegeben von
Uwe Mönnich und Kai-Uwe Kühnberger

Gesellschaft für linguistische Datenverarbeitung www.gldv.org

Foundations of Ontologies in Text Technology

Impressum

LDV-Forum ISSN 0175-1336 Band 22 - 2007 - Heft 2	Zeitschrift für Computerlinguistik und Sprachtechnologie GLDV-Journal for Computational Linguistics and Language Technology – Offizielles Organ der GLDV
Herausgeber	Gesellschaft für Linguistische Datenverarbeitung e. V. (GLDV) Juniorprofessor Dr. Alexander Mehler, Universität Bielefeld, *alexander.mehler@uni-bielefeld.de* Prof. Dr. Christian Wolff, Universität Regensburg *christian.wolff@sprachlit.uni-regensburg.de*
Anschrift der Redaktion	Prof. Dr. Christian Wolff, Universität Regensburg Institut für Medien-, Informations- und Kulturwissenschaft D-93040 Regensburg
Wissenschaftlicher Beirat	Vorstand, Beirat und Arbeitskreisleiter der GLDV *http://www.gldv.org/cms/vorstand.php*, *http://www.gldv.org/cms/topics.php*
Erscheinungsweise	2 Hefte im Jahr, halbjährlich zum 31. Mai und 31. Oktober. Preprints und redaktionelle Planungen sind über die Website der GLDV einsehbar (*http://www.gldv.org*).
Einreichung von Beiträgen	Unaufgefordert eingesandte Fachbeiträge werden vor Veröffentlichung von mindestens zwei ReferentInnen begutachtet. Manuskripte sollten deshalb möglichst frühzeitig eingereicht werden und bei Annahme zur Veröffentlichung in jedem Fall elektronisch und zusätzlich auf Papier übermittelt werden. Die namentlich gezeichneten Beiträge geben ausschließlich die Meinung der AutorInnen wieder. Einreichungen sind an die Herausgeber zu übermitteln.
Bezugsbedingungen	Für Mitglieder der GLDV ist der Bezugspreis des LDV-Forums im Jahresbeitrag mit eingeschlossen. Jahresabonnements können zum Preis von 25,- € (inkl. Versand), Einzelexemplare zum Preis von 15,- € (zzgl. Versandkosten) bei der Redaktion bestellt werden.
Satz und Druck	Boris Kaiser, Bielefeld, mit *LaTeX* (*pdfeTeX* / *MiKTeX*) und *Adobe InDesign CS 3.0.1* , Druck: Druck TEAM KG, Regensburg

Uwe Mönnich, Kai-Uwe Kühnberger

Editorial

1 Introduction

The rise of the world-wide-web in connection with the tremendous increase of electronically available textual data of all kinds, types, genres, and forms make the scientific study of text resources a trend-setting research endeavor. The joint work of researchers trained in different disciplines and research traditions encompassing the theoretical study of properties of texts, text transformations, markup languages, query languages, and text structures, as well as the practical pursuit of archiving textual information, retrieving background knowledge from texts, and adapting dynamically ontological knowledge to new information can be considered as the birth of *text technology* as a scientific discipline.

It is not very astonishing that the triumphal elevation of hypertexts – powered by the success of the world-wide-web – as a data structure, did play an important role in making text technology a widely recognized scientific subject. Text technology as a scientific discipline has a rather short history, although its origins have their roots in classical academic research traditions like (computational) linguistics, computer science,artificial intelligence, literary sciences, and text sciences. Despite its recent emergenceas a coherent body of research text technology can easily be distinguished from theseneighboring areas and can claim to have become an autonomous discipline of its ownright. Text technology differs from classical computational linguistics and natural language processing in focusing on text as a means in itself, not on text as a container for language expressions (sentences) or text as a representation (or coding mechanism) of utterances. Moreover, text technology considers structures and layouts of texts contrary to literary history or classical linguistics and does not concentrate on finding generative principles for the question of what constitutes a text. Last but not least, it uses and develops markup standards (like XML, RDF, OWL etc.) and algorithms for statistical and symbolic computations on texts, very similarly to computer science and artificial intelligence in the area of the semantic web. But unlike computer science, text technology directs its attention on texts instead of data structures in general, therefore it attempts to structure data rather differently in comparison to, for example, the semantic web tradition, and combines ideas from structure transformations that are not at the center of interest of the computer science community.

As a research area that is crucially located between different disciplines, text technology is a strongly interdisciplinary research attempt. It combines research methodologies from the already mentioned disciplines like linguistics, computational linguistics, artificial intelligence, and computer science. Whereas the domain of interest remains in-the realm of the humanities, namely texts of all sorts and types, from a methodological point of view text technology is primarily concerned with formal sciences. With the emergence of new types of text (mainly based on the success of the world-wide-web), that are no longer linearly structured, but contain hyperlinks and multi-media data, new interaction paradigms and usability aspects are provoked and generate new needs for finding, retrieving, and editing information. Furthermore new techniques for archiving multi-modal linguistic knowledge need to be developed and implemented that have firmly established the importance of the interdisciplinary research endeavor *text technology.*

In comparison to text technology, the term *ontology* as it is used in technical disciplines has a rather different history. Picking up a term that has a history of more than two thousand years in philosophy, researchers in artificial intelligence introduced ontologies into their discipline as a means to represent conceptual background knowledge in expert systems (Brachman and Schmolze, 1985). During the further development it turned out, that many applications, most recently web applications, would strongly benefit from a sound basis on which semantic information can be coded (Daconta et al., 2003). It is a rather natural idea to integrate ontological knowledge into current text technological applications. The result is an enrichment of structural information: for example, taking annotation graphs as structural representation formalisms into account, adding ontological knowledge to annotation graphs enlarges the structural representation of text data by semantic knowledge.

The present volume is the first part of a double volume about “Ontologies in Text Technology” covering the theoretical basis of the topic. It contains a representative sample of cutting-edge work in the foundations of combining text technology and ontologies for state-of-the-art techniques of processing texts in language technology. Volume II entitled “Applications of Ontologies in Text Technology” will be published in January 2008 and will contain more applied work in the area of anaphora resolution, discourse parsing, and extracting synonymy relations and lexico-semantic classes from text.

The origins of this double volume go back to the workshop “Adaptive Ontologies on Syntactic Structures” held in conjunction with the 28th Annual Meeting of the *German Association of Linguistics* (DGfS) at the University of Bielefeld in February 2006. As a follow-up workshop, the editors organized an international workshop in Osnabrück in September 2006 entitled “Ontologies in Text Technology – Approaches to Extract Semantic Knowledge from Syntactic Information”. The proceedings of this workshop contain six page papers of the participants and were published in the PICS series (Publications of the Institute of Cognitive Science). Due to the fact that with Guus Schreiber and Klaus Schulz two distinguished keynote speakers, gave inspiring talks in this workshop, the workshop attracted many internationally well-known researchers working in text and language technology. Because of the great success of this workshop the idea was conceived to provide a possibility to present the results of this workshop to a broader audience. It was decided that the participants of the workshop should be invited to submit full and extended versions of their papers for a journal publication. After a thorough further reviewing process and a revision of the accepted full articles, the result is the present double volume of the *GLDV-Journal for Computational Linguistics and Language Technology.*

2 The Research Unit 437 *Text Technological Information Modeling*

During the last six years the development of text technology in Germany was strongly influenced by the research unit 437 "Text Technological Information Modeling" funded by the *German Research Foundation* (DFG). This research unit is an interdisciplinary research endeavor carried by the Universities of Bielefeld, Gießen, Dortmund, Tübingen, and Osnabrück. Starting in the year 2001, this group constitutes the largest collaborative research project devoted to text technological issues and has provided the basis for text technological research in Germany. Currently this research unit is in its final funding year. In order to get a better impression of the overall project, a concise overview of the involved sub-projects of the second phase of this research unit is given:

- *Secondary Structuring of Information and Comparative Analysis of Discourse.*
  Principal Investigator: Dieter Metzing.

- *Induction of Document Grammars for the Representation of Logical Hypertextual Document Structures.*
  Principal Investigator: Alexander Mehler.

- *Text-Grammatical Foundations for the (Semi-)Automated Text-to-Hypertext Conversion.*
  Principal Investigator: Angelika Storrer.

- *Generic Document Structures in Linearly Organized Texts: Text Parsing Using Domain Ontologies and Text Structure Ontologies.*
  Principal Investigator: Henning Lobin.

- *Adaptive Ontologies on Extreme Markup Structures.*
  Principal Investigators: Uwe Mönnich, Kai-Uwe Kühnberger.

Although the research unit tries to cover all aspects of current text technological activities, it is easily possible to identify certain core aspects that play a central role in all sub-projects. Examples for such vertical topics of the whole research unit are ontologies, annotations, markup standards, and processing aspects of texts. All these topics play an important role in all participating projects. Some aspects of these vertical topics of the research unit are also represented in this double volume of the GLDV-*Journal*. The present volume focuses on the foundations of theories for developing, characterizing, coding, learning, and adapting ontological background knowledge as a crucial challenge for the semantic annotation of text documents. Some of the sub-projects of the collaborative research unit mentioned above are represented in this volume. Others will document aspects of their work in Volume II "Applications of Ontologies in Text Technology". We think that we can provide by this not only a representative documentation of text technology in general, but also a representative collection illustrating the research unit 437 in particular.

3 The Structure of Volume I

This first volume “Foundations of Ontologies and Text Technology” contains articles concerned with the methodological basis of using ontologies in text technology. Two aspects need to be distinguished in this context: the syntactic aspect attempts to focus on the underlying languages and data structures used for coding technologically relevant information, as for example, markup standards like XML and annotation graphs as a means to code linguistic information. Complementary to the syntactic level, the semantic aspect deals with properties of ontological knowledge for text technological applications. Both topics include aspects of learning and adaptation: learning and adaptation of ontologies is a research field that is of great importance for the future, because hand-coded ontologies are tedious, time-consuming, and expensive to create (Perez and Mancho, 2003). But also on the syntactic side, there is the need for the development of learning mechanisms: learning text types based on structural information only, without any information about their content, turns out to be possible in many cases. In the following, we will summarize major aspects of the articles included in this volume.

Lexical-semantic networks like the well-known WordNet (Fellbaum, 1998) together with its versions in other languages like RussNet or GermaNet are not only a de facto standard for several applications in text technology, but can also be seen as prototypical examples where ontological knowledge can successfully be applied in text technology. In their article “Domain Ontologies and Wordnets in OWL: Modelling Options”, *Harald Lüngen* and *Angelika Storrer* question the common conversion standard to interpret synsets and lexical units of WordNet as OWL individuals. The article provides arguments for a different conceptual view, namely that synsets and lexical units need to be interpreted as concepts instead. Technically this results in a different modeling of codingWordNet ontologies in the OWL format. The authors base their claim on an evaluation of OWL representation models for WordNet variants like GermaNet combined with TermNet.

The second article of this volume “Automatic Ontology Extension: Resolving Inconsistencies” by *Ekaterina Ovchinnikova* and *Kai-Uwe Kühnberger* continues the discussion of ontologies by focusing on learning and adaptation aspects of ontologies against the background of new input. The work follows the tradition to represent ontological knowledge using description logic, therefore it considers ontology design from a logical perspective (Baader et al., 2003). Due to the fact that automatically generated and automatically updated ontologies face the problem of becoming inconsistent, the paper provides an automatic procedure for resolving occurring inconsistencies in ontology design. Potential inconsistencies in ontology design are restricted to logical ones, in particular, the overgeneralization of concepts and polysemy problems are discussed in detail. The authors propose an algorithmic solution for an automatic resolution based on the minimal non-conflicting substitute.

Related to the question of how to consistently extend ontologies by dynamic updates is the question of how the population of ontologies with existing data sources can be achieved. The article “Integration Languages for Data-Driven Approaches to Ontology

Population and Maintenance" by *Eduardo Torres Schumann*, *Uwe Mönnich*, and *Klaus Schulz* proposes a new integration language that is capable of generating new entries in large-scale ontologies based on structured data. By an intelligent user interface an efficient way to supervise the population of the ontology can be provided. The authors embed their work into a system that is able to encode large amounts of data like encyclopedic and common purpose knowledge: this large-scale knowledge base is called EFGT net (Schulz and Weigel, 2003), a precisely defined framework designed for various NLP applications.

Text technology is concerned with different types of text. Not only that different types of text have different content, often they differ significantly on a structural level as well. Concerning webpages one can distinguish, for example, homepages of scientists, blogs, or online stores from each other by structural features (Lindemann and Littig, 2006). The article "Structural Classifiers of Text Types: Towards a Novel Model of Text Representation" by *Alexander Mehler*, *Peter Geibel*, and *Olga Pustylnikov* discusses possibilities to learn text types solely on the basis of structural information without having any content information. The authors show that the document object model (DOM) can be used, in order to code structural information of texts. The authors propose different learning mechanisms for achieving this task like quantitative structure analysis (QSA) and several variants of tree kernels. The article adds also an evaluation of these learning algorithms based on a large newspaper corpus.

Graph structures play an important role in coding linguistic and textual information. Prominent examples are annotation graphs, which are used to represent multi-layered information about language, like phonological, grammatical, semantical, and pragmatical information, as well as non-linguistic information (gestures or cultural background). On the other hand, tree structures can be used in order to analyze text types. The article "Towards a Logical Description of Trees in Annotation Graphs" by *Jens Michaelis* and *Uwe Mönnich* focuses on logical descriptions of annotation graphs, one of the major data resources for text technological applications. The authors present results for characterizing a large class of annotation trees, namely, single time line, multiple tiers (STMT) models, which constitute a subclass of annotation graphs in the sense of Bird and Liberman (2001), and from which multi-rooted trees can be constructed. Besides other technical results, the article provides also a spelled-out algorithm for tree-like graph transduction from a given STMT model into a multi-rooted tree. The result is a uniform and mathematically rigorous format for the syntactic representation of annotation graphs. Taking into account that multi-rooted trees and Bird-Liberman annotation graphs play a prominent role in archiving and coding texts, this work can be considered as a theoretical basis for annotation tasks in general.

4 Acknowledgments

This volume would not have been possible without the help of many people. Most importantly the guest editors want to thank the editors-in-chief of the GLDV-*Journal for Computational Linguistics and Language Technology*, Alexander Mehler and Christian Wolff. Their encouragement and support in all phases of the emergence of this double

volume has been irreplaceable in completing it. Furthermore we want to thank the German Research Foundation for financial support of the research unit 437 "Text Technological Information Modeling" and particularly the speaker of this research unit, Dieter Metzing.

Last but not least, the editors want to thank the program committee for their careful evaluations of the submitted papers. The quality of this volume is also a direct consequence of the work these reviewers invested. The program committee consisted of the following researchers (in alphabetical order): Irene Cramer, Thierry Declerck, Stefan Evert, Pascal Hitzler, Wolfgang Höppner, Helmar Gust, Marcus Kracht, Edda Leopold, Alessandro Moschitti, Larry Moss, Rainer Osswald, Olga Pustylnikov, Georg Rehm, Hans-Christian Schmitz, Bernhard Schröder, Uta Seewald-Heeg, Manfred Stede, Markus Stuptner, Frank Teuteberg, Yannick Versley, Johanna Völker, Armin Wegner, and Christian Wolff.

References

Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., and Patel-Schneider, P. F., editors (2003). *The Description Logic Handbook: Theory, Implementation, and Applications.* Cambridge University Press, New York.

Bird, S. and Liberman, M. (2001). *A Formal Framework for Linguistic Annotation.* Speech Communication, (33):23–60.

Brachman, R. and Schmolze, J. (1985). *An Overview of the KL-ONE Knowledge Representation System.* Cognitive Science, 9:171–216.

Daconta, M., Obrst, L., and Smith, K. (2003). *The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management.* John Wiley and Sons.

Fellbaum, C., editor (1998). *WordNet. An Electronic Lexical Database.* MIT Press.

Lindemann, C. and Littig, L. (2006). *Coarse-Grained Classification of Web Sites by their Structural Properties.* In Proc. of WIDM'06, pages 35–42.

Perez, G. A. and Mancho, M. D. (2003). *A Survey of Ontology Learning Methods and Techniques.* OntoWeb Delieverable 1.5.

Schulz, K. and Weigel, F. (2003). *Systematics and Architectures for a Resource Representing Knowledge about Named Entities.* In Proceedings Workshop on Principles and Practice of Semantic Web Reasoning, pages 189–207.

Inhaltsverzeichnis

LDV FORUM - Band 22(2) - 2007
Foundations of Ontologies in Text Technology

Harald Lüngen, Angelika Storrer

Domain ontologies and wordnets in OWL: Modelling options

1 Project framework and goals

Wordnets are lexical reference systems that follow the design principles of the Princeton WordNet project[1] (Fellbaum, 1998). Domain ontologies (or domain-specific ontologies such as GOLD[2], or the GENE Ontology[3]) represent knowledge about a specific domain in a format that supports automated reasoning about the objects in that domain and the relations between them (Erdmann, 2001). In this paper, we will discuss how the Web Ontology Language OWL can be used to represent and interrelate the entities and relations in both types of resources. Our special focus will be on the question, whether synsets should be modelled as individuals (we use *individual* and *instance* as synonyms and will refer to this option as *instance model*) or as classes (we will refer to this option as *class model*). We will present three OWL models, each of which offers different solutions to this question. These models were developed in the context of the research group "Text-technological Modelling of Information"[4] as a collaboration of the projects SemDok and HyTex. Since these projects are mainly concerned with German documents and with corpora that contain documents of a special technical or scientific domain, we used subsets of the German wordnet GermaNet (Kunze and Lemnitzer, 2002), henceforth referred to as GN, and the German domain ontology TermNet (Beißwenger et al., 2004), henceforth referred to as TN, to develop and evaluate the three models. To relate the general vocabulary of GN with the domain specific terms in TN, we developed an approach that was inspired by the plug-in model proposed by Magnini and Speranza (2002). In this approach, which has been developed in cooperation with the GermaNet research group (see Kunze et al. (2007) for details), we adapted the OWL model for the English Princeton WordNet suggested by van Assem et al. (2006) to GN, i.e. we modelled German synsets as instances of word-class-specific synset classes. For the reasons explained in section 3, we wanted to experiment with alternative models that implement the class model. In section 4 we will present three alternative OWL representations for GN and TN and discuss their benefits and drawbacks.

2 Basic entities and relations in GermaNet and TermNet

Wordnets and domain ontologies have been used in various applications of text processing (cf. Fellbaum, 1998; Kunze et al., 2003; Hirst, 2004, for an overview). Although

[1] http://wordnet.princeton.edu
[2] http://www.linguistics-ontology.org/gold.html
[3] http://www.geneontology.org/
[4] cf. http://www.text-technology.de

the Princeton WordNet was initially not conceived as an ontology but rather as a psychologically motivated model of lexical knowledge (Miller and Hristea, 2006, p.1), ontology textbooks often mention the Princeton WordNet as an ontological resource. (Sowa, 2000, p.497) distinguishes between *terminological ontologies*, the categories of which need not be fully specified by axioms and definitions, and *axiomatized ontologies*, the categories of which are distinguished by axioms and definitions stated in logic or in some computer-oriented language that could be automatically translated to logic. A similar distinction is drawn in Erdmann (2001), p.72: he differentiates between *light-weight ontologies*, which consist primarily of a representation schema providing means to specify taxonomies and to define additional features and relations, and *heavy-weight ontologies*, which are specified in a logic-based representation language. In this sense, PWN would be classified as a light-weight ontology; and PWN is, indeed, mentioned in the list of possible ontology resources (Erdmann, 2001, p.71).

Designing an OWL representation for a wordnet-style resource implies that one interprets the semantics of the entities and relations used in the original lexical resource with respect to the semantics of OWL.

In this interpretation process, the choice between one modelling option and the other is highly dependent upon the application context in which the ontology is to be used. The models discussed in section 4 have been developed with the following application framework in mind:

- The models are designed to be used in our research group's text processing applications, such as anaphora resolution (Goecke et al., this volume), discourse parsing (Bärenfänger et al., this volume), text-to-hypertext conversion (Holler et al., 2004; Storrer, 2008), and text classification (Mehler, this volume).

- Since some of these applications may deal with documents in a specific domain, we aim at a common representation format for domain-specific and general vocabulary.

For readers not familiar with wordnet-style lexical representations, the following paragraphs contain a brief introduction to the main types of entities and relations that have to be captured in our models.

The basic entities in GermaNet are disambiguated words, called *lexical units*.[5] Lexical units denoting the same or a very similar concept are grouped together in *synsets*, where the word *synset* is an abbreviation for *synonym set*. Lexical units and synsets are connected by two types of binary relationships: (1) *conceptual relations* like hyponymy and meronymy hold between synsets, and (2) *lexical-semantic relations* like antonymy hold between pairs of lexical units.

The basic entities in our domain ontology TermNet are technical terms, used to refer to well-defined concepts in the specialised domain. In many cases they form a taxonomy in which terms are represented as classes, and more specific terms are defined

[5] The OWL models of the Princeton WordNet discussed in section 3 use the term "word sense" for disambiguated words; the corresponding term in GermaNet is "lexical unit". Since our models are based on GermaNet data we will use the term "lexical unit" (abbreviated by LU) in this paper.

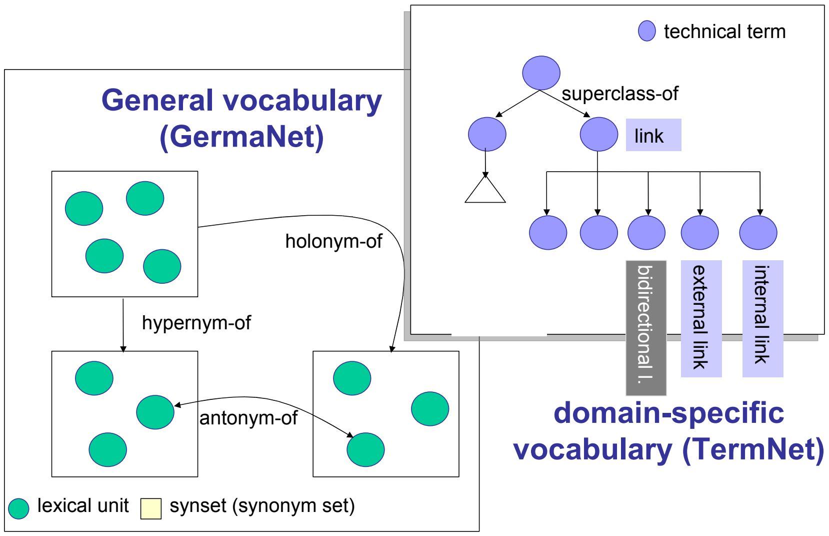


Figure 1: Entities and relations in GermaNet and TermNet.

as subclasses of broader terms. In the terminology of hypertext research, which is represented in TN for example, the technical terms *InternalLink*, *ExternalLink*, and *BidirectionalLink* are all subclasses of the broader term *Link*. Figure 1 illustrates these basic entities and relations in GN and TN.

3 OWL models for the Princeton WordNet

Although the Princeton WordNet was initially not conceived as an ontology, it has proven to be a useful resource for ontology-driven NLP applications. Following the Semantic Web initiative, several approaches for the conversion of the Princeton WordNet into OWL or RDFS have been put forward. We took three approaches that use OWL as their target representation language and examined them more closely: (1) The W3C approach ("W3C") (a working draft has been published by the Semantic Web Best Practices and Deployment Working Group, van Assem et al., 2006); (2) the "Neuchâtel approach" ("NCH") (Ciorăscu et al., 2003), and (3) the "Amsterdam approach" ("AMST") (van Assem et al., 2004). NCH has partly been considered in W3C, and the group of authors representing AMST overlaps with that of W3C; thus, AMST seems to be a predecessor of W3C. The three approaches differ in their goals: W3C aims at providing a standard conversion of the Princeton WordNet into OWL that can be used directly by Semantic Web applications. In this approach the OWL version should not deviate from the original PWN, i.e. the PWN data model should be reflected in OWL without further interpretations. The goal of AMST was also to provide an OWL-encoded version of PWN. The main objective of NCH, though, was to create a test domain for the ontology-based information system *knOWLer* and to demonstrate its performance by means of sample queries in a document retrieval scenario. A discussion of modelling alternatives did not play a role in this effort. The W3C approach used version 2.0 of the Princeton WordNet, while NCH converted the version 1.7.1 of the Princeton WordNet in OWL.

In W3C and NCH, the actual ontology (i.e. the class hierarchy without the instances, cf. Erdmann, 2001, p.74), which is called the “WordNet RDF/OWL schema” in W3C, consists of the class of synsets (W3C: *Synset*, NCH: *LexicalConcept*) and its subclasses *Noun(Synset)*, *Adjective(Synset)*, *Adverb(Synset)*, and *Verb(Synset)*, where *Adjective(Synset)* has a further subclass called *AdjectiveSatellite(Synset)*. Lexical units are modelled by the class *WordSense* in W3C and by the class *WordObject* in NCH. In W3C, *WordSense* is further subdivided into part of speech-specific subclasses like *NounWordSense*; in NCH, it is not. Moreover, for purely formal units (not associated with a meaning), the class *Word* exists.[6] In NCH, a corresponding class called *StemObject* exists only in an external ontology which is used for document retrieval.

The single synsets – e.g. the synset {horse, nag, steed} – are modelled as individuals, i.e. as instances of *NounSynset*, *VerbSynset* etc., in all three approaches. Likewise, the single lexical units (e.g. *horse*) are modelled as individuals in W3C as well as in NCH. Consequently, the lexicalisation relation (the relation that connects synsets and lexical units) is an OWL `ObjectProperty` with the domain *Synset* (*LexicalConcept*) and with the range *WordSense* (*WordObject*; the relation is called *synsetContainsWordSense* in W3C and *wordForm* in NCH), thus connecting a synset individual to one or more lexical unit individuals. In AMST, lexical units are modelled neither as classes nor as individuals, but as literals which appear as values of the multiple-valued DatatypeProperty *wordForm* (domain: *Synset*). Furthermore, in all three approaches, further ObjectProperties with *Synset* as domain and range exist, which model the PWN conceptual relations (e.g. *hyponymOf*, *entails*, and partly their POS-specific restrictions) in OWL. In a similar fashion, the PWN lexical relations (e.g. *antonymOf*, *participleOf*) are represented as ObjectProperties *WordSense* in the OWL versions of W3C and NCH, with *WordSense* as domain and range. Moreover, the W3C approach contains instructions on how to interpret the PWN *hyponymOf* relation by declaring it a subproperty of the *subclassOf* property in OWL. In AMST, the lexical relations are encoded by dint of “helper classes” such as *SynSetVerb*.

Considering that the conversion mainly aims at preserving the original structure and providing an OWL representation that can be easily processed and integrated in SW applications, these models seem to be quite suitable.

[6] The homonymic lexical units *nArtefakt.248.Schloss* and *nOrt.862.Schloss* of GermaNet, for example, share certain formal (i.e. orthographic, phonological, and morphological) properties which could be represented as properties of one *Word* instance.

From a linguistic viewpoint, however, it is striking that all of the approaches model synsets, i.e. sets of quasi-synonymous units, and their members, the disambiguated lexical units, as individuals. This is striking because synsets are frequently considered to be concepts which can be referenced linguistically by the lexical units contained in the synsets; e.g. a synset formed by {horse, nag, steed} denotes the horse concept. This suggests that at least the synsets should be conceived as classes, the instances of which are individual objects (e.g. the horse "Fury"). However, in principle, the lexical unit "horse" can also generically refer to the whole class (e.g. in meaning postulates like "A sorrel is a reddish horse").

All in all, the decision to model synsets and lexical units as individuals, i.e. to implement the instance model, is not at all obvious. Instead, both options – for which we introduced the short terms *instance model* and *class model* in section 1 – capture two different perspectives that one may have on wordnets:

1. In the instance model, a wordnet is conceived primarily as a lexicon describing properties of lexical units. The categories of the model represent linguistic classes and subclasses. Thus, synsets and word senses are modelled as instances of word class categories, e.g. the classes *NounWordSense* or *NounSynset*.

2. In the class model, a wordnet is conceived primarily as an ontology describing properties of the concepts that are denoted by the lexical units. The categories captured in the ontology represent concepts and their properties. Thus, synsets and word senses are modelled as classes in which the instances are individual entities.

There are two arguments which motivate, in our view, the implementation of the class model:

1. The Princeton WordNet, in its version PWN 2.1, draws an explicit distinction between the relation of hyponymy on the one hand (e.g. the subordinate synset containing "peach" is a hyponym of the superordinate synset containing "drupe") and the class-instance relation on the other (e.g. the proper name "Berlin" is an instance of the synset containing "city").[7] Over 7,600 PWN synsets were manually classified as instances and tagged as such. Despite this introduction of the class-instance distinction, the PWN version 2.1 may still be converted to OWL using the instance model, e.g. by ignoring the class-instance distinction among synsets by skipping those synsets that are tagged as instances. However, Miller and Hristea (2006) (p.1) introduced this distinction with the aim to help ontologists "to distinguish between a concept-to-concept relation of subsumption and an individual-to-concept relation of instantiation". We believe that this aim implies that synsets (like "peach") are conceived as concepts denoting classes with numerous instances, while proper names (like "Berlin") denote instances of synset classes (in the case of "Berlin" the class synset containing "city"). In our opinion,

[7] Examples from (Miller and Hristea, 2006, p.3).

this perspective is captured more adequately by the class model than by the instance model, because when class synsets are already modelled as instances, the class-instance relations would have to be defined between pairs of instances; this is not a very intuitive interpretation of such relations.

2. The second argument is concerned with domain-specific vocabulary in domain ontologies. Domain ontologies often represent taxonomies of technical terms that mirror superclass-subclass-relations between their instances. This may be illustrated by the example in figure 1: the term *externalLink* is a subclass of the broader term *Link*. In the class model one may represent these relations using the <`rdfs:subClassOf`> property and benefit from its related mechanisms of feature inheritance. Another aspect that may be nicely captured by the class model is that in such taxonomies, subclasses with the same classification feature (e.g. *InternalLink* and *ExternalLink* in the example illustrated in figure 1) are disjoint: an individual link may either be an instance of *InternalLink* or an instance of *ExternalLink*. This restriction can be neatly represented using the OWL <`owl:disjointWith`> construct. Since <`rdfs:subClassOf`> and <`owl:disjointWith`> can only be defined for classes, the class model is better suited to represent taxonomies than the instance model. All in all, the class model seems to be more appropriate to capture domain-specific terminology in OWL than the instance model.

For our domain ontology TermNet, we developed an OWL model that implements the class model: the main entities of TN, the technical terms, are represented as classes. Specific terms are related to broader terms by means of the <`rdfs:subClassOf`> property. Disjointness of technical terms, e.g. between *internal link* and instance of *external link* in our example, is represented by using the OWL <`owl:disjointWith`> construct.[8]

If one chooses the class model for the domain ontology one still may follow the instance model when representing the general vocabulary of GermaNet. Indeed, in the approach described in Kunze et al. (2007) we related the class model of TN with an instance model of GN; this option will be described in section 4.1. In addition, we experimented with alternative models for GN: one that implements GN following the class model and one that combines both options in OWL Full. The three models as well as their respective combination with TermNet will be discussed and compared in the following section.

4 Three alternative models for representing GN and TN in OWL

In this section, we discuss three alternative representations of GN and its plug-in connections with TN in OWL: the first representation we call *The OWL DL Instance Model* (GN synsets and lexical units are OWL individuals), the second encoding we call *The OWL DL Class Model* (GN synsets and lexical units are classes), and the third

[8] The model is described in Kunze et al. (2007); the subset of TermNet considered in the model comprises 141 NounTerms from the domain of hypertext research.

encoding we call *The OWL Full Metaclass Model* (GN synsets and lexical units are both OWL classes *and* individuals).

For each model, a basic hierarchy of classes is declared using <`owl:Class`> and <`rdfs:subClassOf`> statements. The basic hierarchy includes *Synset* with its subclasses *NounSynset*, *AdjectiveSynset*, *VerbSynset*, and *AdverbSynset*, as well as the class *LexicalUnit* with its subclasses *NounUnit*, *AdjectiveUnit*, *VerbUnit*, and *AdverbUnit*, cf. also Kunze et al. (2007).

In each model, we define the general lexicalisation relation (describing the relation between one synset and its lexical units) as an OWL Object Property called *hasMember*. Listing 1 shows that this property has the general class *Synset* as its domain and the general class *LexicalUnit* as its range.

```
<owl:InverseFunctionalProperty rdf:about="#hasMember">
  <rdfs:range rdf:resource="#LexicalUnit"/>
  <rdfs:domain rdf:resource="#Synset"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <owl:inverseOf rdf:resource="#memberOf"/>
</owl:InverseFunctionalProperty>
```

Listing 1: OWL code introducing the lexicalisation relation *hasMember* in all three models

Likewise in each model, the GN hyponym relation is an OWL Object Property called *isHyponymOf* and defined with *Synset* as both domain and range, cf. listing 2.

The POS-specific restrictions for *hasMember* and *isHyponymOf* are encoded in both models by use of the <`owl:allValuesFrom`> construction. Listing 3 illustrates how such restrictions are defined for the class*NounSynset*: the OWL code in this listing specifies the restriction of the range of the property *hasMember* to *NounUnit*, which is the POS-corresponding subclass of *LexicalUnit*.

In each model, we also wanted to relate GN synsets to terms of the domain ontology TermNet (TN). Since the OWL representation of TN remains constant (terms are always represented as OWL classes for reasons stated in section 3), each model implies a different way of encoding the plug-in relations between GN and TN.

In the following section, we will compare these three models and describe how they can be related to TermNet within OWL. This will be discussed by the examples of how the lexicalisation relation between individual synsets and lexical units, the hyponymy relation between individual synsets, and the plug-in relation *attachedToNearSynonym* between individual synsets in GN and terms in TN are encoded. Furthermore, for each model, we discuss our experiments with queries to the resulting knowledge bases formulated in Prolog and nRQL.

4.1 The OWL DL Instance Model

As discussed in section 3, previous approaches to the representation of the PWN in OWL conform to an instance model, where the single synsets are rendered as OWL

```
<owl:ObjectProperty rdf:about="#conceptualRelation">
  <rdfs:domain rdf:resource="#Synset"/>
  <rdfs:range rdf:resource="#Synset"/>
</owl:ObjectProperty>

<owl:TransitiveProperty rdf:about="#isHyponymOf">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:subPropertyOf rdf:resource="#conceptualRelation"/>
  <owl:inverseOf>
    <owl:TransitiveProperty rdf:about="#isHypernymOf"/>
  </owl:inverseOf>
</owl:TransitiveProperty>
```

Listing 2: OWL code introducing conceptual relations and the relation *isHyponymOf* in all three models

```
<owl:Class rdf:ID="NounSynset">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class rdf:about="#NounUnit"/>
      </owl:allValuesFrom>
      <owl:onProperty>
        <owl:InverseFunctionalProperty rdf:ID="hasMember"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  ...
</owl:Class>
```

Listing 3: OWL code for a restriction of the lexicalisation relation *hasMember* in all three models

individuals and where the conceptual and lexical-semantic relations are OWL property instances. The instance model for representing GermaNet in OWL, which adopted the W3C strategy (see section 3) for the representation of wordnets, was introduced in Kunze et al. (2007).

Relation instances of the *hasMember* relation are encoded as property instances of individual synsets. Listing 4 shows the XML code representing the individual that corresponds to the synset {eitel, selbstherrlich, selbstgefällig, selbstgerecht}.

A relation instance of *isHyponymOf* is also encoded as a property instance of an individual synset in XML, cf. listing 5.[9]

To check the consistency of the OWL DL Instance Model, the GermaNet ontology was populated with 37 synset and 83 lexical unit individuals and all relation instances that hold between them. The ontology was checked for consistency using the ontology editor Protégé 3.1.1[10] in connection with the reasoner/classifier software RacerPro 1.9.1[11].

[9] All characteristics of the OWL object and datatype properties as well as OWL code representing further lexical relations in the instance model are described in Kunze et al. (2007).

[10] http://protege.stanford.edu, visited 24 May 2007

```
<AdjectiveSynset rdf:ID="aVerhalten.235">
   <hasMember rdf:resource="#aVerhalten.235.eitel"/>
   <hasMember rdf:resource="#aVerhalten.235.selbstherrlich"/>
   <hasMember rdf:resource="#aVerhalten.235.selbstgefällig"/>
   <hasMember rdf:resource="#aVerhalten.235.selbstgerecht"/>
   <!-- ... (Further properties of aVerhalten.235] -->
<AdjectiveSynset>
```

Listing 4: OWL code for relation instances of *hasMember* in the Instance Model

```
<AdjectiveSynset rdf:ID="aVerhalten.235">
   <!-- ... (Further properties of aVerhalten.235] -->
   <isHyponymOf rdf:resource="#aVerhalten.225"/>
<AdjectiveSynset>
```

Listing 5: OWL code for relation instances of *isHyponymOf* in the Instance Model

To test how queries to the ontology can be formulated and processed, we parsed the owl file using the triple store SWI Prolog Semantic Web library[12] in combination with the Thea OWL library for Prolog (Vassiliadis, 2006). For the ontology, we subsequently implemented in Prolog several query types for the ontology, which are typical of text-technological applications:

- List the set of synsets that a given lexical unit is a member of
- List the set of lexical units a given synset has as its members
- List the set of synonyms of a given lexical unit
- List the set of direct hyponym (hypernym) synsets of a given synset
- List the set of direct hyponym (hypernym) lexical units of a given lexical unit
- List the set of direct or transitive hyponym (hypernym) synsets of a given synset
- List the set of direct or transitive hyponym (hypernym) lexical units of a given lexical unit

Queries of these types could be successfully run on the GN subset encoded as OWL DL Instance Model ontology.

In the approach described in Kunze et al. (2007), we related a subset of TN technical terms with a subset of GN synsets. Since that implies that an ontology which follows the instance model is related to an ontology which follows the class model, we call the approach to connect these a mixed model. We defined three plug-in relations called *attachedToNearSynonym*, *attachedToGeneralConcept*, and *attachedToHolonym* with domain *tn:Term* and range *gn:Synset*. Plug-ins are relations to connect the specialised

[11] http://www.racer-systems.com, visited 24 May 2007
[12] http://www.swi-prolog.org, visited 24 May 2007

concepts (concepts from domain-specific vocabulary or terminologies) of a domain ontology with the more general concepts of a PWN style lexical-semantic network. The original plug-in approach yields a common hierarchy in which the top concepts of the specialised ontology are eclipsed while the subordinate concepts, the terms, are imported into the general language ontology. The plug-ins defined in Kunze et al. (2007) are inspired by, but not identical to, the ones originally introduced in (cf. Magnini and Speranza, 2002). The plug-in relation *attachedToNearSynonym*, for example, is defined as an OWL Object Property in the mixed model with domain *tn:Term* and range *gn:Synset* as shown in listing 6. *Gn, tn*, and *plg* are namespace prefixes for the URIs of the three ontologies involved (GermaNet, TermNet, and plug-in relations), which are ideally kept in separate files.

```
<owl:ObjectProperty rdf:about="#plg:attachedToNearSynonym">
  <rdfs:domain rdf:resource="#tn:Term"/>
  <rdfs:range rdf:resource="#gn:Synset"/>
</owl:ObjectProperty>
```

Listing 6: OWL code for introducing the plug-in relation *attachedToNearSynonym* in the Instance Model

To plug the class *tn:Term_Link* into its corresponding synset *gn:Link*, the former is declared to be a subclass of a local restriction that assigns every individual of the class *tn:Term_Link* the individual *gn:Link* as the value on the property *plg:attachedToNearSynonym*, using the <`owl:hasValue`> construction (listing 7) (cf. also Kunze et al., 2007).

```
<owl:Class rdf:ID="tn:Term_Link">
  <rdfs:subClassOf rdf:resource="#tn:NounTerm"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:hasValue rdf:resource="#gn:Link"/>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="plg:attachedToNearSynonym"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Listing 7: OWL code for relation instances of *attachedToNearSynonym* in the Instance Model

Since in this mixed model TermNet terms are modelled as classes and GermaNet synsets are modelled as instances and since in OWL DL classes cannot be specified as values of property instances, the plug-in relations proposed in Kunze et al. (2007) cannot have inverse properties, i.e. cannot be defined using the <`owl:inverseOf`> construction. A declaration of inverse relations is strictly speaking not necessary for making inferences, but it is still desirable because it can speed up processing.

4.2 The OWL DL Class Model

When modelling the conceptual relations between synsets according to a class model (synsets are OWL classes), our first preference would have been to relate classes by declaring pairs of them as relation instances of a conceptual relation like *isHyponymOf*. However, when classes are assigned as values of properties, they must function as individuals at the same time, which goes beyond the scope of OWL DL (cf. Smith et al., 2004). Thus, we decided to relate classes with one another by employing local property restrictions using the <`owl:allValuesFrom`> construction, such as in the example in listing 8, where the synset containing *Webdokument* is declared to be a hyponym of the synset containing *Hypertextsystem*. When a synset has more than one hypernym, we declare the <`owl:allValuesFrom`> restriction such that all values have to be taken from a *union* of classes.

```
<owl:Class rdf:about="#gn:Synset_Webdokument">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class>
          <owl:unionOf rdf:parseType="Collection">
            <owl:Class rdf:about="#gn:Synset_Dokument"/>
            <owl:Class rdf:about="#gn:Synset_Hypertextsystem"/>
          </owl:unionOf>
        </owl:Class>
      </owl:allValuesFrom>
      <owl:onProperty>
        <owl:TransitiveProperty rdf:about="#gn:isHyponymOf"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:class>
```

Listing 8: OWL code for relation instances of *isHyponymOf* in the OWL DL Class Model

Within a class model, it is tempting to model lexical units as the instances of the synsets. However, we have indicated above that we want the instances of synsets to include individuals of a discourse model, such as *Berlin* (i.e. named entities) or *Horse_351*. Lexical units do not represent semantic units but linguistic expression types; thus, it is adequate to model lexical units as classes, too. (Their instances should represent the tokens (occurrences) of lexical units in text.) Accordingly, the relation instances of the lexicalisation relation *hasMember* and all of the lexical relations are encoded as local property restrictions on synset classes, too, cf. listing 9.

Again, we parsed the ontology using the Semantic Web library with the Thea OWL library for Prolog. To run the set of queries listed in section 4.1, a new set of Prolog predicates had to be implemented. Still, the queries could all be run in a straightforward manner, and Prolog provided the correct answer sets.

```
<owl:InverseFunctionalProperty rdf:about="#hasMember">
  <owl:inverseOf rdf:resource="#memberOf"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:range rdf:resource="#LexicalUnit"/>
  <rdfs:domain rdf:resource="#Synset"/>
</owl:InverseFunctionalProperty>
```

Listing 9: OWL code for relation instances of *hasMember* in the OWL DL Class Model

To test the consistency and general queryability of the combination of both GN and TN as class models in OWL, we also implemented a set of test queries using the query language nRQL (Haarslev et al., 2004) in connection with RacerPro. We tested the query types listed in section 4.1 as well as more complex queries across GN and TN, e.g. querying hyponymy also along the plug-in relation *attachedToGeneralConcept*. The results showed that, when using RacerPro, only information about (sets of) individuals can be queried, but not about classes as such, which would be necessary for a wordnet ontology in the class model. Thus, we had to first introduce one pseudo-individual for each synset and lexical unit. Consequently, the original class model became somewhat corrupted, but, unlike in the case of the SWI SemWeb Library, additional coding of query predicates was not required when using nRQL. We could formulate queries as listed in section 4.1 to the GN+TN Class Model represented in OWL.[13] Listing 10 shows nRQL code for a query for all hyponyms (TN terms or GN synsets) of the GN synset *Navigationshilfe*. RacerPro would infer the right answer, i.e. a list of synset IDs from both GermaNet and TermNet including those in listing 10.

Listing 10: nRQL query to Class Model

```
(retrieve (?y) (or (and (?x ?y |gn_isHyperonymOf|) (?x
  |gn_Synset_Navigationshilfe|)) (and (?x |gn_Synset_Navigationshilfe|)
  (?x ?z |gn_isHyperonymOf|) (?z ?y |plg_inverseOfAttachedToGeneralConcept|))))

((?Y |tn_ObjektiverLink|))
((?Y |tn_Eins-zu-n-Link|))
((?Y |tn_VerborgenerLink|))
((?Y |gn_SS_Link|)))
```

4.3 The OWL Full Metaclass Model

As already indicated, we tested a third modelling option for GermaNet in OWL. The Instance Model can be converted into a *Metaclass Model* simply by adding the following line to the definitions of the class *Synset*:[14]

[13] We would like to thank Bianca Selzam, who built the TermNet OWL model and conducted the query experiments using nRQL.

```
<rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
```

This makes *Synset* a *metaclass*, i.e. a class in which the instances are classes, too. Accordingly, lexical units were declared as metaclasses. Thus, all of the single synsets and lexical units are simultaneously classes and instances in this ontology, which now seems to offer all advantages of the instance *and* the class model at the same time: since synsets and lexical units are instances, straightforward and simple XML element relation instances as in the instance model shown in section 4.1, are used to represent the conceptual and lexical-semantic relations, and the lexicalisation relation. On the other hand, since synsets and lexical units are classes, they can now be populated with token occurrence individuals as needed in text-technological applications, cf. figure 2. Interestingly, it seems that a metaclass mechanism for OWL could also be used in many other domains and applications, (cf. Schreiber, 2002; Noy (ed.), 2005).

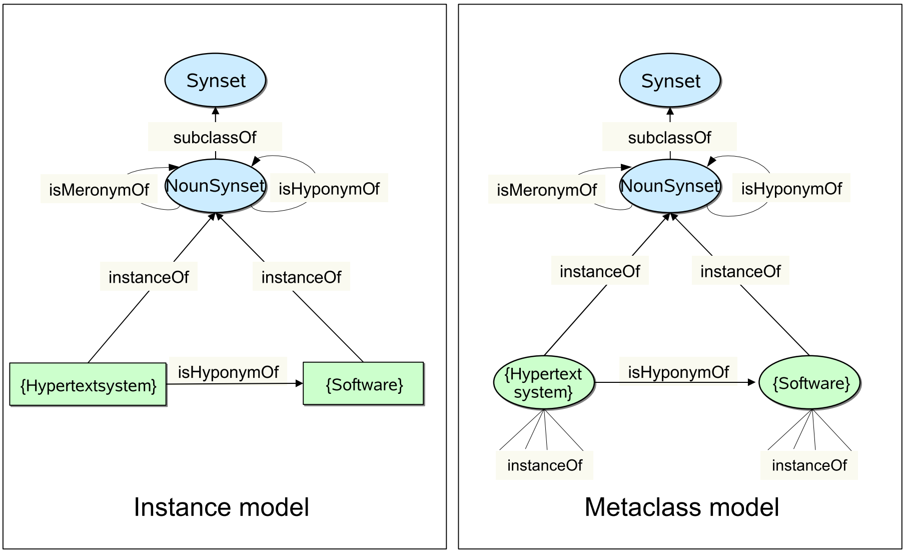


Figure 2: Instance vs. Metaclass Model.

Unfortunately the ontology described above lies outside the scope of OWL DL, i.e. it is in OWL Full (cf. Smith et al., 2004). Thus, standard DL-based reasoners cannot be used on it.[15]

[14] Line is shown as added using the Protégé editor.

[15] Outside the DL reasoning community, Pan et al. (2005) introduced a non-standard OWL variant called OWL FA. This is a well-defined *metamodelling* extension of OWL DL, and, unlike OWL Full, it is still decidable.

However, this metaclass model ontology of GermaNet in OWL Full could be parsed using the SWI Prolog Semantic Web library in combination with the Thea OWL library for Prolog and successfully queried with queries of the types listed in section 4.1. The queries had to be coded in Prolog and are basically the same as the ones one can code for the instance model. We implemented a set of queries in Prolog that can infer hyponym relationships in the OWL Full Metaclass representation of GN.[16] For querying the set of hyponyms of a given lexical unit, for example, the Prolog predicates shown in listing 11 were written: HasElements/2 and IsMemberof/2 are user-defined predicates that check the GN lexicalisation relation (*hasMember* and its inverse property *memberOf*, respectively), and owl_parser:uri_split/4 is a predicate provided by the Thea OWL library for deleting or adding a namespace URI. (For demonstration purposes, we provide a "readable" version of the predicate *unitIsHyperonymOf* here, i.e. one where the answer does not contain namespace URIs.) Listing 12 shows a query for the set of lexical unit hyponyms of the lexical unit *aVerhalten.235.eitel* and the response of the Prolog interpreter.

Listing 11: Prolog predicates for querying the Metaclass Model

```
unitIsHyperonymOf_readable(Input,Output) :-
        isMemberOf(Input,Set),
        setIsHyperonymHelper2(Set,Set2),
        hasElements(Set2,C),
        owl_parser:uri_split(C,_,Output,'#').

setIsHyperonymHelper2(Input,Output) :-
        individual(Input,_,_,PList),
        member(value('http://www.owl-ontologies.com/unnamed.owl#isHypernymOf',Output),
        PList).
```

Listing 12: Prolog query to Metaclass Model

```
?- unitIsHyperonymOf_readable('aVerhalten.235.eitel', A).

A = 'aVerhalten.231.geckenhaft' ;
A = 'aVerhalten.236.affektiert' ;
A = 'aVerhalten.236.geziert' ;
```

Further predicates that can be used for the hypernymy and hyponymy-related queries listed in section 4.1 are *unitIsDirectHyponymOf, setIsHyperonymOf* and *setIsDirectHyperonymOf.*

[16] We would like to thank Christian Kullmann, who implemented the query predicates and conducted the query experiments in Prolog using Thea.

5 Conclusions and outlook

Existing conversions of the Princeton WordNet into the Semantic Web ontology language OWL (Ciorăscu et al., 2003; van Assem et al., 2004, 2006) as well as a description to convert GermaNet to OWL (Kunze et al., 2007) apply what we have dubbed an instance model to the representation of wordnets in OWL: the single synsets and lexical units are rendered as OWL individuals, and relation instances of the conceptual and lexical-semantic relations appear as property instances. Nevertheless, from a linguistic point of view, synsets are concepts (classes) whose instances are individuals or discourse entities, and lexical units are types of linguistic expressions, whose instances can be interpreted as the token occurrences of these expressions in text, which seems appropriate at least in text-technological applications. Moreover, domain-specific ontologies or terminologies encoded in OWL such as TermNet (Kunze et al., 2007) or GOLD (Farrar and Langendoen, 2003) rather apply a *class model* of representation, in which terms are rendered as OWL classes, and a taxonomic hierarchy is imposed by means of the OWL *subClassOf* relation. In text-technological applications, it is often desirable to integrate such domain-specific ontologies with a general wordnet, and it would clearly ease the integration process if the two resource types corresponded to the same representation model.

Thus, in section 4, we described our evaluation of three OWL representation models for wordnets, using the example of GermaNet, and how they can be combined with the domain ontology TermNet, applying plug-in approach suggested by Magnini and Speranza (2002). Firstly, in the instance model of GN, conceptual and lexical-semantic relations as well as the lexicalisation relation appear as property instances, i.e. XML elements. Plug-in relations have as their domain a term class and as their values a GN individual synset. As a consequence, inverse relations of plug-ins could not be defined. Secondly, in the class model of GN, all conceptual and lexical-semantic relations, the lexicalisation relation, as well as plug-in relations were rendered as property restrictions on classes representing individual synsets and terms. Thirdly, in the metaclass model of GN, synsets and lexical units are both classes and individuals in OWL. All lexical-semantic and conceptual relations as well as the lexicalisation relation are thus XML property instances like in the instance model. Plug-in relations can either be represented as property instances like in the instance model or as local property restrictions like in the class model.

In our view, it would be clearly desirable to encode wordnets in a metaclass model in OWL: it allows for a linguistically adequate representation of synsets and lexical units as OWL classes while the somewhat clumsy representation of lexical-semantic and conceptual relations as local property restrictions, such as in the class model, are not necessary. Its only drawback is that DL-based reasoning software cannot be used on a metaclass model, because it is outside of the sublanguage OWL DL. As a possible way out, we found that the processing of queries for wordnet relations in the OWL metaclass model could be realised using the triple store SWI Prolog Semantic Web library[17] and

[17] We would like to thank Guus Schreiber for pointing out this possibility to us.

the Thea OWL library (Vassiliadis, 2006) for Prolog.

A total conversion of GermaNet into all of the three models examined is under way. We also aim at plugging in more domain ontologies in GermaNet based on their OWL representations; thus far, only connections from TermNet have been tested. The set of Prolog predicates for querying wordnets in OWL according to the metaclass model has to be extended for the remaining wordnet relations. Finally, we plan to develop a metaclass model for TermNet. On this basis, we aim to experiment with alternative plug-in properties that relate the GN metaclass model to the TN metaclass model (and possibly other domain ontologies represented according to this model). We will inform about future work on our project website, cf. `http://www.hytex.info/040_werkstatt/030_owlmodellierung`.

References

Beißwenger, M., Storrer, A., and Runte, M. (2004). Modellierung eines Terminologienetzes für das automatische Linking auf der Grundlage von WordNet. *LDV-Forum*, 19(1-2):113–125.

Cioră scu, C., Cioră scu, I., and Stoffel, K. (2003). knOWLer – Ontological support for information retrieval systems. In *Proceedings of the 26th Annual International ACM-SIGIR Conference, Workshop on Semantic Web*, Toronto, Canada.

Erdmann, M. (2001). *Ontologien zur konzeptuellen Modellierung der Semantik von XML*. Books on Demand, Karlsruhe.

Farrar, S. and Langendoen, D. T. (2003). A linguistic ontology for the semantic web. *GLOT International*, 7(3):97–100.

Fellbaum, C., editor (1998). *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA.

Haarslev, V., Möller, R., and Wessel, M. (2004). Querying the Semantic Web with Racer + nRQL. In *Proceedings of the KI-2004 International Workshop on Applications of Description Logics (ADL'04), Ulm, Germany, September 24*, Ulm.

Hirst, G. (2004). Ontology and the lexicon. In Staab, S. and Studer, R., editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 209–229. Springer, Heidelberg.

Holler, A., Maas, J.-F., and Storrer, A. (2004). Exploiting coreference annotations for text--to-hypertext conversion. In *Proceedings of LREC 2004, Volume II*, pages 651–654, Lisboa.

Kunze, C. and Lemnitzer, L. (2002). GermaNet - representation, visualization, application. In *Proceedings of LREC*, volume V, pages 1485–1491, Las Palmas.

Kunze, C., Lemnitzer, L., Lüngen, H., and Storrer, A. (2007). Repräsentation und Verknüpfung allgemeinsprachlicher und terminologischer Wortnetze in OWL. *Zeitschrift für Sprachwissenschaft*, 26(2).

Kunze, C., Lemnitzer, L., and Wagner, A. (2003). *Anwendungen des deutschen Wortnetzes in Theorie und Praxis. Sonderheft der Zeitschrift für Computerlinguistik und Sprachtechnologie.* Bonn.

Magnini, B. and Speranza, M. (2002). Merging Global and Specialized Linguistic Ontologies. In *Proceedings of Ontolex 2002*, pages 43–48, Las Palmas de Gran Canaria, Spain.

Miller, G. A. and Hristea, F. (2006). Word net nouns: Classes and instances. *Computational Linguistics*, 32(1):1–3.

Noy (ed.), N. (2005). Representing classes as property values on the semantic web. W3C Working Group Note. `http://www.w3.org/TR/swbp-classes-as-values/`, visited 23 May 2007.

Pan, J. Z., Horrocks, I., and Schreiber, G. (2005). OWL FA: A metamodeling extension of OWL DL. In *Proceedings of the Workshop OWL: Experiences and directions*, Galway, Ireland.

Schreiber, G. (2002). The web is not well-formed. *IEEE Intelligent Systems*, 17(2):79–80.

Smith, M. K., Welty, C., and Deborah L. McGuinness, e. (2004). OWL Web Ontology Language – Guide. Technical report, W3C (World Wide Web) Consortium. `http://www.w3.org/TR/2004/REC-owl-guide-20040210/`, visited 24 May 2007.

Sowa, J. F. (2000). *Knowledge Representation. Logical, Philosophical, and Computational Foundations.* Brooks/Cole, Pacific Grove.

Storrer, A. (2008). Mark-up driven strategies for text-to-hypertext conversion. In Metzing, D. and Witt, A., editors, *Linguistic Modelling of Information and Markup Languages. Contributions to Language Technology*, Text, Speech and Language Technology. Kluwer, Dordrecht. To appear.

van Assem, M., Gangemi, A., and Schreiber, G. (2006). Conversion of WordNet to a standard RDF/OWL representation. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy.

van Assem, M., Menken, M. R., Schreiber, G., Wielemaker, J., and Wielinga, B. (2004). A method for converting thesauri to RDF/OWL. In *Proceedings of the 3rd International Semantic Web Conference (ISWC 2004)*, number 3298 in Lecture Notes in Computer Science, Hiroshima, Japan.

Vassiliadis, V. (2006). Thea. A web ontology language - OWL library for [SWI] Prolog. Web-published manual, `http://www.semanticweb.gr/TheaOWLLib/index.htm`, visited 15 July 2006.

Ekaterina Ovchinnikova, Kai-Uwe Kühnberger

Automatic Ontology Extension: Resolving Inconsistencies

Ontologies are widely used in text technology and artificial intelligence. The need to develop large ontologies for real-life applications provokes researchers to automate ontology extension procedures. Automatic updates without the control of a human expert can generate potential conflicts between original and new knowledge. As a consequence the resulting ontology can yield inconsistencies. We propose a procedure that models the process of adapting an ontology to new information by repairing several important types of inconsistencies.

1 Introduction

There is an increasing interest in augmenting text technological and artificial intelligence applications with ontological knowledge. Since the manual development of large ontologies has been proven to be time-consuming, many current investigations are devoted to automatic ontology learning methods (Perez and Mancho, 2003).

The most important existing markup language for ontology design is the Web Ontology Language[1] (OWL), with its popular versions (OWL Lite and OWL DL) based on the logical formalism called Description Logic (DL). DL was designed for the representation of terminological knowledge and reasoning devices (Baader et al., 2003). Although most of the tools extracting or extending ontologies automatically output the knowledge in the OWL-format, they usually use only a small subset of the corresponding DL representation. The core ontologies generated in most practical applications contain the subsumption relation defined on concepts (taxonomy) and a few general relations (such as part-of and other). At present complex ontologies making use of the full expressive power and advances of the various versions of DL can be achieved only manually or semi-automatically. However, several recent approaches not only attempt to learn taxonomic and general relations, but also state which concepts in the knowledge base are equivalent or disjoint (Haase and Stojanovic, 2005).

The storage of ontological information within a logical framework entails inconsistency problems, because pieces of information can contradict each other, making the given ontology unsatisfiable and therefore useless for reasoning purposes. The problem of inconsistency becomes even more important with regard to large-scale ontologies: resolving inconsistencies in large ontologies by hand is time-consuming and tedious, therefore automatic procedures to debug ontologies are required.

The approach presented in this paper focuses on logical inconsistencies in terminological knowledge base: after a rough sketch of DLs (Section 2), we discuss informally

[1] A documentation can be found at `http://www.w3.org/TR/owl-features/`.

inconsistencies in ontologies (Section 3) and related work (Section 4). In addition to extending some existing ontology debugging methods, we provide formal criteria to distinguish different types of logical inconsistencies (overgeneralization and polysemy) in Section 5 and present an adaptation algorithm resolving logical inconsistencies that may appear in ontology extensions (Section 6). Section 7 adds some remarks concerning the order of the update and Section 8 concludes the paper.

2 Description Logic

In this section, we define description logics (DL) underlying the ontology representation considered in this paper[2] (cf. Baader et al., 2003, for an overview). A DL ontology contains a set of terminological axioms (called *TBox*), a set of instantiated concepts (called *Assertion* or *ABox*), and a set of role axioms (called *RBox*). In the present paper, we focus on the *TBox*, leaving the *ABox* and the *RBox* aside for further investigation.

A TBox is a finite set of axioms of the form $A_1 \equiv A_2$ (equalities) or $A \sqsubseteq C$ (inclusions), where A stands for a concept name and C (called *concept description*[3]) is defined as follows (R denotes a role name, A denotes an atomic concept): $C \rightarrow A \mid \neg A \mid \forall R.A$.

The semantics of concepts and axioms is defined in the usual way in terms of a model theoretic *interpretation function* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty set of individuals and the function $\cdot^{\mathcal{I}}$ maps every concept name A to $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and every role name R to $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. Negation and universal restriction is defined as usual: $(\neg A)^{\mathcal{I}} = \Delta^{\mathcal{I}} \backslash A^{\mathcal{I}}$ and $(\forall R.A)^{\mathcal{I}} = \{x \mid \forall y. \langle x, y\rangle \in R^{\mathcal{I}} \rightarrow y \in A^{\mathcal{I}}\}$. An interpretation $\mathcal{I}$ is a *model* of a TBox $\mathcal{T}$, if for every inclusion $A \sqsubseteq C$ in the TBox, $A^{\mathcal{I}} \subseteq C^{\mathcal{I}}$ and for every equality $A_1 \equiv A_2$ in the TBox, $A_1^{\mathcal{I}} = A_2^{\mathcal{I}}$ holds. A concept description D *subsumes* C in $\mathcal{T}$ ($\mathcal{T} \models C \sqsubseteq D$), if for every model $\mathcal{I}$ of $\mathcal{T}$: $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. A concept C is called *satisfiable towards* $\mathcal{T}$, if there is a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}}$ is nonempty. Otherwise C is *unsatisfiable towards* $\mathcal{T}$. The algorithms for checking satisfiability of concept descriptions are described in Baader et al. (2003) and implemented in several reasoners[4]. A TBox $\mathcal{T}$ is called *unsatisfiable* iff there is an atomic concept A defined in $\mathcal{T}$ that is unsatisfiable.

An important DL concept for this paper is the *least common subsumer (lcs)* (cf. Baader and Küsters, 2006) for the definition and algorithms for computing lcs). Intuitively, the lcs for concept descriptions C_1 and C_2 is a concept description that collects all common features of C_1 and C_2 and is most specific towards subsumption.

Definition 1 *A concept description L is a least common subsumer (lcs) of concept descriptions $C_1, ..., C_n$ towards a TBox $\mathcal{T}$ iff it satisfies the following two conditions:*

1. $\forall i \in \{1, ..., n\}: \ \mathcal{T} \models C_i \sqsubseteq L$ *and*
2. $\forall L'$: *if* $\forall i \in \{1, ..., n\}: \mathcal{T} \models C_i \sqsubseteq L'$ *and* $L' \neq L$ *then* $\mathcal{T} \not\models L' \sqsubseteq L$.

[2] In the following definitions, we closely follow Haase and Stojanovic (2005) who present an approach using one of the most powerful DL-versions for ontology learning.

[3] Hereinafter concept descriptions are referred to as concepts.

[4] Some of the DL reasoners are listed at `http://www.cs.man.ac.uk/~sattler/reasoners.html`.

3 Inconsistent Ontologies

The notion of an *inconsistent ontology* has several meanings. For example, three types of inconsistencies are distinguished in Haase and Stojanovic (2005):

- *Structural inconsistency* is defined with respect to the underlying ontology language. An ontology is structurally inconsistent, if it is syntactically inconsistent, i.e. if it contains axioms violating syntactical rules of the representation language (e.g. OWL *DL*).

- *Logical inconsistency* of an ontology is defined on the basis of formal semantics: An ontology is logically inconsistent, if it has no model.

- *User-defined inconsistency* is related to application context constraints defined by the user.

In this paper, we consider logical inconsistencies only. In particular, we focus on unsatisfiable terminologies. Notice that an ontology can become logically inconsistent only if its underlying logic allows negation. Ontologies share this property with every logical system. For the approaches concerned with core ontologies (lacking negation) contradictions in the ontological knowledge base cannot arise. But for approaches using more powerful logics, the problem of inconsistency becomes important (Haase and Stojanovic, 2005).

Terminological unsatisfiability can have several reasons: first, errors in the automatic ontology learning procedure or mistakes of the ontology engineer, second, polysemy of concept names, and third, generalization mistakes. The polysemy problem is particularly relevant for automatic ontology learning. If an ontology is learned automatically, then it is hardly possible to distinguish between word senses: If, for example, a concept *Tree* is declared to be a subconcept of both, *Plant* and *Data structure* (where *Plant* and *Data structure* are subconcepts of disjoint concepts, e.g. *Object* and *Abstraction*), then *Tree* will be unsatisfiable.

Generalization mistakes causing unsatisfiability are connected with definitions of some concepts that are too specific: such definitions contradict with their subconcepts, representing exceptions to these definitions. Here is a classical example:

Example 1

TBox:	$\{1.\ Bird \sqsubseteq CanFly,\quad 2.\ CanFly \sqsubseteq CanMove,$
	$3.\ Canary \sqsubseteq Bird,\quad 4.\ Penguin \sqsubseteq Bird\}$
New axiom:	$\{5.\ Penguin \sqsubseteq \neg CanFly\}$

The statement *all birds can fly* in Example 1 is too specific. If an exception *penguin*, that cannot fly, is added, the terminology becomes unsatisfiable.

Example 2 below demonstrates a case where two overgeneralized definitions of the same concept conflict with each other:

Example 2
TBox: $\{1.\ Child \sqsubseteq \forall likes.Icecream, 2.\ Icecream \sqsubseteq Sweetie, 3.\ Chocolate \sqsubseteq Sweetie,$
$4.\ Icecream \sqsubseteq \neg Chocolate, 5.\ Chocolate \sqsubseteq \neg Icecream\}$
New axiom: $6.\ Child \sqsubseteq \forall likes.Chocolate$

In this example, both definitions of $Child$ (i.e. $\forall likes.Icecream$ and $\forall likes.Chocolate$) are too specific. $Icecream$ and $Chocolate$ being disjoint concepts produce a conflict, if a modeled child likes at least one of their instances. Strictly speaking, the TBox in Example 2 is satisfiable. But we consider contradictions in scopes of universal quantification also as problematic, since such definitions are unusable in practice.

4 Related Work

A technique to find a minimal set of axioms that is responsible for inconsistencies in an ontology was first proposed in Baader et al. (2005). In order to detect a set of problematic axioms, assertions are labeled and traced back, if a contradiction is found in a tableau expansion tree. In Schlobach and Cornet (2003), an advanced approach of this idea is presented by introducing the notion of a *minimal unsatisfiability-preserving sub-TBox* (MUPS): An axiom pinpointing service for $\mathcal{ALC}$ is proposed identifying the exact parts of axioms that are causing a contradiction. Several present approaches to ontology debugging are concerned with explanation services that are integrated into ontology developing tools. For example, Wang et al. (2005) present a service explaining unsatisfiability in OWL-DL ontologies by highlighting problematic axioms and giving natural language explanations of the conflict. In Haase and Stojanovic (2005), an approach to automatic ontology extraction is described. Every extracted axiom receives a confidence rating witnessing how frequent the axiom occurs in external sources.

The approaches sketched above either do not give solutions of how to fix the discovered contradictions or just propose to remove a problematic part of an axiom, although removed parts of axioms can result in a loss of information. Considering, for example, Example 1 again, if the concept $CanFly$ is removed from axiom 1, then the entailments $Bird \sqsubseteq CanMove$ and $Canary \sqsubseteq CanFly$ are lost.

In Fanizzi et al. (2004), inductive logic programming techniques are proposed to resolve inconsistencies. If a concept C is unsatisfiable, then the axiom defining C is replaced by a new axiom, constructed on the basis of positive assertions for C. The information previously defined in the ontology for C gets lost. Kalyanpur (2006) extends the OWL-DL tableau algorithm with a tracing technique to detect conflicting parts of axioms. It is suggested to rewrite axioms using frequent error patterns occurring in ontology modeling. Lam et al. (2006) revise the technique proposed in Baader and Hollunder (1995) and support ontology engineers in rewriting problematic axioms in $\mathcal{ALC}$: Besides the detection of conflicting parts of axioms, a concept is constructed, that replaces the problematic part of the chosen axiom. This approach keeps the entailment $Bird \sqsubseteq CanMove$, but not $Canary \sqsubseteq CanFly$ in Example 1. An approach to resolve overgeneralized concepts conflicting with exceptions is presented in Ovchinnikova and

Kühnberger (2006) for $\mathcal{ALE}$. Besides rewriting problematic axioms, a split of an overgeneralized concept C into a more general concept (not conflicting with exceptions) and a more specific one (capturing the original semantics of C) is proposed.

5 Proposed Approach

5.1 Tracing Clashes

In this section, we revise the tableau-based algorithm presented in Lam et al. (2006) for tracing clashes in unsatisfiable terminologies and rewriting problematic axioms. We adapt this tracing technique to our simple logic. The proposed algorithm detects the relevant parts of the axioms that are responsible for the contradiction.

Following Lam et al. (2006) suppose that a terminology $\mathcal{T}$ contains axioms $\{\alpha_1, ..., \alpha_n\}$, where α_i refers to an axiom $A_i \sqsubseteq C_i$ or $A_i \equiv C_i$ ($i \in \{1, ..., n\}$). In checking the satisfiability of a concept description C the tableau algorithm constructs a model of C represented by a tree $\mathbf{T}$. Each node x in this tree is labeled with a set $\mathcal{L}(x)$ containing elements of the form $(a : C, I, a' : C')$, where C and C' are concept descriptions, a and a' are individual names, and I is a set of axiom indices. An element $(a : C, I, a' : C')$ has the following intended meaning: individual a belongs to concept description C due to the application of an expansion rule on C' and I contains the indices of the axioms where $a : C$ originates from.

$\mathbf{T}$ is initialized with a node x and $\mathcal{L}(x) = \{(a : C, \varnothing, nil)\}$. The algorithm expands $\mathbf{T}$ according to the rules in Table 1[5]. Concept descriptions involved in the expansion are converted to negation normal form. The algorithm terminates if no more expansion rules can be applied to tree nodes. $\mathbf{T}$ contains a *clash* if an individual a belongs simultaneously to concept descriptions C and $\neg C$, i.e. $(a : C, -, -) \in \mathcal{L}(x)$ and $(a : \neg C, -, -) \in \mathcal{L}(x)$.[6]

A clash in an expansion tree does not always mean unsatisfiability. If an individual a in a model tree for a concept A belongs to a value restriction $\forall R.C$, where C is unsatisfiable, but a has no R-successors, then this value restriction does not cause unsatisfiability of A. On the other hand, with the advent of instances or subconcepts of A which have R-successors this value restriction invokes unsatisfiability (cf. Wang et al., 2005).

Definition 2 *Minimal clash-preserving sub-TBox (MCPS)*
Let A be a concept for which its model tree obtained relative to a terminology $\mathcal{T}$ contains clashes. A sub-TBox $\mathcal{T}' \subset \mathcal{T}$ is a MCPS of A, if a model tree for A towards $\mathcal{T}'$ contains clashes and a model tree of A towards every $\mathcal{T}'' \subset \mathcal{T}'$ contains no clashes.

The union of the axiom indices in I of all clash elements in the model tree of A corresponds to MUPS[7] of A in Schlobach and Cornet (2003) and constitutes MCPS of A in Lam et al. (2006). Given a specific clash (e_1, e_2), MCPS$_{(e_1,e_2)}(A)$ is similarly

[5] Tags are provided to avoid circularity in the expansion of concept definitions.
[6] "$-$" is a placeholder. It stands for any value.
[7] Concerning unsatisfiablitily of A.

Table 1: Tableau expansion rules.

Rule $\equiv+$	if $A_i \equiv C_i \in \mathcal{T}$, and $(a : A_i, I, a' : A')$ is not tagged, then $tag((a : A_i, I, a' : A'))$ and $\mathcal{L}(x) := \mathcal{L}(x) \cup \{(a : C_i, I \cup \{i\}, a : A_i)\}$
Rule $\equiv-$	if $A_i \equiv C_i \in \mathcal{T}$, and $(a : \neg A_i, I, a' : A')$ is not tagged, then $tag((a : \neg A_i, I, a' : A'))$ and $\mathcal{L}(x) := \mathcal{L}(x) \cup \{(a : \neg C_i, I \cup \{i\}, a : \neg A_i)\}$
Rule $\sqsubseteq$	if $A_i \sqsubseteq C_i \in \mathcal{T}$, and $(a : A_i, I, a' : A')$ is not tagged, then $tag((a : A_i, I, a' : A'))$ and $\mathcal{L}(x) := \mathcal{L}(x) \cup \{(a : C_i, I \cup \{i\}, a : A_i)\}$
Rule $\forall$	if $(a : \forall R.C, I, a' : A') \in \mathcal{L}(x)$, and the above rules cannot be applied, then if there is $(b : D, J, a : \forall R.D) \in \mathcal{L}(x)$, then $\mathcal{L}(x) := \mathcal{L}(x) \cup \{b : C, I, a : \forall R.C\}$ else $\mathcal{L}(x) := \mathcal{L}(x) \cup \{b : C, I, a : \forall R.C\}$, where b is a new individual name

defined as in Definition 2 except for $\mathrm{MCPS}_{(e_1,e_2)}(A)$ preserving only one clash (e_1, e_2), but not all clashes as $\mathrm{MCPS}(A)$. To trace clashes we need to introduce the following definition.

Definition 3 *Trace*
Given an element $e = (a_0 : C_0, I_0, a_1 : C_1)$ in a set $\mathcal{L}(x)$, the trace of e is a sequence of the form $\langle(a_0 : C_0, I_0, a_1 : C_1), (a_1 : C_1, I_1, a_2 : C_2), \ldots, (a_{n-1} : C_{n-1}, I_{n-1}, a_n : C_n), (a_n : C_n, \varnothing, nil)\rangle$, where $I_{i-1} \subseteq I_i$ for each $i \in \{1, ..., n\}$ and every element in the sequence belongs to $\mathcal{L}(x)$.

Note that the expansion rules in Table 1 coincide with Lam et al. (2006), except for the Rule $\forall$, which obviously does not change crucial properties of the algorithm like complexity, decidability etc. Therefore, the properties of the original algorithm in Lam et al. (2006) are also relevant for our algorithm.

5.2 Types of Clashes

First of all, it is important to understand which solution for resolving clashes is appropriate from a pragmatic point of view. In order to achieve this, we return to our running examples. Concerning Example 1 it seems to be obvious that the axiom $Bird \sqsubseteq CanFly$ has to be modified, since this axiom contains overgeneralized knowledge. Simply deleting this axiom would result in the loss of the entailments $Bird \sqsubseteq CanMove$ and $Canary \sqsubseteq CanFly$, although both entailments do not contradict with the axiom $Penguin \sqsubseteq \neg CanFly$. A natural idea is to replace the problematic part of the overgeneralized definition of the concept $Bird$ (namely $CanFly$) with its least subsumer, that does not conflict with $Penguin$. In our example, the concept description $CanMove$ is precisely such a subsumer. Unfortunately, the simple replacement of $CanFly$ by $CanMove$ in Axiom 1 is not sufficient to preserve the entailment $Canary \sqsubseteq CanFly$.

We suggest therefore to introduce a new concept $FlyingBird$ that preserves the previous meaning of $Bird$ and subsumes its former subconcepts:

1.	$Bird \sqsubseteq CanMove$	2.	$CanFly \sqsubseteq CanMove$
3.	$Canary \sqsubseteq FlyingBird$	4.	$Penguin \sqsubseteq Bird$
5.	$Penguin \sqsubseteq \neg CanFly$	6.	$FlyingBird \sqsubseteq Bird$
7.	$FlyingBird \sqsubseteq CanFly$		

The situation is different for multiple overgeneralizations. A relevant solution for Example 2 is to replace the overgeneralized definitions $\forall likes.Icecream$ and $\forall likes.Chocolate$ with their least common subsumer $\forall likes.Sweetie$. The resulting axiom $Child \sqsubseteq \forall likes.Sweetie$ claims that children like only sweeties without specifying it:

1. $Child \sqsubseteq \forall likes.Sweetie$ 2. $Icecream \sqsubseteq Sweetie$ 3.$Chocolate \sqsubseteq Sweetie$
4. $Icecream \sqsubseteq \neg Chocolate$ 5. $Chocolate \sqsubseteq \neg Icecream$

The examples make clear that it is a non-trivial practical question of how multiple and single overgeneralizations can be distinguished. A single overgeneralization occurs, if some concept is too specifically defined and an exception contradicts with this definition. In the case of multiple overgeneralizations, two or more definitions of the same concept are too specific and conflict with each other. Unfortunately, it seems to be impossible to define this distinction purely logically, since this distinction is just a matter of human expert interpretation.

From a practical perspective it turns out that multiple overgeneralizations occur, if a concept is subsumed by two or more concepts that are explicitly defined as disjoint in the ontology (cf. Example 2). This case has a certain structural similarity to the polysemy problem, where an unsatisfiable concept is also subsumed by different disjoint concepts (cf. the tree example in Section 3). Practically, polysemy can be distinguished from multiple overgeneralizations by taking into account the level of abstraction of the disjoint concepts. In the case of polysemy, the disjoint superconcepts of the unsatisfiable concept usually occur in the upper structure of the taxonomy tree, whereas multiple overgeneralizations occur on lower levels of the taxonomy.

Definition 4 defines the abstraction level of concepts. The abstraction level of a concept towards a model tree is the number of steps in the shortest path from this concept to a most general (undefined) concept in the tableau extension procedure.

Definition 4 *Given a set $\mathcal{L}(x)$ that was obtained relative to a terminology $\mathcal{T}$ and an element $(a : C_1, I_1, a_0 : C_0) \in \mathcal{L}(x)$, the abstraction level $\mathbf{L}(C_1)$ is defined as the minimal cardinality of the sequences $\langle (a : C_1, I_1, a_0 : C_0), ..., (a : C_n, I_n, a : C_{n-1}) \rangle$ where $\forall i \in \{1, ..., n\} : [(a : C_i, I_i, a_{i-1} : C_{i-i}) \in \mathcal{L}(x)$ and $I_i \subseteq I_{i-1}]$ and there is no concept D such that $C_n \sqsubseteq D \in \mathcal{T}$ or $C_n \equiv D \in \mathcal{T}$.*

Using Definition 4 it is possible to distinguish the two types of inconsistencies formally (cf. Definition 5): If a concept is subsumed by two other concepts that are defined to be disjoint[8] and the abstraction level of these concepts is higher than a user-defined

[8]For the sake of simplicity Definition 5 concerns multiple overgeneralization with only two concepts.

distinctive abstraction level, then this case is considered to be polysemous. If the abstraction level is below the user-defined abstraction level, then we are dealing with multiple overgeneralizations. Finally, if the clash is not produced by explicitly disjoint concepts, then we face the case of single overgeneralization.

Definition 5 *Given a clash* $(a : C, -, -), (a : \neg C, -, -)$ *from a set* $\mathcal{L}(x)$ *that was obtained relative to a terminology* $\mathcal{T}$ *and a distinctive abstraction level* l*, the following cases can be distinguished:*

- *If there exists a concept* D *such that* $(a : D, -, -), (a : \neg D, -, -) \in \mathcal{L}(x)$ *and* $D \sqsubseteq \neg C \in \mathcal{T}$ *and* $C \sqsubseteq \neg D \in \mathcal{T}$*, then*
  - *If* $\mathtt{max}(\mathbf{L}(C), \mathbf{L}(D)) \geq l$ *this clash is polysemous,*
  - *Else this clash is a multiple overgeneralization,*
- *Else this clash is a single overgeneralization.*

In the following subsection, we will discuss resolution aspects of the mentioned types of clashes.

5.3 Resolving Clashes

Unfortunately, it is impossible to resolve polysemy problems automatically without an appeal to external knowledge. After splitting the problematic concept (e.g. *Tree* in the example of Section 3) into two concepts with different names (e.g. *TreeStructure* and *TreePlant*) it is necessary to find out which one of the definitions and subconcepts of the original concept refers to which of the new concepts. This can be done either by the ontology engineer or with the help of additional knowledge about the usage context of this concept in external resources. Since this paper is concerned with logical aspects of ontology adaptation only, we do not consider this problem here.

As already mentioned above, multiple overgeneralizations can be repaired by replacing conflicting definitions with their least common subsumer. In order to find a least common subsumer, we need to calculate subsumers for concepts. Fact 1 characterizes subsumers computationally.

Fact 1 *Given a set* $\mathcal{L}(x)$ *obtained relative to a terminology* $\mathcal{T}$ *and concept* C *such that* $(a : C, -, -) \in \mathcal{L}(x)$*, a concept* C' *is a subsumer of* C *towards* $\mathcal{T}$ *if*

- $\exists e = (a : C', -, -) : (a : C, -, -) \in Trace(e)$ *or*
- $\exists e = (a : \forall R.D, -, -) : (a : C, -, -) \in Trace(e)$ *and* $C' \doteq \forall R.D'$ *such that* D' *is atomic and a subsumer of* D*.*

If C *is satisfiable towards* $\mathcal{T}$*, then the other direction of the implication does also hold.*

Fact 1[9] claims that a concept C' is a subsumer of a concept C, if it was added to a node a in the tableau expansion procedure after C or if C is subsumed by a relational

[9]The proofs of Fact 1 and further facts below are not presented in detail due to space limitations. We will rather provide sketches of proof ideas.

restriction $\forall R.D$ and C' is a relational restriction on R with a scope D' subsuming D. General axioms with a complex concept definition on the right side cannot occur in our restricted logic. Therefore if C is satisfiable, then no inferred subsumption that is not explicitly expressed in the TBox is possible. If C is unsatisfiable, then it is subsumed by any concept. As the reader will see hereinafter, we are interested only in cases where C is satisfiable.

According to Fact 2, given a set $\mathcal{L}(x)$, a lcs for two satisfiable concepts C_1 and C_2 occurring in $\mathcal{L}(x)$ can be characterized as a minimal concept subsuming both, C_1 and C_2 towards $\mathcal{T}$.

Fact 2 *Given a set $\mathcal{L}(x)$ obtained relative to a terminology $\mathcal{T}$ and concepts C_1 and C_2 satisfiable towards $\mathcal{T}$, such that $(a : C_1, -, -) \in \mathcal{L}(x), (a : C_2, -, -) \in \mathcal{L}(x)$ a concept L is a least common subsumer for C_1 and C_2 towards $\mathcal{T}$ if and only if the following two conditions hold:*

1. *$L \in subsumers_{\mathcal{T}}(C_1) \cap subsumers_{\mathcal{T}}(C_2)$.*
2. *For every concept L' that satisfies condition 1 and is not equal to L: $\mathcal{T} \not\models L' \sqsubseteq L$.*

Fact 1 ensures that L satisfies condition 1 of Definition 1 (recall that the set $subsumers_{\mathcal{T}}(C)$ is the exhaustive set of concept descriptions subsuming C towards $\mathcal{T}$ provided C is satisfiable towards $\mathcal{T}$). Obviously, L satisfies also condition 2 of Definition 1. Thus, in order to resolve a clash produced in a model tree of a concept A by two overgeneralized definitions C_1 and C_2, it is sufficient to delete axioms $A \sqsubseteq C_1$ and $A \sqsubseteq C_2$ from the terminology and add axioms from the set $\{A \sqsubseteq L \mid L \textit{ is a } lcs(C_1, C_2)\}$. If concepts C_1 and C_2 themselves are unsatisfiable, then they should be repaired before A. Compare Section 7 for more remarks concerning this issue.

Now we examine the problem of resolving the case of single overgeneralization. Lam et al. (2006) shows that removing any of the axioms appearing in the clash traces is sufficient to resolve the clash. An important practical question concerns the choice of the axiom to be removed or modified. In the literature, a lot of ranking criteria were suggested for this task on ontology debugging (Schlobach and Cornet, 2003; Kalyanpur, 2006; Lam et al., 2006; Haase and Stojanovic, 2005):

- *Arity* of an axiom α denotes in how many clashes α is involved. The higher the arity is, the lower is the rank of α.
- *Semantic impact* of α denotes how many entailments are lost if α is removed. Axioms with a high semantic impact are ranked higher.
- *Syntactic relevance* denotes how often concept and role names occurring in an axiom α are used in other axioms in the ontology. Axioms containing elements that are frequently occurring in the ontology are ranked higher.
- *Manual ranking* of α can be provided by the ontology engineer.

- *Frequency ranking* of α is used in approaches to semi-automatic ontology extraction and denotes how often concepts and roles in α occur in external data sources.

In this paper, we do not discuss ranking strategies and suppose that one of these strategies has been applied and the problematic axiom to be removed or rewritten has been chosen. Assume a concept description C is chosen to be removed from an axiom α, in order to resolve a clash in a model tree for a concept A. The next questions is: How can we find an appropriate concept description C' that can resolve the clash by replacing C? We are looking for a replacement that resolves the clash, does not cause new clashes or entailments, and preserves as many entailments implied by $\mathcal{T}$ as possible. Definition 6 defines such a replacement.

Definition 6 *Minimal nonconflicting substitute (MNS)*
Assume the following is given: a terminology $\mathcal{T}$, a clash $e_1 = (a : X, -, -), e_2 = (a : \neg X, -, -)$ in the model tree for a concept A, a concept C satisfiable towards $\mathcal{T}$[10] *that is chosen to be removed from an axiom α_i ($\alpha_i \in \mathcal{T}$ and $\exists j \in \{1, 2\} : (a : C, \{i, -\}, -) \in Trace(e_j)$), and $\mathcal{T}'' := \mathcal{T} \setminus \{\alpha_i\}$. Let an axiom α' be obtained from α_i by replacing C with a concept C' and $\mathcal{T}' := \mathcal{T} \setminus \{\alpha_i\} \cup \{\alpha'\}$. C' is a minimal nonconflicting substitute (MNS) of C if the following conditions hold:*

1. *A model tree for A towards $\mathcal{T}'$ contains the same number of clashes as a model tree for A towards $\mathcal{T}''$.*

2. *If $C' \neq \top$, then there exists an entailment β, such that $\mathcal{T} \models \beta$, $\mathcal{T}'' \not\models \beta$, and $\mathcal{T}' \models \beta$.*

3. *There exists no entailment β such that $\mathcal{T} \not\models \beta$ and $\mathcal{T}' \models \beta$.*

4. *There exists no concept description C'' with the same properties of C', such that C'' preserves more entailments from $\mathcal{T}$.*

Condition (1) guarantees that MNS resolves the clash (e_1, e_2) in which α_i and C are involved and does not introduce new clashes. Due to condition (2) MNS preserves at least one entailment from $\mathcal{T}$ that would be lost with the removal of C. Condition (3) excludes new entailments that are not implied by $\mathcal{T}$ and condition (4) guarantees that MNS preserves as much information as possible.

Fact 3 *Given a clash $e_1 = (a : X, -, -), e_2 = (a : \neg X, -, -)$ obtained from a set $\mathcal{L}(x)$ relative to a terminology $\mathcal{T}$ and a concept C satisfiable towards $\mathcal{T}$ that is chosen to be removed from an axiom α_i ($\alpha_i \in \mathcal{T}$ and $\exists j \in \{1, 2\} : (a : C, \{i, -\}, -) \in Trace(e_j)$), a concept C' is a MNS of C if and only if the following conditions hold:*

1. *C' subsumes C towards $\mathcal{T}$.*

[10] Notice again: if C is unsatisfiable, then it should be repaired before A. Compare Section 7 for more information.

2. *If C is subsumed by X or $\neg X$ towards $\mathcal{T}$, then C' is not subsumed by X or $\neg X$ towards $\mathcal{T}$.*

3. *C' preserves at least one entailment from $\mathcal{T}$ or $C' := \top$.*

4. *There exists no concept description $C'' \neq C'$ with the same properties, such that C' subsumes C'' towards $\mathcal{T}$.*

Due to conditions (1) and (2) C' satisfies conditions (1) and (3) of Definition 6 (since C' is not subsumed by a conflicting definition, it does not reconstruct the clash). Condition (3) guarantees that C' satisfies condition (2) of Definition 6. Finally condition (4) corresponds to (4) in Definition 6.

We reconsider Example 1. The model tree for the concept *Penguin* in this example consists of the following elements:

$$\begin{aligned}\mathcal{L}(x) \quad = \quad & \{(a : Penguin, \varnothing, nil), (a : Bird, \{4\}, a : Penguin),\\ & (a : \neg CanFly, \{5\}, a : Penguin), (a : CanFly, \{4, 1\}, a : Bird),\\ & (a : CanMove, \{4, 1, 2\}, a : CanFly)\}\end{aligned}$$

Thus, the set of problematic axioms is $\{1, 4, 5\}$. Suppose the concept *CanFly* is chosen to be removed from axiom 1. According to Fact 3 *CanMove* is MNS of *CanFly*. If *CanFly* is replaced by *CanMove* in Axiom 1, then the entailments of the form $X \sqsubseteq CanFly$, where X is a subconcept of *Bird* (for example, *Canary*), would be lost. Such situations are undesirable, because the clash $(CanFly, \neg CanFly)$ concerns only the conflict between the overgeneralized concept *Bird* and the exception *Penguin*. In order to keep the entailments, we suggest to introduce a new concept *FlyingBird* to the terminology which will capture the original meaning of *Bird* (cf. Section 5.2).

6 Adaptation Algorithm

If a new axiom α is added to a terminology $\mathcal{T}$, then the proposed algorithm constructs model trees for every atomic concept X that is defined in $\mathcal{T}$. Model trees containing clashes are used for ontology repairing. Using Definition 5 the algorithm distinguishes between inconsistency types. Since polysemy can not be repaired without external knowledge, this problem is only reported to the user. Multiple overgeneralizations are repaired by replacing the conflicting definitions with their least common subsumer.

With respect to single overgeneralization, for every clash (e', e'') a concept description F from the trace of some clash element $e \in \{e', e''\}$ is chosen to be rewritten in an axiom β ($\beta \doteq E \sqsubseteq F$ or $\beta \doteq E \equiv F$) according to a given ranking (see section 5.3). F is replaced with its minimal nonconflicting substitutes in β. A new concept E^{new} is introduced to capture the original semantics of E. The name E^{new} is constructed automatically from the original name E and the problematic concept F. The set T consists of elements from $\mathcal{L}(x)$ that are contained in the trace of the element e between the unsatisfiable concept X and the rewritten concept E.

The **split&replace** procedure splits atomic concepts from $\mathcal{L}(x)$ that are involved in the clash above. Concepts appearing in the trace of e earlier are split first. If a definition of B_1 was rewritten and B_1 was split into B_1^{new} and B_1 and a concept B_2 is going to be split immediately after B_1, then B_1 in the definition of a new concept name B_2^{new} is replaced with B_1^{new}.

Algorithm Adapt a satisfiable terminology $\mathcal{T}$ to a new axiom α, given a distinctive abstraction level l

```
$\alpha \doteq A \sqsubseteq B$ or $\alpha \doteq A \equiv B$, add $\alpha$ to $\mathcal{T}$
for all axioms $\alpha' \doteq X \sqsubseteq Y$ or $\alpha' \doteq X \equiv Y$, $\alpha' \in \mathcal{T}$
if $X$ is unsatisfiable towards $\mathcal{T}$ then
  for all clashes $(e', e'')$ in the sets $\mathcal{L}(x)$ of the model tree for $X$ where
                                   $e' = (a : C, -, -), e'' = (a : \neg C, -, -)$
    if $\exists D : (a : D, -, -), (a : \neg D, -, -) \in \mathcal{L}(x)$ and $\{D \sqsubseteq \neg C, C \sqsubseteq \neg D\} \subset \mathcal{T}$
    then if $\mathtt{max}(\mathbf{L}(C), \mathbf{L}(D)) \geq l$ then report polysemy
         else remove $D_1, D_2$ where
                          $D_{i \in \{1,2\}}$ are last but one elements in the traces of $e', e''$
              for all $lcs(D_1, D_2)$ add $A \sqsubseteq lcs(D_1, D_2)$ to $\mathcal{T}$ end for
    else
      choose an $\beta \in \mathcal{T}$ ($\beta \doteq E \sqsubseteq F$ or $\beta \doteq E \equiv F$) such that
      $\exists e \in \{e', e''\}$ : $(b : F, -, -) \in Trace(e)$ to be rewritten acc. to ranking
      remove $\beta$ from $\mathcal{T}$
      for all MNS($F$)
        $\beta^{new}$ is obtained from $\beta$ by replacement of $F$ with MNS($F$)
        add $\beta^{new}$ to $\mathcal{T}$
      end for
      add $E^{new} \sqsubseteq E, E^{new} \sqsubseteq F$ to $\mathcal{T}$
      let $T$ be a subsequence of $Trace(e)$
      between the elements $(a : E, -, -)$ and $(b : X, -, -)$ (not inclusive)
      split&replace($E$, $E^{new}$, $T$)
  end for

Subroutine split&replace($A, A^{new}$, $T$)
$(b : B', -, a : B)$ is the next element of $T$ and $B$ is atomic
$B''$ is obtained by replacing $A$ with with $A^{new}$ in $B'$
if $B \sqsubseteq B' \in \mathcal{T}$ then add $B^{new} \sqsubseteq B''$ to $\mathcal{T}$
else add $B^{new} \equiv B''$ to $\mathcal{T}$
for all $\gamma \in \mathcal{T}$ such that $\gamma$ is not the next axiom in $T$
    replace $B$ with $B^{new}$ in the right part of $\gamma$
end for
split&replace($B, B^{new}$, $T$)
```

Example 3 shows the application of the algorithm. The concept *Transport* in axiom

2 is chosen to be rewritten. It is easy to see that the proposed algorithm extends the semantics of the "split" concepts, whereas the semantics of other concepts remains unchanged. New concept names ($TransportAirplane$, $AviatesTransportAirplanePilot$) are constructed automatically.

Example 3
Original terminology:

1. $Pilot \sqsubseteq \forall aviates.Airplane$ 2. $Airplane \sqsubseteq Transport$
3. $PassengerPlane \sqsubseteq Airplane$ 4. $FighterPilot \sqsubseteq Pilot$
5. $FighterPilot \sqsubseteq \forall aviates.FightingMachine$ 6. $FightingMachine \sqsubseteq \neg Transport$

Changed terminology:

1. $Pilot \sqsubseteq \forall aviates.Airplane$
2. $PassengerPlane \sqsubseteq TransportPlane$
3. $FighterPilot \sqsubseteq Pilot$
4. $FighterPilot \sqsubseteq \forall aviates.\neg FightingMachine$
5. $FightingMachine \sqsubseteq \neg Transport$
6. $TransportAirplane \sqsubseteq Airplane$
7. $TransportAirplane \sqsubseteq Transport$
8. $AviatesTransportAirplanePilot \sqsubseteq Pilot$
9. $AviatesTrasportAirplanePilot \sqsubseteq \forall aviates.TrasportAirplane$

7 Root and Derived Concepts

It is easy to verify that the result of the application of our algorithm is dependent on the order the concepts are input into the debugger. If axioms are added one by one, then nothing in the procedure needs to be changed. But if a set of axioms is added to the ontology, then it is interesting to see whether it is reasonable to reorder axioms in this set. Unsatisfiable concepts can be divided into two classes:

1. *Root concepts* are atomic concepts for which a clash found in their definitions does not depend on a clash of another atomic concept in the ontology.

2. *Derived concepts* are atomic concepts for which a clash found in their definitions either directly (via explicit assertions) or indirectly (via inferences) depends on a clash of another atomic concept (Kalyanpur, 2006).

In order to debug a derived concept, it is enough to debug corresponding root concepts. Thus, it is reasonable to debug only the root concepts. The technique of distinguishing between root and derived concepts was proposed in Kalyanpur (2006). We integrate this technique into our approach.

Definition 7 *An atomic concept A is* **derived** *from an atomic concept A' if there exists a clash (e_1, e_2) in the model tree of A such that $MCPS_{(e_1,e_2)}(A')$ is a subset of*

$MCPS_{(e_1,e_2)}(A)$. If there is no concept A' from which A is derived, then A is a **root** *concept.*

Fact 4 shows how to find dependencies between problematic concepts.

Fact 4 *Given a clash (e_1, e_2) obtained from a set $\mathcal{L}(x)$ for a concept A, A is derived from a concept A' (towards this clash) if and only if $\exists(b : A', -, -) \in Trace_{\mathcal{L}(x)}(e_1) : (b : A', -, -) \in Trace_{\mathcal{L}(x)}(e_2)$.*

$\mathrm{MCPS}_{(e_1,e_2)}(A')$ is a subset of $\mathrm{MCPS}_{(e_1,e_2)}(A)$ if and only if the model tree for A' is a subset of the model tree for A and both of these trees contain the clash (e_1, e_2) (modulo differences in the axiom indices set). Obviously, A' is in the traces of the clash elements in its own model tree. Therefore, if A is derived from A' towards (e_1, e_2), then A' is in the traces of e_1 and e_2 in $\mathcal{L}(x)$. On the other hand, if A' occurs in the traces of e_1 and e_2, then the model tree of A' contains this clash and is a subset of $\mathcal{L}(x)$.

Using Fact 4 it is possible to construct a directed graph with atomic concepts as nodes and arrows denoting derivation. It can happen that two concepts are simultaneously derived from each other (for example $\{A \sqsubseteq D,\ A \sqsubseteq \neg D,\ B \sqsubseteq D,\ B \sqsubseteq \neg D,\ A \equiv B\}$). In this case, it is necessary to debug both of the derived concepts.

8 Conclusion and Future Work

In this paper, we presented an approach for dynamically resolving conflicts appearing in automatic ontology learning. This approach is an integration of ideas proposed in Lam et al. (2006) and Ovchinnikova and Kühnberger (2006) extended for the subset of description logics used in practically relevant systems for ontology learning (Haase and Stojanovic, 2005). Our algorithm detects problematic axioms that cause a contradiction, distinguishes between different types of logical inconsistencies and automatically repairs the terminology. This approach is knowledge preserving in the sense that it keeps as many entailments implied by the original terminology as possible.

In Ovchinnikova et al. (2007), a prototypical implementation of the idea of splitting overgeneralized concepts in $\mathcal{ALE}$-DL was discussed. This implementation was tested on the famous wine-ontology[11] that was automatically extended with new classes extracted from text corpora with the help of the Text2Onto[12] tool. Several cases of overgeneralization were detected and correctly resolved[13].

In the near future, we plan to test the prototype implementation of the proposed algorithm on existing real-life ontologies. It is of particular interest to see to what extent statistical information about the distribution and co-occurrence of concepts in texts can help to improve the adaptation procedure for making it more adequate to human intuition.

[11] `http://www.w3.org/TR/owl-guide/wine.owl`

[12] `http://ontoware.org/projects/text2onto/`

[13] For example, the class *LateHarvest* originally defined to be a sweet wine was claimed to be overgeneralized after an exception *RieslingSpaetlese* which was defined to be a late harvest wine and a dry wine appeared.

Acknowledgments

This research was partially supported by grant MO 386/3-4 of the German Research Foundation (DFG).

References

Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., and Patel-Schneider, P. F., editors (2003). *The Description Logic Handbook: Theory, Implementation, and Applications.* Cambridge University Press, New York.

Baader, F. and Hollunder, B. (1995). Embedding Defaults into Terminological Representation Systems. *J. Automated Reasoning*, 14:149–180.

Baader, F. and Küsters, R. (2006). Non-standard Inferences in Description Logics: The Story So Far. In Gabbay, D. M., Goncharov, S. S., and Zakharyaschev, M., editors, *Mathematical Problems from Applied Logic I. Logics for the XXIst Century*, volume 4 of *International Mathematical Series*, pages 1–75. Springer.

Baader, F., Lutz, C., Milicic, M., Sattler, U., and Wolter, F. (2005). Integrating Description Logics and Action Formalisms: First Results. In *Proceedings of the 2005 International Workshop on Description Logics (DL2005)*, CEUR-WS.

Fanizzi, N., Ferilli, S., Iannone, L., Palmisano, I., and Semeraro, G. (2004). Downward Refinement in the $\mathcal{ALN}$ Description Logic. In *HIS '04: Proc. of the Fourth International Conference on Hybrid Intelligent Systems (HIS'04)*, pages 68–73, Washington, DC, USA. IEEE Computer Society.

Haase, P. and Stojanovic, L. (2005). Consistent evolution of owl ontologies. In *Proc. of the Second European Semantic Web Conference*, pages 182–197.

Kalyanpur, A. (2006). *Debugging and Repair of OWL Ontologies. Ph.D. Dissertation.* University of Maryland College Park.

Lam, S. C., Pan, J. Z., Sleeman, D. H., and Vasconcelos, W. W. (2006). A Fine-Grained Approach to Resolving Unsatisfiable Ontologies. In *Web Intelligence*, pages 428–434.

Ovchinnikova, E. and Kühnberger, K.-U. (2006). Adaptive $\mathcal{ALE}$-Tbox for Extending Terminological Knowledge. In *19th Australian Joint Conference on ArtificialIntelligence*, pages 1111–1115.

Ovchinnikova, E., Wandmacher, T., and Kühnberger, K.-U. (2007). Solving Terminological Inconsistency Problems in Ontology Design. *International Journal of Interoperability in Business Information Systems (IBIS)*, 2(1):65–80.

Perez, G. A. and Mancho, M. D. (2003). A Survey of Ontology Learning Methods and Techniques. OntoWeb Delieverable 1.5.

Schlobach, S. and Cornet, R. (2003). Non-Standard Reasoning Services for the Debugging of Description Logic Terminologies. In *IJCAI*, pages 355–362.

Wang, H., Horridge, M., Rector, A. L., Drummond, N., and Seidenberg, J. (2005). Debugging OWL-DL Ontologies: A Heuristic Approach. In *International Semantic Web Conference*, pages 745–757.

Eduardo Torres Schumann, Uwe Mönnich, Klaus U. Schulz

Integration Languages for Data-Driven Approaches to Ontology Population and Maintenance

Populating an ontology with a vast amount of data and ensuring the quality of the integration process by means of human supervision seem to be mutually exclusive goals that nevertheless arise as requirements when building practical applications. In our case, we were confronted with the practical problem of populating the EFGT Net, a large-scale ontology that enables thematic reasoning in different NLP applications, out of already existing and partly very large data sources, but on condition of not putting the quality of the resource at risk. We present here our particular solution to this problem, which combines, in a single tool, on one hand an integration language capable of generating new entries for the ontology out of structured data with, on the other hand, a visualization of conflicting generated entries with online ontology editing facilities. This approach appears to enable efficient human supervision of the population process in an interactive way and to be also useful for maintenance tasks.

1 Introduction

Ontologies play a key role in the Semantic Web and Knowledge Management research as a way to model the domain of application and in order to achieve an integrated access to heterogeneous data sources. Building an ontology has also become usual when developing resources for Natural Language Processing (NLP) applications, due to the need of representing meaning traditionally compiled in lexica or thesauri in a form suitable for manipulation by the computer. Although there are some differences between ontologies used in the different scenarios – e.g. some ontologies used in NLP applications are referred to as *light weight ontologies* because their lack of a specification in a formal ontology language –, they all share the fact that the process of ontology development requires considerable human effort. Moreover, ontologies have to be refined and maintained regularly in an iterative, data-driven process called the ontology learning cycle (Maedche and Staab, 2001). This tasks constitute a determining factor in the development of NLP applications and have also been identified as the "bottleneck on the way to the semantic web". If a large amount of existing data is intended to be integrated through ontologies, approaches need to work in a data-driven, automatic way but at the same time enable efficient human supervision.

The work we report on here is carried out in the framework of a larger project that aims to encode vast encyclopedic and common purpose knowledge deposited in a knowledge base called the EFGT Net, which is intended to be used for different NLP applications like semantic annotation and thematic reasoning. The EFGT Net is managed by means of a formal language been specially designed for this purpose. Rather than providing an exhaustive, precise definition of the formal meaning of a concept, the aim of this language is to place concepts in the thematic space represented by the EFGT Net. One main advantage of the EFGT Net language it can easily handle nets with more than 10^5 concepts. Further details on motivations, design guidelines and scientific ideas behind the EFGT Net are provided by Schulz and Weigel (2003).

As one branch of development, we wished to populate the EFGT Net with large amounts of already available data, such as legacy data from previous projects, public available data like GeoNames (2007) and data automatically extracted from document resources like the Wikipedia (Wikipedia, 2007). These data sources were given in different formats, such as tables and XML. The translation of data into entries of the net turned out to be a non-trivial and interesting task. Two questions became central:

1. How to specify a computable mapping from data of distinct formats to possible concepts of the ontology?
2. How to support the user in the task of deciding which of the generated concept candidates harmonize with the resource and should be incorporated to the ontology?

Here, we introduce our technical solution to these problems, the so-called Upload Tool. As to the first question, our approach allows the user to define templates of a specific form. In the simplest case, a template can be seen as a formal definition of a new concept, expressed in an extension of the EFGT Net language with the help of variables. The template specifies how the sequence of template variables is mapped to some tuple of concept names or attributes found in the textual input data. From each image tuple found in the data, a new instantiation of the template is obtained, which gives rise to a new concept definition which can be added to the ontology. More complex templates simultaneously define several new concepts from one tuple of data. Variants of the template syntax address the problem of defining data tuples and mappings for input data coming in different formats (tables, XML). This kind of extension of the ontology specification language for template definition and data integration is what we call here an *integration language*.

In many cases, some of the "new" concepts that are obtained from template instantiation will be already defined in the ontology. Hence, to avoid inconsistencies, data integration needs some form of manual control. Note that both the amount of data to be integrated as well as the number of concepts in the resource may be very large. As a consequence, methods and interfaces that facilitate efficient human supervision of the population process are essential when providing an appropriate answer to the second question. In our approach, candidate entries are aligned with the ontology, calculating existing entries that seem close to the candidate concept, either from a logical or linguistic point of view. A visualization of the alignment results with integrated ontology editing facilities allow the user to handle conflicts on-the-fly. This kind of immediate visual feedback about the state of the ontology with respect to certain data is also useful for tracking changes in the modeling of entries and other maintenance tasks.

An implementation of the Upload Tool has been applied for substantially extending a core ontology. By using already existing lists and data extracted specially for this purpose, we were able to incorporate to the net some 10^4 new concepts representing common named entities, most of them geographical names and names of famous people.

The paper is structured as follows. Section 2 covers related work. Section 3 introduces the formal language for concept definition in the EFGT Net, which is extended in Sec. 4 to a language for defining templates. Section 5 explains how to instantiate templates given suitable tables or XML data. Possible conflicts arising during alignment of generated concept definitions are described in Sec. 6. The Upload Tool and interactive support mechanisms are described in Sec. 7. Section 8 discusses on future work and other possible applications for templates.

2 Related Work

Obviously, the kind of integration language proposed here has many commonalities with data transformation languages such as XSLT (XSLT, 2006), since we use it to transform data from different formats into entries of the EFGT Net. Some distinctions should be made though. While transformation languages are designed to bring easily some specific data model or format into other formats, as e.g. XSLT is designed for transforming XML, our integration language is designed for the converse goal, i.e. bringing syntactically heterogeneous data into a single target model. As explained later, this is reflected in the syntax of our integration language, which has an "intensional" part based on the EFGT Net language used for constructing and querying the EFGT Net and an "extensional" part for extracting data given in different formats. The intensional part our integration language is comparable with known ontology query languages such as OWL-QL (Fikes et al., 2003) for querying ontologies stated in OWL or SPARQL (Prud'hommeaux and Seaborne, 2006) for RDF ontologies, which can also be regarded as extensions of the corresponding ontology specification format. In fact, in our approach the ontology is also queried during alignment in order to determine already existing concepts with the same logical representation as the generated entries. Since the logical representations generated by our templates are (almost) fully specified, the use of a full-fledged ontology query language like the ones mentioned above is unnecessary.

A way of obtaining structured data from heterogeneous, semi-structured data sources like web pages consists in defining wrappers (Laender et al., 2002). The extensional part of our integration language has an analogous function to the many existing specialized wrapper definition languages (WebL, 2005; Huck et al., 1998) and is used for specifying how to extract tuples from specific data files.

In work focusing on learning and populating ontologies from text (Buitelaar et al. (2005) gives an overview), integrated frameworks a have been described (Maedche and Volz, 2001) where the ontology engineer can edit the ontology and perform other maintenance tasks as she inspects the results from the learning component. This constitues a practical solution, as learning ontologies from text remains a difficult task and automatically produced results cannot be always simply adopted. In constrast, we don't learn new concepts but generate them out a given data set, sharing with the mentioned tools the idea of assisting the user during the population process.

Sophisticated interactive methods for user guidance can also be found in work on methods for merging already existing ontologies. Although fully automatic approaches to this problem exist, see e.g. Ehrig et al. (2005), other approaches bank on human supervision of the integration process, such as PROMPT (Noy and Musen, 2000) and OntoMap (Maier et al., 2003). They share the idea of identifying anchors, i.e., corresponding concepts in the target ontologies, and the merge the structure of both ontologies basing on heuristics. Unclear cases together with possible pertinent merging operations are are displayed and the user has to take a decision for the next merging step. Anchors are manually stated (OntoMap) or identified automatically based on linguistical similarity (PROMPT). Although we also exploit linguistical simliarity during the alignment of the generated entries, the integration approach presented here is fundamentally different because, instead of heuristically merging ontology structures, we map structural relationships encoded in data files to ontological relationships.

3 The EFGT Net Language - An Overview

The EFGT Net formalism is presented in this section on an informal basis. We focus on its logical language, which will be extended to an integration language in the next sections. Concepts are captured in a EFGT Net by creating an entry consisting in

1. a unique identifier for the concept, its *ID String*, that determines the position of the concept in the net, and

2. *a set of attributes* holding a linguistical representation of the concept including the name of the concept in at least one of the supported languages, a list of synonyms, orthographical variants and flection forms, etc. as well as other data like a related URL, an associated period of time, etc.

For a set of entries, a sound underlying deduction calculus determines a directed acyclic graph (DAG) for the corresponding set of ID String, where nodes in the DAG represent the concepts and edges binary relations between concepts. Schulz and Weigel (2003) provide more details.

Figure 1 summarizes the syntax of ID Strings. Starting from the root element `()`, there are two operation for creating complex ID Strings for a new concept out of a given one. The first operation, a *local introduction*,

$$\begin{array}{lcl}
\mathbb{IDS}\text{tr} & := & (\,)\,|\,(\,\mathbb{T}\text{ype}\,\mathbb{IDS}\text{tr}\,.\,\mathbb{N}\text{um}\,)\,|\,(\,\mathbb{IDS}\text{tr}\,\&\,\mathbb{IDS}\text{tr}\,) \\
\mathbb{T}\text{ype} & := & (\,\mathtt{e}\,|\,\mathtt{E}\,|\,\mathtt{F}\,|\,\mathtt{g}\,|\,\mathtt{G}\,|\,\mathtt{t}\,|\,\mathtt{T}\,) \\
\mathbb{N}\text{um} & := & \mathbb{D}\,(\,\mathtt{0}\,|\,\mathbb{D}\,)^* \\
\mathbb{D} & := & (\,\mathtt{1}\,|\,\ldots\,|\,\mathtt{9}\,)
\end{array}$$

Figure 1: ID String syntax.

is used when the new concept can be sufficiently characterized as a set or a set element that narrows the meaning of a more general concept. For example, the concept "Cities" can be regarded as a set of geographical locations (different cities) narrowing the more general concept "Locations", and "Oslo" could be an element of the set "Cities". This is represented in the formal language by marking the ID String of the refined concept with a special type which corresponds to the kind of specialization (i.e. the "kind" of set or set element), as well as an index for enumeration. In the example, "Locations" could be coded as a refinement of the top node `()` by marking it with the type `G` of sets of geographical entities and the index `1`, i.e. `(G().1)`. Analogously, "Cities" could be coded as `(G(G().1).1)` and "Oslo" as the (third) element of "Cities" by using type `g` for geographical elements, `(g(G(G().1).1).3)`. The complete set of types, which motivates the name EFGT Net, contains:

- `E,e` Type `E` denotes a set of *E*ntities like `composers` whereas type `e` denotes a singleton entity like *J. S. Bach*.
- `F` Type `F` denotes a thematic *F*ield like *quantum physics*. Since every thematic field can be regarded as a set of subfields, there is no type `f`.
- `G,g` As mentioned before, type `G` denotes *G*eographical sets like *rivers* whereas type `g` stands for singleton geographic instances like *the Alps*.
- `T,t` Finally, type `T` denotes a set of *T*emporal periods like *epochs in art*, and type `t` denotes an individual temporal interval like *September 11th 2001*.

The other way to create a new ID String is by combining two ID Strings with the operator `&` to form a new one. This is called a *concept intersection*. If the component ID Strings are sets of the same type, the new concept stands for the intersection of the sets. When combining a set with an individual, the new ID String represents a subset where the individual acts as a modifier. In the remaining cases, the intersection represents a new thematic field, the exact meaning is left open. E.g. combining "Persons", `(E().1)`, and "Science", `(F(F().1).2)`, would yield "Persons in Science/ Scientists"), `((E().1)&(F(F().1).2))`. "European Countries" may result from joining the identifier for "Europe", `(g(...).1)`, and "Country", `(G(...).2)`, to `((g(...).1)&(G(...).2))`.

The attributes used in addition to the ID String for characterizing the concept are qualified by means of a *semantic type* indicating which kind of information it holds (birth date of a person, the number of inhabitants of a city, a company's web-page, etc.). Additionally, a *syntactic type* specifies how the information is conveyed (as a date, proper name, URL, etc.) and a *language type* the language used. The list of semantic and syntactic types is open and can be extended to accommodate different requirements.

4 From the EFGT Net Language to an Integration Language

In this section we start developing our particular integration language for the EFGT Net. We use it for stating *templates*, which specify how data can be integrated in the ontology. As a starting point, consider the very simple data example about Switzerland's geography compiled in Table 1. Suppose

Canton	District	Capital
Thurgau	Bezirk Weinfelden	Weinfelden
Thurgau	Bezirk Bischofszell	Bischofszell
Wallis	Bezirk Brig	Brig-Glis

Table 1: Geographical data about Switzerland

you want to add each district in the table to an EFGT Net already holding the cantons. For each canton, one may want to introduce a new set of type `G` for receiving the canton districts and then add each district with type `g` to the corresponding district set.

We first explain how templates without variables may be used to define new candidate entries. In our integration language, the template

$$({}_{districts}\,\texttt{G}[\text{Thurgau}].\texttt{n})$$
$$\textit{districts}.\texttt{name.en.name} = \text{“districts in canton Thurgau”}$$

constructs a suitable ID String for a new concept named "districts in canton Thurgau" under the concept "Thurgau" (we may later add the corresponding districts in a second step). The first line provides the skeletal structure of the ID String for the new concept expressed in the extended ID String language. It queries the net for the concept between the square brackets, “Thurgau”, takes its ID String and defines a new set of type `G` as a local introduction with a fresh index, represented by `n` in the expression. Assuming the ID String for “Thurgau” is `(g(G().2).3)` in the net, the ID String generated by this template would be `(G(g(G().2).3).1)`, where, for simplicity, it is also assumed that there are no further local introductions of type `G` under “Thurgau” and taking `n` = 1 provides a fresh index. The string ‘*districts*’ in tiny letters, a *concept anchor*, marks the expression between the parentheses as the ID String for a new concept and is used to assign an attribute representation to the generated ID String. Here fore, the second line of the template specifies the name of the concept as “districts in canton Thurgau”, where `name` is (incidentally) both the semantic and syntactic type and `en` the language type. The entry set generated by this template is then {(`(G(g(G().2).3).1)` , {`name.en.name` = “districts in canton Thurgau”})}, which contains a single entry of the form (*ID String*, *attribute set*). In the general situation, each anchor in the template generates an entry that can be aligned with the ontology, so only templates with at least one concept anchor with its corresponding attribute specification are allowed. One template produces as many ontology entries as anchors are introduced in the template.

The real power of templates comes from the use of variables. Figure 2 shows an elaboration of the simple template above. The local introduction

$$\begin{array}{rl} \multicolumn{2}{c}{(_{capital}\ (_{district}\ \mathtt{g}(_{districts}\ \mathtt{G}[\mathtt{\#Canton}].\mathtt{n}).\mathtt{n})\&[\text{capitals}])} \\ \textit{districts}.\mathtt{name.en.name} & = \text{“districts in canton } \mathtt{\#Canton}\text{”} \\ \textit{district}.\mathtt{name.en.name} & = \text{“}\mathtt{\#District}\text{”} \\ \textit{capital}.\mathtt{name.en.name} & = \text{“}\mathtt{\#Capital}\text{”} \end{array}$$

Figure 2: A more elaborated entry template

of type g (inner term) shows that each *district* is encoded as an element of the corresponding district set. Each *capital* (cf. outermost term) is modeled as an intersection of its district with the concept “capitals” which already exists in the net. Instead of writing a new template for each capital, we use variables that point to values in the table. These variables are represented in Fig. 2 by the strings `#Canton`, `#District` and `#Capital`. When executing the template for the given data set, candidate entries are produced, the description using the concepts found in the table. Details of the mapping from template variables to data will be discussed in the next section. For the moment, if we assume that the template in Fig. 2 is correctly instantiated with the values found in the first row of Table 1, three candidate entries are generated, which are depicted in Fig. 3. Here we assume that there are already two districts in the district set, so “Bezirk Weinfelden” receives the

```
{   ( (G(g(G().2).3).1), {name.en.name = "districts in canton Thurgau"} ),
    ( (g(G(g(G().2).3).1).3), {name.en.name = "Bezirk Weinfelden"} ) ,
    ( ((g(g(G(g(G().2).3).3).1).3)&(G((G().1)&(F().2)).3)),
         {name.en.name = "Weinfelden"} )    }
```

Figure 3: Entry set generated by the elaborated template for the first row in Table 1

index 3. Note that for the following instantiation, which uses the second row of Table 1, a redundant entry would be generated for "districts in canton Thurgau". Such redundancies are handled later during the alignment of entries with ontology.

The complete extended ID String syntax for template definition is summarized in Fig. 4. Compare it with Fig. 1. The last three alternatives for $\mathbb{IDS}\mathrm{tr}'$ constitute the core of the extension, where the first of them represents

$$\begin{array}{lcl}
\mathbb{IDS}\mathrm{tr}' & := & (\,)\mid(\,\mathbb{T}\mathrm{ype}\,\mathbb{IDS}\mathrm{tr}'\,.\,\mathbb{N}\mathrm{um}\,)\mid(\,\mathbb{IDS}\mathrm{tr}'\,\&\,\mathbb{IDS}\mathrm{tr}'\,)\mid \\
 & & [\,\mathbb{Q}\mathrm{uery}\,]\mid(_{\mathbb{A}\mathrm{nchor}}\mathbb{T}\mathrm{ype}\,[\,\mathbb{Q}\mathrm{uery}\,]\;.\mathtt{n})\mid(_{\mathbb{A}\mathrm{nchor}}\mathbb{IDS}\mathrm{tr}'\,\&\,\mathbb{IDS}\mathrm{tr}'\,) \\
\mathbb{A}\mathrm{nchor} & := & \mathrm{Alphanum}^{+} \\
\mathbb{Q}\mathrm{uery} & := & \mathbb{IDS}\mathrm{tr}\mid\mathbb{L}\mathrm{it} \\
\mathbb{L}\mathrm{it} & := & \mathrm{String}\mid\mathbb{R}\mathrm{ef}
\end{array}$$

Figure 4: The extended ID String syntax.

a concept query and the other two are variants of the local introduction and concept intersection rules that introduce concept anchors. We allow ID Strings or *literals* as queries. A literal (rule $\mathbb{L}\mathrm{it}$) is an arbitrary string value specified in the template and interpreted as a concept name or, alternatively, a value taken from data, symbolized by $\mathbb{R}\mathrm{ef}$ for reference in the syntax. References are discussed in the next section. Names for concept anchors are alphanumeric strings (rule $\mathbb{A}\mathrm{nchor}$).

$$\begin{array}{lcl}
\mathbb{A}\mathrm{ttr} & := & \mathbb{A}\mathrm{nchor}\,.\,\mathbb{S}\mathrm{em}\,.\,\mathbb{L}\mathrm{ang}\,.\,\mathbb{S}\mathrm{yn}=\mathbb{L}\mathrm{it} \\
\mathbb{S}\mathrm{em} & := & (\,\mathtt{name}\mid\mathtt{syn}\mid\mathtt{url}\mid\ldots) \\
\mathbb{L}\mathrm{ang} & := & (\,\mathtt{de}\mid\mathtt{en}\mid\ldots) \\
\mathbb{S}\mathrm{yn} & := & (\,\mathtt{nom}\mid\mathtt{adj}\mid\ldots)
\end{array}$$

Figure 5: Attribute specification syntax.

The grammar of attribute assignments via concept anchors is summarized in Fig. 5. For each anchor, literals are assigned to attributes specified by their semantic, language and a syntactic type.

5 Referring to Data for Entry Template Instantiation

In the previous section we indicated that the variables of a template are mapped to tuples of textual data for instantiation. In general, many distinct instantiations can be obtained from one data source. Here we explain the details of this mapping for two kinds of input data, tables and XML data. This clarifies the procedural meaning of a template in presence of data encoded in these formats.

The following observation leads us to the main idea: Once a template is instantiated by some fixed tuple, the outermost anchor generates a concept of the EFGT Net which is more specific than (a descendant of) all concepts generated by other anchors of the template. More generally, if the template contains two anchors A and B, and if B is in the syntactic scopus of A, then the concept introduced by A is more specific than the one introduced by B. E.g., in the template in Fig. 2, each concept resulting from an instantiation of the anchor *district* is more specific than (a descendant of) a concept instantiation of the reference `Canton`. The anchor *capital* represents the most specific concept for each instantiation of the template. Since for a given anchor A the set of ancestors B represented in the template is unequivocal, the ancestors marked in the template *functionally depend* on the enclosing anchor. E.g., functional dependencies in our example template constitute a chain: the *capital* functionally determines the *district*, which determines the *districts* set, which determines the canton.

As explained in further detail below, each data format also encodes functional dependencies in a different way. The main idea for variable instantiation will be to extract tuples that represent functional dependencies in the data file and use these tuples in order to instantiate the functional dependencies represented in the template. This principle constraints the data files we can process using our integration language: functional dependencies represented in templates must also be encoded as functional dependencies in the data files. In this sense we say that data files must "naturally" encode the target relationships in the ontology. In practice, we found that this was not a severe restriction, allowing to leave data in its original appearance. This way, the user can define templates interactively and try different alternatives without a need of transforming data each time.

Table Data. Tables usually have at least one key column that functionally determines the values in the other columns. E.g., in Table 1 both the `Capital` and `District` columns functionally determine the other two columns, so that a chain of dependencies can be constructed. Thus, each row provides a tuple representing functional dependencies that can be used for instantiating the template. The choice here is to let the outermost anchor range over the key column and to let the other references in the template point to adequate columns, and then to evaluate the template successively for each row of the table. We mentioned that template variables are marked by the symbol `#`. If the columns in the table are named, a reference can be specified by `#` followed by the name of the column. Columns are internally numbered, so the user can use numbers #1, #2, etc. for reference. Figure 6 shows the template of Fig. 2 with references to Table 1, so that the candidate entry set generated for the first row is exactly the set depicted in Fig. 3.

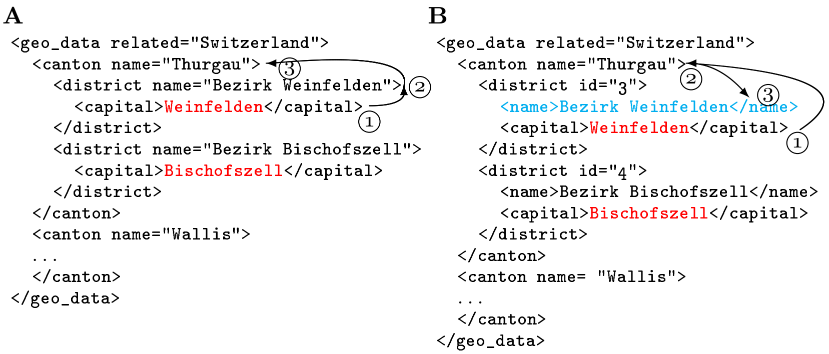
$$
\begin{array}{rl}
\multicolumn{2}{c}{(_{capital}\ (_{district}\ \mathtt{g}(_{districts}\ \mathtt{G}[\mathtt{\#Canton}].\mathtt{n}).\mathtt{n})\&[\text{capitals}])} \\
\mathit{districts}.\mathtt{name.en.name} & = \text{“districts in canton } \mathtt{\#Canton}\text{”} \\
\mathit{district}.\mathtt{name.en.name} & = \text{“}\mathtt{\#District}\text{”} \\
\mathit{capital}.\mathtt{name.en.name} & = \text{“}\mathtt{\#Capital}\text{”}
\end{array}
$$

Figure 6: A more elaborated entry template

Sometimes, one wants to skip some rows of the table or to apply different templates depending on the value of some fields. This is achieved by using `if-then-else` statements with conditions on references, also included in the integration language. In other cases, a field of a table may contain an enumeration of concepts. This helps to collapse many rows that only differ in one column to a single row. Our language has special devices for dealing with such variations.

XML data. Each XML document can be represented as a labeled tree. In this kind of tree, some structural relationships can be regarded as encoding functional dependencies. Most prominently, an element functionally determines its parent node. If a label a occurs exactly once in each path ending with a label b, then each b element functionally determines a unique a element among its ancestors. If an element comes with a unique textual contents, then the text functionally determines the element. Furthermore, each element node functionally determines its its attribute nodes.

Figure 7 shows two panes of two different XML files encoding the data about Switzerland from Table 1. On pane A, functional dependencies in Table 1 are represented as structural relationships that are also inherently functional in XML. The text node “Weinfelden” functionally determines its element node of type `capital`, which has an unequivocal parent (with type `district`) that has an unique attribute `name`, “Bezirk Weinfelden”. So in pane A, “Bezirk Weinfelden” functionally depends on “Weinfelden”. It is easy to see that “Thurgau” is also a function of “Bezirk Weinfelden”. Of course, there a different ways to encode functional dependencies by structural relationships in an XML representation. For the extraction of tuples that represent functional dependencies, we rely on XPath expressions (XPath, 2006). The procedure is a follows. The first step is to determine by means of an absolute XPath expression a set of nodes acting as keys. The outermost concept anchor is let to range over this key set. There is a path in the document tree from each node in the key set to the root document node. References for the ancestors of the concept anchor in the



A

```
<geo_data related="Switzerland">
  <canton name="Thurgau">
    <district name="Bezirk Weinfelden">
      <capital>Weinfelden</capital>
    </district>
    <district name="Bezirk Bischofszell">
      <capital>Bischofszell</capital>
    </district>
  </canton>
  <canton name="Wallis">
  ...
  </canton>
</geo_data>
```

B

```
<geo_data related="Switzerland">
  <canton name="Thurgau">
    <district id="3">
      <name>Bezirk Weinfelden</name>
      <capital>Weinfelden</capital>
    </district>
    <district id="4">
      <name>Bezirk Bischofszell</name>
      <capital>Bischofszell</capital>
    </district>
  </canton>
  <canton name= "Wallis">
  ...
  </canton>
</geo_data>
```

Figure 7: Two XML representations of the data in Table 1

template are defined as relative paths, interpreted on the path from the key node to the document root. In XPath terms, relative paths are evaluated against nodes in the ancestor axis of each key node or the key node itself (`ancestor-or-self` axis).

Figure 8 shows a variant of the elaborated template in Fig. 2 including XPath references that generate adequate instantiations out of pane A. The absolute path `//capital` selects the capitals of the XML document, that are identified in turn with the name of the outermost anchor concept *capital*. The other references are evaluated on each path starting at each capital node and ending on the document node. For the capital node "Weinfelden", the references `district/@name` and `canton/@name` extract "Bezirk Weinfelden" and "Thurgau" respectively from the `ancestor-or-self` axis, which is implicitly assumed and doesn't have to be stated in the template. This method is quite flexible, allowing to deal with different XML

$$\left(_{capital}\left(_{district}\ \mathtt{g}\left(_{districts}\ \mathtt{G}[\mathtt{\#canton/@name}].\mathtt{n}\right).\mathtt{n}\right)\&[\text{capitals}]\right)$$

$$\begin{aligned}
\mathit{districts}.\mathtt{name.en.name} &= \text{“districts in canton }\mathtt{\#canton/@name}\text{”}\\
\mathit{district}.\mathtt{name.en.name} &= \text{“}\mathtt{\#district/@name}\text{”}\\
\mathit{capital}.\mathtt{name.en.name} &= \text{“}\mathtt{\#//capital}\text{”}
\end{aligned}$$

Figure 8: The elaborated template adapted for pane A in Fig. 7

renditions of the same data and even with those that encode functional dependencies with structural relationships that are not inherently functional in XML. E.g., pane B in Fig. 7 encodes the name of the district in an element node `name` instead as an attribute like in pane A. The relationship between the district and name element is also functional in this case, because there is only one child element of type `name`. But in the example, this is only incidentally so. This kind of structural relationship is not inherently functional in XML, because the XML data model allows that an element has more that one element child of the same type, although, of course, that `district` element may have only on `name` element can be regulated by means of a DTD or XML Schema. We call this kind of encoding of functional relationships with not inherently functional structural relationships in XML a "pseudo-attribute". Our approach also can handle pseudo-attributes. In the template we can simply replace the reference `district/@name` by `district/name` in order to obtain the same tuples as for pane A.

6 Alignment of Generated Entries

In general, generated entries can be aligned with the ontology by examining first, whether there is another concept in the ontology with an equivalent logical characterization (*logical existence*), and second, whether the concept is already *linguistically present* in the ontology. In the case of the EFGT Net, it is enough to match the generated ID String against all ID Strings in the net to decide the logical existence. Whether a concept is linguistically present in the ontology can be decided by performing a search over the attributes holding linguistic information. The cases in Fig. 9 can then be distinguished. A generated candidate entry can be considered a *new*

Log. existent	**Ling. present**	**Case name**	**Interpretation**
no	no	*Potential new entry*	Generated new entry
yes	no	*Logical clash*	1) Complementary lex. representation 2) Logical modeling too coarse
yes	yes	*Concept match*	Entry exists already
no	yes	*Name clash*	1) Logically differing concepts with same name (homonym entries) 2) Same concept but different logical modeling

Figure 9: Alignment cases

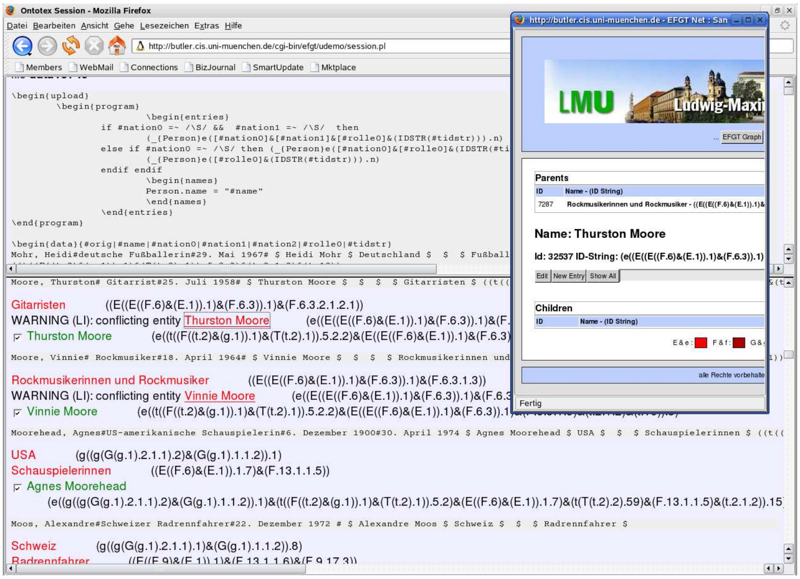
concept to be added to the ontology when there is simply nothing indicating that it collides with another concept in the ontology. *Logical clashes* can be obliterated by merging the attribute representation of both concepts. This makes sense when the colliding concept name is just a variant not included in the linguistic representation of the existing concept. It may also be the case that the semantic analysis of two different concepts is too coarse to distinguish between them. A *concept match* is given when the generated entry is indistinguishable from another concept in the ontology. *Name clashes* also have two possible interpretations. It may occur that two semantically different concepts share their linguistical representation (homonym entries), in which case the new candidate entry may be considered for adding it to the ontology. The converse interpretation is that an already existing concept in the ontology is modeled by the template in a different way than in the net.

7 Ontology Population and Inspection with the Upload Tool

We have developed a prototype called the *Upload Tool* that interprets EFGT Net entry templates. It has a client-server architecture, where the EFGT Net resides in a RDBS backend queried by the web client. The client interprets the template language and performs the alignment. Figure 10 shows a screen-shot of the Upload Tool. The upper part contains the entry template, submitted as a file in a previous step. The entry template can be edited and evaluated online. The results of the alignment are displayed in the lower part of the window as a list of entries for each template instantiation. When there is a concept match, entries are colored red, while potential new entries are displayed in green. Check-boxes allow for selecting green entities and uploading them to the database, where inference takes place and the structure of the net is rearranged for accommodating new concepts. Blue entries have already been considered in a previous template instantiation and don't need to be considered again.

When conflicts appear, different facilities that correspond to the different interpretations listed in Table 9 are provided to user for handling the conflict. In the case of logical clashes, a warning appears and the user can decide to run a facility for merging the attribute (lexical) representation, or, corresponding to the other interpretation, a tool for editing and refining the ID String of the conflicting concepts. For efficiently handling name clashes, there are two modes available that respectively enable or disable the creation of homonym entries. This is useful depending on the data considered. E.g., enabling homonym entries is self-evident when uploading geographical data, where homonyms are usual. Different people can also have homonym names, but when integrating names in the EFGT Net one may want to check if incoming data corresponds to a truly new person or represents a refinement in the modeling of an existing entry, so disabling homonyms is better choice in a first step. Fig. 10 shows a name clash for "Thurston Moore", where an additional window showing the parents of the already existing entry has been opened for clarification. When the name clash is interpreted as a difference in the logical modeling between the existing and the generated entry, a further tool for merging the corresponding ID Strings can be started directly from the client interface.

Posterior changes in uploaded data can be easily tracked with the Upload Tool, because once the net has been populated with a specific file, the same


Ontotex Session - Mozilla Firefox
Datei Bearbeiten Ansicht Gehe Lesezeichen Extras Hilfe
http://butler.cis.uni-muenchen.de/cgi-bin/efgt/udemo/session.pl
Members WebMail Connections BizJournal SmartUpdate Mktplace
http://butler.cis.uni-muenchen.de - EFGT Net : San
LMU
Ludwig-Maxi
... EFGT Graph
Edit
New Entry
Show All
alle Rechte vorbehalte
Fertig

```
\begin{upload}
        \begin{program}
                        \begin{entries}
                if #nation0 =~ /\S/ &&  #nation1 =~ /\S/  then
                        (_{Person}e([#nation0]&[#nation1]&[#rolle0]&(IDSTR(#tidstr))).n)
                else if #nation0 =~ /\S/ then (_{Person}e([#nation0]&[#rolle0]&(IDSTR(#ti
                        (_{Person}e([#rolle0]&(IDSTR(#tidstr))).n)
                endif endif
                        \begin{names}
                        Person.name = "#name"
                        \end{names}
                \end{entries}
\end{program}

\begin{data}{#orig|#name|#nation0|#nation1|#nation2|#rolle0|#tidstr}
Mohr, Heidi#deutsche Fußballerin#29. Mai 1967# $ Heidi Mohr $ Deutschland $  $  $ Fußball
```

```
Moore, Thurston# Gitarrist#25. Juli 1958# $ Thurston Moore $  $  $  $ Gitarristen $ ((t((
```

Gitarristen ((E((E((F.6)&(E.1)).1)&(F.6.3)).1)&(F.6.3.2.1.2.1))
WARNING (LI): conflicting entity Thurston Moore (e((E((E((F.6)&(E.1)).1)&(F.6.3)).1)&(F.
☑ Thurston Moore (e((t((F((t.2)&(g.1)).1)&(T(t.2).1)).5.2.2)&(E((E((F.6)&(E.1)).1)&(F.6.3)

```
Moore, Vinnie# Rockmusiker#18. April 1964# $ Vinnie Moore $  $  $  $ Rockmusikerinnen und
```

Rockmusikerinnen und Rockmusiker ((E((E((F.6)&(E.1)).1)&(F.6.3)).1)&(F.6.3.1.3))
WARNING (LI): conflicting entity Vinnie Moore (e((E((E((F.6)&(E.1)).1)&(F.6.3)).1)&(F.6.3
☑ Vinnie Moore (e((t((F((t.2)&(g.1)).1)&(T(t.2).1)).5.2.2)&(E((E((F.6)&(E.1)).1)&(F.6.3)).1

```
Moorehead, Agnes#US-amerikanische Schauspielerin#6. Dezember 1900#30. April 1974 $ Agnes Moorehead $ USA $  $  $ Schauspielerinnen $ ((t((
```

USA (g((g(G(g.1).2.1.1).2)&(G(g.1).1.1.2)).1)
Schauspielerinnen ((E((F.6)&(E.1)).1.7)&(F.13.1.1.5))
☑ Agnes Moorehead
(e((g((g(G(g.1).2.1.1).2)&(G(g.1).1.1.2)).1)&(t((F((t.2)&(g.1)).1)&(T(t.2).1)).5.2)&(E((F.6)&(E.1)).1.7)&(t(T(t.2).2).59)&(F.13.1.1.5)&(t.2.1.2)).15

```
Moos, Alexandre#Schweizer Radrennfahrer#22. Dezember 1972 # $ Alexandre Moos $ Schweiz $  $  $ Radrennfahrer $
```

Schweiz (g((g(G(g.1).2.1.1).1)&(G(g.1).1.1.2)).8)
Radrennfahrer ((E((F.9)&(E.1)).1)&(F.13.1.1.6)&(F.9.17.3))

Parents	
ID	Name - (ID String)
7287	Rockmusikerinnen und Rockmusiker - ((E((E((F.6)&(E.1)).1)&

Name: Thurston Moore

Id: 32537 ID-String: (e((E((E((F.6)&(E.1)).1)&(F.6.3)).1)

Children	
ID	Name - (ID String)

E&e: 🟥 F&f: 🟥 G&

Figure 10: Alignment results in the Upload Tool

file always can be retried. If some entries have changed in the meanwhile, they will appear as name clashes when aligning them again with the original template. This can also be regarded as focused view to the ontology on the basis of some data, allowing to inspect the ontology thematically.

8 Future work

Templates are useful for maintaining and populating ontologies with pertinent data that is already available. We see additional applications for templates we want to investigate in future. Storing data together with related templates could be an easy way to create thematic ontology modules one can then combine in order to obtain customized ontologies. Maintaining a template library also could be useful for further automatizing the data integration process as well as for providing support when data acquisition and document browsing take place in an integrated scenario, as we proposed in Weigel et al. (2006).

References

Buitelaar, P., Cimiano, P., and Magnini, B., editors (2005). *Ontology Learning from Text: Methods, Evaluation and Applications*, volume 123 of *Frontiers in*

Artificial Intelligence and Applications. IOS Press.

Ehrig, M., Staab, S., and Sure, Y. (2005). Bootstrapping Ontology Alignment Methods with APFEL. In Gil, Y., Motta, E., Benjamins, V. R., and Musen, M. A., editors, *Int. Semantic Web Conference*, volume 3729 of *LNCS*, pages 186–200. Springer.

Fikes, R., Hayes, P., and Horrocks, I. (2003). OWL-QL: A Language for Deductive Query Answering on the Semantic Web. Technical Report KSL 03-14, Stanford Univ.

GeoNames (2007). GeoNames. http://www.geonames.org/.

Huck, G., Fankhauser, P., Aberer, K., and Neuhold, E. (1998). JEDI: Extracting and synthesizing information from the web. In *Proc. of COOPIS*.

Laender, A., Ribeiro-Neto, B., Silva, A., and Teixeira, J. (2002). A brief survey of web data extraction tools. In *SIGMOD Record*, volume 31.

Maedche, A. and Staab, S. (2001). Learning ontologies for the semantic web. In *Workshop on the Semantic Web (SemWeb)*.

Maedche, A. and Volz, R. (2001). The Ontology Extraction and Maintenance Framework Text-To-Onto. In *2001 IEEE Int. Conf. on Data Mining (ICDM'01). Workshop on Integrating Data Mining and Knowledge Management*.

Maier, A., Schnurr, H.-P., and Sure, Y. (2003). Ontology-based Information Integration in the Automotive Industry. In *Proc. of the 2nd Int. Semantic Web Conference (ISWC2003)*, volume 2870 of *LNCS*, pages 897–912. Springer.

Noy, N. F. and Musen, M. A. (2000). PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In *Proc. of the National Conf. on Artificial Intelligence (AAAI)*, pages 450–455, Austin, TX.

Prud'hommeaux, E. and Seaborne, A. (2006). SPARQL Query Language for RDF. http://www.w3.org/TR/rdf-sparql-query/. W3C Candidate Recommendation.

Schulz, K. U. and Weigel, F. (2003). Systematics and Architecture for a Resource Representing Knowledge about Named Entities. In *Proc. Workshop on Principles and Practice of Semantic Web Reasoning*, pages 189–207.

WebL (2005). Automating the web language. http://research.compaq.com/SRC/WebL.

Weigel, F., Schulz, K. U., Brunner, L., and Torres-Schumann, E. (2006). Integrated Document Browsing and Data Acquisition for Building Large Ontologies. In *Proc. of the 10th Int. Conf. on Knowledge-Based & Intelligent Information & Engineering Systems (KES)*.

Wikipedia (2007). Wikipedia, the free encyclopedia. http://www.wikipedia.org.

XPath (2006). XML Path Language (XPath) Version 1.0. W3C Recommendation 16 November 1999. http://www.w3.org/TR/xpath.

XSLT (2006). XSL Transformations (XSLT), Version 1.0. W3C Recommendation 16 November 1999. http://www.w3.org/TR/xslt.

Alexander Mehler, Peter Geibel, Olga Pustylnikov

Structural Classifiers of Text Types: Towards a Novel Model of Text Representation

Texts can be distinguished in terms of their *content*, *function*, *structure* or *layout* (Brinker, 1992; Bateman et al., 2001; Joachims, 2002; Power et al., 2003). These reference points do not open necessarily orthogonal perspectives on text classification. As part of explorative data analysis, text classification aims at automatically dividing sets of textual objects into classes of maximum internal homogeneity and external heterogeneity. This paper deals with classifying texts into text types whose instances serve more or less homogeneous functions. Other than mainstream approaches, which rely on the vector space model (Sebastiani, 2002) or some of its descendants (Baeza-Yates and Ribeiro-Neto, 1999) and, thus, on *content-related lexical features*, we solely refer to *structural* differentiae. That is, we explore patterns of text structure as determinants of class membership. Our starting point are tree-like text representations which induce feature vectors and tree kernels. These kernels are utilized in supervised learning based on cross-validation as a method of model selection (Hastie et al., 2001) by example of a corpus of press communication. For a subset of categories we show that classification can be performed very well by structural differentia only.

1 Introduction

The basic idea of text classification is that the content, structure and shape of textual units vary, though not deterministically with the communicative situation or function they manifest. As this variation is not stochastic, we can build classes or types of textual units where members of the same class share class constitutive differentiae. Varying reference points of clarifying the ontological status of these differentiae lead to different notions of text types: If we focus on functional or situative criteria of class membership, we deal with so called *genres* (Martin, 1992; Ventola, 1987) or *registers* (Biber, 1995; Halliday and Hasan, 1989), respectively. Analogously, we speak of *hypertext sorts*, *digital genres* or *web genres* in the case of web documents (Santini, 2007). If we consider the composition of classes in terms of their *extension*, that is, from the point of view of enumerating their elements, we deal with *sorts* of documents – e.g. *text sorts* in the sense of Heinemann (Heinemann, 2000). If in contrast to this, class membership is defined in *intensional* terms, we deal with *text patterns* (Heinemann, 2000) or *superstructures* (van Dijk and Kintsch, 1983) as prototypical representations of class members, whose expectation-driven production/processing they support.

In this paper we focus on functionally demarcated text types for which we investigate to which degree class membership is manifested by structural differentia. The idea to predict the function of a text by the patterns it instantiates comes from the quantitative approach according to which distributional patterns vary with the text function (Biber, 1995). Starting from the weak contextual hypothesis (Miller and Charles, 1991) one might state that structural differences reflect functional ones while similar functions tend to be manifested by similarly structured texts. With a focus on registers, Biber (1995, p.59) puts this as follows: "preferred linguistic forms of a register are those that are best suited functionally to the situational demands of the variety [...]." As there is a many-to-many relation of structure and function (neither can we deterministically infer a unique function based on observing some text pattern, nor is the same function always manifested by the same pattern), learning text types by exploring text structures is a nontrivial task.

In the present paper, we focus on the logical document structure (Power et al., 2003) as a source of feature selection while we disregard layout and any content indicating lexical units of the texts to be classified. Since we focus on *text* types, we leave out hypertext and, especially, web documents.[1] Further, since we aim at modelling text *types*, we go beyond classical approaches which learn classifiers in order to enumerate class members without any effort in interpreting these classifiers as representations of text patterns. Rather, we perform our experiments as a preliminary step towards learning classifiers as representations of such patterns. As far as the classifiers being learnt allow deriving representations of patterns which, in turn, allow computing the similarity of texts with respect to these patterns, we contribute to a *prototype ontology* in the sense of Sowa (2000). For a remarkably large set of text types of press communication, we show that classification of their instances performs very well when disregarding any lexical features.

The paper is organized as follows: Section 2 discusses some related approaches; Section 3 presents two novel text representation models which are evaluated and discussed in Section 4. Finally, Section 5 concludes and prospects future work.

2 Related Work

In recent years, feature selection attracted many researchers in the field of classification. The aim is to find alternatives to the *bag-of-words* approach (Biber, 1995; Kessler et al., 1997; Karlgren, 1999; Lee and Myaeng, 2002; Wolters and Kirsten, 1999). Although lexical features are selective with respect to text content, this IR model generally disregards text structure. Now, modeling document structure comes into reach of machine learning (Dehmer, 2005). Some approaches even show that structural patterns allow to classify texts in the absence of any lexical information (Dehmer, 2005; Lindemann and Littig, 2006; Pustylnikov, 2006). Baayen et al. (1996) present a pioneering approaches in this field. They achieve good results in authorship attribution by focussing on frequencies

[1]For a related approach to web-documents cf. (Mehler, 2007; Mehler et al., 2006).

of constituent types (e.g., NP, VP). These observations indicate that authors have idiosyncratic syntactic signatures by which they can be classified. Further, Lindemann and Littig (2006) report good results in classifying web sites of different web genres (*blogs*, *personal*, and *academic homepages* as well as *online shops* and *corporate sites*). One of their resources of structural features is the link structure of the sites. Note that Mehler et al. (2006) have shown that such classifications are problematic when looking for web genres of a much higher resolution. The genres analyzed by Lindemann and Littig (2006) are in a sense general that one might expect their instances to be well separable in terms of their structure. This paper shows that such a structural classification is even possible for a wide range of more homogeneous rubrics of press communication.

Biber (1995) generally claims that "no single linguistic parameter is adequate in itself to capture the range of similarities and differences among spoken and written registers" and, thus, pleads for a multidimensional approach which takes a multitude of lexical and syntactical features into account. This is confirmed by Lewis (1992) who reports that compared to the bag-of-words approach there is no improvement if single features (e.g., *phrase pattern counts*) are taken into account. However, the extraction of a multitude of such features is time consuming and error-prone. Thus, an easy processable resource of expressive features is needed instead. The logical document structure and its quantitative characteristics is such a resource. It can be automatically computed for a wide range of genres and registers and is certainly easier accessible than either, e.g., rhetorical structure or syntactic structure. Evidence for this assumption comes from previous studies (Pustylnikov, 2006; Gleim et al., 2006) with respect to several registers and two languages (English and German). In this work we extend the structural framework by introducing *Quantitative Structure Analysis* (QSA) as a formal model of structural text representation models.

3 Text Representation Models Based on Structure-Sensitive Features

In recent experiments (Mehler et al., 2006), we have studied the selectivity of structural features in text classification. Our findings have shown that unsupervised learning of web document structures performs above the baseline scenario of random classification. As a complementary approach, we now tackle the question what "golden standard" can be achieved by using supervised methods. In order to do that, we investigate two structure-oriented text representation models as input to SVM-based machine learning:

1. *Quantitative structure analysis:* Our starting point is to represent texts by a set of quantitative features as a model of their structure. That is we build feature vectors whose coefficients do no longer stand for lexical units, but represent structural text characteristics. Section 3.1 presents a formal account of this approach which is inspired by Tuldava (1998) who clusters texts by means of simple quantitative characteristics. A further source of inspiration is synergetic linguistics (Köhler,

```
<!-- ... --> <DIV id="DIV1">
    <P id="P1">
        <S id="S1">
            <T c="l">
                <O>Montag</O><L p="Weekday">Montag</L>
            </T>
            <PUN>,</PUN><E/>
            <N>3</N>
            <PUN>.</PUN><E/>
            <T c="l">
                <O>Januar</O><L p="Month">Januar</L>
            </T><E/>
            <N>1994</N><E/>
            <T c="l">
                <O>NACHRICHTEN</O>
                <L p="N">Nachricht</L>
            </T><E/>
            <T c="l"><O>S\"{u}ddeutsche</O><L p="A">s\"{u}ddeutsch</L></T><E/>
            <T c="l"><O>Zeitung</O><L p="N">Zeitung</L></T>
        </S> <!-- ... -->
    </P> <!-- ... -->
</DIV> <!-- ... -->
```

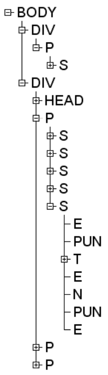


Figure 1: Outline of a sample text document (left) and its corresponding DOM-representation (right) – generated by the TextMiner system (Mehler, 2002) – in the form of an ordered rooted tree as input to feature selection (all XML element contents deleted).

1999) which develops reference systems of quantitative variables of dynamic linguistic systems.

2. *The tree kernel approach:* Secondly, we elaborate SVM learning which uses kernels that operate on pairs of examples (Vapnik, 1995). The theory of SVMs ensures that kernels can be defined for tree-like structures (Haussler, 1999; Jain et al., 2005; Schoelkopf and Smola, 2002). A large class of structure kernels is formed by convolution kernels (Haussler, 1999) including the one defined by Collins and Duffy (2001) for labeled, ordered trees. This tree kernel has previously been applied to structure based classification of sentences described by their parse trees (Collins and Duffy, 2001; Moschitti, 2004). In order to classify texts, we apply an extended version of this kernel in Section 3.2. This new kernel allows a variable number of descendants for some tree nodes.

Input to these two approaches are tree-like instances of the *Document Object Model* (DOM). That is, starting from a text corpus $C = \{x_1, \dots, x_r\}$, each text $x_k \in C$ is mapped onto a DOM-tree representing its logical document structure. This is done by means of the TextMiner system (Mehler, 2002) which uses an element name-adapted version of XCES (Ide et al., 2000) in order to explore the paragraph structure of texts down to the level of their lexical tokens (by disregarding sentence structure). Figure (1) illustrates a sample text and its DOM-based representation in the form of an ordered rooted tree. Trees of this sort are subsequently input of feature selection, that is, of quantitative structure analysis (Section 3.1) and of tree kernel methods (Section 3.2).

3.1 Quantitative Structure Analysis

The *Vector Space Model* (VSM) is one of the most successful quantitative text representation models. It is the starting point of *Latent Semantic Analysis* (LSA) which takes an appropriately weighted term-document matrix as input and aims at eliminating as much of its noise as possible. From a linguistic point of view, the VSM performs a *bag-of-words* approach which focuses on *lexical* cohesion as a source of text similarity measuring (e.g., by the well-known cosine approach). LSA complements an effort in exploring indirect relations, e.g., of texts which may be said to be similar not because of having the same, but because of sharing similar lexical items whose similarity is computed in terms of their co-occurrence patterns. In this section, we present a likewise quantitative text representation model in terms of a *bag-of-structural-features approach* henceforth called *Quantitative Structure Analysis* (QSA). QSA is no longer based on lexical, but on structural features of the input texts. Thus, although we map texts onto feature vectors, their coefficients represent quantitative characteristics of text structure. Needless to say that both the VSM or LSA and the QSA approach can be combined (Mehler, 2002). However, in this paper we will concentrate on the separability potential of structural text features.

Generally speaking, QSA is based on a set $T_S = \{T_1, \ldots, T_m\}$ of structure types (e.g. constituency types) of some level S of text structuring (e.g. of the level of logical document structure or of intentional structure). For some structure type (e.g. sentence, paragraph, phrase) $T_i \in T_S$ of the level S we can ask, among other things, for (i) the *frequency*, (ii) (absolute) *length* (in terms of the number of leaf nodes), (iii) complexity (in terms of the number of immediate daughter nodes or some other mediate level if existing), (iv) *depth*, (v) *extension* (i.e. relative length), (vi) *proportion* (of text formation), (vii) *text position*, (viii) *distance*, (ix) (e.g. Markov) *order* or *arrangement* or (x) for the *characteristic repetition* (e.g. positional repetition) of instances of this type within a given text x of the corpus $C = \{x_1, \ldots, x_r\}$ and, thus, for different quantitative features. More specifically, if $T_i \in T_S$ is a structure type of level S and F_i is one of the latter features, we need to specify a measuring unit in order to calculate its value for a given instance of T_i in x. Let, for example, T_i be the structure type named title as part of the *Logical Document Structure* (LDS), then we can ask for its length in terms of lexical tokens or its frequency in terms of its number of occurrences. Analogously, we may ask for a title's depth in terms of the number of subtitles it is dominating. Alternatively, we may calculate its depth as the depth of its phrase structure tree. Another example is rhetorical structure in the sense of *rhetorical structure theory* (Mann and Thompson, 1988). In this case, we may ask for the text position of contrast relations. Now, the measuring unit is less clear so that we may define, for example, that the position of a contrast relation in a text equals the number of elementary text spans to its left. Following this procedure, we get a different positioning number for each title of the input text. In order to keep the presentation of our algorithm abstract, we resist defining the space of all possible measuring units for each of the features, but will define them as soon as needed.

The idea behind QSA is to, firstly, collect all values of a given structure feature in a text x where these values are, secondly, input to some aggregation functions in order to, thirdly, derive x's *Quantitative Structure Profile* (QSP). As we have to put apart the choice of text structure types and their features, we come up with a quadripartite approach:

Segmentation Let S be a description level of text structure and $T_S = \{T_1, \dots, T_m\}$ a set of structure types of S. For each structure type $T_i \in T_S$ and each text $x_k \in C = \{x_1, \dots, x_r\}$ we build a separate vector of instances of T_i in x_k. Using a functional notation, we write:

$$T_i(x_k) = \left(I^1_{x_k}, \dots, I^h_{x_k}\right)' \tag{1}$$

where $I^l_{x_k}, l \in \{1, \dots, h\}$, is the lth instance of T_i in x_k. We assume that the linear order $(1, \dots, h)$ is defined by the order of occurrences of T_i's instances in x_k. Note that different texts may differ in the number of their instances of T_i. Now, let

$$\mathbb{T}_i(C) = \bigcup_{k=1}^{|C|} \{T_i(x_k)\} \tag{2}$$

Thus, we can write

$$T_i \colon C \to \mathbb{T}_i(C) \tag{3}$$

Feature Validation Now, let $F = \{F_1, \dots, F_n\}$ be a set of numerical features (e.g. length, depth, complexity as enumerated above) and $F_j \in F$. Using once more a functional notation, we define

$$F_j \colon \mathbb{T}_i(C) \to \cup_{h=1}^{\infty} \mathbb{R}^h \tag{4}$$

by setting

$$F_j((I^1_{x_k}, \dots, I^h_{x_k})') = (F_j(I^1_{x_k}), \dots, F_j(I^h_{x_k}))' = \vec{v}(T_i, F_j, x_k) \in \mathbb{R}^h \tag{5}$$

$F_j(I^l_{x_k})$, $l \in \{1, \dots, h\}$, is the F_j-value of the lth instance of T_i in x_k. So far, each text $x_k \in C$ is mapped for each feature $F_j \in F$ onto a separate vector of the F_j-values of T_i's instances in x_k. As these vectors may differ with respect to the number of their coefficients, we do not yet get a matrix.[2] The next step is to aggregate each of these vectors separately to get a single value for feature F_j of all instances of T_i in x_k. Thus, we conceive the vectors $F_j((I^1_{x_k}, \dots, I^h_{x_k})') = \vec{v}(T_i, F_j, x_k) = \vec{v}_{ijk}$ as value distributions. In order to make these distributions comparable, we perform standardization by means of z-scores so

[2] An alternative would be to operate with empty coefficients – we do not follow this approach.

that random variables are derived with means of 0 and variances of 1. Without any loss of generality we assume henceforth that all vector coefficients are standardized and write

$$\mathbb{F}_j(\mathbb{T}_i(C)) = \bigcup_{k=1}^{|C|} \{\vec{v}_{ijk}\}$$

Feature Aggregation Now, let $O = \{O_1, \ldots, O_o\}$ be a set of parameters of location or statistical spread and $O_p \in O$ one of these aggregation functions. Then we define

$$O_p \colon \mathbb{F}_j(\mathbb{T}_i(C)) \to \mathbb{R} \tag{6}$$

where $O_p(\vec{v}_{ijk}) \in \mathbb{R}$ is the value of O_p when performed on the vector of the values of feature F_j of T_i's instances in x_k. So far, we mapped each of the features F_j onto a single number $O_p(\vec{v}_{ijk}) \in \mathbb{R}$. The final stage is to collect these numbers to get a quantitative structure profile for each text.

Text Representation For each text $x_k \in C$, we define a quantitative structure profile as

$$\begin{aligned} \mathrm{qsp}(x_k) &= \langle \quad O_1(\vec{v}_{11k}), \ldots, O_1(\vec{v}_{1nk}), \ldots, O_1(\vec{v}_{m1k}), \ldots, O_1(\vec{v}_{mnk}), \\ & \qquad \ldots, \\ & \qquad O_o(\vec{v}_{11k}), \ldots, O_o(\vec{v}_{1nk}), \ldots, O_o(\vec{v}_{m1k}), \ldots, O_o(\vec{v}_{mnk}) \quad \rangle \\ &\in \quad \mathbb{R}^{m \cdot n \cdot o} \end{aligned} \tag{7}$$

That is, qsp is a function $\mathrm{qsp} \colon C \to \mathbb{R}^{m \cdot n \cdot o}$ which allows to build a $(|C|, m \cdot n \cdot o)$-matrix $\mathrm{qsp}(C) = (a_{ij})$ where a_{ij}, $j = (s-1)mn + (t-1)n + v$, is the value of aggregation function $O_s \in O$ performed on the feature distribution induced by the tth feature $F_t \in F$ with respect to instances of the vth text structure type $T_v \in T_S$ in text x_i. We call $\mathrm{qsp}(C)$ *the quantitative structure profile* of corpus C.

Note that matrix $\mathrm{qsp}(C)$ can be input to single value decomposition with subsequent noise reduction so that QSA is complemented by a latent variable analysis.

So far, we described a bag-of-features approach as input to supervised text categorization or unsupervised classification. In the following section, we describe alternative approaches to building kernels for mapping tree-like structures. In Section 4, these two approaches are evaluated.

3.2 Tree Kernels for XML Documents

In order to investigate the structure-based classification of XML documents based on their DOM trees (Document Object Model), we might apply the SVM (Vapnik, 1995) after defining an appropriate *tree kernel*. This can either be accomplished by directly defining an appropriate function that is positive-semidefinite (PSD, e.g., Schoelkopf and Smola 2002) or by explicitly defining an appropriate feature mapping for the structures considered, e.g., by means of patterns. An example of a kernel is the *parse tree kernel*

(Collins and Duffy, 2001; Moschitti, 2004), which is applicable to parse trees of sentences with respect to a given grammar.

In contrast to parse trees in which a grammar rule applied to a non-terminal determines number, type and sequence of the children, structural parts of a text represented by its DOM tree might have been deleted, permuted or inserted compared to a text considered similar. This higher flexibility should be taken into account in the similarity measure represented by the tree kernel, because otherwise the value for similar documents might be unreasonably small. Moreover, we might want to include textual information present in nodes by plugging in suitable kernels operating, e.g., on the usual TFIDF representation of the respective text, or more elaborate ones like string kernels (Lodhi et al., 2002) operating on the word sequence or even additional tree kernels operating on the parsed sentence structure.

We therefore extended previous work on tree kernels suitable for XML data in several respects that are useful in the context of HTML and XML documents. The *DOM tree kernel* (DomTK) is a straightforward generalization of the parse-tree kernel to DOM trees. The *set tree kernel* (SetTK) allows permutations of child subtrees in order to model document similarity more appropriately, but can still be computed relatively efficiently.

3.2.1 The Parse Tree Kernel

In the following, we consider trees whose nodes $v \in V$ are labeled by a function $\alpha : V \longrightarrow \Sigma$, where Σ is a set of node labels. The elements of Σ can be thought of as tuples describing the XML tag and attributes of a non-leaf node in the DOM tree. Leaves are usually labeled with words or parts of texts. We will incorporate node information by using a kernel k^{Σ} operating on pairs of node labels, i.e., on tags, attributes, and/or texts. Two trees T and T' are called isomorphic if there is a bijective mapping of the nodes that respects the structure of the edges, the labellings specified by α and α', and the ordering of the nodes.

Collins and Duffy (2001, 2002) defined a tree kernel that can be applied in the case of parse trees of natural language sentences (see also Moschitti 2004), in which non-leaf nodes are labeled with the non-terminal of the node, and leaves with words. The production applied to a non-leaf node determines the number, type, and ordering of the child nodes.

Collins and Duffy showed that $k(T, T')$ can be computed efficiently by determining the number of possible *mappings* of isomorphic partial parse trees (excluding such consisting of a single node only). Partial parse trees correspond to incomplete parse trees, in which leaves might be labeled with non-terminals. Let $v \in V$ and $v' \in V'$. The function $\Delta(v, v')$ is defined as the number of isomorphic mappings of partial parse trees rooted in v and v', respectively. Collins and Duffy stated in their article the fact that

Δ is a so-called convolution kernel (Haussler, 1999) having form

$$k(T,T') = \sum_{v \in V, v' \in V'} \Delta(v,v') \,. \tag{8}$$

The Δ-function can be computed recursively by setting $\Delta(v,v') = 0$ for *any* words and if the productions applied in v and v' are different. Different productions mean different non-terminals, or identical non-terminals but different grammar rules (i.e., the number or type of corresponding child nodes do not correspond). If the productions in v and v' are identical and both nodes are pre-terminals, we set $\Delta(v,v') = 1$. For non-terminals with identical productions, Collins and Duffy use the recursive definition

$$\Delta(v,v') = \prod_{i=1}^{n(v)} (1 + \Delta(v_i, v'_i)) \,, \tag{9}$$

where v_i is the i-th child of v, and v'_i is the i-th child of v'. $n(v)$ denotes the number of children of v (corresponding to that of v').

It is possible to down-weight deeper trees using a factor $\lambda \in [0,1]$. The corresponding recursive computation is $\Delta(v,v') = \lambda \prod_{i=1}^{n(v)}(1 + \Delta(v_i, v'_i))$ together with the modified base case $\Delta(v,v') = \lambda$ for pre-terminals with identical productions.

3.2.2 Kernels for DOM trees

The *DOM tree kernel* (DomTK) is a relatively straightforward extension of the parse tree kernel that allows to incorporate node labels by means of k^{Σ}. This achieved by defining $\Delta_{\text{DomTK}}(v,v') = \lambda \cdot k^{\Sigma}(\alpha(v), \alpha(v'))$ for nodes. If not both v and v' are leaves, we, in contrast to the parse tree kernel, compare just as many children as possible using the given order $\leq$ on the child nodes.

It can be seen from a corresponding feature mapping that when comparing two trees T and T', we have two take into account shorter prefixes of the child tree sequences of two nodes v and v' as well. This is done by defining the Δ-function as

$$\Delta_{\text{DomTK}}(v,v') = \lambda \cdot k^{\Sigma}(\alpha(v), \alpha(v')) \Big(1 + \sum_{k=1}^{\min(n(v), n'(v'))} \prod_{i=1}^{k} \Delta_{\text{DomTK}}(v_i, v'_i)\Big) \tag{10}$$

for all nodes v and v'.

The DOM tree kernel does not allow the child trees of v and v' to be permuted without a high loss in similarity as measured by the kernel value $k(T,T')$. This behavior can be improved, however, by considering the child tree sequences as *sets* and applying a so-called set kernel to them, which is also an instance of the convolution kernel. This

mean number of articles per rubric	1426.8
standard deviation	2315.9
$\mu - \frac{\sigma}{2}$	268.8303
$\mu + \frac{\sigma}{2}$	2584.8

Table 1: Values of the parameters of the procedure of category selection.

results in the definition

$$\Delta_{\mathrm{SetTK}}(v, v') = \sum_{i=1}^{n(v)} \sum_{i'=1}^{n'(v')} \Delta_{\mathrm{SetTK}}(v_i, v'_{i'}) \,, \tag{11}$$

i.e., all possible pairwise combinations of child trees are considered.

When looking for a suitable feature space in the case $\lambda = 1$ and $k^{\Sigma} = k^{id}$ where $k^{id} = 1$ for nodes with identical labels, and $k^{id} = 0$ otherwise, we find that the definition in (11) corresponds to considering *paths* from the root to the leaves. This is a well-known technique for characterizing labeled graphs (see, e.g., Geibel and Wysotzki 1996), which can also be applied to trees. We also investigated tree kernels based on string kernels as in Kashima and Koyanagi (2002) and Moschitti (2006), but found them too inefficient for the application at hand.

4 Evaluation

The main hypothesis of our approach is that structure-based classification is a serious alternative to the bag-of-lexical-features approach. Thus, we expect a high selectivity of structural features with respect to functionally delimitable text types. In order to support this hypothesis, we process a corpus of press communication. The corpus is built as follows: We start from a ten years release of the German newspaper Süddeutsche Zeitung (SZ) and select *all* articles of *all* rubrics within this corpus. This gives a corpus of $135,546$ texts of 96 rubrics. Note that each text is mapped onto exactly one rubric. As the frequency distribution of the rubrics is unbalanced (it ranges from a rubric with only 2 instances to rubrics with more than 10,000 instances) and since the number of categories is – compared to other experiments in the field of text classification – large, we decided to select a subset of rubrics as target categories to be learnt. This was done as follows: We computed the mean μ and standard deviation σ of each rubric in terms of the number of its text instances and chose those rubrics R whose cardinality $|R|$ behaves as follows (cf. Table 1):

$$\mu - \sigma/2 < |R| < \mu + \sigma/2$$

As a result, we select 31 rubrics as target categories (cf. Table 2). This generates a corpus C of $31,250$ texts.

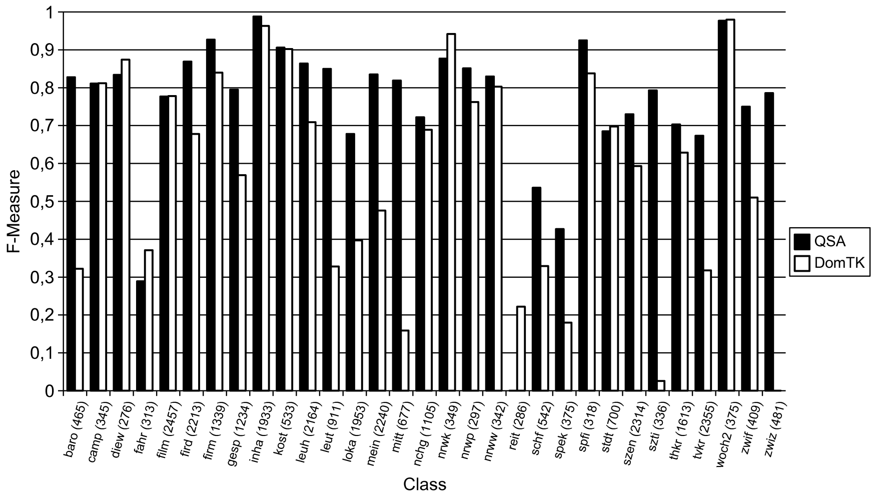


Table 2: Results of two categorization experiments using 31 rubrics of the SZ: *QSA* (black bars) and *DomTK* (gray bars). Bars are ordered alphabetically according to the code name of the category. Numbers in parentheses denote the size (number of instances) of the corresponding category.

Next, we perform an SVM-experiment in the framework of QSA. That is, we use $\mathrm{qsp}(x)$ as the vector representation of texts $x \in C$. More specifically, we refer to logical document structure (LDS) as the focal level of text structuring and set $T_{LDS} = \{$division, paragraph, sentence, headline sentence, headline paragraph$\}$. Next, we set $F = \{$complexity, length$\}$ and $O = \{$mean, standard deviation, entropy$\}$. Thus, for each input text we get a vector $\mathrm{qsp}(x)$ with exactly 30 features.

The design parameters of the subsequent SVM-experiment are as follows: We use an RBF-kernel with $\gamma = 0.00001$ and a trade-off between training error $c = 1000$. Further, we train a binary classifier for each of the 31 categories. Thus, for input corpus C and any rubric c_i of the set of target categories $\mathbb{C}$ the set of negative examples of c_i is set to $C \setminus [c_i]$ where $[c_i] \subseteq C$ is the set of all instances of c_i in C. In the present experiment, all sets $[c_i]$ are pairwise disjunct. Next, we utilize the *leave-on-out cross-validation* method (Hastie et al., 2001) and get a recall and precision value for each of the categories trained by means of the SVMlight (Joachims, 2002). This allows us to compute an F-score (FS) for each of the rubrics $c_i \in \mathbb{C}$ separately as (Hotho et al., 2005):

$$FS_i = \frac{2}{\frac{1}{\mathrm{recall}_i} + \frac{1}{\mathrm{precision}_i}}$$

The F-scores of the 31 categories are summarized in Table 2. Finally, we set $\mathbb{L} = \{[c_i] \mid c_i \in \mathbb{C}\}$ – obviously, $\mathbb{L}$ is a partition of C – and compute the F-Measure as a weighted mean of the F-scores of all categories as:

$$\text{F-Measure}(\mathbb{L}) = \sum_{i=1}^{|\mathbb{C}|} \frac{|[c_i]|}{|C|} FS_i$$

In the framework of QSA, this gives an overall F-measure of SVM-based learning of 0.78 – a remarkably good results for a rather large set of different categories to be learnt. Note that we took all $31,250$ texts of the corpus into account in order to compute this result.

Next, we perform a comparable experiment using the tree kernels as introduced in Section 3.2. The complexity of computing $k(T, T')$ based on (8) depends on the product of the node numbers $n(v)n(v')$, see Collins and Duffy (2001). Since some of the trees in the corpus are relatively large, we had to down-sample the corpus to a subset containing only 6250 examples. Instead of using leave-one-out cross-validation, we used 10-fold cross validation, which is more efficient, but known to produce reliable results, too. In order to make up for this loss in data to some extend, we performed a coarse search for optimal values of λ (parameter for tree depth, see above) and C. We varied λ additively in the interval $[0.0, 2.0]$ and C multiplicatively in the interval $[0.0001; 400.0]$. For SetTK we only used a subset of the parameters, because its complexity depends quadratically on the branching factor. For DomTK, choosing λ around 0.5 and C around 50 produced reasonable results for many classes. In addition to DomTK and SetTK, we also tested an implementation of a tree kernel based on a string kernel (cf. Moschitti 2006). The computation of a single kernel matrix took more than three days, so we are not able to present results for this third kernel in this article. Notice that for the second part of our SVM experiment based on tree kernels we used the LIBSVM (Chang and Lin, 2001).

4.1 Discussion

Table 2 presents the results of the experiment for QSA and the DomTK approach. Every category is identified by a short-cut representing a rubric (e.g., `woch2` = '*Wochenchronik*' – '*chronicle of the week*'). The corresponding F-Score values demonstrate the separability of most of the categories. Although DomTK performs better for a few classes, it is usually outperformed by QSA. This confirms results also found in other areas where tree kernel methods often perform worse than feature-based methods. Note that DomTK had to operate on a down-sampled data set (and fewer parameter combinations could be tried, too), while QSA explored the whole spectrum of the input corpus.

In the case of QSA, half of the categories perform with an F-score above 0.8 (vs. nine in the case of DomTK) – five (four in the case of DomTK) categories lead to F-score values above 0.9. The combined F-measure value of the QSA approach is 0.78. This shows that there are many categories which can be reliably attributed to their category by only looking for a small set of their quantitative structural features. Obviously, the set of 30 features taken into account by the present instance of QSA is much smaller than by VSM which may take several thousand lexical dimensions into account. Results for other text types (genres and registers) give a comparable result so that the method is, obviously, not restricted to the area of press communication (Pustylnikov and Mehler, 2007). However, in the case of the DomTK and the QSA approach there are poorly performing categories. This is *not* surprising as we do not expect that structure is the only reliable manifestation of text types. Rather we shed light on its potential which in future work will be combined with content-related approaches to text classification.

5 Conclusion

This paper evaluated logical document structure as a source of feature selection in text classification. It has shown that structure-based classifications come into reach and produce very promising results. This finding is all the more important as, e.g., QSA provides an easy to compute and space efficient text representation model. Thus, the paper is a first step towards the far-reaching goal of developing a prototype ontology of text types. Future work will focus on elaborating the present approach, especially in terms of a sensitivity analysis of the whole spectrum of quantitative text characteristics. Further, we will develop a corresponding graph model of web documents.

Acknowledgments

We thank Alessandro Moschitti (University of Rome) for helpful discussions. We thank Sebastian Herold (University of Osnabrück) for performing the kernel experiments. AM and OP's work is supported by the Research Group 437 *Text Technological Modeling of Information* (`http://www.text-technology.de/`) and by the SFB 673 *Alignment in Communication* (`http://ariadne.coli.uni-bielefeld.de/sfb/`) both funded by the German Research Foundation (DFG).

References

Baayen, H., van Halteren, H., and Tweedie, F. (1996). Outside the cave of shadows: Using syntactic annotation to enhance authorship attribution. *Literary and Linguistic Computing*, 11(3):121–131.

Baeza-Yates, R. and Ribeiro-Neto, B., editors (1999). *Modern Information Retrieval.* Addison-Wesley, Reading, Massachusetts.

Bateman, J. A., Kamps, T., Kleinz, J., and Reichenberger, K. (2001). Towards constructive text, diagram, and layout generation for information presentation. *Computational Linguistics*, 27(3):409–449.

Biber, D. (1995). *Dimensions of Register Variation: A Cross-Linguistic Comparison.* Cambridge University Press, Cambridge.

Brinker, K. (1992). *Linguistische Textanalyse. Eine Einführung in Grundbegriffe und Methoden*. Erich Schmidt, Berlin.

Chang, C.-C. and Lin, C.-J. (2001). *LIBSVM: a library for support vector machines*. Software available at`http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Collins, M. and Duffy, N. (2001). Convolution kernels for natural language. In *NIPS*, pages 625–632.

Collins, M. and Duffy, N. (2002). New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *ACL*, pages 263–270.

Dehmer, M. (2005). *Strukturelle Analyse Web-basierter Dokumente*. Multimedia und Telekooperation. DUV, Berlin.

Geibel, P. and Wysotzki, F. (1996). Learning relational concepts with decision trees. In Saitta, L., editor, *Machine Learning: Proceedings of the Thirteenth International Conference*, pages 166–174, San Fransisco, CA. Morgan Kaufmann Publishers.

Gleim, R., Mehler, A., and Dehmer, M. (2006). Web corpus mining by instance of wikipedia. In Kilgariff, A. and Baroni, M., editors, *Proceedings of the EACL 2006 Workshop on Web as Corpus, April 3-7, 2006, Trento, Italy*, pages 67–74.

Halliday, M. A. K. and Hasan, R. (1989). *Language, Context, and Text: Aspects of Language in a Socialsemiotic Perspective*. Oxford University Press, Oxford.

Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The elements of statistical learning. Data Mining, Inference, and Prediction*. Springer, Berlin/New York.

Haussler, D. (1999). Convolution kernels on discrete structures. Technical Report UCS-CRL-99-10, UC Santa Cruz.

Heinemann, W. (2000). Textsorte – Textmuster – Texttyp. In Brinker, K., Antos, G., Heinemann, W., and Sager, S. F., editors, *Text- und Gesprächslinguistik. Linguistics of Text and Conversation*, pages 507–523. De Gruyter, Berlin/New York.

Hotho, A., Nürnberger, A., and Paaß, G. (2005). A Brief Survey of Text Mining. *LDV-Forum*, 20(1):19–62.

Ide, N., Bonhomme, P., and Romary, L. (2000). XCES: An XML-based standard for linguistic corpora. In *Proc. of LREC 2000, Athens*, pages 825–830.

Jain, B. J., Geibel, P., and Wysotzki, F. (2005). SVM learning with the Schur-Hadamard inner product for graphs. *Neurocomputing*, 64:93–105.

Joachims, T. (2002). *Learning to classify text using support vector machines*. Kluwer, Boston.

Karlgren, J. (1999). Non-topical factors in information access. In *WebNet (1)*, pages 27–31.

Kashima, H. and Koyanagi, T. (2002). Kernels for semi-structured data. In *ICML*, pages 291–298.

Kessler, B., Nunberg, G., and Schütze, H. (1997). Automatic detection of text genre. In *Proceedings of the 35th ACL and 8th EACL, Madrid*, pages 32–38.

Köhler, R. (1999). Syntactic structures. properties and interrelations. *Journal of Quantitative Linguistics*, 6:46–57.

Lee, Y.-B. and Myaeng, S. H. (2002). Text genre classification with genre-revealing and subject-revealing features. In *Proc. of the 25th Annual International ACM SIGIR Conf. on Research and Development in IR*, pages 145–150. ACM Press.

Lewis, D. D. (1992). Feature selection and feature extraction for text categorization. In *Proceedings of Speech and Natural Language Workshop*, pages 212–217.

Lindemann, C. and Littig, L. (2006). Coarse-grained classification of web sites by their structural properties. In *Proc. of WIDM'06*, pages 35–42.

Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., and Watkins, C. J. C. H. (2002). Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444.

Mann, W. C. and Thompson, S. A. (1988). Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8:243–281.

Martin, J. R. (1992). *English Text. System and Structure.* John Benjamins, Philadelphia.

Mehler, A. (2002). Hierarchical orderings of textual units. In *Proc. of COLING'02*, pages 646–652.

Mehler, A. (2007). Structure formation in the web. In Witt, A. and Metzing, D., editors, *Linguistic Modeling of Information and Markup Languages.* Springer, Dordrecht.

Mehler, A., Gleim, R., and Dehmer, M. (2006). Towards structure-sensitive hypertext categorization. In *Proc. of the 29th Annual Conf. of the GfKl, March 9-11, 2005, Universität Magdeburg*, pages 406–413.

Miller, G. A. and Charles, W. G. (1991). Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1):1–28.

Moschitti, A. (2004). A study on convolution kernels for shallow semantic parsing. In *Proc. of the 42th Conf. of the ACL*, pages 335–342.

Moschitti, A. (2006). Efficient convolution kernels for dependency and constituent syntactic trees. In *ECML*, pages 318–329.

Power, R., Scott, D., and Bouayad-Agha, N. (2003). Document structure. *Computational Linguistics*, 29(2):211–260.

Pustylnikov, O. (2006). How much information is provided by text structure? Automatic text classification using structural features (in German). Master thesis, University of Bielefeld, Germany.

Pustylnikov, O. and Mehler, A. (2007). A new look on register analysis: text classification by means of structural classifiers. In *Proceedings of the 10th International Pragmatics Conference (Göteborg, 8-13 July 2007).*

Santini, M. (2007). Characterizing genres of web pages: Genre hybridism and individualization. In *Proceedings of the 40th Annual Hawaii International Conference on System Sciences (HICSS'07).*

Schoelkopf, B. and Smola, A. J. (2002). *Learning with Kernels.* The MIT Press, Cambridge, MA.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47.

Sowa, J. F. (2000). *Knowledge Representation: Logical, Philosophical, and Computational Foundations.* Brooks/Cole, Pacific Grove.

Tuldava, J. (1998). *Probleme und Methoden der quantitativ-systemischen Lexikologie.* Wissenschaftlicher Verlag, Trier.

van Dijk, T. A. and Kintsch, W. (1983). *Strategies of Discourse Comprehension.* Academic Press, New York.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory.* Springer, New York.

Ventola, E. (1987). *The Structure of Social Interaction: a Systemic Approach to the Semiotics of Service Encounters.* Pinter, London.

Wolters, M. and Kirsten, M. (1999). Exploring the use of linguistic features in domain and genre classification. In *Proceedings of the EACL*, pages 142–149.

Jens Michaelis, Uwe Mönnich

Towards a Logical Description of Trees in Annotation Graphs

The importance of annotation graphs increases rapidly due to new developments in multi-media applications, text technology, and semantic web technologies. The paper provides a logical specification of trees in annotation graphs commonly used to code documents that are inherently structured on various levels.

1 Introduction

It is a matter of fact that a long history in artificial intelligence and computational linguistics tries to develop tools to extract semantic knowledge from syntactic information. In particular, from a text technological point of view the general research perspective is to extract (semantic) information from annotated documents. Regarding this aim, some of the relevant annotation models used are:

- Multilayer annotations
- Hyperlinks
- Discourse structure
- (Classical) linguistic description levels

An important aspect concerning annotation models is the development of a suitable logic system in order to specify the syntactic and semantic structures of such models. Due to the fact that heterogeneous subsystems must be incorporated, an amalgamation process can be assumed as the underlying architecture. The challenges of such architectures are twofold: first, the dynamic interaction between syntactic and semantic information must be represented and second, the efficiency of algorithms must be guaranteed.

Fortunately, in the case of annotation graphs, the object of these remarks, techniques from parameterized complexity theory can be exploited. This theory provides powerful tools for a detailed investigation of algorithmic problems. As it turns out, the concept of treewidth, indicating the similarity of a graph or a relational structure with a tree, is a parameter which helps to show that many otherwise intractable problems become computable in linear time when restricted to tree-like inputs.

The key feature of annotation graphs is their abstraction from the diversity of concrete formats used for the transcription of text and speech. This feature makes them an ideal candidate for the comparison of different annotation systems, e.g., those currently developed by several linguistic collaborative research centres in Germany.

Translating these different representation schemes into the framework of annotation graphs is a necessary prerequisite for the transfer of the pleasant computational properties of annotation graphs to the original systems. This is particularly important in those circumstances where the natural data structure of the concrete markup system cannot be readily understood as describing trees, or at least, multi-rooted trees, taken into account the possibilty of multiple annotation layers. Our main result can be interpreted as specifying the conditions under which algorithmic methods that were designed for trees can be applied to the realm of annotation systems that are apparently based on underlying graph structures.

A classical domain for multilayered annotations is linguistics. Utterances of speakers can be considered from different perspectives: examples are syntactic, semantic, discourse and intonation aspects, just to mention some of them. Representing these types of data in one representation format yields overlapping hierarchies. Moreover, the resulting structures are naturally considered as graphs rather than trees.

These challenges can be met by representing annotation graphs in logical form. Benefits of such a representation are the low descriptive complexity, the representability as open diagrams, or the abstract format for the interaction of different linguistic levels. Furthermore logical representations are amenable to the arsenal of techniques from logical graph theory.

2 Annotation Graphs

We start this section by explicitly providing the corresponding formal definitions, first, and by looking, in particular, at one example in more detail, afterwards.

2.1 Definitions and Examples

The underlying definition of an annotation graph is specified as follows:

Definition 2.1 (Bird and Liberman 2001) An *annotation graph (AG)*, G, over a label set L and a family of timelines $\langle \langle T_i, \leq_i \rangle \rangle_{i \in I}$, I being some index set,[1] is a 3-tuple $\langle N, A, \tau \rangle$ consisting of a node set N, a collection of labeled arcs $A \subseteq N \times N \times L$, and a partial time function $\tau : N \rightharpoonup \bigcup_{i \in I} T_i$, which satisfies the following two conditions:

(i) $\langle N, A \rangle$ is a labeled acyclic digraph containing no nodes of degree zero, and

(ii) for any path from node n_1 to n_2 in A, if $\tau(n_1)$ and $\tau(n_2)$ are defined, then there is a timeline $\langle T_i, \leq_i \rangle$ such that $\tau(n_1), \tau(n_2) \in T_i$, and such that $\tau(n_1) \leq_i \tau(n_2)$.

Remarks. AGs may be disconnected or empty, and they must not have orphan nodes. It follows from the definition that every piece of *connected* annotation structure can refer to at least one timeline. In Figure 1, an AG from the linguistics domain is depicted.

[1] Thus, for each $i \in I$, timeline $\langle T_i, \leq_i \rangle$ consists of a nonempty set T_i and a total order $\leq_i$ on T_i.

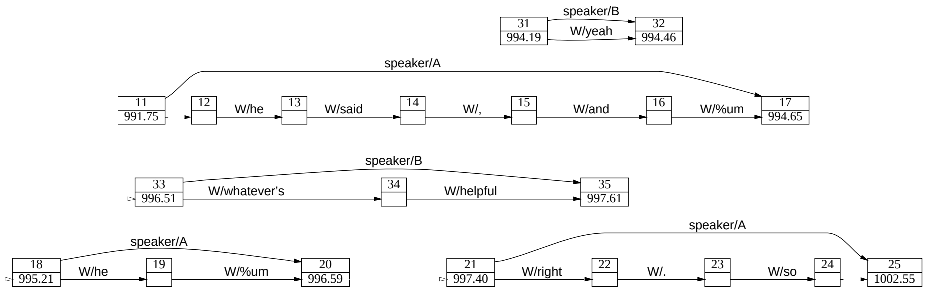


Figure 1: An example of an anchored annotation graph (cf. Bird and Liberman 2001).

Definition 2.2 (Bird and Liberman 2001) An *anchored AG* is an AG, G, defined as in Definition 2.1 and additionally satisfying the following condition:

(iii) if any node n does not have both incoming and outgoing arcs then $\tau : n \mapsto t$ for some time t.

Remarks. For anchored annotation graphs, it follows from the definition that every node has two bounding times, and timelines partition the node set. The AGs depicted in Figure 1 are anchored.

Definition 2.3 (Bird and Liberman 2001) A *totally anchored AG* is an anchored AG, G, defined as in Definition 2.2 such that the time function $\tau : N \rightharpoonup \bigcup_{i \in I} T_i$ is total.

Definition 2.4 A totally anchored AG, G, defined as in Definition 2.3 is *time-crossing arc free* if for all $\langle p, q, l\rangle, \langle r, s, m\rangle \in A$ such that $\tau(p), \tau(r) \in T_i$ for some i, and such that $\tau(p) \leq_i \tau(r)$ then either $\tau(q) \leq_i \tau(r)$ or $\tau(s) \leq_i \tau(q)$ holds.

In particular, the EXMARaLDA annotation tool, developed by the linguistic collaborative research centre in Hamburg, can be seen as strongly relying on the formal concept of AGs (cf. Schmidt 2005). In fact, not using the full range of possibilities provided by the general AG-definition given above, the core model underlying an EXMARaLDA basic transcription provides a so-called *single timeline, multiple tiers (STMT)* model, which in strict AG-terms can be understood as being a totally anchored AG consisting of exactly one timeline (cf. Figure 2). In line with the assumption of Bird and Liberman (2001, p. 12f) that reference to a single timeline implies that nodes with the same time reference should be considered to be identical, the AG depicted in Figure 2 can formally be specified as in the next example.

Example 2.5 For $T_{\text{ex}} = \{0, 1, 2, 3, 4, 5\}$ and $\leq_{\text{ex}} = \leq_{\mathbb{N}} \restriction T_{\text{ex}} \times T_{\text{ex}}$ consider the timeline $\langle T_{\text{ex}}, \leq_{\text{ex}}\rangle$.[2] Then for $\langle T_{\text{ex}}, \leq_{\text{ex}}\rangle$ and the label set L_{ex} implicitly specified via the

[2] $\mathbb{N}$ denotes the set of all non-negative integers, including 0. $\leq_{\mathbb{N}}$ is the canonical order on $\mathbb{N}$, and for each set $M \subseteq \mathbb{N}$, $\leq_{\mathbb{N}} \restriction M \times M$ is the restriction of $\leq_{\mathbb{N}}$ to $M \times M$.

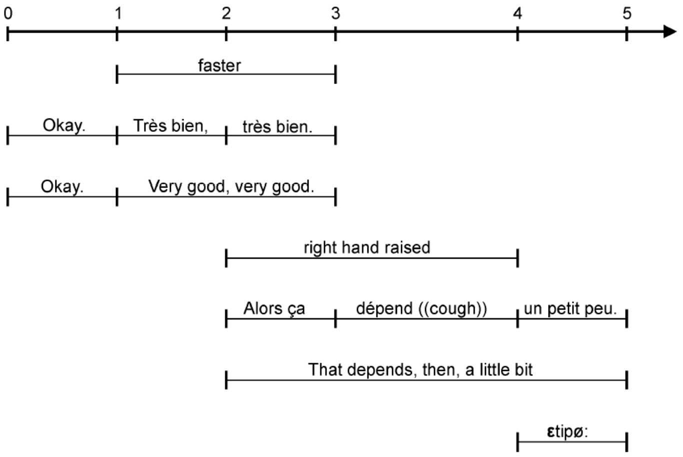


Figure 2: Example of an STMT model (cf. Schmidt 2005).

definition of the arc collection A_{ex}, let G_{ex} be the totally anchored AG $\langle N_{\text{ex}}, A_{\text{ex}}, \tau_{\text{ex}} \rangle$ over $\langle T_{\text{ex}}, \leq_{\text{ex}} \rangle$ (as the single timeline) and L_{ex}, where N_{ex} is the set $\{0, 1, 2, 3, 4, 5\}$, τ_{ex} is the identity function on $\{0, 1, 2, 3, 4, 5\}$ and A_{ex} consists of the following directed labeled edges:[3]

$$a_1^{(1)} = \langle 1\,,\,3\,,\,\text{faster}\rangle\,,$$
$$a_1^{(2)} = \langle 0\,,\,1\,,\,\text{Okay.}\rangle\,,$$
$$a_2^{(2)} = \langle 1\,,\,2\,,\,\text{Très bien,}\rangle\,,$$
$$a_3^{(2)} = \langle 2\,,\,3\,,\,\text{Très bien.}\rangle\,,$$
$$a_1^{(3)} = \langle 0\,,\,1\,,\,\text{Okay.}\rangle\,,$$
$$a_2^{(3)} = \langle 1\,,\,3\,,\,\text{Very good, very good.}\rangle\,,$$
$$a_1^{(4)} = \langle 2\,,\,4\,,\,\text{right hand hand raised}\rangle\,,$$
$$a_1^{(5)} = \langle 2\,,\,3\,,\,\text{Alors ça}\rangle\,,$$
$$a_2^{(5)} = \langle 3\,,\,4\,,\,\text{dépend ((cough))}\rangle\,,$$
$$a_3^{(5)} = \langle 4\,,\,5\,,\,\text{un petit peu.}\rangle\,,$$
$$a_1^{(6)} = \langle 2\,,\,5\,,\,\text{That depends, then, a little bit}\rangle\,,$$
$$a_1^{(7)} = \langle 4\,,\,5\,,\,\varepsilon\text{tipø:}\rangle\,.$$

[3]If the collection of arcs is, in fact, "simply" a set then $a_1^{(2)}$ and $a_1^{(3)}$ are identical. Since the intention here is actually that they are not identical, we assume them to be distinguishable either by some (at least technically) different labeling or by treating the collection of arcs as a multiset.

2.2 Logical Properties

In order to specify some logical properties, we need some additional concepts. First, we define a tree-decomposition of a graph.

Definition 2.6 (Robertson and Seymour 1986) A *tree-decomposition* of a graph $G = \langle V, E\rangle$ is a tree $T = \langle N, F\rangle$ together with a collection of subsets $\{T_u \,|\, u \in N\}$ of V such that $\bigcup_{u\in N} T_u = V$ and the following properties hold:[4]

1. For every edge $e = \langle a, b\rangle$ of G there is a $u \in N$ such that $\{a, b\} \subseteq T_u$
2. For all $u, v, w \in N$, if w lies on a path from u to v in T then $T_u \cap T_v \subseteq T_w$

The *width* of a tree-decomposition is equal to $\max_{u\in T}\{|T_u| - 1\}$

Note that a consequence of condition 2 from the above definition could be formulated as 2', namely, for all $u \in V$, $\langle\{v \in N \,|\, u \in T_v\}, F \cap (\{v \in N \,|\, u \in T_v\} \times \{v \in N \,|\, u \in T_v\}\rangle$ is a tree. The important concept of *treewidth*, defined next, is an indicator for the tree-likeness of a given graph G.

Definition 2.7 (Robertson and Seymour 1986) The *treewidth* of a graph G is the minimum value of k such that G has a tree-decomposition of width k.

The treewidth of a class of graphs C is naturally defined as the smallest number k such that for all graphs G in C, their treewidth is smaller or equal to k. Annotation graphs have unbounded treewidth since arbitrary large grids can be considered as annotation graphs. In practical applications, though, one is mainly concerned with finite families of documents represented as annotation graphs. Trivially, these families are of bounded treewidth. In the following we therefore restrict our attention to such finite families of annotation graphs.

Fact 2.8 *Finite families of anchored annotation graphs are of bounded treewidth.*

Monadic Second-Order Logic (MSO) is a subset of second-order logic. MSO extends first-order logic by allowing quantification over subsets of the universe of discourse. In other words quantification over second-order variables with at most one argument position is allowed.

Annotation graphs can be conceived as finite relational structures with the set of nodes understood as universe of discourse, the family of arcs as binary relations and the times as monadic predicates. A compact representation can be given in terms of an open diagram (cf. Prolog facts).

Theorem 2.9 (Courcelle) *Every property expressible in MSO is verifiable in linear time on graphs of bounded treewidth.*

[4]Here, a *tree* is taken to be a connected acyclic graph. Later we will restrict our attention to the concept of a finite ordered tree as it underlies our Definition 3.2 of a *finite labeled tree.*

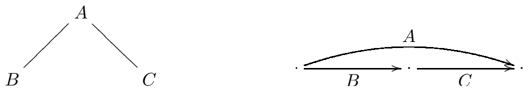
Courcelle's result is contained in Courcelle 1990. The principal tool in the proof of linear time decidability is the annotation of tree-like graph structures with logical types of bounded quantifier rank. These types assume the role of states in a bottom-up tree automaton.

Let MSO_2 designate the extended monadic second-order logic with quantification over sets of nodes and sets of edges.

Theorem 2.10 (Seese 1991) *If a set of graphs G has a decidable MSO_2 theory then it is the subset of the (homomorphic images of) a recognizable set of trees, i.e. it is* tree-definable *in the sense of Courcelle (2006).*

Remark. Trees are constructed from a finite set $\mathcal{F}$ of graph operations. Typical examples are disjoint union, relabeling of edges, addition of edges. Seese's theorem is an outgrowth of a combination of techniques from MSO-definable graph transductions and from the fundamental work of Robertson and Seymour on graph minors.

The representation of trees within the AG-framework is possible in terms of so-called *chart constructions*, where each tree node is mapped to an AG-arc, as outlined in the following easy example (cf. Cotton and Bird 2000):



The representation of trees in MSO_2-terms, relying on an order $\leq$, can be given via the notion of a *matching relation*. A set of edges M is called a *matching relation* if the following conditions are satisfied:

- $e \in M \wedge inc(e, u, v) \rightarrow u \leq v$ (M is compatible with $\leq$)
- $e \in M \wedge e' \in M \wedge inc(e, u, v) \wedge inc(e', w, z) \wedge u \leq w \leq v \rightarrow u \leq z \leq v$ (M is non-crossing)

where *inc* denotes the usual incidence relation.

Recall from the previous section the notion of a *totally anchored AG* that is *time-crossing arc free*. In case such an AG is anchored w.r.t. a single timeline, that notion is a particular instance of a matching relation.

The present section has served to emphasize the advantages that come with a logical description of graphs of bounded treewidth. The fact that a set of finite graphs is of bounded treewidth does not lead automatically to a representation of this set as a family of trees beyond the obvious tree decomposition as defined in Definition 2.6. Restricting the attention to the family of annotation graphs that satisfy the combined conditions characteristic of the single timeline, multiple tiers model, it can indeed be shown that these graphs allow a presentation in the form of multi-rooted trees. The next section is devoted to an elaboration of this claim.

3 Multilayer Annotation and STMT models

3.1 Multi-rooted Trees

Since our final aim will be to formally reconstruct an STMT model as a so-called *multi-rooted tree*, we here give some further explicit definitions setting the stage.

Definition 3.1 A *tree domain* is a nonempty set $D_\tau \subseteq \mathbb{N}^*$ such that for all $\chi \in \mathbb{N}^*$ and $i \in \mathbb{N}$ it holds that $\chi \in D_\tau$ if $\chi\chi' \in D_\tau$ for some $\chi' \in \mathbb{N}^*$, and $\chi i \in D_\tau$ if $\chi j \in D_\tau$ for some $j \in \mathbb{N}$ with $i < j$.[5]

Definition 3.2 A *finite labeled tree*, τ, is a quadruple of the form $\langle N_\tau, \triangleleft_\tau^*, \prec_\tau, \mathit{label}_\tau \rangle$ where the triple $\langle N_\tau, \triangleleft_\tau^*, \prec_\tau \rangle$ is a finite (ordered) tree defined in the usual sense,[6] i.e. up to an isomorphism $\langle N_\tau, \triangleleft_\tau^*, \prec_\tau \rangle$ is the natural (tree) interpretation of some tree domain D_τ,[7] and where label_τ is the *labeling function (of τ)*, i.e. a function from N_τ into some set of *labels*, L_τ.

Note that, if $\langle N_\tau, \triangleleft_\tau^*, \prec_\tau, \mathit{label}_\tau \rangle$, is a finite labeled tree, a tree domain, D_τ, whose natural (tree) interpretation is isomorphic to $\langle N_\tau, \triangleleft_\tau^*, \prec_\tau \rangle$ is uniquely determined. Note further that Definition 3.2 demands $\langle N_\tau, \triangleleft_\tau^*, \prec_\tau \rangle$ to be "only" isomorphic to the natural (tree) interpretation of some tree domain, D_τ, and not to be necessarily a tree domain itself. Such a definition allows us to have two finite labeled trees with disjoint set of nodes, which, of course, is not the case for the corresponding tree domains whose natural (tree) interpretations are isomorphic to the underlying (non-labeled) trees. This possibility is exploited within the next definition.

Definition 3.3 For any finite string $\alpha \in \Sigma^*$ for some finite alphabet Σ, a *multi-rooted tree (over α)* is a finite tuple of the form $\langle \tau_r \rangle_{r<R}$ for some $R \in \mathbb{N}$, where for each $r < R$, τ_r is a finite labeled tree $\langle N_r, \triangleleft_r^*, \prec_r, \mathit{label}_r \rangle$ such that for each $r, s < R$ with $r \neq s$ it holds that $N_r \cap N_s = \emptyset$, and such that for each $r < R$ there are some $k(r) \in \mathbb{N}$, $\alpha_0^{(r)}, \ldots \alpha_{k(r)}^{(r)} \in \Sigma^*$ and $w_0^{(r)}, \ldots w_{k(r)+1}^{(r)} \in \Sigma^*$ for which the yield of τ_r is of the form $\alpha_0^{(r)} \cdots \alpha_{k(r)}^{(r)}$, and for which α is of the form $w_0^{(r)}\alpha_0^{(r)}, \ldots w_{k(r)}^{(r)}\alpha_{k(r)}^{(r)}w_{k(r)+1}^{(r)}$.

Straightforwardly, adding a "super root" immediately dominating the tree components of a multi-rooted tree, $\langle \tau_r \rangle_{r<R}$, gives rise to a tree in the sense of Definition 3.2 again.

3.2 Finding a partition of a given STMT model into subgraphs

Given a totally anchored AG, G, consisting of exactly one timeline such that every two nodes with the same time reference are identical (i.e., G is an STMT model),

[5] For each set M, M^* is the Kleene closure of M, including ϵ, the empty string.

[6] N_τ is the finite, nonempty set of *nodes*, and $\triangleleft_\tau^*$ and $\prec_\tau$ are the respective binary relations of *dominance* and *precedence* on N_τ. Thus, $\triangleleft_\tau^*$ is the reflexive-transitive closure of $\triangleleft_\tau \subseteq N_\tau \times N_\tau$, the relation of *immediate dominance* on N_τ.

[7] In other words, up to an isomorphism N_τ is a tree domain such that for all $\chi, \psi \in N_\tau$ it holds that $\chi \triangleleft_\tau \psi$ iff $\psi = \chi i$ for some $i \in \mathbb{N}$, and $\chi \prec_\tau \psi$ iff $\chi = \omega i \chi'$ and $\psi = \omega j \psi'$ for some $\omega, \chi', \psi' \in \mathbb{N}^*$ and $i, j \in \mathbb{N}$ with $i < j$.

the considerations of this subsection concern the aim of finding a partition of G into subgraphs each of which being without time-crossing arcs in the sense of Definition 2.4. This aim is motivated by the observation that time-crossing of two edges is usually caused by interference of two different layers of annotation, while single layers consist of edges in matching form, i.e. time-crossing arc free. A corresponding partition is not necessarily unique and there are, of course, several possible algorithms in order to calculate a particular instance of such a partition. Heuristics may help to decide which algorithm is to be preferred in terms of its computational properties. As an example, we here present an algorithm which can be seen as being a representative of a "left-to-right, top-down" strategy (cf. Figure 3). The perspective motivating this "tree-traversing" metaphor results from our final aim to reconstruct the given AG as a multi-rooted tree by using the corresponding partition. In the next section we will show, that such a reconstruction can be done "on the fly," when calculating the partition according to the presented algorithm.

For the rest of this section let $G = \langle N, A, \tau \rangle$ be a totally anchored AG over a label set L and a single timeline $\langle T, \leq_T \rangle$ such that every two nodes with the same time reference are identical. W.l.o.g. we may assume T is of the form $\{0, 1, \dots, K-1\}$ for some $K \in \mathbb{N}$ such that τ is surjective, i.e., in particular the order on T is induced by the canonical order on $\mathbb{N}$. Since we are concerned with a single timeline and a totally anchored graph, we can also identify the set of vertices, N, with T. Thus, the cardinality of N is K. We construct the "upper part" of the corresponding adjacency matrix, $\langle A_{p,q} \rangle_{0 \leq p < q < K}$, where each entry $A_{p,q}$ consists of all arcs $a = \langle p, q, l \rangle \in A$ for some $l \in L$. The algorithm which provides us with a partition of A, Part A, is the following:

```
**partition** $A$
1   $r \leftarrow 0$
2   **construct** $A_r$                 % construct first partition set
3   Part $A \leftarrow \langle A_r \rangle$                 % initialize the list of partition sets
4   UNTIL $A_r = \emptyset$ REPEAT
5     $r \leftarrow r + 1$
6     **construct** $A_r$               % construct next (potential) partition
7     IF $A_r \neq \emptyset$ THEN
8       Part $A \leftarrow$ Part $A \cdot \langle A_r \rangle$   % append next partition set to list
9     FI
10  RETURN Part $A$
```

Here the subprocedure **construct** A_r is defined as:

```
**construct** $A_r$
1   $A_r \leftarrow \emptyset$
2   IF $K > 0$ THEN[8]
3     **explore** $A_{0,K-1}$
4   FI
```

[8]Note that, because of (i) of Definition 2.1, $K > 1$ if N is nonempty.

For $0 \leq p < q < K$, **explore** $A_{p,q}$ is defined as follows:

explore $A_{p,q}$

```
1  IF $A_{p,q} \neq \emptyset$ THEN
2    choose $a \in A_{p,q}$
3    $A_r \leftarrow A_r \cup \{a\}$               %add a to partition set $A_r$
4    $A_{p,q} \leftarrow A_{p,q} \setminus \{a\}$               %remove a from set $A_{p,q}$
5  FI
6  $i \leftarrow 0$
7  $j \leftarrow 1$
8  WHILE $p+i < q-j$ and $A_{p+i,q-j} = \emptyset$ REPEAT %searching a leftmost ‘‘child’’
9    IF $p+i < q-j-1$ THEN     %exploring ‘‘top-down’’ interval [p+i,q-j]
10     $j \leftarrow j+1$                %reducing right interval value by 1
11   ELSE
12     $i \leftarrow i+1$                %increasing left interval value by 1
13     $j \leftarrow 0$                  %right interval value back to q
14   FI
15 $i_{p,q} \leftarrow i$
16 $j_{p,q} \leftarrow j$
17 IF $p + i_{p,q} < q - j_{p,q}$ THEN     %there is no ‘‘child’’ at all iff $p+i_{p,q}=q-j_{p,q}$
18   explore $A_{p+i_{p,q},q-j_{p,q}}$       %exploring leftmost ‘‘child’’ -- nonemptiness
                                  of $A_{p+i_{p,q},q-j_{p,q}}$ is guaranteed by WHILE-loop
19   IF $j_{p,q} > 0$ THEN
20     explore $A_{q-j_{p,q},q}$           %searching for right ‘‘sibling’’ of leftmost
                                  ‘‘child’’
21   FI
22 FI
```

Fact 3.4 *For $p, q < K$ with $p < q$, $i_{p,q}$ and $j_{p,q}$ (depending on $i_{p,q}$) are minimal, i.e., $A_{p+i,q-j} = \emptyset$ for all $j < q-(p+i)$ in case $i < i_{p,q}$, and for all $j < j_{p,q}$ in case $i = i_{p,q}$.*

Remark. Note that for $p, q < K$ with $p < q$, the WHILE-loop for potentially finding minimal i and minimal j, depending on i, within the subprocedure **explore** $A_{p,q}$ is a “left-to-right, top-down” search along the “rows” left-to and below matrix entry $A_{p,q}$ (cf. Figure 3). In particular, we have $A_{p+i,q-j} = \emptyset$ if $i < i_{p,q}$ and $q-(p+i_{p,q}) \leq j < q-(p+i)$. Hence **explore** $A_{p,p+i_{p,q}}$ would yield no contribution to the partition set A_r, if it were part of the subprocedure **explore** $A_{p,q}$.

Proposition 3.5 *The time complexity of* **partition** A *is in $O(K^2m)$, m being the cardinality of A.*

Proposition 3.6 *Let $k \in \mathbb{N}$. Then for $K = 2k$, the label set $L = \{l_0, l_1, \ldots, l_{k-1}\}$ and the arc set $A = \{\langle p, p+k, l_p\rangle \mid 0 \leq p < k\}$ the time complexity of* **partition** A *is at least in $O(K^2m)$ (as before, m being the cardinality of A, i.e., $m = k$ in this case).*

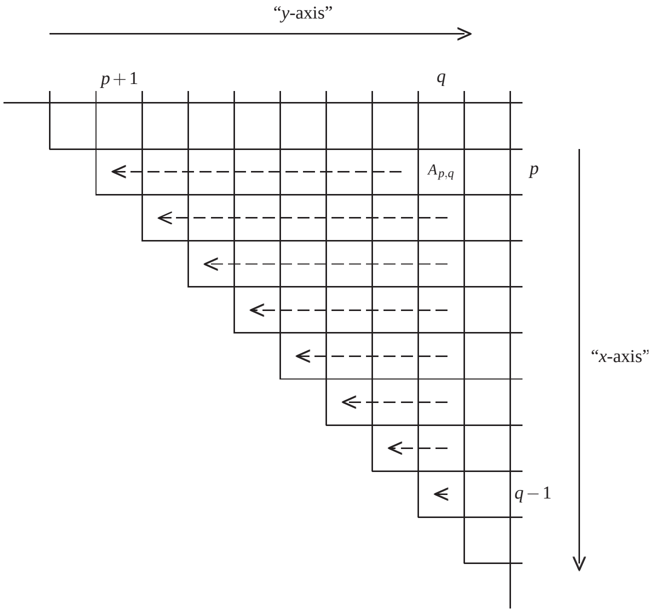


Figure 3: Potential search space for i and j within WHILE-loop of **explore** $A_{p,q}$.

As mentioned above, in order to find an appropriate partition of A, the algorithm given above only represents one of several more general possibilities depending on the strategy pursued. In fact, applying the algorithm to the example above yields a partition into three arc sets which might "contra-intuitively" partition the second, third and fifth layer, depending on which arc is chosen first from the arc subsets $A_{0,1}$, $A_{1,3}$, $A_{2,3}$ and $A_{4,5}$ according to line 2 of the subprocedure **explore**. If the second layer is not partitioned, the fifth will be, and vice versa. This is due to the fact that both $a_2^{(2)}$ and $a_2^{(5)}$ definitely belong to the set A_0, while $a_3^{(2)}$ and $a_1^{(5)}$ never will do so at the same time.

Example 3.7 (continued) Applying **partition** to A_{ex} yields a partition into three sets, A_0, A_1 and A_2, of the form

$$A_0 = \{x_0\,,\, x_1\,,\, a_2^{(2)}\,,\, x_2\,,\, a_2^{(5)}\,,\, x_3\}\,,\; A_1 = \{x'_0\,,\, x'_1\,,\, x'_2\,,\, x'_3\} \text{ and } A_2 = \{a_1^{(6)}\,,\, a_1^{(4)}\}$$

with $x_i, x'_i \in X_i$ such that $x_i \neq x'_i$ for $0 \leq i \leq 3$, where

$$X_0 = \{a_1^{(2)}, a_1^{(3)}\}\,,\; X_1 = \{a_1^{(1)}, a_2^{(3)}\}\,,\; X_2 = \{a_3^{(2)}, a_1^{(5)}\} \text{ and } X_3 = \{a_3^{(5)}, a_1^{(7)}\}.$$

Note that the situation of "algorithmic arbitrariness" immediately changes if we have further access to "external" information which can be used to guide the selection from an arc subset $A_{p,q}$, like hierarchical structure and ontological information in the sense of Bird and Liberman (2001, Section 3.2f), e.g., in terms of type and speaker as in an EXMARaLDA Basic Transcription Model itself based on a pure STMT model (cf. Schmidt 2005 and Figure 4 here).

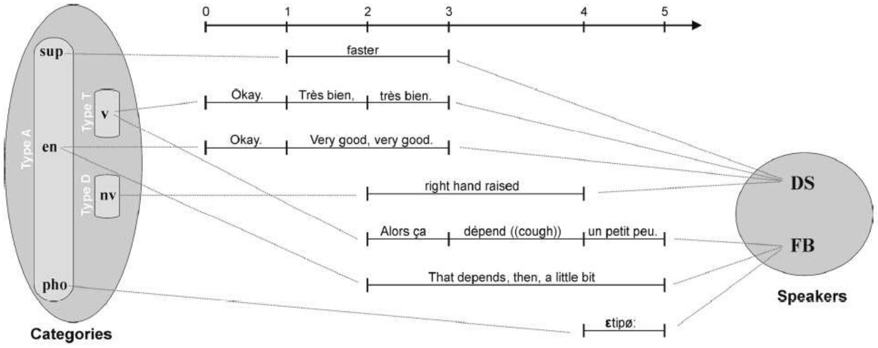


Figure 4: Example of an EXMARaLDA Basic Transcription Model (cf. Schmidt 2005).

3.3 Building a multi-rooted tree from an STMT model - "Reversing the chart construction"

Consistently taking over all formal prerequisites, we, in particular, let $G = \langle N, A, \tau \rangle$ be the totally anchored AG over a label set L and a single timeline $\langle T, \leq_T \rangle$ from the previous section. In order to build—on the fly—a multi-rooted tree, MultTree A, from Part A, the partition of A, created by applying **partition** to A, we slightly adapt and extend the algorithm.[9]

[9] Modified lines are indicated by a prime following the line number (x'). Additional lines are indicated by a subsequent line number counting (x.1, x.2 etc.).

```
partition $A$
% including the construction of a multi-rooted tree, MultTree $A$, from Part $A$
1      $r \leftarrow 0$
2'     construct $A_r, T_r$                          % construct first partition set and
                                                       corresponding tree
3      Part $A \leftarrow \langle A_r \rangle$                       % initialize list of partition sets
3.1    MultTree $A \leftarrow \langle T_r \rangle$                   % initialize multi-rooted tree (list)
4      UNTIL $A_r = \emptyset$ REPEAT
5        $r \leftarrow r + 1$
6        construct $A_r, T_r$                        % construct next (potential) partition
                                                       set and corresponding tree
7        IF $A_r \neq \emptyset$ THEN
8          Part $A \leftarrow$ Part $A \cdot \langle A_r \rangle$            % append next partition set to list
8.1        MultTree $A \leftarrow$ MultTree $A \cdot \langle T_r \rangle$    % append next tree to multi-rooted
                                                       tree (list)
9        FI
10     RETURN Part $A$
10.1   RETURN MultTree $A$
```

The subprocedure **construct** A_r, T_r is "just" an extension of **construct** A_r.

```
construct $A_r, T_r$
1      $A_r \leftarrow \emptyset$
1.1    $T_r \leftarrow \emptyset$
2      IF $K > 0$ THEN[10]
2.1      $\chi_{0,K-1} \leftarrow \epsilon$                 % define (essential part of) root address of $T_r$
2.2      $k_{0,K-1} \leftarrow \epsilon$                 % dummy for line 4.1 of procedure explore $A_{0,K-1}$
2.3      real root$_{0,K-1} \leftarrow$ FALSE     % potentially no arc from node 0 to node K-1
3        explore $A_{0,K-1}$
4      FI
```

For $0 \leq p < q < K$, the modified subprocedure **explore** $A_{p,q}$ is now given by

[10] Recall that, because of (i) of Definition 2.1, $K > 1$ if N is nonempty.

```
explore $A_{p,q}$
1      IF $A_{p,q} \neq \emptyset$ THEN
2        choose $a \in A_{p,q}$
3        $A_r \leftarrow A_r \cup \{a\}$                        % add a to partition set A_r
4        $A_{p,q} \leftarrow A_{p,q} \setminus \{a\}$                      % remove a from set A_p,q
4.1      $\chi_{p,q} \leftarrow \chi_{p,q} \cdot k_{p,q}$                      % define (essential part of) new
                                                 node address for T_r
4.2      $T_r \leftarrow T_r \cup \{\langle\langle\chi_{p,q}, r\rangle, label(a)\rangle\}$        % add to T_r a corresponding new
                                                 node labeled by label(a)[11]
4.3      real root$_{p,q}$ $\leftarrow$ TRUE                      % no dummy node, cf. line 4.10, 11.1
4.4      $k \leftarrow 0$                                % potential next daughter to find
                                                 is leftmost child
4.5      IF $p + 1 = q$ THEN
4.6        $T_r \leftarrow T_r \cup \{\langle\langle\chi_{p,q} \cdot k, r\rangle, \langle p, p+1\rangle\rangle\}$ % yield of T_r comprises (arc cover-
                                                 ing) interval [p,p+1]
4.7      FI
4.8    ELSE
4.9      IF $p = 0$ and $q = K - 1$ THEN
4.10       $T_r \leftarrow T_r \cup \{\langle\langle\epsilon, r\rangle, dummy\text{-}label_r\rangle\}$  % dummy root covering potential non-
                                                 time-crossing, non-inclusive arcs
                                                 (cf. T_0 and T_1 from Example 3.9)
4.11       $k \leftarrow 0$
4.12     ELSE
4.13       $k \leftarrow k_{p,q}$
4.14     FI
5      FI
6      $i \leftarrow 0$
7      $j \leftarrow 1$
```

[11]For each arc $a = \langle p, q, l\rangle \in A$ for some $p, q \in N$ and $l \in L$, we take $label(a)$ to denote its label l.

```
8       WHILE $p+i < q-j$ and $A_{p+i,q-j} = \emptyset$ REPEAT %searching leftmost ‘‘child’’
9         IF $p+i < q-j-1$ THEN
10          $j \leftarrow j+1$
11        ELSE
11.1        IF real $\text{root}_{p,q}$ = TRUE THEN
11.2          $T_r \leftarrow T_r \cup \{\langle\langle\chi_{p,q} \cdot k, r\rangle, \langle p+i, p+i+1\rangle\rangle\}$ % yield of $T_r$ comprises (arc
                                                    covering) interval [p+i,p+i+1]
11.3          $k \leftarrow k+1$
11.4        FI
12          $i \leftarrow i+1$
13          $j \leftarrow 0$
14        FI
15      $i_{p,q} \leftarrow i$
16      $j_{p,q} \leftarrow j$
17      IF $p + i_{p,q} < q - j_{p,q}$ THEN
17.1      $\chi_{p+i_{p,q},q-j_{p,q}} \leftarrow \chi_{p,q}$
17.2      $k_{p+i_{p,q},q-j_{p,q}} \leftarrow k$
17.3      real $\text{root}_{p+i_{p,q},q-j_{p,q}} \leftarrow$ real $\text{root}_{p,q}$
18        **explore** $A_{p+i_{p,q},q-j_{p,q}}$
19        IF $j_{p,q} > 0$ THEN
19.1        $\chi_{p+i_{p,q},q-j_{p,q}} \leftarrow \chi_{p,q}$
19.2        $k_{q-j_{p,q},q} \leftarrow k_{p+i_{p,q},q-j_{p,q}} + 1$
19.3        real $\text{root}_{q-j_{p,q},q} \leftarrow$ real $\text{root}_{p,q}$
20          **explore** $A_{q-j_{p,q},q}$                %searching for right ‘‘sibling’’ of
                                                    leftmost ‘‘child’’
21        FI
22      FI
```

Remark. The previously stated results on the time complexity bounds of **partition** A are not affected (cf. Proposition 3.5 and 3.6).

Proposition 3.8 *For each T_r appearing as a component in MultTree A, let L_r denote the (label) set $L \cup \{dummy\text{-}label_r\} \cup \{\langle p, p+1\rangle \,|\, 0 < p < K-1\}$. Then the set of first components of the first components of the elements of T_r, $\{\chi \in \mathbb{N}^* \,|\, \langle\langle\chi, r\rangle, l\rangle \in T_r$ for some $l \in L_r\}$, constitutes a tree domain. In this sense T_r can straightforwardly be interpreted as a finite labeled tree. All non-terminal nodes of T_r, except for the root node, are necessarily labeled by arc labels from L. The root node is labeled by dummy-label$_r$ in case $A_{0,K-1}$ was empty, while processing* **explore** *$A_{0,K-1}$. Otherwise it is also labeled by an element from L. Each leaf is labeled by $\langle p, p+1\rangle$ for some $p < K-1$.*

Example 3.9 (continued) Applying the modified algorithm **partition** to A_{ex}, in particular yields a multi-rooted tree MultTree $A_{\text{ex}} = \langle T_0, T_1, T_2\rangle$ with

$$
\begin{aligned}
T_0 = \{ & \langle \langle \epsilon , 0 \rangle , \textit{dummy-label}_0 \rangle , \langle \langle 0 , 0 \rangle , \textit{label}(x_0) \rangle , \langle \langle 00 , 0 \rangle , \langle 0 , 1 \rangle \rangle , \\
& \langle \langle 1 , 0 \rangle , \textit{label}(x_1) \rangle , \langle \langle 10 , 0 \rangle , \textit{label}(a_2^{(2)}) \rangle , \langle \langle 100 , 0 \rangle , \langle 1 , 2 \rangle \rangle , \\
& \langle \langle 11 , 0 \rangle , \textit{label}(x_2) \rangle , \langle \langle 110 , 0 \rangle , \langle 2 , 3 \rangle \rangle , \langle \langle 2 , 0 \rangle , \textit{label}(a_2^{(5)}) \rangle , \\
& \langle \langle 20 , 0 \rangle , \langle 3 , 4 \rangle \rangle , \langle \langle 3 , 0 \rangle , \textit{label}(x_3) \rangle \rangle , \langle \langle 30 , 0 \rangle , \langle 4 , 5 \rangle \rangle \}
\end{aligned}
$$

$$
\begin{aligned}
T_1 = \{ & \langle \langle \epsilon , 1 \rangle , \textit{dummy-label}_1 \rangle , \langle \langle 0 , 1 \rangle , \textit{label}(x'_0) \rangle , \langle \langle 00 , 1 \rangle , \langle 0 , 1 \rangle \rangle , \\
& \langle \langle 1 , 1 \rangle , \textit{label}(x'_1) \rangle , \langle \langle 10 , 1 \rangle , \langle 1 , 2 \rangle \rangle , \langle \langle 11 , 1 \rangle , \textit{label}(x'_2) \rangle , \\
& \langle \langle 110 , 1 \rangle , \langle 2 , 3 \rangle \rangle , \langle \langle 2 , 1 \rangle , \textit{label}(x'_3) \rangle \rangle , \langle \langle 20 , 1 \rangle , \langle 4 , 5 \rangle \rangle \}
\end{aligned}
$$

$$
\begin{aligned}
T_2 = \{ & \langle \langle \epsilon , 2 \rangle , \textit{dummy-label}_2 \rangle , \langle \langle 0 , 2 \rangle , \textit{label}(a_1^{(6)}) , \langle \langle 00 , 2 \rangle , \textit{label}(a_1^{(4)}) , \\
& \langle \langle 000 , 2 \rangle , \langle 2 , 3 \rangle \rangle , \langle \langle 001 , 2 \rangle , \langle 3 , 4 \rangle \rangle , \langle \langle 01 , 2 \rangle , \langle 4 , 5 \rangle \rangle \}
\end{aligned}
$$

with $x_i, x'_i \in X_i$ such that $x_i \neq x'_i$ for $0 \leq i \leq 3$, where

$$X_0 = \{a_1^{(2)}, a_1^{(3)}\} , \ X_1 = \{a_1^{(1)}, a_2^{(3)}\} , \ X_2 = \{a_3^{(2)}, a_1^{(5)}\} \text{ and } X_3 = \{a_3^{(5)}, a_1^{(7)}\}$$

(cf. Figure 5).

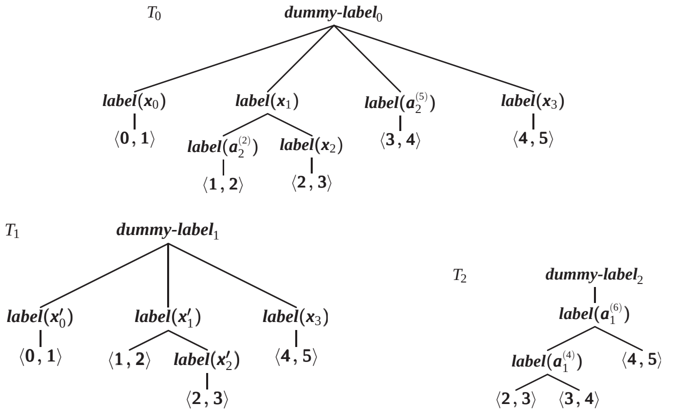


Figure 5: The tree components of the multi-rooted tree $\mathrm{MultTree}\, A_{\mathrm{ex}} = \langle T_0, T_1, T_2 \rangle$.

4 Envoi

In this paper we have sketched the beginnings of a logical theory of annotation graphs. Along the way we have tried to emphasize the following points:

- Abstract logical framework with multilayer capabilities for linguistic annotations
- Compact logical representation
- Efficient MSO theory

- Subset of regular trees
- Internal definability of tree structure
- Partition of annotation layers

While the logical approach towards annotation models provides a unified format for the syntactic level it still has to be complemented with a component that serves to integrate syntactic with semantic structures. Primary candidates for this component are amalgamation techniques from model theory and the assembly of heterogeneous formal specifications via transformation systems. Care must be taken in this effort to preserve the nice complexity properties that are associated with finite graphs of bounded treewidth. On the other hand, annotation graphs offer a minimal formalization of typical transcription needs by means of acyclic graphs with fielded records on the edges. Semantic information is easily integrated into this minimal framework. It is for this reason that we believe that our general perspective on the formal properties of annotation graphs will retain its value if additional types of annotation are added to the current format of transcription schemes.

Acknowledgements

This research was partially supported by grant MO 386/3-4 of the German Research Foundation (DFG).

References

Bird, S. and Liberman, M. (2001). A formal framework for linguistic annotation. *Speech Communication*, 33:23–60.

Cotton, S. and Bird, S. (2000). An integrated framework for treebanks and multilayer annotations. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC 2002),* Las Palmas de Gran Canaria, pages 1670–1677. European Language Resources Association.

Courcelle, B. (1990). The monadic second-order logic of graphs I: Recognizable sets of finite graphs. *Information and Computation*, 85:12–75.

Courcelle, B. (2006). The monadic second-order logic of graphs XV: On a conjecture by D. Seese. *Journal of Applied Logic*, 4:79–114.

Robertson, N. and Seymour, P. D. (1986). Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms*, 7:309–322.

Schmidt, T. (2005). Time-based data models and the text encoding initiative's guidelines for transcription of speech. Arbeiten zur Mehrsprachigkeit, Folge B, Nr. 62 (Working Papers in Multilingualism, Series B, No. 62), Universität Hamburg, Hamburg.

Seese, D. (1991). The structure of the models of decidable monadic theories of graphs. *Annals of Pure and Applied Logic*, 53:169–195.

Autorenverzeichnis

Peter Geibel
Institut für Kognitionswissenschaft
Universität Osnabrück
Albrechtstraße 28
D-49076 Osnabrück
pgeibel@uos.de

Kai-Uwe Kühnberger
Institut für Kognitionswissenschaft
Universität Osnabrück
Albrechtstraße 28
D-49076 Osnabrück
kkuehnbe@uos.de

Harald Lüngen
Institut für Germanistik - ASCL
Justus-Liebig-Universität Gießen
Otto-Behaghel-Straße 10
D-35394 Gießen
luengen@uni-giessen.de

Alexander Mehler
Computerlinguistik und Texttechnologie
Fakultät für Linguistik
und Literaturwissenschaft
Universität Bielefeld
Postfach 10 01 31
Universitätsstraße 25
D-33501 Bielefeld
alexander.mehler@uni-bielefeld.de

Jens Michaelis
Institut für Kognitionswissenschaft
Universität Osnabrück
Albrechtstraße 28
D-49076 Osnabrück
jens.michaelis@uni-osnabrueck.de

Uwe Mönnich
Seminar für Sprachwissenschaft
Arbeitsbereich Theoretische
Computerlinguistik
Universität Tübingen
Wilhelmstraße 19
D-72074 Tübingen
um@sfs.uni-tuebingen.de

Ekaterina Ovchinnikova
Institut für Kognitionswissenschaft
Universität Osnabrück
Albrechtstr. 28
D-49076 Osnabrück
e.ovchinnikova@gmail.com

Olga Pustylnikov
Fakultät für Linguistik und
Literaturwissenschaft
Universität Bielefeld
Universitätsstraße 25
D-33615 Bielefeld
olga.pustylnikov@uni-bielefeld.de

Klaus U. Schulz
Centrum für Informations- und
Sprachverarbeitung
Ludwig-Maximilians-Universität
München
Oettingenstraße 67
D-80538 München
schulz@cis.uni-muenchen.de

Angelika Storrer
Institut für deutsche Sprache und
Literatur
Universität Dortmund
D-44221 Dortmund
angelika.storrer@uni-dortmund.de

Eduardo Torres Schumann
Centrum für Informations- und Sprachverarbeitung
Ludwig-Maximilians-Universität München
Oettingenstraße 67
D-80538 München
torressc@cip.ifi.lmu.de