Content Management and Knowledge Management:

Two Faces of Ontology-based Deep-Level Interpretation of Text

Vom Promotionsausschuss der
Technischen Universität Hamburg-Harburg
zur Erlangung des akademischen Grades

Doktorin der Naturwissenschaften
genehmigte Dissertation

von

Irma Sofía Espinosa Peraldí
aus Morelia, Mexiko

2011

Reviewers:
Prof. Dr. Ralf Möller
Prof. Dr. Menzel
Prof. Dr. Mayer-Lindenberg

Day of the defense:
16th of September 2011

Abstract

In order to make content management systems able to exploit annotations about content semantics for the management of content, this work proposes an approach for text interpretation that is built on the principles of logic-based abduction. The approach is presented in the context of a framework that allows for a deep-level interpretation of text. The research contributions of this work encompass (1) a framework for the integration of shallow-processing techniques of text with logic-based techniques to cope with the extraction of content descriptions that describe media contents in detail; (2) the design of a logic-based process for multimedia fusion that supports the systematic combination of interpretation results obtained from different types of media (e.g., text and image); (3) the description of design patterns for domain ontologies and rules useful for logic-based media interpretation; (4) the use of text interpretation to guide the task of ontology design ("Grounded Ontology Design") as a strategy to support knowledge management in an organization; (5) the design and implementation of an application that shows the advantages of using annotations to support location-aware and situation-specific services as a new kind of content management.

Kurzfassung

Damit Content Management Systeme fähig sind, Daten auf der Basis von semantischen Strukturen systematisch zu verwalten, wird eine logikbasierte Form der Textinterpretation untersucht. Aufbauend auf Prinzipien der logischen Abduktion stellt die Arbeit ein Rahmenwerk vor, das eine "tiefe" Interpretation von Texten ermöglicht. Die wissenschaftlichen Beiträge umfassen (1) die Integration von Techniken zur einfachen Analyse von Textinhalten, gekoppelt mit Techniken zur logikbasierten Interpretation, um Strukturen zu gewinnen, die die Inhalte von Texten in multimedialen Dokumenten genauer beschreiben; (2) die Gestaltung eines logikbasierten Ansatzes zur Integration (Fusion) von Interpretationsergebnissen aus verschiedenen Dokumentteilen (Text und Bild); (3) die Beschreibung von Design Patterns für Ontologien zur Steuerung der Wissensakquisition für die Interpretation; (4) die Steuerung des Ontologie-Designs über die Fähigkeit zur automatischen Textinterpretation ("Grounded Ontology Design") als ein strategisches Konzept zur Unterstützung des Wissensmanagements in einer Organisation; (5) die Beschreibung der Konzeption und Umsetzung einer Anwendung, die die Vorteile der Verwendung von semantischen Strukturen zum Content Management zeigt.

To my dear parents and husband

A mis padres y esposo

Acknowledgments

I would like to thank my advisor Prof. Dr. Ralf Möller for his continuous support on time, ideas and funding to make these five years come to a successful end, and for encouraging me and trusting me with a very exciting topic of research. I would also like to thank Prof. Dr. Menzel and Prof. Dr. Mayer-Lindenberg for reviewing my work.

My acknowledgment also goes to the colleagues of the STS research group at the TUHH, who contributed a lot in my professional and personal life: Sylvia Melzer, Thomas Rahmlow, Hartmut Gau, Andrey Galochkin, Anahita Nafissi, Thomas Sidow, Kamil Sokolski, Tobias Näth and Ulrike Hantschmann. Special thanks to Dr. Atila Kaya and Dr. Michael Wessel for providing me always with directions and valuable time for discussions.

During these five years I also had the opportunity to exchange ideas with numerous scientists around the world. Among others I would especially like to express my gratitude to: Dr. Héctor Francisco Ruiz Paredes, Dr. George Paliouras, Dr. Jerry Hobbs and Dr. Alfio Ferrara.

I want to thank for the very important emotional support to my dear parents Irma Serafina Peraldí León and José Luis Espinosa Hurtado, my brothers Luis Fernando and Daniel, as well as my husband Savvas Katemliadis. Without you I would not be able to get this far.

The first two years of my studies abroad were supported through a common scholarship by the Mexican Council for Science and Technology (CONACYT) and the German Academic Exchange Service (DAAD), which allowed me to launch my academic career for the following six years.

Finally, I also want to thank my friends who supported me during my time in Germany, some of which also helped me to proof-read my final written work: Dr. Sebastian Bossung, Friedemann Lindemann, Dr. Özgür Özcep, Dimitra Gertsaki, Dr. Patrick Hupe, Dr. Ana Gabriela Valladares Juárez, Edalith Guzmán Rivera, Mónica Yadira Narváez Clemente, Freya Carstens and Dr. Volker Carstens, Karina Marx, Margarita Katein, the Rampf family and Christina Brohr.

Contents

List of Figures

Note: Figures containing athletics images and athletics news are reproduced here with permission granted by IAAF.

Chapter 1

Introduction

"Those who *read* books are overtaken by those who *remember* what they read. These are overtaken by those who *understand* what they read and these by those who *apply* what they understood". This proverb can be used as a metaphor between humans and information systems (see Figure 1.1, page 2) to highlight various aspects of information processing. First, to highlight how the information conveyed by media is best exploited with the objective of solving a problem. Second, to highlight that a process of media content understanding is necessary in order to identify the information that helps in solving a problem. Third, to highlight that the processes of media content understanding and problem solving (apply) imply the need of knowledge and intelligence.

In this context, Figure 1.1 depicts the involvement of Computer Science in topics such as information storage (read), information retrieval (remember), media content interpretation (understand) and Decision Support Systems (DSSs). The problems of storage and retrieval of information are long solved in Computer Science, and with it the creation of information systems has been facilitated. Information Systems (ISs) have become an integral part of any organization providing support to business critical processes from a range of classical application areas, e.g., human resources, financial sector, logistics, medicine, etc. Commonly such ISs are based on a three-tier architecture using structured information at the data-tier, i.e., relational databases. On the other hand, the volume of unstructured information, i.e., media such as text documents, images, audio, etc., is rapidly growing and is also becoming a constituent element of ISs, creating the need for using services of Content Management Systems (CMSs). For example, in a medical scenario an information system contains structured information such as a patient's personal information, treatment history, registry of monitoring systems, etc., as well as unstructured information such as text documents with doctor's remarks for a specific disease, images of radiographies, etc. While structured information can be easily accessed by a database system, the access to unstructured information is limited to the services offered

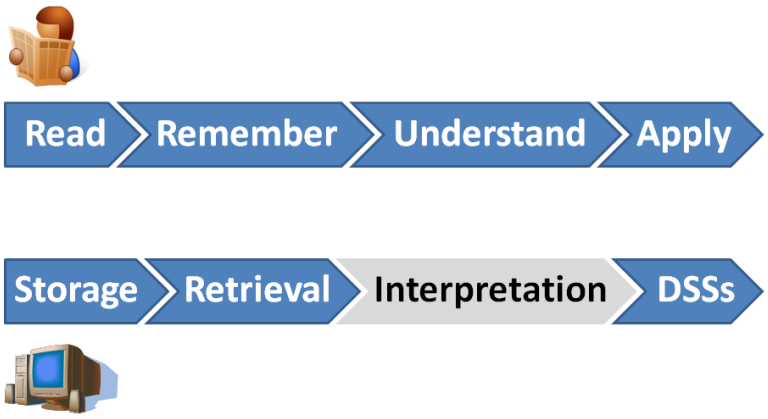


Figure 1.1: Relevant information processes

by current CMSs. Currently, CMSs are able to manage data from media by means of *format*, e.g., ASCII, JPEG, etc., *structure*, e.g., title and paragraphs in text or spatial and temporal segments in video, etc., as well as *editorial information*, e.g., author, edition date, as well as manually provided *metadata*, etc.[1] But content management on the basis of *content semantics*[2] is still beyond the abilities of current CMSs. This means that the media is accessed by ISs as plain objects and the content semantics are only exploited by human end-users. In order to provide systems with the necessary capability to exploit content semantics, annotations are needed. *Annotations* are here defined as machine-processable structures that describe in detail the content of media. The manual creation of annotations is obviously time consuming and difficult to achieve given the high volumes of media being produced. Thus, the automated creation of annotations is required. This requirement motivates the automatic interpretation of media, which, as highlighted before, is a process that requires knowledge and intelligence in order to extract deep-level information, which is beyond the explicit content in media. Deep-level information is obtained by a process of "reading between the lines" which is formally called in this work as Deep-Level Interpretation (DLI). A process that has been studied before in [HSME88] but that lacked automatic means to probe for coherence of its results. The work presented here contributes with practical means to achieve automated DLI.

As modern web-based interfaces become more and more popular even for classical application areas typical for ISs, users expect that the output of an information system is presented in a situation-specific way such that the output can be immediately used to access related material, which, in turn, opens a new focus to a broader information source. The design of human-computer interfaces that support this mode of interaction

[1] These means of content management are hereafter identified with the acronym EMSF (see Figure 1.2, page 4)

[2] A definition of the term *content semantics* will be provided in the next chapter, for now it is sufficient to read it as "meaning of the content" to understand this introduction.

requires new means of content management, i.e., annotations about content semantics in such a way that media (unstructured information) is enriched with structures that are machine-processable, and the support of Knowledge-Based Systems (KBSs) to exploit such annotations. With the support of KBSs, "intelligent" systems can be developed to support situation-specific interaction scenarios that are driven by the content semantics of the media being accessed. For example, in the medical scenario described above, an information system that is able to interpret media (e.g., texts about doctor's remarks) and exploit a KBS, can support a doctor in the process of diagnosis by considering not only structured but also unstructured information. This thesis describes a logic-based approach for the Deep-Level Interpretation (DLI) of text and emphasizes the relevance of DLI for content management and knowledge management. The interpretation of text is relevant given that natural language is highly expressive, therefore conveying large amounts of information compared to visual media, e.g., image and video. Moreover, immense quantities of text are constantly being produced.

In this work, DLI provides the foundation for automatic content annotation and automatic exploitation of annotations in web-based application interfaces for various application services, including *location-aware* and *situation-specific* navigation services. Given these innovative means of content management, one can support knowledge management in larger organizations. Knowledge management is facilitated by a systematic approach to knowledge modeling by directly using modeled knowledge for text interpretation. Thus, rather than having abstract knowledge modeling without a specific application in mind, we argue that knowledge modeling should be managed by grounding it in the direct use of modeled knowledge for automatic content annotation.

1.1 Motivation and Overview

The automation of media interpretation is still a challenging problem given that KBSs are required. Furthermore, the technologies that enable KBSs, i.e., formal and expressive means of knowledge representation and reasoning systems, have only become mature recently. The topic of this work is to solve the problem of DLI of text by using state-of-the-art technology for knowledge representation, more specifically, by exploiting Description Logics (DLs) to design a framework that allows not only logic-based interpretation but also to obtain interpretation results that can be proven for consistency against the background knowledge. Consistency checking is a process that is missing from relevant work [KKT92][HSME88] in the field of interpretation. Thus, for the DLI framework proposed here, knowledge representation and reasoning capabilities are required. To this end, a background knowledge base which is composed of a DL-based ontology and a set of rules

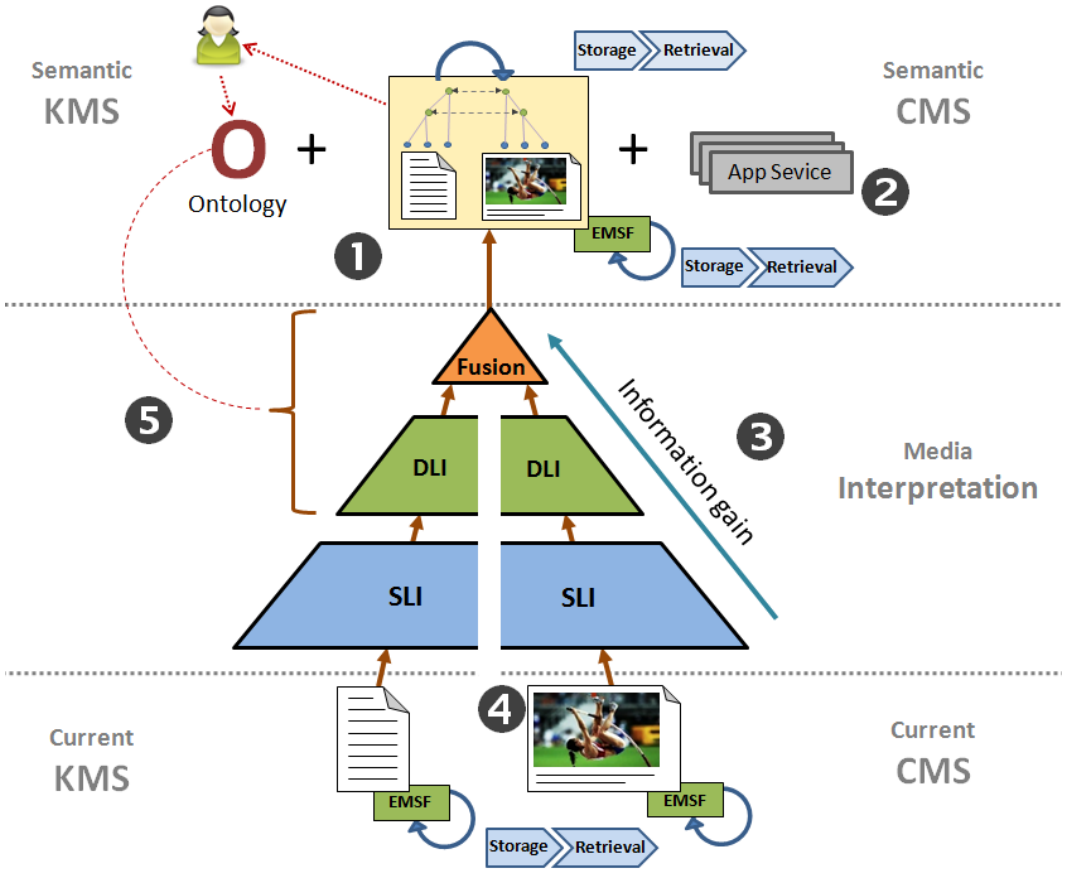


Figure 1.2: Framework for the generation of content descriptions based on media objects with Editorial information, Metadata, Structure and Format (EMSF).

has been designed. Moreover, the Description Logics (DLs) reasoner RacerPro [Rac07] is used in order to exploit the reasoning services that are necessary to execute the DLI process.

Using Figure 1.2[3] we highlight the contributions of this work to media interpretation, Content Management (CM) and Knowledge Management (KM). The design of DLI and fusion are relevant contributions of this work. DLI and fusion processes support the structuring of media through the extraction of so-called deep-level annotations (see no.1) by exploiting a domain ontology and a set of rules as background knowledge. This work distinguishes between surface-level and deep-level annotations. *Surface-level annotations* result from procedural media-dependent interpretation processes (see SLI in Figure 1.2), and can be associated to the corresponding segments of the media object. *Deep-level annotations* result from logic-based interpretation processes (DLI and fusion) and provide for more abstract information. A more detailed description of the difference between both processes is provided in Chapter 2. Hereafter, the term *annotation* refers to the union of surface-level and deep-level annotations. Annotations can be used as

[3]The pyramid's diagram was obtained from [EKM11] and modified for the presentation in this work.

references to media content. Provided a set of Aboxes with assertions that describe the content of corresponding media objects and a domain ontology, an ontology-based query language such as nRQL [HMW04] can be used by CMSs such that reasoning services can be exploited to retrieve media (for further details see Chapters 4 and 5). In this way, CMSs can use not only editorial information, metadata, segments and format (EMSF), but also automatically-generated annotations for the storage and retrieval of media based on content semantics.

In this work it is argued that in order to execute retrieval based on content semantics, reasoning services should be exploited. Thus, during query processing, not only the Abox structures, but also reasoning services w.r.t. a Tbox should be used. It is often the case in the research community that retrieval processes are characterized as being "semantics-based", solely because querying is done on top of assertions expressed with a description language (e.g., RDF) that has a formal semantics (see [Spi09][4]). But obviously, the semantics of the notions used in a query is better captured if reasoning w.r.t. to an appropriate Tbox is provided as well.

The manipulation of deep-level annotations by CMSs is beneficial for the management of application services (see Fig.1.2, no.2). Thus, the deep-level annotations of media can be used to dynamically determine which application services, also called *applicable services*, can be made accessible such that situation-specific scenarios can be supported. In this way, the functionality offered by CMSs (or any other application that manages media) dynamically adapts to the content semantics of the specific media being used. This kind of functionality characterizes what is called here a *semantics-driven application*. The semantics-driven behavior of an application is achieved in this work by describing the parameters of every available service through logic-based representations, which are later used for service discovery via reasoning (see Sections 4.1.3 and 5.3).

As the diagram for the media interpretation process illustrates (see Fig.1.2, no.3), three levels of information extraction are considered which help to compute representations in a multi-staged process. The first level is called Surface-Level Interpretation (SLI) which uses media-dependent techniques. Since the main focus of this work is text interpretation, Natural Language Processing (NLP) techniques are of interest, and in particular existing shallow-processing techniques [PKG+02, Gal08] are used in order to obtain the required input for the DLI process. The results of this level take the form of Abox assertions and are called surface-level annotations or *observations*. In the second level, DLI takes place, which uses the background knowledge and reasoning services on top of the observations to extract deep-level annotations. As illustrated in Figure 1.2, see no.4, SLI is media-dependent; therefore different techniques are required according to the types of media,

[4]http://videolectures.net/iswc09_spivack_ppap/

e.g., text, image, video, audio, etc. In contrast, DLI works on a logic level and therefore it has been used not only for text interpretation but also for image interpretation. The third level of extraction is called *fusion*; here the annotations extracted from different types of media are fused with the use of *same-as* assertions to unify different individuals. As will be described in Section 2.6, the deep-level annotations obtained from DLI provide for the necessary level of abstraction, such that observations of a different nature (extracted from different media) are put into a common context to allow for fusion. Finally, the design patterns described in this work allow the knowledge engineer to create domain ontologies and rules (see Fig.1.2, no.5) that are applicable not only for supporting deep-level interpretation, but are also useful in other applications (see Chapter 4) that manage media content. Here the thesis supports that an ontology useful for media interpretation, is also useful for other applications that deal with media. Moreover, interpretation results can be used as feedback on missing background knowledge. In this way, the engineer is able to identify the knowledge that should be defined. From this perspective, the knowledge engineer is *guided* through the task of designing an ontology. This process is called *grounded ontology design*. Grounded design in the context of media interpretation means that the choice of terminological elements are influenced by the content semantics of media.

1.2 Contributions

As the previous sections have already introduced, the present work contributes to the areas of text understanding, content management and knowledge management as follows. With respect to text interpretation, a solution to this problem is provided:

- by designing a process of abductive reasoning [PKM$^+$07a, PKM09b];
- by designing an interpretation process [PKM$^+$07b] that uses DLs as knowledge representation formalism and exploits abduction and other reasoning services to extract deep-level information (annotations) and to ensure consistency w.r.t. the domain of interest;
- and by presenting in [PKMM08] an approach that combines state-of-the-art shallow-processing techniques to extract surface-level annotations from large-scale corpora, with logic-based techniques to provide for deep-level interpretation.

The work presented in [PKM09a] contributes to CM by:

- creating the media annotations that are required for CMSs to have access to content semantics and support retrieval processes;

- showing that an ontology used to support the process of DLI is appropriate to support other applications that manage media;
- demonstrating that not only media can be managed but also application services. By using annotations and reasoning services, a system can dynamically activate so-called applicable services according to the content semantics of the media in use.

The contributions to KM are:

- A set of design patterns for the engineering of background knowledge.
- A new perspective of knowledge engineering called grounded ontology design is presented in this work and shortly described in [CEF+09]. In this perspective the knowledge engineer is guided through the design task by using interpretation results as feedback about missing background knowledge. Thus, if no interpretations of a certain part of a document can be provided, hints to the knowledge engineer can be automatically generated such that one becomes aware of gaps in knowledge representation. In this way, this work shows that DLI is useful to manage the knowledge of the engineer while performing ontology design.
- The DLI of text contributes to knowledge management by providing support to software engineers in the automatic identification of object models for evolving applications. In this way, the knowledge required for the design of an object model is managed by the deep-level interpretation of text documents.

1.3 List of Publications

Book Chapters

S. Espinosa, A. Kaya, R. Möller. *Logical Formalization of Multimedia Interpretation.* In G. Paliouras, C. D. Spyropoulos, G. Tsatsaronis, editors, Knowledge-Driven Multimedia Information Extraction and Ontology Evolution 2011, pages 110-133, Volume 6050. Springer LNAI Series.

Journals

S. Castano, S. Espinosa, A. Ferrara, V. Karkaletsis, A. Kaya, R. Möller, S. Montanelli, G. Petasis, and M. Wessel. *Multimedia Interpretation for Dynamic Ontology Evolution.* Journal of Logic and Computation 2009, 19(5), pages 859–897. Oxford University Press.

Conferences

S. Espinosa Peraldi, A. Kaya, S. Melzer, R. Möller. *On Ontology Based Abduction for Text Interpretation.* In A. Gelbukh, editor, In Proc. of CICLing-2008, Volume 4919. Springer LNCS Series, pages 194-205. February 2008. Haifa, Israel.

S. Espinosa Peraldi, A. Kaya, S. Melzer, R. Möller, M. Wessel. *Towards a Media Interpretation Framework for the Semantic Web.* In Proc. of WI'07, number 1331876, pages 374–380. IEEE Computer Society, 2007. October 2007. Silicon Valley, USA.

Workshops

S. Espinosa Peraldi, A. Kaya, Möller. *Formalizing Multimedia Interpretation based on Abduction over Description Logic Aboxes.* In Proc. of DL2009, July 2009, Oxford, UK.

S. Espinosa, A. Kaya, R. Möller. *The BOEMIE Semantic Browser: A Semantic Application Exploiting Rich Semantic Metadata.* In Proc. of AST-2009, October 2009. Lübeck, Germany.

S. Espinosa, A. Kaya, R. Möller. *Ontology and Rule Design Patterns for Multimedia Interpretation.* In Proc. of the 2nd BOEMIE Workshop, December 2008. Koblenz, Germany.

S. Espinosa, A. Kaya, S. Melzer, R. Möller, M. Wessel. *Multimedia Interpretation as Abduction.* In Proc. of DL-2007, June 2007. Brixen, Italy.

S. Castano, S. Espinosa, A. Ferrara, V. Karkaletsis, A. Kaya, S. Melzer, R.Möller, S. Montanelli, G. Petasis. *Ontology Dynamics with Multimedia Information: The BOEMIE Evolution Methodology.* In Proc. of IWOD, June 2007. Innsbruck, Austria.

1.4 Outline

As a knowledge-based system, DLI of text requires reasoning capabilities and a formal language to represent background knowledge. Chapter 2 provides a section of preliminaries to describe; the syntax and semantics of a DL language formalism used in this work for the construction of ontologies, together with reasoning services useful for DLI. In addition, a DL-based ontology used as a running example, and a system for shallow-processing of text is also briefly described in order to demonstrate that current NLP tools can provide the input that DLI requires. After the preliminaries, we describe the term content semantics for media objects and analyze how it can be represented through content descriptions. We also describe the DLI process as an abduction-based process by providing a step-by-step example for the interpretation of a text paragraph. Finally, the process of multimedia fusion is presented. Chapter 3 focuses on the creation of a useful background knowledge. The concept of grounded knowledge design and a set of design patterns are developed. Chapter 4 focuses on the contributions to the area of CM. Situation-specific and location-aware services for semantics-driven applications are introduced and Chapter 5 describes a generic architecture useful to develop semantics-driven applications. The purpose is to show that the process of DLI and its contributions to CM can be realized. Insights gained as a result of this research are brought together in Chapter 6 and future work is presented in Chapter 6 as well.

Chapter 2

Deep-Level Interpretation of Text

It has been estimated in [Hat07] that unstructured content (hereafter called *media*), such as text documents, audio, images, etc., reaches eighty percent of the total amount of data produced in companies. Most of this information is found in text documents. The rather large percentage of media information brings to view the fact that media, and more specifically text documents, are important assets of organizations. Therefore, integrating media as a constituent element of ISs in organizations has become an important research goal.

An IS typically has three main components: data sources, application programs and user interfaces. The integration of media into ISs affects all of these components. The most-obvious integration occurs in connection with the data sources, from which media can be retrieved and, finally, rendered in a user interface. The integration of media into user interfaces of ISs has been facilitated due to the advances of web technology [CV01]. Given these types of interfaces, integrating structured and unstructured information, users expect that classical GUI-based interaction scenarios in application programs are seamlessly extended with media shown in situation-specific ways. Graphical elements of application GUIs are associated with commands, which, in turn, can be used to execute application services. One can easily imagine a specific application service, such as, "find related information". This service can be realized by presenting media objects found, for instance, within data sources and/or web resources. Naturally, users expect that presented media can also be associated with services. This tight mode of interaction between media and application GUIs has not been achieved yet. In this work, the fundamentals for realizing this style of interaction are defined.

The association of services with parts of presented media objects must be based on the semantics of the media content. Media are currently managed as unstructured data, possibly augmented with metadata, and, in general, these metadata do not capture the *content semantics*. Thus, we need to automatically extend media with content descriptions

representing content semantics (informally referred to as "meaning of the content" for now). This chapter describes a process of text interpretation which aims at the automatic extraction of content descriptions that can be associated to text documents.

In order to formally define the notion of content semantics, we introduce description logics (DLs). Using the model theory introduced with DLs in Section 2.1, we are able to formally define the notion of content semantics in Section 2.2. Content semantics are represented through content descriptions attached to media as annotations. Based on the definition of content semantics, Section 2.3 describes content-based services by providing a specific example, namely, content retrieval. With the retrieval example we highlight the need of content descriptions associated to media. Content descriptions are derived in a process that we call Deep-Level Interpretation (DLI). Before describing DLI in Section 2.5, the requirements of DLI are described in Section 2.4. Section 2.6 describes how DLI can also be applied to image content, and how explanations support the logic-based fusion of multimedia. Finally, Section 2.7 concludes this chapter.

2.1 Description Logics as a Representation Language

Description Logics (DLs) are of interest in this work because they provide the syntax and semantics that content descriptions require in order to represent content semantics. Ontologies can be built based on DLs; they provide the vocabulary that is used by content descriptions for a specific domain. This section describes a DL language and the inference services, supported by DLs, that are relevant to this work. Provided the description of DLs, we are able to describe a specific domain ontology that will be used as a running example throughout this work.

Research in DLs has a long tradition within the AI community in the study of formal logic-based semantics, more specifically in the study of subsets of First-Order Logic (FOL). DLs are a family of highly expressive language formalisms, where a language is generally chosen as to guarantee the decidability of decision problems (described later in Section 2.1.1). Algorithms for decision problems have been implemented as inference services by various DL systems such as RacerPro [HM01]. In this work the term "semantics" refers to DL-based semantics and is described as follows.

Semantics of first-order languages such as DLs are based on the notions of set-theoretic interpretations, $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, in short called *interpretations*. The interpretation function $\cdot^{\mathcal{I}}$ (see Figure 2.1) assigns to every concept A a set of objects from the domain ($A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$) and to every role R a set of tuples of objects from the domain ($R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$). A concept is therefore interpreted as a set of objects, and roles are interpreted as sets of

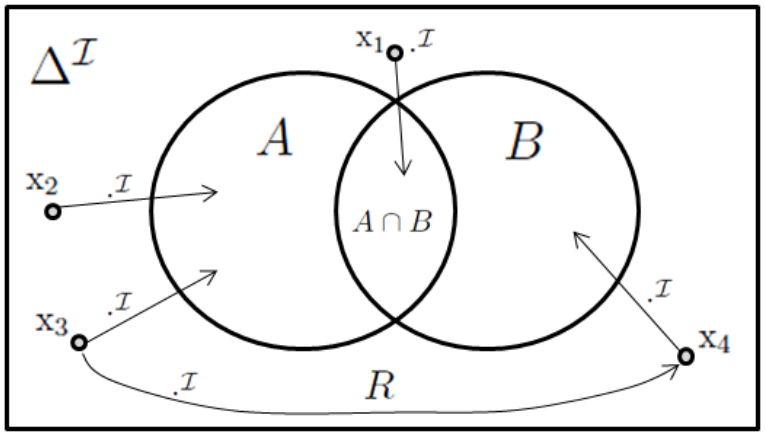


Figure 2.1: DLs define semantics through set-theoretic interpretations.

pairs of objects. A specific interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ *satisfies* a concept description A iff $A^{\mathcal{I}} \neq \emptyset$. In this case, $\mathcal{I}$ is called a *model* for A.

Syntax and semantics of $\mathcal{ALCHIQ}$

For a given application context, the design of an ontology starts by choosing a set of elementary descriptions (or *atomic descriptions*) which are representative for the domain of interest. The elementary descriptions are grouped in the so-called *signature*. A signature $\mathcal{S}$ is a triple composed of the following disjoint sets: the set of Concept Names (CN), also called *atomic concept descriptions*, the set of Role Names (RN), also called *atomic role descriptions* and the set of Individual Names (IN) used to name specific objects of the domain. For example, for the *Athletics* domain, the following signature can be used:

$$\begin{aligned}(\text{CN, RN, IN})_{Athletics} := (&\{SportsTrial,\ HighJumpTrial,\ SportsRound\},\\ &\{hasParticipant,\ hasRoundName,\ hasSportsName\},\\ &\{highjump_1,\ firstround_1\})\end{aligned}$$

In this work, the application scenarios that exploit content descriptions do not require reasoning on concrete domains, more specifically, on strings. For this reason, we intentionally do not introduce the syntax and semantics of concrete domains. Therefore, assertions, such as (date_1, "13 August 2002"):hasValue, are read as a tuple of individual names related within a role, such as *hasValue*.

The next step is to determine the expressivity requirements of the language. This can be achieved by analyzing requirements on complex concepts and role terms. Complex concepts are built with the use of CN and RN names as well as operators whose meaning is defined in terms of a set-theoretic semantics. For example, the language $\mathcal{ALCQHI}$, has an expressivity specified with a syntax which follows the grammar of Figure 2.2. In the grammar, the letter A is used to refer to atomic concept descriptions and the letters R and

S refer to atomic role descriptions. Descriptions for complex concepts can be inductively built.

$$
\begin{array}{rcll}
C, D & \longrightarrow & A & \mid \text{atomic concept description} \\
 & & C \sqcap D & \mid \text{conjunction} \\
 & & C \sqcup D & \mid \text{disjunction} \\
 & & \neg C & \mid \text{negation} \\
 & & \exists R.C & \mid \text{existential restriction} \\
 & & \forall R.C & \mid \text{value restriction} \\
 & & \exists_{\geq n} R.C & \mid \text{qualified minimum restriction} \\
 & & \exists_{\leq n} R.C & \mid \text{qualified maximum restriction}
\end{array}
$$

Figure 2.2: The grammar of $\mathcal{ALCQHI}$.

The concept descriptions $\top$ and $\bot$ are defined to be abbreviations for $A \sqcup \neg A$ and $A \sqcap \neg A$, respectively. Concept descriptions may be written in parentheses in order to avoid scoping ambiguities. With this expressivity it is possible to build complex descriptions such as

$$HighJumpTrial \sqcap \exists_{\leq 1} hasParticipant.Athlete$$

whose intended meaning is, informally speaking, the set of domain objects of the concept *HighJumpTrial* that are related with at most one object, found in the set of objects of the concept *Athlete* through a role *hasParticipant*. For complex concept descriptions the interpretation function is extended as shown in Figure 2.3, where $\sharp$ is used to denote the cardinality of a set.

A *Tbox* is a finite set of axioms called *inclusions* of the form $C \sqsubseteq D$ ($R \sqsubseteq S$) and *equalities* of the form $C \equiv D$ ($R \equiv S$), where C, D are concepts and R, S are roles. In concepts, inverse roles R^- (or S^-) may be used instead of role names R (or S) and $(R^-)^{\mathcal{I}} := \{(x, y) \mid (y, x) \in R^{\mathcal{I}}\}$. Inclusion axioms for concepts are called Generalized Concept Inclusions (GCIs). GCIs play an important role in this work to represent so called aggregates, which will be described later in Section 2.5.1. GCIs can be divided into two parts, i.e., the left-hand side and the right-hand side of the symbol $\sqsubseteq$. It is possible to use an atomic description on each side (e.g., $HighJumpTrial \sqsubseteq SportsTrial$) or to use a complex description on the right-hand side. For example, the following GCI defines a specialization of $SportsTrial$ called $PoleVault$ for the complex concept description on the right-hand side of $\sqsubseteq$.

$$
\begin{array}{rcl}
PoleVault & \sqsubseteq & SportsTrial \sqcap \forall hasParticipant.PoleVaulter \\
 & & \sqcap \exists_{\leq 1} hasPart.Pole \\
 & & \sqcap \exists_{\leq 1} hasPart.HorizontalBar
\end{array}
$$

GCIs specify so called *necessary conditions*, which are conditions that are minimum requirements but still not sufficient to exhaustively define a concept. Equalities specify so called *necessary and sufficient conditions* ($\equiv$), which indicate that a concept is defined exhaustively. For this reason they are called *definitions* (only if the GCIs are non-cyclic).

$$\begin{aligned}
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}} \\
(\exists R.C)^{\mathcal{I}} &= \{x \mid \exists y.(x,y) \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\} \\
(\forall R.C)^{\mathcal{I}} &= \{x \mid \forall y. \text{ if } (x,y) \in R^{\mathcal{I}} \text{ then } y \in C^{\mathcal{I}}\} \\
(\exists_{\leq n} R.C)^{\mathcal{I}} &= \{x \mid \sharp\{y \mid (x,y) \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\} \leq n\} \\
(\exists_{\geq n} R.C)^{\mathcal{I}} &= \{x \mid \sharp\{y \mid (x,y) \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\} \geq n\}
\end{aligned}$$

Figure 2.3: The semantics of $\mathcal{ALCQHI}$.

An interpretation $\mathcal{I}$ *satisfies* a GCI $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. An interpretation is a *model* of a Tbox if it satisfies all GCIs in the Tbox. A concept description C *is subsumed by* a concept description D w.r.t. a Tbox if the GCI $C \sqsubseteq D$ is satisfied in all models of the Tbox. In this case, it can also be said that D *subsumes* C.

An *Abox* is a set of *assertions* of the form $i : C$, $(i, j) : R$, or (i, j) :same-as. Let $i,\ j \in$ IN. A concept assertion, $i : C$, is satisfied by an interpretation if $i^{\mathcal{I}} \in C^{\mathcal{I}}$. An individual i is an *instance* of a concept C w.r.t. a Tbox and an Abox if for all interpretations $\mathcal{I}$ of the Tbox and Abox $i^{\mathcal{I}} \in C^{\mathcal{I}}$. A role assertion, $(i, j) : R$, is satisfied by an interpretation $\mathcal{I}$ if $(i^{\mathcal{I}}, j^{\mathcal{I}}) \in R^{\mathcal{I}}$. A same-as assertion, (i, j):same-as, is satisfied by an interpretation $\mathcal{I}$ if $i^{\mathcal{I}} = j^{\mathcal{I}}$ and (i, j) : $\neg$same-as is satisfied by an interpretation $\mathcal{I}$ if $i^{\mathcal{I}} \neq j^{\mathcal{I}}$. An interpretation satisfying all assertions in an Abox $\mathcal{A}$ is called a model for $\mathcal{A}$. An Abox $\mathcal{A}$ is called *consistent* if such a model exists, it is called *inconsistent* otherwise.

An *ontology* $\mathcal{O}$ is a triple $(\mathcal{S}, \mathcal{T}, \mathcal{A})$ composed of a Signature, a Tbox, and an Abox. Let α be concept or role assertion. $\mathcal{O} \models \alpha$ if for all models $\mathcal{I}$ of $\mathcal{O}$ it holds that $\mathcal{I}$ satisfies α. $\mathcal{O} \models \mathcal{A}'$ if for all $\alpha \in \mathcal{A}'$ it holds that $\mathcal{O} \models \alpha$. We write $\mathcal{O} \cup A'$ to mean $(\mathcal{S}, \mathcal{T}, \mathcal{A} \cup A')$

In the following section a description is presented about decision problems which represent standard reasoning services that are useful to this work.

2.1.1 Decision Problems and their Reductions

As will be described in Section 2.5.1 and 2.5 the following decision problems play a central role in the DLI process.

- Concept satisfiability
- Tbox satisfiability
- Concept subsumption
- Abox consistency
- Instance test
- Instance retrieval

The *concept satisfiability* problem is to check whether a model for a concept description exists. The *Tbox satisfiability* problem is to determine whether a model for the Tbox exists. The *concept subsumption* problem is to check whether $C \sqsubseteq D$ holds in all models of the Tbox. The *Abox consistency problem* for an Abox $\mathcal{A}$ w.r.t. a Tbox is the problem of determining whether there exists a model of $\mathcal{A}$ that is also a model of the respective Tbox. The *instance test* problem is to probe whether an individual i is an instance of a concept description C w.r.t. a Tbox and an Abox. The *instance retrieval* problem w.r.t. a concept description C is to find all individuals i mentioned in the assertions of an Abox such that i is an instance of C. For roles and pairs of individuals, similar definitions can be given. In order to solve the instance problem for an individual i and a concept description C w.r.t. an Abox $\mathcal{A}$ one can check if the Abox $\mathcal{A} \cup \{i : (\neg C)\}$ is inconsistent [BN03]. Furthermore, the satisfiability problem for a concept description C can be reduced to the consistency problem for the Abox $\{i : C\}$. In theory, all problems introduced above can be reduced to the Abox consistency problem. In practical systems, e.g. RacerPro, specific optimization techniques are used to solve a certain decision problem.

In addition to the basic retrieval inference services, expressive query languages are required in practical applications such as the ones described later in Section 4. For this reason, in the following sections we will describe retrieval inference services and a specific type of queries called grounded conjunctive queries.

2.1.2 Retrieval Inference Services

Before describing retrieval inference services is necessary to define the concepts of *sequences* and *substitutions*.

Let $\underline{X}$, be a sequence of variables, $\underline{Y_1}, \ldots, \underline{Y_n}$ be sequences of variables and individuals. $\underline{z}$ denotes a sequence of individuals. For this work, sequences of length 1 $\langle X \rangle$ or 2 $\langle X, Y \rangle$ are considered. Furthermore, X and Y are variables. The function *getinds* retrieves a set of individuals from an Abox.

A *substitution* $\sigma = [X \leftarrow i, Y \leftarrow j, i \leftarrow i]$ is defined as a function that maps individuals to variables and individuals to itself.

The application of a variable substitution σ to a sequence of variables $\langle X \rangle$ or $\langle X, Y \rangle$ is defined as $\langle \sigma(X) \rangle$ or $\langle \sigma(X), \sigma(Y) \rangle$, respectively, with $\sigma(X) = i$ and $\sigma(Y) = j$. In this case, a sequence of individuals is defined. If a substitution is applied to a variable X for which there exists no mapping $X \leftarrow k$ in σ then the result is undefined. A variable substitution for which all required mappings are defined is called *admissible*.

The *retrieval* inference problem w.r.t. a Tbox $\mathcal{T}$ is defined as *instance retrieval* (see page 16), such that it allows the retrieval of all instances for a concept or a role name in the Tbox. In practical applications, more complex retrieval scenarios are required that demand expressive query languages, in this context, conjunctive queries are well-established.

2.1.3 Standard and Grounded Conjunctive Queries

A *conjunctive query* consists of a *head* and a *body*. The head contains variables called *distinguished variables*, for which the user would like to compute bindings. The body consists of query atoms in which all variables from the head must be mentioned. If the body contains additional variables, they are seen as existentially quantified and are called *non-distinguished*. Let $Q_1, \ldots, Q_n$ denote atomic concept or atomic role descriptions from a Tbox $\mathcal{T}$. A query is defined by the following syntax

$$\{(\underline{X}) \mid Q_1(\underline{Y_1}), \ldots, Q_n(\underline{Y_n})\}$$

The left hand side of the sign | represents the head and the right hand side denotes the body. The sequence $\underline{X}$ contains variables that must also appear in the body $as\_set(\underline{X}) \subseteq as\_set(\underline{Y_1}) \cup \ldots \cup as\_set(\underline{Y_n})$. Informally speaking, $Q_1(\underline{Y_1}), \ldots, Q_n(\underline{Y_n})$ defines a conjunction of so-called *query atoms* $Q_i(\underline{Y_i})$.

There are different types of query atoms. *Concept* query atoms have a sequence of variables $\underline{Y_n}$ of length one and are read as $C(X)$. *Role* query atoms have a sequence on variables $\underline{Y_n}$ of length two and are read as $R(X, Y)$. *Same-as* query atoms are written as $= (X, Y)$, sometimes abbreviated as $X = Y$. Complex queries are built from query atoms using boolean constructs for conjunction (indicated with comma) or union ($\vee$). Parentheses may be used to indicate the intended scope.

In the literature (e.g. [HSTT00, GHLS07, WM06]), two different semantics for conjunctive queries are discussed, called *standard* and *grounded*. In *standard* conjunctive queries, non-distinguished variables are bound to domain objects and do not have to be bound to individuals (named domain objects). A system supporting (unions of) standard conjunctive queries is QuOnto [ACG+05].

In so-called *grounded* conjunctive queries, non-distinguished variables are bound to named domain objects, i.e., answering a query with respect to an ontology $\mathcal{O}$ means finding admissible variable substitutions σ such that $\mathcal{O} \models \{(\sigma(\underline{Y_1})) : Q_1, \ldots, (\sigma(\underline{Y_n})) : Q_n\}$. Given all possible variable substitutions σ, the *result* of a query is defined as

$$\{\sigma(\underline{X}) \mid \mathcal{O} \models \{(\sigma(\underline{Y_1})) : Q_1, \ldots, (\sigma(\underline{Y_n})) : Q_n\}\}$$

A variable substitution is said to provide bindings for the head variables of a query. Note that the variable substitution σ is applied before checking whether $\mathcal{O}$ entails the substitutions σ, i.e., the query is *grounded* first. For example, for the following query and Abox:

Query:
$\{(X) \mid SportsTrial(Y),\ hasParticipant(Y, X),\ Person(X)\}$
Abox:
$\{ind_1 : SportsTrial,\ ind_2 : Person,\ (ind_1,\ ind_2) : hasParticipant\}$
Result: $\{ind_2\}$

the substitution $[X \leftarrow ind_2, Y \leftarrow ind_1]$ allows for answering the query, and defines bindings for X.
A *boolean* query is a query with an empty head as follows

$$\{() \mid Q_1(\underline{Y_1}), \ldots, Q_n(\underline{Y_n})\}$$

If for a boolean query there exists a variable substitution σ such that $\mathcal{O} \models \{(\sigma(\underline{Y_1})) : Q_1, \ldots, (\sigma(\underline{Y_n})) : Q_n\}$ holds, then the query is answered with *true*, otherwise the answer is *false*.

2.1.4 Rules

Conjunctive queries can also be written as non-recursive rules as a mean to name subqueries for later reuse. A rule r has the following form

$$P(\underline{X}) \leftarrow Q_1(\underline{Y_1}), \ldots, Q_n(\underline{Y_n})$$

The predicate symbols used in the atoms of a rule should use concept or role names from the Tbox $\mathcal{T}$ of $\mathcal{O}$. Similar to conjunctive queries the restriction $as\_set(\underline{X}) \subseteq as\_set(\underline{Y_1}) \cup \ldots \cup as\_set(\underline{Y_n})$ holds. Rules are used to derive new Abox assertions once they are applied to an Abox $\mathcal{A}$.

The function $apply(\mathcal{O}, r, \mathcal{A})$ returns a set of Abox assertions $\{(\sigma(\underline{X})) : P\}$ for all admissible variable substitutions σ such that the answer to the boolean query

$$\{() \mid Q_1(\underline{Y_1}), \ldots, Q_n(\underline{Y_n})\}$$

is *true* with respect to $\mathcal{O} \cup \mathcal{A}$. If no such σ can be found, the result of the call to $apply(\mathcal{O}, r, \mathcal{A})$ is the empty set. Thus, *apply* processes rules in a forward way, following the implication sign ($\leftarrow$) from right to left, i.e., from the body (antecedent) to the head (consequent). The application of a set of rules $\mathcal{R} = \{r_1, \ldots r_n\}$ to an Abox is defined as follows.

$$apply(\mathcal{O}, \mathcal{R}, \mathcal{A}) = \bigcup_{r \in \mathcal{R}} apply(\mathcal{O}, r, \mathcal{A})$$

Forward chaining is an inference method in which a set of rules is applied starting from antecedents to consequents, until a goal is reached or the Abox is saturated, i.e., there are no new Abox assertions that can be obtained. In order to guarantee termination, the restriction $as\_set(\underline{X}) \subseteq as\_set(\underline{Y_1}) \cup \ldots \cup as\_set(\underline{Y_n})$ should hold, i.e, each variable that appears in the head of a rule must also appear in the body of the same rule. In this way, the result of $forward\_chaining(\mathcal{O}, \mathcal{R}, \mathcal{A})$ is $\emptyset$ if $apply(\mathcal{O}, \mathcal{R}, \mathcal{A}) \cup \mathcal{A} = \mathcal{A}$ and $apply(\mathcal{O}, \mathcal{R}, \mathcal{A}) \cup forward\_chaining(\mathcal{O}, \mathcal{R}, \mathcal{A} \cup apply(\mathcal{O}, \mathcal{R}, \mathcal{A}))$ otherwise.

2.1.5 Knowledge Bases

A *knowledge base* $\Sigma = (\mathcal{O}, \mathcal{R}, A_x)$ is composed of an ontology $\mathcal{O}$, a set of rules $\mathcal{R}$ and an Abox A_x. We write $\Sigma \cup A'$ to means $(\mathcal{O}, \mathcal{R}, A_x \cup A')$. The Abox part of an ontology $\mathcal{A}$ of $\mathcal{O}$ contains strategically relevant individuals in the domain, and therefore, are sharable. $\mathcal{A}$ is disjoint from the Abox A_x. The Abox A_x contains the interpretation results of a specific media object, therefore the assertions in A_x are application dependent and not sharable. Differencing between a sharable Abox and Aboxes from a knowledge base is relevant to support KM (see Section 3). The set of rules $\mathcal{R}$ are used to define the space of possible interpretations used by Abox abduction as described in Section 2.5.3. We say that a knowledge base Σ entails an Abox assertion α and write $\Sigma \models \alpha$ to mean $(\mathcal{S}, \mathcal{T}, \mathcal{R}, \mathcal{A} \cup A_x) \models \alpha$ if

$$(\mathcal{S}, \mathcal{T}, \mathcal{A} \cup A_x \cup forward\_chaining(\mathcal{S}, \mathcal{T}, \mathcal{R}, \mathcal{A} \cup A_x)) \models \alpha$$

2.2 Content Semantics and Content Descriptions

After providing a formal description of DL-based semantics and inference services, we are now able to define the term content semantics as follows.

Definition. Content Semantics *Let $\mathcal{O}$ be an ontology, A_x an Abox, and d_x a text document. If A_x is associated to d_x, the semantics of the document d_x is defined as follows*

$$Sem(d_x, A_x) = \{\mathcal{I} \mid \mathcal{I} \models \mathcal{O},\ \mathcal{I} \models A_x\}$$

where Sem is a function that associates the semantics of A_x to the document d_x.

In this way, the semantics of media content is defined by the set of interpretations that satisfy all assertion of A_x w.r.t. to an ontology $\mathcal{O}$. In other words, the possible models of A_x represent the semantics of the document d_x.

To clarify this definition, consider the following example. The domain is an infinite set of domain objects $\Delta^{\mathcal{I}} = \{x_1, x_2, x_3, \ldots\}$, $\mathcal{T}$ and A_x contain the following axioms and assertions, respectively.

$$\mathcal{T} = \left\{ \begin{array}{rcl} CountryName & \equiv & Russia \sqcup Finland \\ Russia & \sqsubseteq & \neg Finland \end{array} \right\}$$

$$A_x = \{c_1 : CountryName,\ p_1 : Person,\ (p_1, c_1) : hasNationality\}$$

Possible models of A_x are:

$c_1^{\mathcal{I}} = \{x_1\}$	$CountryName^{\mathcal{I}} = \{x_1\}$
$p_1^{\mathcal{I}} = \{x_2\}$	$Person^{\mathcal{I}} = \{x_2\}$
	$Russia^{\mathcal{I}} = \{x_1\}$
	$Finland^{\mathcal{I}} = \{\}$
	$hasNationality^{\mathcal{I}} = \{(x_2, x_1)\}$

$c_1^{\mathcal{I}} = \{x_1\}$	$CountryName^{\mathcal{I}} = \{x_1\}$
$p_1^{\mathcal{I}} = \{x_2\}$	$Person^{\mathcal{I}} = \{x_2\}$
	$Russia^{\mathcal{I}} = \{\}$
	$Finland^{\mathcal{I}} = \{x_1\}$
	$hasNationality^{\mathcal{I}} = \{(x_2, x_1)\}$

If the axiom $c_1 : \neg Russia$ is added to the Abox above, then the model on the left-hand side (see above) is not a model anymore. In this way, provided the syntax and semantics of DLs, the axioms of the Tbox and an Abox help to restrict the set of possible models of a domain. By associating individuals of the Abox to specific segments of a text document (see Figure 2.4), it is possible to manage media on the basis of content descriptions, (also called Aboxes) with formal semantics.

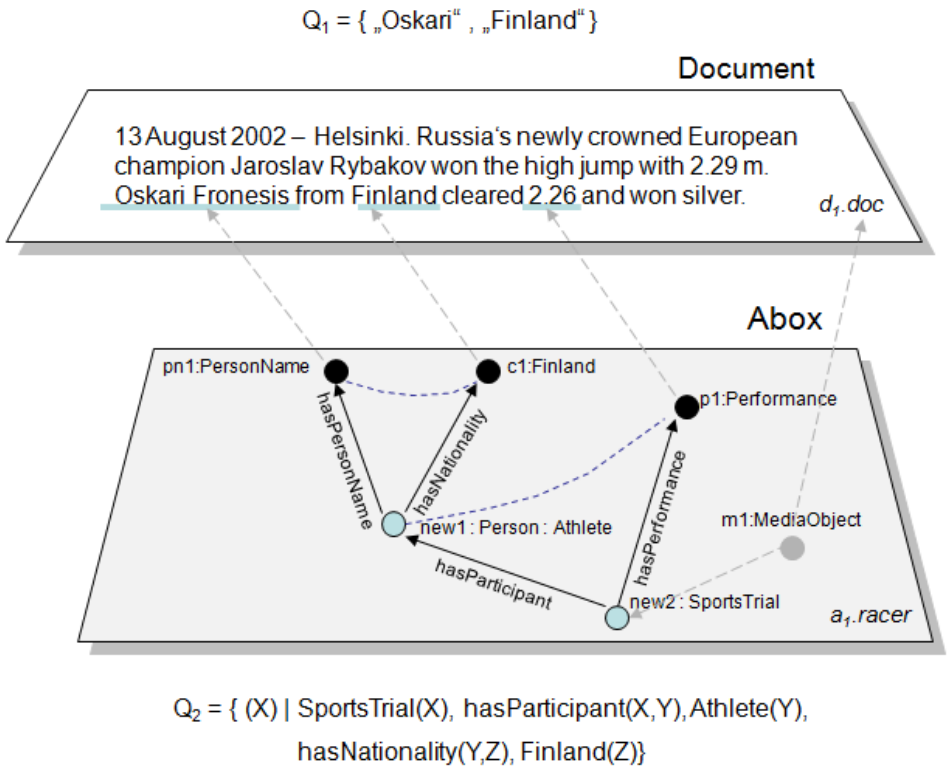


Figure 2.4: Retrieval of text documents

2.3 Content-Based Services using Content Descriptions

As previously mentioned in this chapter, it is desirable to create application systems with GUIs that provide a tight mode of interaction combining standard interface elements and media. To achieve this, the application services — associated to the GUI elements — should have access to the content descriptions of media (Section 4.1.3 proposes a way to achieve this). The access to content descriptions by application services allows for the construction of a system that provides content-based services. Many of such services can be implemented through media retrieval. For example, a service that suggests related information called “Read related news articles”. In this section we describe the use of content descriptions to support media retrieval.

String matching algorithms are commonly used for the retrieval of text documents — with obvious limitations. Instead, a process of media retrieval is desirable that uses queries based on content descriptions that represent content semantics. Different from classical IR processes [MRS08], the retrieval process exemplified here is based on content semantics represented by content descriptions. Content descriptions are advantageous since they use a vocabulary (defined through the signature of an ontology) composed of abstract terms. For example, consider a user interested in retrieving text documents of the athletics

$$Query_x := \{(\underline{X}) \mid Q_1(\underline{Y_1}), \ldots, Q_n(\underline{Y_n})\}$$

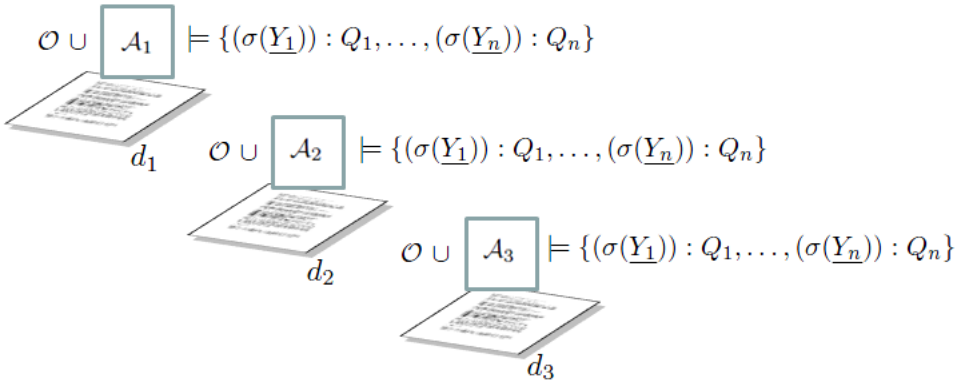


Figure 2.5: Semantics-based content retrieval

domain. Figure 2.4 shows two queries. Q_1 exemplifies a query to be answered by string matching algorithms. Given the poor expressivity of Q_1, less restrictions are imposed during the search process. For this reason, documents of various domains are retrieved. Q_2 is a grounded conjunctive query (see definition in Section 2.1.3). It imposes specific restrictions about the domain of interest by using abstract terms, e.g., "SportsTrial" and "Athlete", and specific relations between terms, e.g., "hasParticipant", "hasNationality". Note that abstract terms used in the query are not always found explicitly in the text content (see the text in Figure 2.4), and therefore they are inaccessible to string matching algorithms. For Q_2 to be executed, an ontology $\mathcal{O}$ is required and an Abox which contains descriptions for the text document "d1.doc". Figure 2.4 (see page 21) shows a graph representing the Abox content. By executing Q_2, a variable substitution for X, $(\sigma(X)) := new_2$ is returned. Provided information about segment offsets, as well as a reference to the document id (see m_1), a function δ can be used to retrieve the documents and specific segments of text (in Section 2.4.2 a technology is described which allows to relate text segments with individuals in an Abox).

In this way, provided a knowledge base containing an Abox A_x for each text document d_x (see Figure 2.5) and an ontology $\mathcal{O}$, queries that use abstract terms can be executed to compute substitutions $\{\sigma(\underline{X})\}$ that satisfy $\mathcal{O} \cup A_x \models \{(\sigma(\underline{Y_1})) : Q_1, \ldots, (\sigma(\underline{Y_n})) : Q_n\}$. Finally, the resulting variable substitutions are used by a function δ to obtain related documents d_x. For $\mathcal{O} = (\mathcal{S}, \mathcal{T}, \mathcal{A})$ and an Abox A_x we define $\mathcal{O} \cup A_x$ as $(\mathcal{S}, \mathcal{T}, \mathcal{A} \cup A_x)$ Given the definition of content semantics provided in the previous section and having described the usefulness of content descriptions to support content-based services, the following thesis is underlying this work.

> **Thesis:** *Aboxes associated to media objects are the optimal means for the representation of content semantics. Aboxes allow CM and KM to provide content-based services on the basis of content semantics.*

The specific architecture presented in Chapter 5 shows that the definition of content semantics as presented in Section 2.2 is useful in practice and helps to support this thesis. Certainly a process that allows the automatic extraction of content descriptions in the form of Aboxes is required. One of the contributions of this work is the design of an interpretation process called Deep-Level Interpretation (DLI), that aims at extracting content descriptions from media. The following sections describe the DLI process in detail.

2.4 Interpretation for Deriving Content Descriptions

With respect to the vast amounts of media that an organization already owns, and the amounts of documents constantly being produced with no annotations to fulfill the requirements of content management, it is impractical to create annotations manually. Instead, a process of media interpretation should be created that automatically extracts content descriptions, which can be used as annotations for media content.

One of the application areas of the research in Natural Language Processing (NLP) is the extraction of structured information. NLP studies techniques which can be (broadly) divided into shallow NLP and deep NLP. They are distinguished by the level of deepness in considering linguistic phenomena with repercussion in performance [All95]. Performance, in terms of efficiency and robustness, turns relevant when dealing with large-scale corpora. Performance has been the main incentive in the proliferation of shallow processing techniques, which are less time consuming than deep NLP. Shallow NLP relies on machine learning techniques, e.g., [BMSW97] and [Sod97] or pattern-based grammar approaches. But, the tendency is to use hybrid approaches that combine deep and shallow NLP since they have demonstrated better results. For example, [MGM98] uses a pipeline paradigm such that several tools can apply different strategies for each stage of the pipeline.

For the aim of text interpretation for the annotation of media, shallow processing techniques are useful to deal with vast amounts of media, but on the other side, extractable annotations have a superficial (less restrictive) character w.r.t. the content semantics they represent. This is true given that, independently of the technique used, e.g., machine-learning or pattern-based grammars, their basis is string matching. This can be seen in Section 2.4.2 where a shallow process is described.

If the aim is to integrate media as structured information into ISs, annotations that represent a deep character (more restrictive) of the content semantics are useful.

*'**Saturday October 14, 2006.** Now in its fourth year, Morelia's International Film Festival is taking place this week. **Martin Scorcese**'s The Departed, Francoise Ozon's El tiempo que resta, and Abel Ferrara's Mary (Grand Prix winner at the Venice Film Festival) are a few of the films that will be screened at this year's festival. Special guests will include Guillermo del Toro, attending the Mexican premiere of his film, Pan's Labyrinth, and Diego Luna, who will present his latest feature, Fade to Black.'*

Figure 2.6: Sample news about film festivals.

For example, to illustrate the difference between a superficial and a deep character of content semantics see the text in Figure 2.6[1]. Consider the terms *PersonName* and *FilmDirectorName* as annotations for the string "Martin Scorcese". *PersonName* has a superficial character and *FilmDirectorName* a deep character. The superficial character of an annotation is one that is generic and less context dependent. To extract an annotation with a deep character of content semantics it is required to consider different relations between a string and other pieces of data w.r.t. a domain of interest. Shallow-processing techniques can extract annotations with a deep character, but this involves specifying more complex, syntax-dependent and domain-dependent grammars or more complex domain-dependent training scenarios (to support machine learning). These are expensive tasks, and, at the same time, remove the advantage of shallow processing techniques, which is their generic and domain-independent character. Thus, instead of indirectly representing domain semantics through pattern-based grammars or training data, a more natural means of representation for domain semantics are required.

Domain ontologies are useful means to represent domain semantics. In this way, a process that uses a domain ontology as a declarative program to determine annotations with a deep semantic character is desirable. A process that exploits domain ontologies to extract annotation with a deep level character is investigated in this work and is called Deep-Level Interpretation (DLI).

When considering the integration of media to ISs, application programs are also affected, and since the contents of new media are unforeseeable, the object models required to support the integration of media to the user interface of an IS are also unforeseeable. Hence, ISs are required to evolve according to the new incoming media. Software engineers are therefore expected to contribute to the evolution of the systems by keeping a track of the required object models, which is a task that is strategic for knowledge management within an organization.

[1]Taken from http://gomexico.about.com/b/2006/10/14/morelias-international-film-festival-kicks-off-today.html

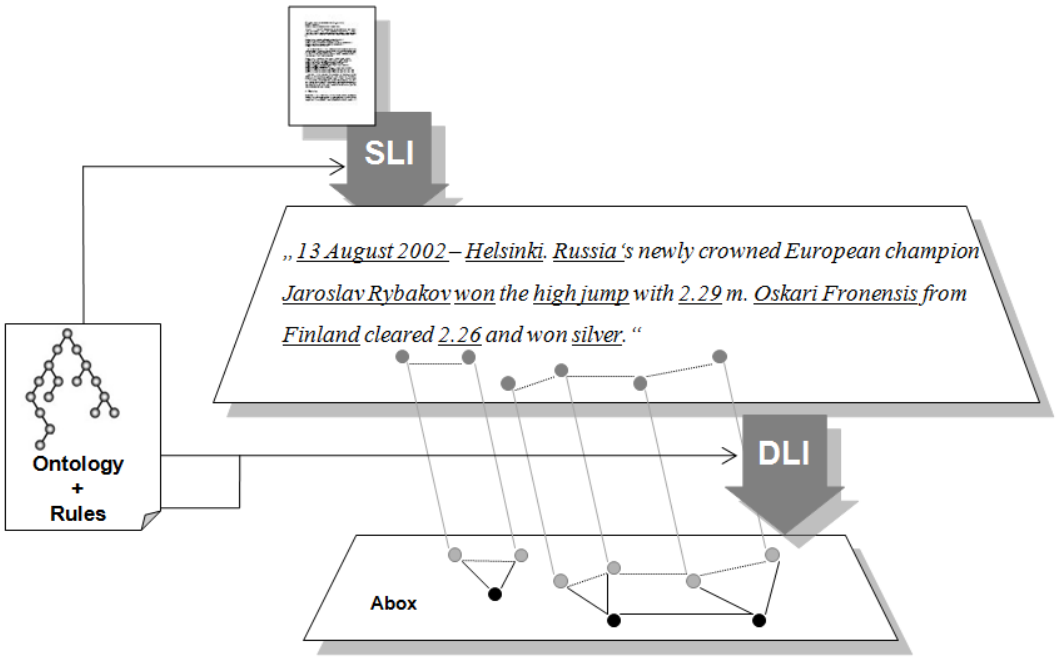


Figure 2.7: The DLI framework

In this chapter a framework is proposed for the deep-level interpretation (DLI) of text documents that provides a foundation for:

- the automatic creation of annotations to allow content management to work on the basis of deep content semantics,
- the exploitation of annotations in web-based application interfaces for various application services, including location-aware and situation-specific navigation services,
- knowledge management with a systematic approach to knowledge modeling by directly using modeled knowledge for text interpretation,
- knowledge management for semantics-based service invocation.

The objective of the DLI framework is to cope with vast amounts of media, more specifically with text documents. To achieve this, a hybrid approach is used, but compared to other approaches which combine deep NLP and shallow NLP such as [MGM98], in this framework a combination of a pattern-based grammar approach and a logic-based approach is used. Qualifying interpretation with the adjective *deep-level* implies the consideration of different levels of interpretation. Thus, as shown in Figure 2.7 the framework distinguishes between two levels of interpretation called surface-level (SLI) and deep-level interpretation (DLI).

Surface-level interpretation (SLI) refers to the process of shallow analysis in which annotations are assigned to specific segments of data found in media. The annotations that result from SLI are called surface-level annotations or *observations* given that

the framework can be seen as an intelligent agent and SLI is the component responsible for perception. SLI is therefore a component that changes according to the media being analyzed, and in this work, text documents are of interest. For example, consider again the text presented in the previous section in Figure 2.6 (see page 24), the segment related to the string "Francoise Ozon" can be interpreted as the name of a person. SLI creates a corresponding description to represent the intended meaning (a person's name) and associates the description with the corresponding segment (the string "Francoise Ozon") of the media object. As a result an annotation is created. An annotation can be represented as an entry in a relational data base, in a structured language such as XML or can be expressed with a logic-based formal language with well defined semantics, such as Description Logics (DLs). The latter allows to exploit reasoning services, and in this way supports the creation of "intelligent" applications. As considered in the field of Artificial Intelligence (AI), the term "intelligent" in this work refers to the ability of a system to find implicit consequences of its explicitly represented knowledge. Such a system is therefore characterized as a KBS [NB03], which is at the core of an intelligent agent. Although in this work we do not completely build such an agent, SLI techniques are designed in such a way that SLI results (also called "observations" in the sequel) are represented with a DL-based language formalism using Abox assertions (see Section 2.1 for a description of DLs and Abox assertions).

Deep-level interpretation (DLI) performs what colloquially is expressed as "reading between the lines". This means that DLI aims at extracting the information that is implied by the text but is not explicitly expressed on the surface. Therefore, the results of DLI can not be directly associated with specific segments of media. The results of DLI are obtained from the interpretation of various segments of media. Therefore, this type of information can only be obtained by considering the observations (previously obtained from SLI) and by considering how they are related to each other. For example, consider again Figure 2.6 (see page 24), the second sentence implies the existence of three show events which are part of a film festival, but there are no explicit references to the show events that can be directly associated to segments of data. Provided the knowledge about the film festival domain, the show events can be hypothesized to explain the observations about person names, film names, a festival name and how these observations are related to each other. In this way DLI aims at extracting additional information (in this example the three show events) that was not represented explicitly before and that helps to *explain* the observations (obtained from SLI) to justify their existence — in a way that is consistent w.r.t. the domain knowledge. This is achieved by using: a domain ontology, SLI results (in the form of Aboxes) and reasoning services.

As mentioned before, by using reasoning services the framework illustrated in Figure

2.7 can be considered to be part of an intelligent agent which uses SLI as perception and DLI as KBS. DLI is logic-based and has the following requirements:

- **DLI's input:**
  1. An ontology and a set of rules that describe a terminology for a specific domain of interest.
  2. A set of observations that result from SLI processes, represented as Abox assertions that are consistent with respect to the domain ontology (see Section 2.1 for a definition of Abox and consistency).
- **DLI's output:** Content descriptions, in the form of Abox assertions, representing the content semantics of a text document. Content descriptions help to *explain* the observations w.r.t. the domain ontology. DLI results are therefore called *explanations*.

Before describing the DLI process in detail, the following sections present a example domain ontology and describe a specific SLI process. The example ontology will be used to illustrate examples throughout this work. The SLI process is described to show that the input requirements of the DLI process (see above) are fulfilled by state-of-the-art technology.

2.4.1 An Example Domain Ontology

After having described the characteristics of SLI and the input requirements of DLI, we are able to introduce an example domain ontology that can be used by these two processes.

A *domain* in knowledge representation [RN03a] is a part of the world about which we wish to express some knowledge, e.g., medicine, engineering, law, athletics, etc. In this way, domain ontologies provide not only a vocabulary ($\mathcal{S}$) to name concepts and roles that are specific to a domain, but also a terminology ($\mathcal{T}$) to specify the semantics of that vocabulary. Appendix A shows an excerpt of the Athletics Event Ontology (AEO)[2] [DEDT07] that will be used to illustrate examples throughout this work. A detailed explanation on the design of the AEO ontology is provided in Section 3.2. For now only some characteristics are described.

As stated in [DEDT07], the goal of ontological engineering is the construction of a consistent knowledge base efficient to serve the requirements of an application context. The design of the AEO ontology follows specific design patterns, such that it serves the requirements of the interpretation process described in Section 2.5.

[2]Downloadable from: http://www.boemie.org/ontologies

'13 August 2002 - Helsinki. Russia's newly crowned European champion Jaroslav Rybakov won the high jump with 2.29 m. Oskari Fronensis from Finland cleared 2.26 and won silver.'

Figure 2.8: Text excerpt with relevant information from the athletics domain.

Since the goal of SLI and DLI is to support the automatic annotation of media, the domain ontology should include the following elements:

- A signature that provides the required vocabulary of the domain of interest by means of atomic concept descriptions and atomic role descriptions used by SLI and DLI processes.
- A Tbox containing GCIs involving complex concept descriptions and role descriptions useful to represent the semantics of more abstract information identified by the DLI process.

Note that the Abox part of the ontology is considered empty, since it is not used by the SLI and DLI processes, instead these processes use Aboxes that aim to describe the content of specific media objects. The elements above specify the extraction requirements imposed on SLI and DLI processes. In other words, the SLI process should be able to find media segments that contain information suitable for their formal representation with the use of atomic concept and role descriptions of the domain ontology. The DLI process should be able to interpret the SLI results with the help of reasoning on rules and GCIs to extract additional information with more abstract content semantics.

To determine the signature of a domain ontology, the ontology engineer should observe the content of a media corpus. The ontology engineer starts by identifying relevant domain-related facts found explicitly in media content, and later by understanding the more abstract information induced by those relevant facts. Explicit facts help to define names for atomic concept descriptions. For example, Figure 2.8 shows an abstract of a document about athletics news. The underlined words highlight the information that is commonly found explicitly in athletics news. Concept names such as *Date, PersonName, Performance, CountryName,* etc., are relevant. In this way, the signature can have the following concept names.

$$(CN)_{Athletics} = \{Date,\ PersonName,\ Performance,\ CountryName\}$$

After identifying the relevant explicit facts, more abstract information should be identified. Abstract information is composed of attributes represented by the explicit facts. For example, consider again the text in Figure 2.8 and the concept names above. *PersonName* and *CountryName* can be modeled as attributes of a more abstract concept such as

Athlete, where an athlete is a person which has a name and a nationality. Thus, the concept names *Athlete* and *Person* and the role names *hasName* and *hasNationality* are identified and added to the signature.

$$(CN)_{Athletics} = \{Date,\ PersonName,\ Performance,\ CountryName,\ Athlete,\ Person\}$$
$$(RN)_{Athletics} = \{hasName,\ hasNationality\}$$

As will be explained in Section 2.5, the DLI process requires concept and role assertions as input. Thus, SLI should also recognize relations between media segments. For example, consider again Figure 2.8 (page 28). SLI should be able to extract the relations of the segment containing the string "Oskari Fronensis" with the segments containing the strings "Finland" and "2.26". To express such relations, role names are required such as *personNameToCountryName* and *personNameToPerformance*. These role names are generic names that express an association between two attributes of a more complex object. Notice that they are more specific names than a role name such as *associatedWith*. More specific role names, such as *personNameToPerformance*, provide a hint on the required domain and range restrictions that should also be modeled in the ontology. Domain and range restrictions contribute to the performance of the DLI process. In this way the signature is augmented as follows.

$$(CN)_{Athletics} = \{Date,\ PersonName,\ Performance,\ CountryName,\ Athlete,\ Person\}$$
$$(RN)_{Athletics} = \{hasName,\ hasNationality,\ personNameToCountryName,\ personNameToPerformance\}$$

As Section 2.6 describes, the DLI process can also be used to interpret image content as long as the input is a set of concept and role assertions. Thus, provided a SLI process able to extract information from image content about objects and spatial relations between them, the ontology should also model concept names and role names for image content. For example, consider the image in Figure 2.9 (page 30), concept names such as *PersonFace, PersonBody, FinishLine,* etc., can be identified. Also spatial relations between objects can be identified such as *adjacent, near,* etc. Thus, the signature of the AEO ontology is augmented with those names.

$$(CN)_{Athletics} = \{Date,\ PersonName,\ Performance,\ CountryName,\ Athlete,\ Person,\ PersonFace,\ PersonBody,\ FinishLine\}$$
$$(RN)_{Athletics} = \{hasName,\ hasNationality,\ personNameToCountryName,\ personNameToPerformance,\ adjacent,\ near\}$$

The concept names SLC and DLC, stand for, Surface-Level Concept and Deep-Level Concept, respectively. The role names SLR and DLR stand for Surface-Level Role and Deep-Level Role, respectively. They are also included in the signature and represent the

Figure 2.9: Information from visual modality in the domain of athletics events.

most generic concepts (roles), after $\top$, in the Tbox hierarchy. Thus, the ontology contains various inclusion axioms such as the following:

$$\begin{aligned} PersonName &\sqsubseteq SLC \\ CountryName &\sqsubseteq SLC \\ Person &\sqsubseteq DLC \\ Athlete &\sqsubseteq DLC \\ hasNationality &\sqsubseteq DLR \\ personNameToPerformance &\sqsubseteq SLR \\ &\ldots \end{aligned}$$

These generic concepts and roles proved to be practical for applications, such as in [PKM09a]. They are used while querying the knowledge base, to distinguish between explicit and implicit media content. They are also added to the signature (see Appendix 2.4.1 in page 27). Moreover, this distinction simplifies the work of an ontology engineer in the modeling of disjointness axioms between descriptions extracted by SLI and DLI processes. This is explained in Section 3.2 (page 84).

After identifying most of the elements in the signature, complex concept descriptions can be defined. As previously described, abstract concepts are composed of various attributes. Thus, defining the concept *Athlete* can be done with the help of a complex concept descriptions using the signature and some constructs as follows.

$$\begin{aligned} Person \sqsubseteq DLC \sqcap & \exists_{\leq_1} hasName.\top \\ & \sqcap \exists_{\leq_1} hasNationality.\top \\ & \sqcap \exists_{\leq_1} hasPart.PersonBody \\ & \sqcap \exists_{\leq_1} hasPart.PersonFace \\ Athlete \sqsubseteq & \ Person \end{aligned}$$

The Tbox containing various complex concept descriptions can be seen in Appendix 2.4.1 (page 27). The DLI process uses the Tbox as a restriction mechanism to select between consistent and inconsistent interpretation results. DLC concepts are not exhaustively defined given that media content is usually incomplete. Thus, only some attributes are actually found within the media content. Moreover, SLI processes can fail to identify information. Notice that if SLI and DLI processes interpret text and image content, the attributes of an abstract object, e.g., *Person*, are characteristic from different content modalities. For example, the description of the concept *Person* (see above) is composed of SLC concepts that represent the content semantics from segments found in text, e.g., *PersonName* and from segments found in an image, e.g., *PersonFace*.

Domain and range restrictions for SLR roles should also be modeled. For example, for the roles *personNameToCountryName* and *personNameToPerformance*, the following restrictions are added.

$$\begin{aligned} \exists personNameToPerformance.\top \sqsubseteq & \ PersonName \\ \top \sqsubseteq & \ \forall personNameToPerformance.Performance \\ PersonName \sqcap & \ (\leq_1 personNameToPerformance.Performance) \\ \exists personNameToCountryName.\top \sqsubseteq & \ PersonName \\ \top \sqsubseteq & \ \forall personNameToCountryName.CountryName \\ PersonName \sqcap & \ (\leq_1 personNameToCountryName.CountryName) \end{aligned}$$

The AEO ontology (see an excerpt in Appendix 2.4.1) models athletics events, competitions, rounds, trials and athletes of many sport disciplines. Only the axioms required to illustrate the running example are shown in the appendix.

After giving a short description of the AEO ontology, in the next section the SLI process is described. The aim is to show that state-of-the-art shallow-processing techniques are good enough to fulfill the input required by the DLI process.

2.4.2 Surface-Level Text Interpretation

Information Extraction (IE) focuses on the extraction of structured information, that is considered relevant to a specific problem. As described in [Cun05], an IE process is designed to fulfill some *pre-specified and precise* information need. In this work, the signature and the terminology of an ontology specify the IE requirements imposed on SLI and DLI processes (as previously described in page 28).

In the context of textual content, according to Russell and Norvig [RN03b], IE systems are mid-way between information retrieval (IR) systems and full-text parsers, in that they need to do more than just considering a document as a bag of words, but less than completely analyzing every sentence. A characteristic that is also applicable for the SLI and DLI processes, and which is sufficient to fulfill media annotation. In [RN03b] a generic classification of IE systems is provided which distinguishes between attribute-based and relation-based systems. *Attribute-based systems* assume that the text content refers to one object and the objective is to extract attributes of that single object. *Relation-based systems* assume more than one object and their objective is to extract the attributes and the relations between the attributes of an object. The DLI process presented in Section 2.5 is inspired by the work of Neumann and Möller [NM06, NM08, MN08]. In their work, *aggregates* (see Section 2.5.2, page 52 for a definition) are considered as building blocks for the interpretation process. The DLI process assumes a relation-based system at the SLI level (see Figure 2.7, page 25). The results of SLI are the attributes of an aggregate. The relations between the attributes represent the constraints that should be satisfied in order for DLI to determine the specific aggregate that is to be extracted.

Finite-state transducers are the basis of many relation-based extraction systems. An example of such a system is the system called General Architecture for Text Engineering (GATE) [Cun02]. In the following section an SLI application is described which uses the GATE platform. Note that even off-the-shelf tools exist in the market that can be used as an SLI process, for example OpenCalais[3]. The objective of the following section is to demonstrate that the DLI process, presented in Section 2.5, works on the output that state-of-the-art NLP technologies are able to produce. The following section shows that shallow processing is enough to produce the required DLI input. Furthermore, shallow processing is advantageous given the smaller amounts of resources used, compared to full NL processing.

[3]see OpenCalais at http://www.opencalais.com/

An Ontology-Based SLI Process

The SLI process described here is based on the work of Galochkin [Gal08]. It encompasses a set of basic steps that characterize contemporary text IE systems, which can be classified into two phases, named *preprocessing* and *analysis*. Both phases are executed as a chain of finite-state transducers. Moreover, SLI is an ontology-based process given that it represents the extracted information as Abox assertions w.r.t. to a domain ontology, such as the AEO ontology introduced in Section 2.4.1.

Provided a text document as input containing news about athletics events, the output is an Abox containing the following

- concept assertions representing the attributes of a set of abstract objects, and
- role assertions representing relations between attributes of abstract objects (called aggregates in [NM06]).

Prepocessing phase

In our setting, the finite-state transducers which are used to execute each step of this phase are obtained from built-in processing resources offered by GATE [CMB+09] called ANNIE. These steps can be applied to any natural language text since they are generic enough to be considered as domain independent. For this work the following steps were applied in a pipeline:

> *Tokenizer* → *Sentence splitter* → *Part-of-Speech tagger* → *Lemmatizer* → *Gazetteer*

Each step is executed by a processing resource from ANNIE identified with the same name. Each step produces reference annotations. Differently to embedded markup, in which annotations are embedded in the analyzed document, reference annotations are stored in a database that associates the references to the corresponding segments of data identified by spans with start and end offsets.

Each of the resources has a predefined set of rules. A set of rules is called a *phase* and a set of phases is called a *grammar*. Phases are used to declare two aspects, the input type, i.e., the category of annotations accepted as input, and the sequence in which a set of phases will be applied. For a detailed description of the syntax of the JAPE language (Java Annotation Patterns Engine) for the definition of rules, phases and grammars refer to [CMB+09]. In the JAPE syntax, rules are built from two elements, i.e., a complex pattern and an action. They are distinguished as the left-hand-side (LHS) and right-hand-side (RHS) elements, respectively.

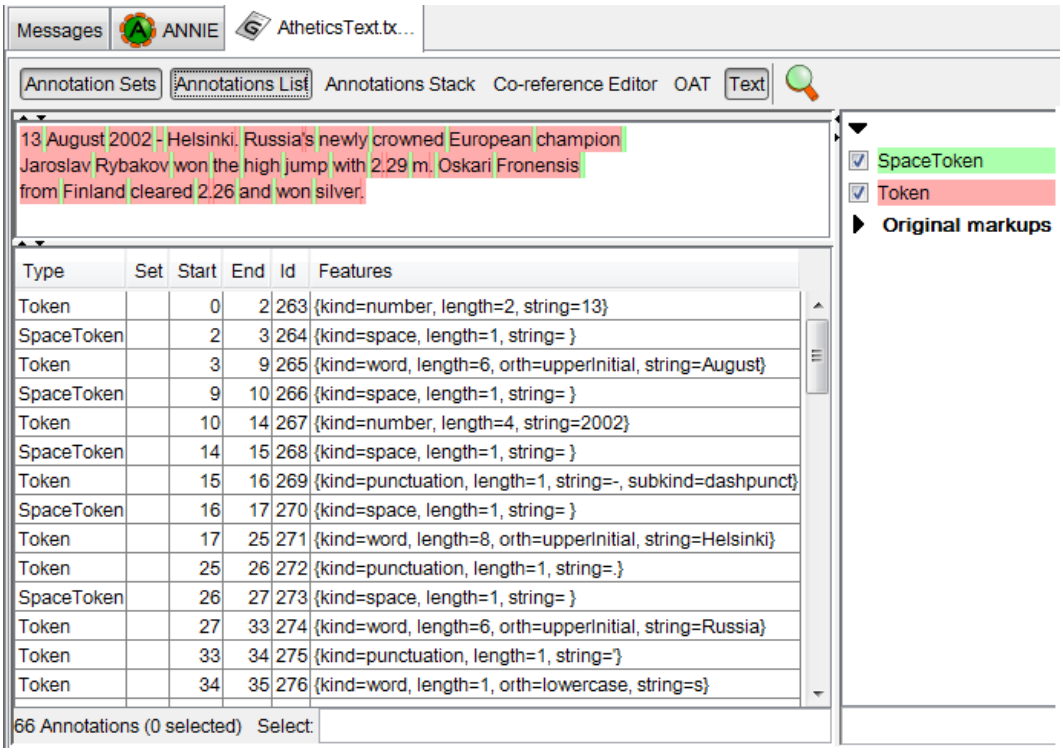


Figure 2.10: Screenshot of the GATE interface showing results of the English tokenizer

```
Rule: AgeBraces  // Tim Hortons (33)
( ({PersonName})({Token.root=="("}) (AGE):cur
({Token.string == "yrs"})?({Token.root == ")"}) )
-->
:cur.Age = {rule = "AgeBraces", class = "Age"}
```

A complex pattern (see the example above) is defined with the use of strings (see `{Token.root=="("}`, `{Token.string == "yrs"}`), previously defined macros (see `(AGE) :cur`) and annotations previously assigned by some GATE module (see `{PersonName}`) together with regular expression operators, e.g., `? | ==`. An action is executed if a pattern is found that matches the LHS of the rule. An action involves annotation statements (see `class = "Age"`) or Java code. The annotation statements create new annotations, consisting of a type, start and end offsets, id and features described as attribute-value pairs (see Figure 2.10).

The *tokenizer*, or more specifically, the English tokenizer breaks the text into different tokens of type number, word, space, punctuation and symbol. It considers special English constructs such as cardinals, abbreviations, etc., that are helpful for later part-of-speech tagging. An example is shown in Figure 2.10. The *sentence splitter* identifies the limits of a sentence and uses regular expressions to differentiate between full stop tokens, such as punctuation, line break, etc., and non-full stop tokens such as abbreviations followed by a punctuation. The default set of rules of the sentence splitter was enhanced with a rule

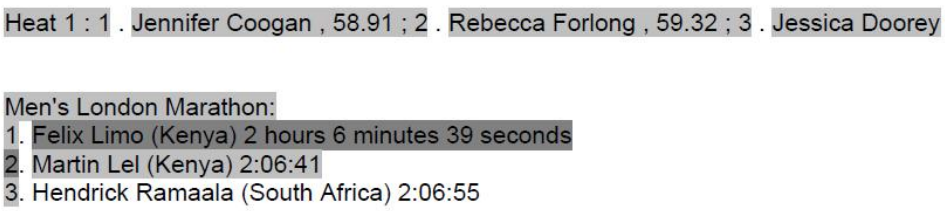


Figure 2.11: Incorrect sentence splitting

that prevents false positives. For example, in news about athletics events it is common to find lists that relate athletes with corresponding ranking and performance information. As Figure 2.11 shows, when using the default set of rules, the punctuation tokens are used for sentence splitting, while in this specific case the punctuation is part of an enumerated list. To facilitate later relation extraction during the analysis phase, it is convenient to consider such enumerated lists as one sentence. For this reason the following rule was added to the GATE sentence splitter

```
Rule: notFullStop
( ({Token.string == ":"} | {Token.string == ";"} | (NEWLINE))
({SpaceToken})* {Token.kind == number} ({SpaceToken})*
(FULLSTOP) ):cr
-->
:cr.NoSplit = {rule = "notFullStop"}
```

The *part-of-speech tagger* (POS) is used with the default set of rules. It is used in order to identify lexical categories on top of word and symbol tokens obtained from the tokenizer within the limits of a sentence. The objective is to distinguish between verbs, nouns, articles, etc., which is a requirement for the lemmatizer (see below) to produce useful results. As shown in Figure 2.12, the results are expressed as a feature called *category* with syntactic categories, e.g., VP, NNP, etc.

The *lemmatizer* (a.k.a. stemmer) extracts lemmas as a result of morphological analysis. Lemmas are helpful for the next step in the pipeline given the assumption that gazetteer lists do not contain words with all possible inflections. In this way, ignoring the inflection of a word is a measure of relaxation to support gazetteer lookup. As Figure 2.13 shows, the lemmatizer adds a feature called *root* whose value is the lemma of the corresponding token.

The last step of the preprocessing phase is executed by the *gazetteer*, which contains lists of words classified under a major and a minor type. If a word on the text is identified within a gazetteer list, then the gazetteer uses the major or minor type of the list to annotate the corresponding word. For this process, both the lemma (previously obtained from the lemmatizer) and the original string of a word are used. Thus, as previously described

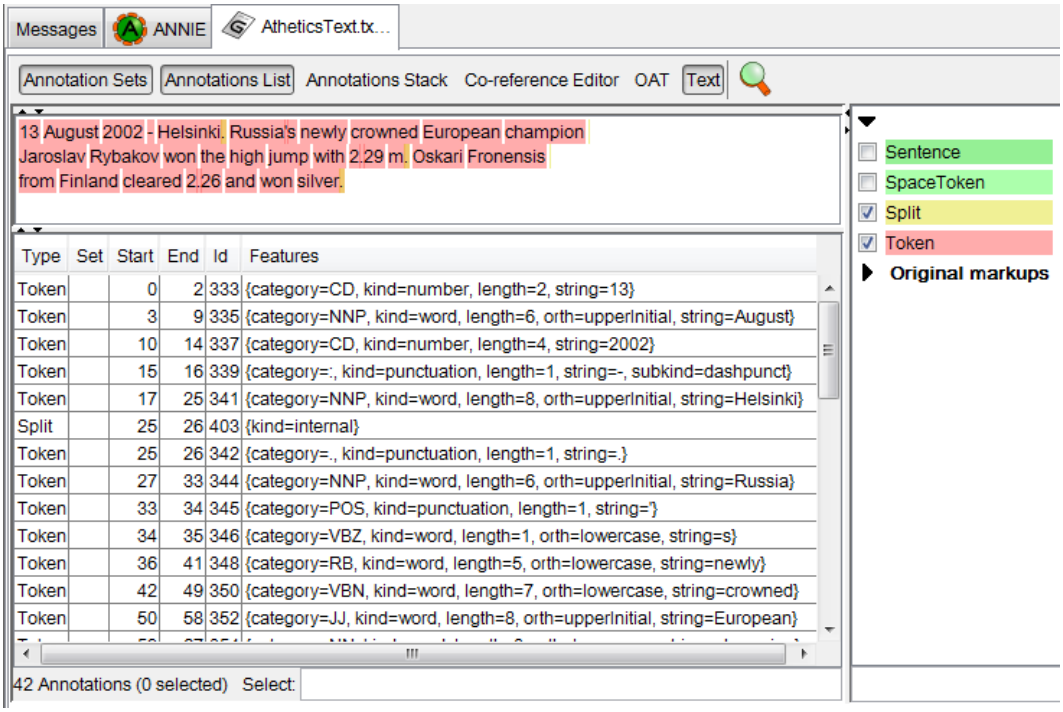


Figure 2.12: Screenshot of the GATE interface showing syntactic categories obtained after part-of-speech tagging.

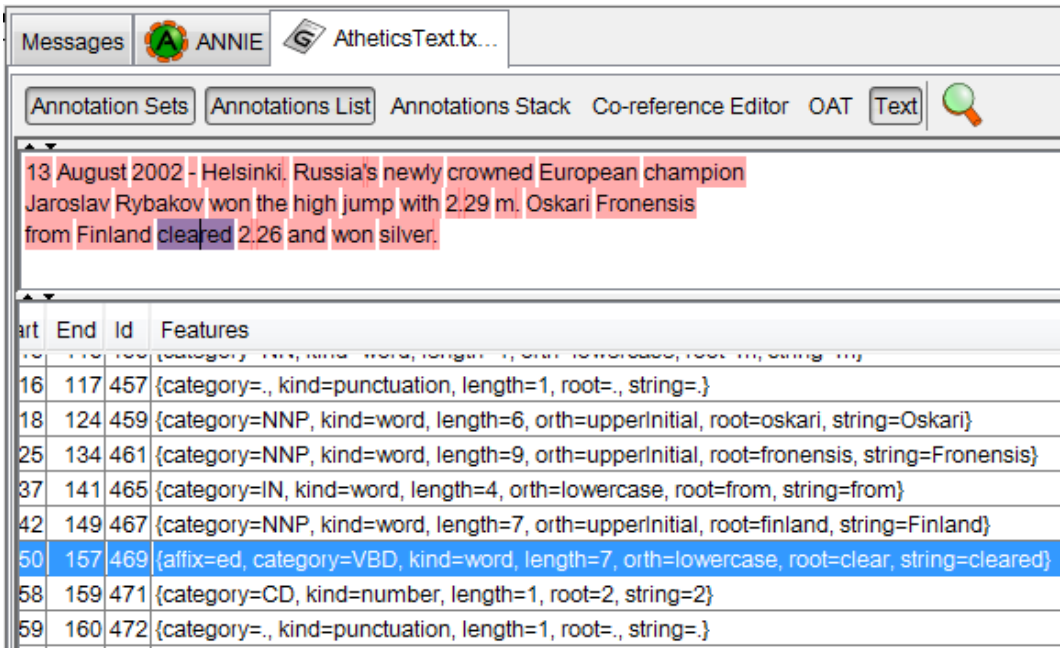


Figure 2.13: Screenshot of the GATE interface showing the root feature containing the lemmas of a token.

it is assumed that gazetteer lists do not contain words with all their inflections; in this case the lemma is helpful. But also the original string is necessary since lemmas alone are not enough. Thus, the lemmatizer produces lemmas such as *Cayman island* from *Cayman islands* or *Athen* from *Athens*, etc., that could prevent the gazetteer from finding them in a list. Also this process should be case sensitive in order to avoid identifying words such as *nice, bar* and others as of type *city.*

Analysis phase

The steps in this phase are executed with the help of the following finite-state transducers

NER → *Co-reference resolution* → *NER* → *Relation extraction*

The analysis phase relies on gazetteer lookups in a balanced way, this means, that gazetteer lists are used for well-known names and grammar rules to exploit the word context in cases where gazetteer lookups may produce ambiguity. The annotations obtained from this pipeline are used to produce ontology assertions. Named Entity Recognition (NER) involves identifying certain occurrences of words or expressions as belonging to particular categories of Named Entities (NEs) [MMG99]. As the pipeline of the analysis phase above shows, NER is performed before and after co reference resolution. This is necessary given that the extraction of some NEs depend on the previous extraction of other NEs. For example, in the following text

"Bungei (24yrs) wins 800m. Wilfred Bungei from Kenya."

the string "24yrs" can be extracted as *Age* (of a person) only after the string "Bungei" is extracted as a *PersonName*. Moreover, placing co-reference resolution in the pipeline between both NER steps is helpful to support the second NER process. Thus, some NEs serve as context for context-dependent and context-independent entities.

The first step in the analysis phase is *NER* of context-independent entities. In our setting the NEs *Nationality, Gender, CountryName, StadiumName, CityName, PersonName* and *Date* can be easily extracted with the help of gazetteer annotations (obtained from the previous phase) since they are highly unambiguous. For this, a JAPE grammar was designed with a set of phases for each of the NEs mentioned above. Given that these NEs are context-independent there is no need to enforce an order on the execution of the rules. As shown in Figure 2.14, this step assigns a type to each token and adds a feature called *class* with its value being the name of the SLC concept that should be instantiated at the end of the analysis phase.

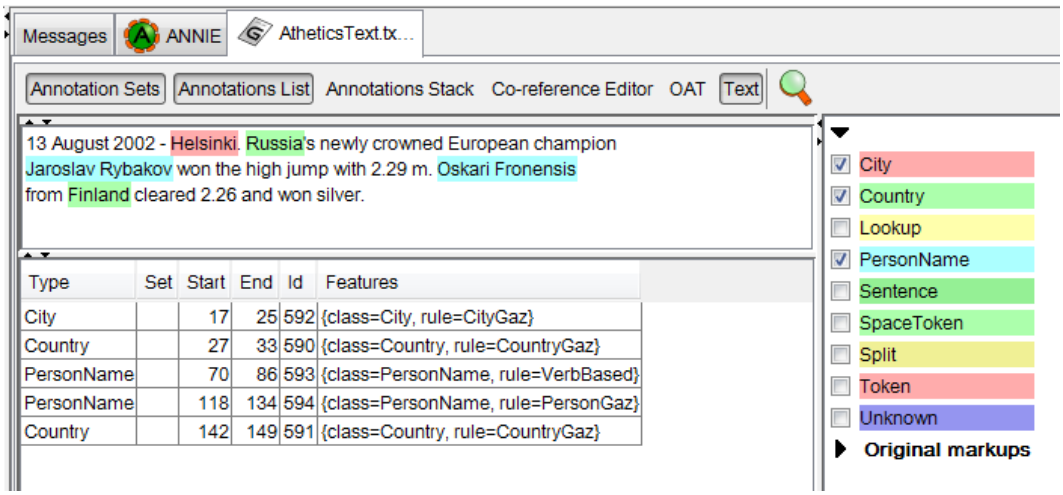


Figure 2.14: Screenshot of the GATE interface showing context-independent named entities.

The next step aims to solve *co-reference* problems. Co-reference arises whenever the same real world entity is referred to in various ways in a text fragment. As explained by [SGC09], this problem may arise due to the use of: (i) different names describing the same entity, (ii) classification expressions such as "the world's richest man" and (iii) the use of pronouns. To solve co-reference problems, two GATE components, the so called OrthoMatcher and the ANNIE pronominal co-referencer are used to deal with items (i) and (iii), respectively. Both, OrthoMatcher and the pronominal co-referencer, require NEs as input. The OrthoMatcher finds identity relations between NE of the same type based on string matching. The pronominal co-referencer exploits NEs and OrthoMatcher results. For a description of the algorithm behind the pronominal co-referencer see [CMB+09]. In the athletics events domain, co-reference resolution is executed for the NEs *PersonName*, *CountryName* and *CityName*. The results of this step are expressed as a feature called *matches* containing a pair of numeric values representing the offset of the matching string as shown in Figure 2.15 (see page 39).

The third step is *NER of context-dependent entities*. For this task a JAPE grammar is designed with an ordered set of phases. Similar to the first NER step, each phase corresponds to a different type of named-entity — setting the following order: *Age* - *Performance* - *SportsName* - *Ranking* - *SportsEventName* - *SportsRoundName*. Figure 2.16 (see page 40) shows the results of this step using the GATE user interface. The results of the first three steps in this phase are used to create instances of the corresponding SLC concepts, as shown in Figure 2.17 (page 40).

Finally, the last step of this phase is *relation extraction*. For this purpose an algorithm was designed that requires as input NEs and is based on the following considerations. The set of NEs found within the limits of a sentence are related to each other, such that the

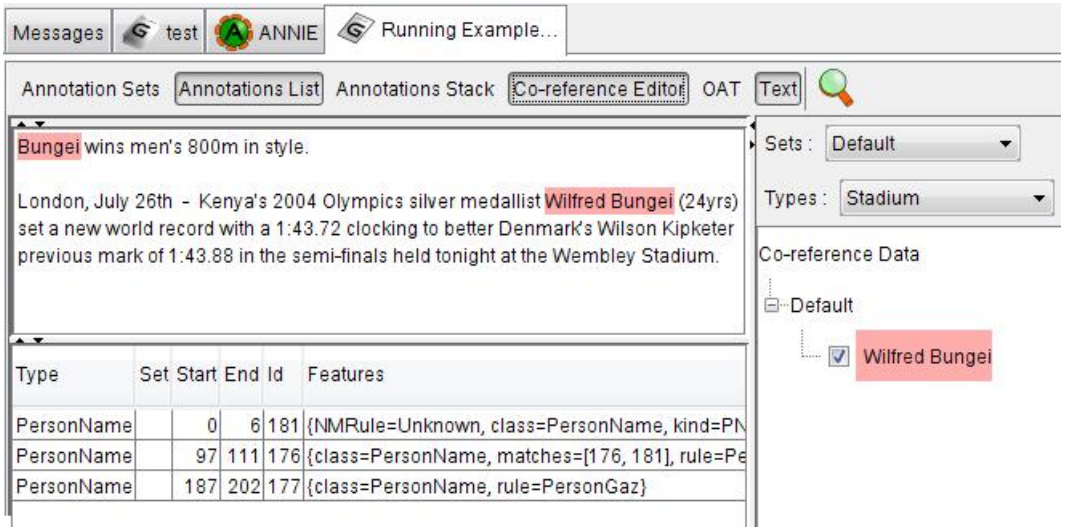


Figure 2.15: Screenshot of the GATE interface showing the results of the OrthoMatcher.

objective is to find pairs of NEs that satisfy domain and range restrictions of the SLR roles defined in the ontology (see Appendix 2.4.1). If pairs of NE are found that satisfy the restrictions of a specific SLR role, the corresponding role assertion is created, akin to a forward-chaining rule. In case that more than one combination of NE pairs are found and cardinality restrictions exist, a combination should be chosen based on token proximity within the text. For example, as shown in Figure 2.18 (page 41), four NEs are candidates to be in a role *personNameToCountryName*.

According to the cardinality restrictions, a *personName* can be related to at most one *CountryName* by the role *personNameToCountryName*, such that the options are to relate "Jaroslav" with "Russia" or "Finland" and similarly for "Oskari". The priority is set to the pairs that are closer to each other by considering an ordered set of NEs within a sentence, such that starting from left to right in the set of NEs the next closest NE should be chosen. For example, from the text in Figure 2.18 a set of NE was extracted and later used to create concept instances such that the following ordered set of ids is obtained $\{date_1,\ city_1,\ country_1,\ pName_1,\ rank_1,\ hjName_1,\ perf_1,\ pName_2,\ country_2,\ perf_2,\ rank_1\}$. According to the set, $country_1$ is closer to $pName_1$ and $pName_2$ is closer to $country_2$. The results of this step are shown in Figure 2.19. As explained before in Section 2.4.1 (see page 2.4.1), the role names used by SLI from textual content, provide hints on the domain and range restrictions that are required.

Having described the type of output that a SLI process should provide, the following section will focus on DLI.

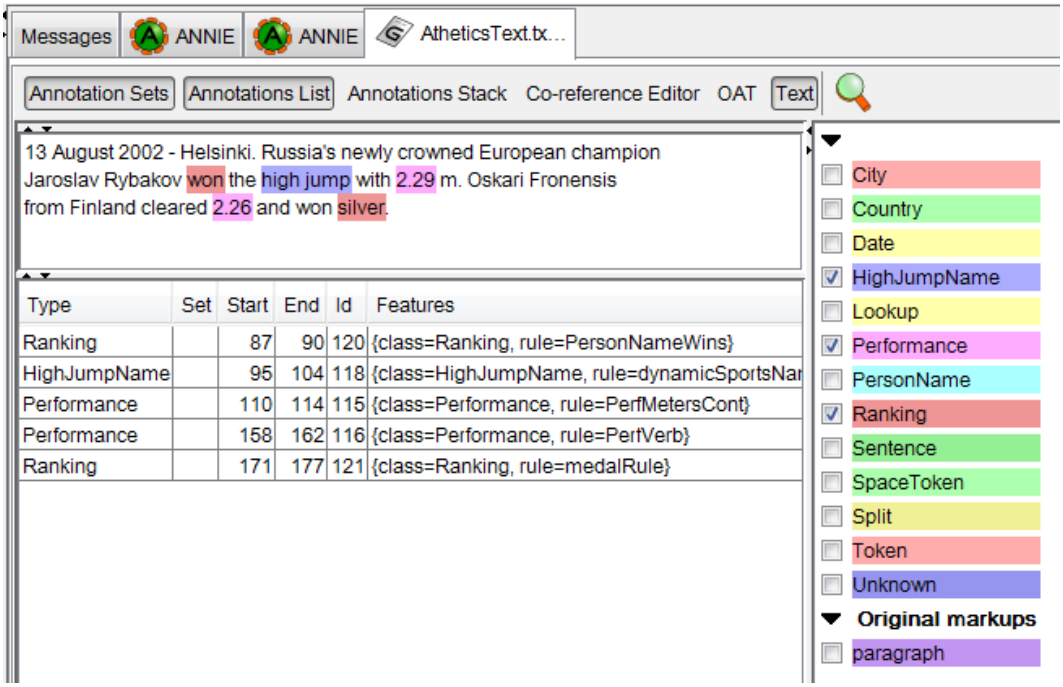


Figure 2.16: Screenshot of the GATE interface showing context-dependent named entities.

$$
\begin{aligned}
date_1 &: Date \\
(date_1,\ \text{“13 August 2002”}) &: hasValue \\
city_1 &: CityName \\
(city_1, \text{“Helsinki”}) &: hasValue \\
country_1 &: CountryName \\
(country_1, \text{“Russia”}) &: hasValue \\
pName_1 &: PersonName \\
(pName_1, \text{“Jaroslav Rybakov”}) &: hasValue \\
rank_1 &: Ranking \\
(rank_1, \text{“won”}) &: hasValue \\
hjName_1 &: HighJumpName \\
(hjName_1, \text{“high jump”}) &: hasValue \\
perf_1 &: Performance \\
(perf_1, \text{“2.29”}) &: hasValue \\
pName_2 &: PersonName \\
(pName_2, \text{“Oskari Fronensis”}) &: hasValue \\
country_2 &: CountryName \\
(country_2, \text{“Finland”}) &: hasValue \\
perf_2 &: Performance \\
(perf_2, \text{“2.26”}) &: hasValue \\
rank_2 &: Ranking \\
(rank_2, \text{“silver”}) &: hasValue
\end{aligned}
$$

Figure 2.17: Abox containing the results of the named-entity recognition process.

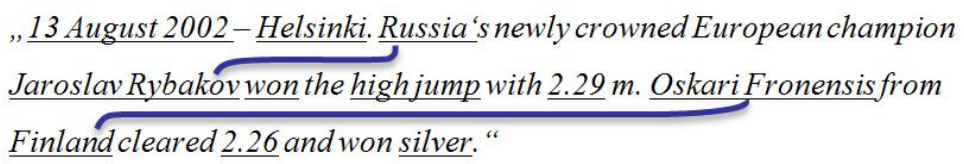


Figure 2.18: Tuples denoted by the role *personNameToCountryName*

$$
\begin{aligned}
(pName_1,\ perf_1) &\ : \ personNameToPerformance \\
(pName_2,\ perf_2) &\ : \ personNameToPerformance \\
(hjName_1,\ perf_1) &\ : \ sportsNameToPerformance \\
(hjName_1,\ perf_2) &\ : \ sportsNameToPerformance \\
(hjName_1,\ date_1) &\ : \ sportsNameToDate \\
(hjName_1,\ city_1) &\ : \ sportsNameToCityName \\
(pName_1,\ country_1) &\ : \ personNameToCountryName \\
(pName_2,\ country_2) &\ : \ personNameToCountryName \\
(rank_1,\ perf_1) &\ : \ RankingToPerformance \\
(rank_2,\ perf_2) &\ : \ RankingToPerformance \\
&\ldots
\end{aligned}
$$

Figure 2.19: Results of relation extraction.

2.5 Deep-level Text Interpretation as Abduction

This section starts by describing abduction, since it is at the core of the DLI process. Later, we can proceed to describe DLI as an abduction-based process.

2.5.1 Abduction

DLI is built on the basis of explanatory reasoning. It aims at extracting information that explains the observations obtained from SLI. The objective is to justify the existence of the observations (in other words, the correctness of SLI results) by explaining the coherence of the observations. Explanatory reasoning is also known as *abductive reasoning* or simply *abduction*.

Abduction is the construction of hypotheses that help to understand a world perceived through observations typically applied in situations with incomplete information. A common application example is medical diagnosis where from symptoms (observations) one wants to find the disease (cause) that explains the symptoms. It is a process where two main tasks are involved:

1. Construction of hypotheses — that aim at explaining the observations.

2. Hypothesis selection.

Given this two tasks, abduction is also described as *inference to the best explanation* and as *reasoning from effects (observations) to causes (explanations).* This characterization of abduction dates back to 1878 when Charles S. Pierce described abduction[4] as a logic inference process that corresponds to the first stage of any interpretive process, which is necessary for the formulation of hypotheses. Since Pierce's contribution, researchers from different disciplines, e.g., philosophy, linguistic, psychology, (see the work of Wirth in [Wir95] for an overview on the application of abduction in various disciplines) have pursued the implementation of abduction, contributing to its further study while trying to address problems such as the definition of parameters for hypothesis selection. The work of Thagard in [Tha78] addresses the problem of hypothesis selection, by proposing simplicity, consilience and analogy as criteria to support this problem.

Consilience measures to what extent a theory (set of hypotheses) explains a set of facts (observations), such that it is possible to compare between theories and be able to state that one theory is more consilient than another if it explains more facts than the other. This is similar to the explanatory characterization of a hypothesis described by Pierce. Simplicity refers to the number of hypotheses required to create a theory that explains a set of facts. According to Thagard "a *simple consilient* theory is desired, such that it not only must explain a range of facts but it must explain those facts without making a host of assumptions with narrow application". With this he means that ideally, theories must not achieve consilience at the expense of simplicity. Analogy allows to increase the value of an explanation. For example, given two objects or classes A and B that are similar due to the properties P, Q and R, and S is the explanation of A having P, Q and R, then S as an explanation for B having P, Q and R is more valuable. In this way, inference to the best explanation is inference to the theory that best satisfies the criteria of consilience and simplicity, as well as analogy.

These notions of Pierce and Thagard [Tha78] have set the basis for further research in abductive reasoning in the area of AI where their work is often cited and that are also relevant in this work as it will be described in Section 2.5.2. But first related work on abduction in the context of artificial intelligence is described.

Related Work: Abduction in Artificial Intelligence

The research done in AI has been focused on the formalization of abduction in the computational realm through logic-based languages such as the work of Kakas et al. [KKT92, KD02], which focuses on the formalization of abduction with logic programming, and Elsenbroich et al. which in [EKS06] focus on abduction with DLs on both Tbox and Abox abduction.

[4]See "Deduction, Induction, and Abduction" at the Standford Encyclopedia of Philosophy.

Given that Abox assertions are the regular means to explicitly represent objects of a domain of interest, it is natural to express observations (from media) as Abox assertions. In consequence a theory that explains a set of observations, represented with Abox assertions, should be expressed through Abox assertions as well, such that it is possible to check for consistency w.r.t. a terminology (Tbox). Thus, Abox abduction is of interest in this work to support DLI of media.

Until now, the problem of Abox abduction has been formally described in the work of Elsenbroich et al. in [EKS06], but it is not shown how to compute explanations. Thus, Abox abduction is a problem that has not been solved until now. A solution to this problem, in the context of DLI, is proposed in this work in Section 2.5.2. In contrast to the approach followed by Kakas et al. [KKT92, KD02] which uses abduction in the context of rules in logic programming only, the approach of DLI proposed here, combines existing DL reasoning mechanisms and rules in a coherent framework as illustrated in Figure 2.21 (see page 52).

The research on abduction in AI has also been focused on its use in different applications, such as knowledge acquisition [KM91, EKS06], robot perception [Sha05], scene interpretation [NM08] and natural language processing [HSAM93]. The works of Hobbs et al. [HSAM93], Shanahan [Sha05] and Neumann and Möller [NM08] have served as inspiration for this work to apply abductive reasoning as basis for DLI. In a nutshell, the work of Hobbs et al. influences the design of the DLI approach as an abduction based process, the work of Shanahan provides a formalization of logic-based abduction adopted in this work to formalize DLI as Abox abduction, and the work of Neumann and Möller has provided notions of knowledge representation with DLs applied on interpretation, more specifically the notion of object configurations has been used for the DLI approach. A short description of these works is provided in the following subsections before describing Abox abduction and DLI in Section 2.5. By describing their work, it is possible to highlight the ideas adopted from their work as well as the differences.

Interpretation as Abduction. The work "Interpretation as Abduction" of Hobbs et al. in [HSAM93] has been influential in conceptualizing text interpretation as a problem that necessarily requires abduction in order to be solved. They developed a linguistic and knowledge-intensive framework to solve the problem of text interpretation, that extracts first-order logic formulas from English sentences. Their approach can be summarized in the following steps, i) derive the logical form of a sentence, together with the constraints that predicates impose on their arguments, ii) merging redundancies where possible and iii) making assumptions where necessary. The result is a first-order logical formula for each sentence in a text document which represents the interpretation of the sentence.

The process of interpretation in [HSAM93] can be divided into three phases[5]. In the first phase, text interpretation is done by means of syntactic and semantic analysis in order to identify *pragmatic problems* such as: reference resolution, syntactic and lexical ambiguity resolution, metonymy resolution, and compound nominal interpretation.

These pragmatic problems can be identified due to different types of knowledge, i.e., from syntactic knowledge, semantic knowledge, domain knowledge, to pragmatic knowledge. See [All95] for a description of these types of knowledge. The result of the first analysis phase is a non-grounded logical expression (also called *goal expression*) composed of a conjunction of atoms. Each atom results from solving one of the pragmatic problems introduced above. Thus, the ability to extract a logical expression is the result of solving the corresponding local pragmatic problem. For example, consider the sentence "Disengaged compressor after lube-oil alarm." To solve the reference of "compressor", the following goal expression is generated:

$$(1)\ \exists x : compressor(x)$$

In the second phase the goal expression should be proved for entailment against axioms in the knowledge base. For example, given a knowledge base that contains the following:

$$starting\_air\_compressor(c_1)$$
$$\forall x : starting\_air\_compressor(x) \Rightarrow compressor(x)$$

The goal expression extracted from the sentence above can be resolved to the instance $c1$ as follows:

$$compressor(c_1)$$

By deriving that c_1 is a compressor the entailment of expression (1) is proved w.r.t. the knowledge base. And in this case no additional assumptions are required. The result of this second phase is a grounded logical expression of the goal expression representing an interpretation. In the case that no derivation is found, assumptions have to be made, and the goal expression must be proved abductively. A more detailed example that illustrates the interpretation of the whole sentence can be seen in [EKM11].

The third phase aims at computing the "cost" of the resulting interpretation. It is considered that during the process of proving a logical form (phase two) different proofs can be found, to choose the "least expensive" proof they developed a method called weighted abduction. *Weighted abduction* aims at providing costs to the different atoms of a goal expression. The costs are given according to syntactic characteristics of the

[5]This division is proposed here for clarification purposes. The phases do not claim an order of application but just a classification of tasks.

sentence. A high cost is provided to syntactic constructs that convey no new information, e.g., pronoun. On the contrary, structures that convey new information and therefore have to be proved abductively have a lower cost. Hobbs et al. have analyzed how likely it is that a syntactic construct conveys or does not convey new information. For example, the main verb is more likely to convey new information than a definite noun phrase which is generally used referentially. Failing to prove a definite noun phrase is therefore expensive.

Full syntactic analysis is done during the first phase to produce a goal expression. Thus, the form of the *goal expression* obtained after the first phase of analysis depends on the syntactic characteristics of a sentence, which will influence the process of hypothesis selection.

After this short description, it is possible to observe that the DLI approach presented here is influenced by the concept of Hobbs et al. on considering interpretation as a task that requires abduction. But different to the work in [HSAM93], in which linguistic knowledge is used, the DLI process presented in Section 2.5 uses only domain knowledge in order to extract and select hypotheses. In this way, the results of DLI are not influenced by sentence construction. The DLI approach requires only results from shallow processing techniques and does not require full syntactic analysis which is advantageous when dealing with large-scale corpora. Full syntactic analysis is obviously necessary for other applications such as language translation.

We would like to argue in this work that for content annotation, the services of shallow-processing techniques are enough to comply with the requirements of DLI. Moreover, the DLI approach exploits the services that DLs reasoners offer such that it is possible to prove the consistency of the SLI and DLI results w.r.t. to a domain of interest expressed through a terminology.

Perception as Abduction The work "Perception as Abduction: Turning Sensor Data Into Meaningful Representation" of Murray Shanahan in [Sha05] presents a formal theory of robot perception as a form of abduction. Here, abduction is used to transform low-level sensor data into a symbolic representation of the world in the form of first-order logic formulas. Shanahan formalizes abduction as follows:

$$\Sigma \cup \Delta \models \Gamma$$

Given a stream of low-level sensor data, represented by the conjunction Γ which is a set of observation sentences, the task of perception is to find one or more explanations of Γ in the form of a consistent logical description Δ of the locations and shapes of hypothesized objects. Σ is a background theory describing how the robots interactions with the world impact on its sensors. The approach of Abox abduction proposed in Section 2.5.2, adopts Shanahan's formalization of abduction but uses it on media interpretation as follows.

The following elements are required: a set of Abox assertions Γ representing observations (obtained from SLI), a knowledge base represented with an ontology Σ, and a set of rules. Abox abduction is a retrieval inference service which aims to derive all sets of Abox assertions Δ (explanations) such that the abduction formula above holds.

Moreover, Shanahan describes four aspects that play a role in abduction applied to perception, i.e., incompleteness and uncertainty, top-down information flow, active perception and sensor fusion.

Incompleteness and uncertainty, are unavoidable in robot perception given the physical limitations of any electrical device that works as a sensor. Uncertainty arises due to noise and incompleteness arises due to a limited vision, and, given the individual goals of a robot, such that Σ not necessarily attempts to capture every facet of the external world. Shanahan refers to mathematical logic as ideal means to express incompleteness and uncertainty, which at the same time possesses a precise semantics. He highlights existential quantification as useful to express incompleteness and uncertainty. The work presented in Section 3.2 about ontology design patterns, highlights the use of existential quantification ($\exists$) and necessary conditions ($\sqsubseteq$) to express incompleteness. Where incompleteness in DLI results from SLI processes failing to extract information. Uncertainty is not considered in DLI. Probabilistic approaches to logic reasoning could be used to represent uncertainty, but this is a type of logic that is beyond the work presented here. Currently there exists formalizations of probabilistic DLs e.g., [Luk07], but research on its application on logic-based media interpretation has just started [GMN+09]. Another concept described in [Sha05] is *top-down information flow*, which has its origins in cognitive psychology but is applied in machine vision, and refers to the influence that high-level cognitive processes have on low-level perception. Shanahan adapts this concept to what he calls *expectation*. In his work, a set of Δ_s that could explain a given Γ, are conjoined with the background theory Σ such that a number of other observation sentences (the expectations) are entailed, which were not present in the original sensor data Γ. Having determined the expectations for each Δ, each Δ can be accepted or rejected by checking whether or not its expectations are entailed.

Top-down information flow is also of special relevance for the DLI approach in the design of the ontology. As explained in Section 3.2, special ontology and rule design patterns are proposed (see *Aggregate Specialization Pattern* (page 93) and *Specific Abduction Rules Pattern* (page 89)). The objective of these patterns is to keep SLI processes generic, while modeling DLI as a domain specific process. In this way, when reasoning with a domain specific ontology, DLI can recognize more specialized DLC concepts which influence SLI results via deduction. For example, from a person name observed from the surface of a text and hypothesizing that it is the name of a person that is participant of a high jump

trial, recognize that it is in fact the name of a high jumper. Similarly to Shanahan, the results of top-down information flow are considered as parameters for hypothesis selection (see Section 2.5, page 63), to support the selection between competing Δs.

Finally, another relevant concept for DLI described by Shanahan is *sensor fusion*. In robot perception, the problem of sensor fusion is relevant given the existence of data coming from: different sensors, or the same sensor but preprocessed in different ways, or from the same sensor but extracted at different times. In order to handle this problem, Shanahan highlights the requirement of designing a background theory Σ which is able to capture how a configuration of objects in the world gives rise to each kind of low-level sensor data. In Section 2.6 we propose a process of multimedia fusion. It is described how the results of DLI are helpful to provide the required level of abstraction to put observations coming from different types of media in a context that is suitable for fusion. Similar to Shanahan, where Σ captures the type of input obtained from different sensors, the ontology used by DLI and the multimedia fusion process is grounded not only on text but also on image content. As explained before in Section 2.4.1, page 31, complex concept descriptions use concept names that represent semantics from text and image content.

On Scene Interpretation with Description Logics The work "On Scene Interpretation with Description Logics" of Neumann and Möller [NM08] presents DLs as a knowledge representation and reasoning system for high-level scene interpretation. They identify various characteristics of a high-level scene interpretation, such that an interpretation is defined as a conceptual structure that:

- deals with several objects and occurrences,
- depends on the temporal and spatial relations between parts of the scene,
- describes scenes in qualitative terms, omitting geometric detail,
- exploits contextual information,
- includes inferred facts unobservable in the scene, and
- uses conceptual knowledge and experiences about the world.

Neumann and Möller emphasize that, to solve the task of artificial vision systems, technologies should not rely only on the performance of low-level vision, but also on high-level knowledge and experiences. In this way, providing for a suitable knowledge representation is one of the main objectives of the work in [NM08]. Their work in knowledge representation, more specifically in the definition of the so called aggregates, is of interest to the DLI approach presented here. *Aggregates* are used to express the properties and constraints

which make a particular set of objects worth being recognized as a whole. They are defined as representational units for object configurations, occurrences and episodes, and consists of a set of parts tied together to form a concept that satisfies certain constraints.

A scene is composed of a set of occurrences, where each occurrence involves various objects. Occurrences have spatio-temporal configurations and objects have a final spatial configuration at the end of each occurrence and also at the end of a scene, such that a qualitative term can be used to describe a final spatial configuration. In [NM08], an example is provided about a table laying scene. A table laying scene has various occurrences, i.e., placing a plate, placing a saucer, followed by placing a cup and so on, such that the last occurrence of the scene has a final spatial configuration of objects called "cover". Thus, a scene has a temporal structure that describes a sequence of occurrences, and a spatial structure to describe the position of the objects in each occurrence (configuration). Furthermore, identity constraints are necessary to identify objects from different occurrences as the same individual, as well as qualitative constraints to define the order of the occurrences.

Given the previous characteristics of a scene, it is obvious that time plays an important role. But in text and image content a stationary state is present, imposing less requirements on knowledge representation. The conceptualization of aggregates, presented in [NM08], is of interest for the DLI approach and has been used for the definition of ontology and rule design patterns. As previously described in Section 2.4.2, the framework for the DLI process, presupposes SLI as being a relation-based system (see page 32). As such, SLI assumes the existence of more than one object such that the aim is to extract the attributes of the assumed objects and the relations between the attributes (configurations). In this way, the objective of DLI is to extract objects according to the attributes and relations between them. In this context the notion of an aggregate as described in [NM08] is used, more specifically the notion of object configurations.

Similar to the work of Neumann and Möller, a compositional and taxonomical hierarchy of aggregate concepts is used as the main structure for the background knowledge. Aggregate concepts are here called DLC concepts. To see examples of specific DLC concept definitions refer to Section 2.4.2 (page 33) and to read a description on DLC concepts, refer to Section 3 (page 3) on ontology design patterns.

The DLI process is different from the scene interpretation process in [NM08] given the absence of a temporal axis, even though similar notions of basic interpretation steps are also applied, e.g., aggregate instantiation, instance refinement and instance merging. In the DLI process, aggregate instantiation is driven by abduction where concrete criteria for hypothesis selection is provided. Instance refinement is also considered and specific rule and ontology patterns (see *Aggregate Specialization Pattern* on page 93 and *Specific*

Abduction Rules Pattern in page 94) are proposed for this purpose. Instance merging is used for fusion between individuals obtained from different types of content in what is called multimedia fusion. Finally, a major contribution of the DLI approach presented here is that, different to Hobbs et al., Shanahan, and Neumann and Möller, a set of design patterns that guide the knowledge engineer in the design of domain knowledge useful for its application on interpretation as abduction is proposed in Section 3.2.

After providing an overview on the related work about abduction and the use of DLs for the definition of content semantics, the following section proposes a formalization for Abox abduction which is at the hearth of the DLI process presented in Section 2.5.4.

2.5.2 Abox Abduction

Abduction is qualified in [EKS06] as an *ampliative* service since it provides for more knowledge than can be obtained deductively. For example, from the following knowledge base (Σ),

$$\mathcal{T}: \ C \sqsubseteq D,\ D \sqsubseteq E,\ H \sqsubseteq E,\ E \sqsubseteq F$$
$$\mathcal{A}: \emptyset$$

and a concept assertion $\gamma: \ i : E$.
The assertion $i : F$ is entailed $\Sigma \cup \gamma \models i : F$ due to deductive reasoning (in every model in which $i : E$ is true, $i : F$ is also true). In abductive reasoning, the assertion $\gamma: \ i : E$ might be explained through some hypothesis (Δ) such that $\Sigma \ \cup \ \Delta \ \models \gamma$ holds. For the previous knowledge base, various hypotheses such as $\Delta_1 = i : E$, $\Delta_2 = i : H$, $\Delta_3 = i : D$ and $\Delta_4 = i : C$ are possible, since for each of them $\Sigma \cup \Delta_i \models \gamma$ holds. As can be noticed from this example, criteria for hypothesis selection is required to reduce the number of possible explanations as much as possible.

Abduction is formalized in this work as a type of non-standard retrieval inference service. More formally, for a given Abox assertion γ (observation) and a knowledge base Σ, the abductive retrieval inference service aims at deriving all sets of Abox assertions Δ (explanations) such that the following holds

$$\Sigma \cup \Delta \models \gamma \tag{2.1}$$

In this way, the abductive retrieval inference service has the task of determining what should be added (hypothesized) to the knowledge base such that the formula above holds. Remind that $\Sigma = (\mathcal{O}, \mathcal{R}, A_x)$. As explained in Section 2.1.5, the Abox A_x of the knowledge base contains the interpretation results of a specific media object. Therefore, in the formula above γ corresponds to one assertion in A_x that should be explained. Determining which assertion in A_x should be explained is described in Section 2.5.4.

Moreover, various Δs can be obtained, therefore a strategy to choose preferred explanations should be developed. In this context, there is a set of constraints that should be considered as criteria in order to reduce the set of possible explanations to one that contains *preferred explanations*. The constraints are classified as *must* and *recommended*, and are suitable in the context of media interpretation.

- Must-constraints:
  - **Consistency:** $\Sigma \cup \Delta \nvDash \bot$
  - **Relevance:** $\Delta \nvDash \gamma$
- Recommended constraints:
  - **Simplicity:** $\inf(\text{poset}(\Delta s, \lambda(x,y) \bullet S(x) < S(y))),$ $S =\mid \{\alpha \in \Delta \mid \mathcal{O} \cup A_x \nvDash \alpha)\} \mid$
  - **Consilience:** $\sup(\text{poset}(\Delta s, \lambda(x,y) \bullet S(x) > S(y))),$ $S =\mid \{\alpha \in \Delta \mid \mathcal{O} \cup A_x \models \alpha)\} \mid$
  - **Informativeness:** Δ is preferred if there exists no other explanation Δ' such that $\Sigma \cup \Delta \vDash \gamma$ with $\Sigma = (\mathcal{O}, \mathcal{R}, \mathcal{A})$ and $\mathcal{O} \cup \Delta' \vDash \Delta$

Elsenbroich et al. [EKS06] provide a thorough discussion of abductive reasoning tasks in DLs including Abox abduction and provide also a set of constraints which overlap with the ones presented above. The overlap applies to consistency and relevance.

Consistency is a criterion that must be fulfilled and is application independent. Thus, according to classical logic, an inconsistent knowledge base (in this case $\Sigma \cup \Delta$) can entail anything. In other words, without satisfiability any explanation fulfills $\Sigma \cup \Delta \models \gamma$.

Relevance is considered as optional in [EKS06], but in this work it is proposed (similar to consistency) as a necessary and application-independent constraint. Relevance means that the observations (γ) only should not be the explanation (Δ) ($\Delta \nvDash \gamma$). An explanation that does not fulfill this constraint is irrelevant. For example, in a medical diagnosis scenario an irrelevant explanation is one that explains sneezing as symptom by the same sneezing as its cause.

Simplicity means that we want to minimize the number of assertions (α) in the explanation (Δ) that are not entailed in ($\mathcal{O} \cup A_x \nvDash \alpha$) and therefore are hypothesized.

From a poset of all possible explanations (Δ_s), ordered according to the criteria S, minimum Δs are preferred. In other words, the criteria S of simplicity is to prefer the explanation Δ with the least number of hypotheses necessary to explain γ.

This criterion is application-dependent. For example, for image interpretation, simplicity is recommended. If Figure 2.20[6] is explained as a *dinner table* or as a *dinner*

[6]The image is taken from [NM06]

Figure 2.20: Which is the best interpretation? A dinner table or a dinner table for two?

table for two, the first explanation is preferred since the second explanation needs to hypothesize a second set of object configurations. In medical diagnosis it might be more reasonable not to fulfill simplicity since one is interested in knowing all possible explanations including those in which the number of hypothesis are high, since discarding some explanations (causes) due to simplicity is risky.

Consilience in this work means that we want to maximize the number of assertions in the explanation ($\alpha \in \Delta$) that are entailed by $\mathcal{O} \cup A_x \models \alpha$.

In other words, we want to maximize the propositions from the background knowledge that are used to create an explanation Δ, in order to explain the highest number of observations γ. For example, in the context of scene interpretation the image in Figure 2.20 can be explained as a *western style dinning table* or as an *asian style dinning table* (where cutlery is not used). One would prefer western style dinning table as explanation since it considers cutlery, while the second explanation ignores the cutlery and only consider the plate and the coffee set in its explanation.

To explain ***informativeness*** consider the background knowledge presented at the beginning of this section (see page 49). From the following two explanations, $\Delta_1 = \{i : D\}$ and $\Delta_2 = \{i : C\}$, if $\mathcal{O} \cup \Delta_2 \models \Delta_1$ then Δ_2 is preferred since it is the more specific one, and there is no other Δ' such that $\mathcal{O} \cup \Delta' \models \Delta_2$. The more specific an explanation is, the more restrictions on the domain are imposed. This results in a more informative explanation. The less specific an explanation is, the less informative it is. A more generic information is more likely to be correct since less assumptions are imposed. In other words, we want to find the $\Delta \in \Delta$s that is subsumed by all the other explanations in Δs w.r.t. $\mathcal{O}$. For example, consider again the image in Figure 2.20. Provided a background knowledge base containing the following axioms $\mathcal{T} = \{DinnerTT \sqsubseteq TableTop,\ BreakfastTT \sqsubseteq TableTop\}$, and two interpretations Δ_1 and Δ_2. Δ_1 explains the observations as a dinner table ($DinnerTT$) and Δ_2 explains the observations as $TableTop$. Δ_1 provides more informa-

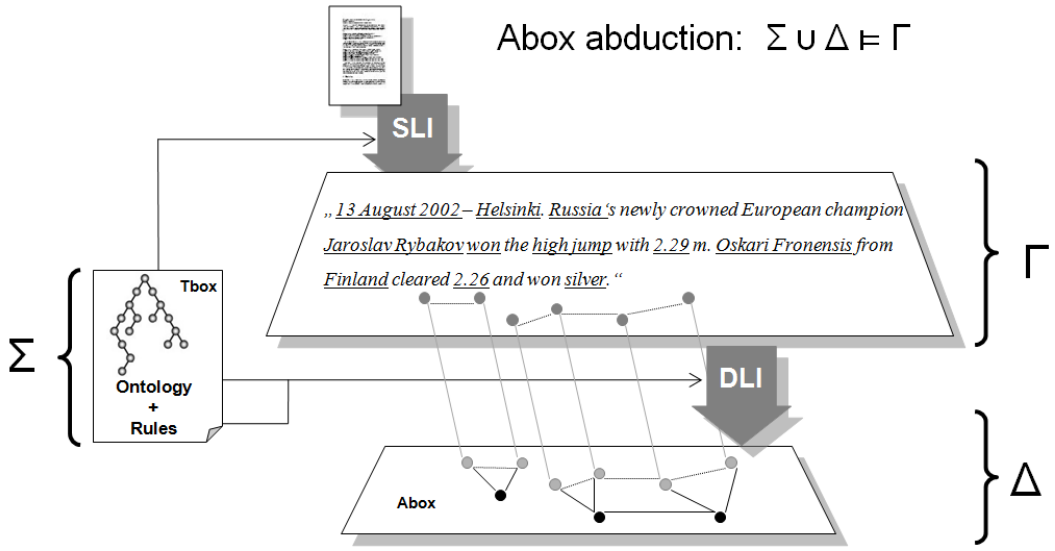


Figure 2.21: Formalizing deep-level interpretation of text as Abox abduction.

tion, while Δ_2 is more generic. Thus, informally speaking, Δ_2 has a higher probability of being correct than Δ_1 since less hypotheses are required. In [HSAM93], this is called the "informativeness-correctness trade off".

It should be noted that informativeness and simplicity compete with each other. Thus, in the example above, if one prefers simplicity, then Δ_1 is preferred since the hypothesis is minimal compared to Δ_2. Considering both criteria might seam a contradiction. For this reason a preference order for the application of criteria is required. In the implementation of the abductive reasoning service, the must-constraints are first applied, followed by simplicity and consilience to obtain a preference score, and finally, the criterion for "informativeness" is applied to further reduce the number of preferred explanations in case more than one preferred explanation exists.

In the context of the deep-level text interpretation framework (see Figure 2.21) the elements of the abduction formula are characterized as follows. Γ represents the information extracted from a text document through SLI processes. SLI processes formally encode their results as a set of Abox assertions, such that, for every word recognized in the text, a corresponding concept assertion γ is made explicit in Γ, and for every relation between words that is recognized in the text, a role assertion γ is added to Γ.

Δ represents the result of the abductive process performed by DLI, which extends the Abox obtained from SLI, with new concept and role assertions describing the content of the text document (expressed in Γ) at a deep-level. In this work the representation of deep-level content semantics is done by instantiating DLC concepts and relating such instances with SLC instances through SLR roles (see Section 2.4.2 for a short description of SLC, DLC and SLR).

In the process of Abox abuction the $\mathcal{O}$ and $\mathcal{R}$ part of the knowledge base are used as follows

- The set of rules $\mathcal{R}$ are used to define the space of possible explanations also called *abducibles*. The rules are practical mechanisms used to produce new assertions that will compose an explanation. Moreover, they provide representation means to express the configuration of an aggregate concept (to capture constraints among parts of aggregates) to cope with the expressivity limitations in the ontology part.
- The ontology $\mathcal{O}$ is formally represented through DLs. The $\mathcal{T}$ part is used to model aggregate concepts through GCIs and complex concept descriptions, called DLC concepts, and exploits reasoning mechanisms to ensure that explanations are consistent. Thus, the ontology helps in *reducing* the number or possible explanations that result from the application of rules via abduction.

2.5.3 The Abduction Algorithm

The abduction algorithm (see Algorithm 1, p. 54) is at the heart of the DLI process and has two main tasks, which are hypothesis construction and hypothesis selection. The main function, called *compute_explanations*, is in charge of finding explanations for an observation (γ). This function requires the following elements as input: a domain ontology $\mathcal{O}$, a set of rules $\mathcal{R}$, an Abox A_x associated to the document object d_x being interpreted, an Abox assertion γ representing the observation that should be explained, an Abox Γ_2 with the rest of the observations obtained by SLI, and a preference function Ω.

As will be explained in Section 2.5.4, DLI distinguishes between *bona fide* (Γ_1) and *fiat* (Γ_2) observations from Γ. Observations in Γ_2 should be explained. Distinguishing between bona fide and fiat observations is a design decision. This design decision is specified by means of rule heads in $\mathcal{R}$ which define the space of abducibles. A_x contains Γ_1.

The preference function Ω is used to configure the ampliative capabilities of the abduction process, i.e., to reduce (or enlarge) the number of possible explanations by influencing the variable substitution function σ. This will be explained later. As an example, consider the following input

$\mathcal{O}$ = The AEO ontology from Appendix A (see page 145)

R = The rules in Appendix B (see page 149)

A_x = all assertions from Figure 2.17 (page 40)

$\gamma = (hjName_1, city_1) : sportsNameToCityName$

Γ_2 = all assertions from Figure 2.19 (page 41) minus γ

Ω = :reuse-old :one-new-ind

Algorithm 1 The Abduction Algorithm

function compute_explanations($\mathcal{O}, \mathcal{R}, A_x, \gamma, \Gamma_2, \Omega$)
$\gamma' := transform\_into\_query(\gamma)$, //where $\gamma = (\underline{z}) : P$ and $\gamma' = P(\underline{z})$
$\Theta s := \{\{Q_1(\sigma(\underline{Y_1})), \ldots, Q_n(\sigma(\underline{Y_n}))\} \mid \exists r \in \mathcal{R}$ with $r := P(\underline{X}) \leftarrow Q_1(\underline{Y_1}), \ldots, Q_n(\underline{Y_n})$,
$\exists \sigma$ such that $P(\sigma(\underline{X})) = \gamma'\}$
$\Delta s := \{\Delta \mid \exists \Theta \in \Theta s$,
$individuals = \{inds(\Gamma_2) \cup inds(A_x) \cup inds(\gamma) \cup \{new_i \mid i = \{1, \ldots, n\}, n = |vars(\Theta)|\}\}$,
$\exists \sigma \in get\_substitutions(vars(\Theta), \Omega, individuals)$ such that
$\Delta = \{Q_1(\sigma(\underline{Y_1})), \ldots, Q_n(\sigma(\underline{Y_n}))\}, \Sigma \cup transform(\Delta) \cup A_x \nvDash \bot\}$
$\Delta s_1 := \sup(\text{poset}(\Delta s, \lambda(x, y) \bullet S(x, \mathcal{O}, A_x) < S(y, \mathcal{O}, A_x)))$
$\Delta s_2 := \{\}$
for all $\Delta \in \Delta s_1$ **do**
  $\Delta' := \{\}$
  for all $\alpha \in \Delta$ **do**
    if $\{() \mid \alpha\}$ is false w.r.t. $\mathcal{O}$ **then**
      $\Delta' := \Delta' \cup transform(\alpha)$
    end if
  end for
  $\Delta s_2 := \Delta s_2 \cup \{\Delta'\}$
end for
$\Delta s_3 := \{\Delta \in \Delta s_2 \mid \neg\exists \Delta' \in \Delta s_2$ such that $\mathcal{O} \cup \Delta' \models \Delta\}$
return Δs_3

function S($\Delta, \mathcal{O}, A_x$)
$S_c =| \{\alpha \in \Delta \mid \mathcal{O} \cup A_x \vDash \alpha\} |$
$S_s =| \{\alpha \in \Delta \mid \mathcal{O} \cup A_x \nvDash \alpha\} |$
$S = S_c - S_s$
return S

The function *transform_into_query* is applied to the Abox assertion γ and returns the corresponding query atom of the form $P(\underline{z})$.

$$\gamma' := sportsNameToCityName(hjName_1, city_1)$$

The next step (see line 3 of Algorithm 1, page 54) is to identify rules that apply in a backward way, i.e., from the head ($P(\underline{X})$) to the body of a rule ($Q_1(\underline{Y_1}), \ldots, Q_n(\underline{Y_n})$). They are identified if there exists a substitution ($P(\sigma(\underline{X}))$) that matches γ'.

For this example, and according to the set of rules in Appendix B (page 149), three matching heads are found, such that body substitutions are obtained for each rule. The result is a set Θs of atom sets Θ as follows

$$\Theta s := \{\{HighJumpCompetition(Z),$$
$$hasName(Z, hjName_1),\ HighJumpName(hjName_1),$$
$$takesPlace(Z, city_1),\ CityName(city_1)\}$$

$$\{PoleVaultCompetition(Z),$$
$$hasName(Z, hjName_1),\ PoleVaultName(hjName_1),$$
$$takesPlace(Z, city_1),\ CityName(city_1)\}$$

$$\{SportsCompetition(Z),$$
$$hasName(Z, hjName_1),\ SportsName(hjName_1),$$
$$takesPlace(Z, city_1),\ CityName(city_1)\}\ \}$$

The next step (see line 4) is to find substitutions for the variables in each $\Theta \in \Theta s$ to obtain a corresponding Δ. For this purpose the function *get_substitutions* is used. It gets as input the variables from Θ, obtained from a call to the function *vars*, a set of individuals composed of all individuals in the knowledge base, i.e, individuals from Γ_2, A_x and γ, obtained with the help of the function *inds*, and a set of freshly created individuals for each variable in Θ ($\mid vars(\Theta) \mid$), and finally the function Ω.

The function Ω is used to set a priority over the elements of the set *individuals* used by the substitution function σ. In this example the preference function (Ω**:** :reuse-old :one-new-ind[7]) prioritizes the use of individuals that already exist in the knowledge base (individuals from Γ_2, A_x, γ) over the newly created individuals, as long as such a substitution is consistent w.r.t. to the background knowledge ($\Sigma \cup \sigma(\underline{Y}) : Q \not\models \bot$). In

[7]In RacerPro, the preference function Ω can be configured by using the following commands: ":c-mode (:reuse-old :one-new-ind)" for concepts, and ":r-mode (:reuse-old :one-new-ind) " for roles. ":reuse-old" refers to the use of existing assertions and has a higher priority than ":one-new-ind" that allows to create a new individual.

this way, the result of *get_substitutions* is a set of Δ containing a set of substitutions $\{Q_1(\sigma(\underline{Y_1})), \ldots, Q_n(\sigma(\underline{Y_n}))\}$, such that once they are transformed into assertions of the form $\{(\underline{z}_1) : Q_1, \ldots, (\underline{z}_n) : Q_n\}$, with the help of the function $transform$, then $\Sigma \cup transform(\Delta) \cup A_x \not\models \bot$ should hold. In this way, the set Δs has only explanations Δ wich are consistent w.r.t. the knowledge base.

In the running example, all individuals in Figure 2.17 (page 40) are used in the substitution process. But all substitutions result in Aboxes which are inconsistent with respect to the Tbox due to disjointness axioms ($DLC \sqsubseteq \neg SLC$). For this reason a freshly created individual (new_1) is used for substitution, such that the following Δs is created

$\Delta_s = \{$
$\Delta_1 = \{new_1 : HighJumpCompetition,$
$(new_1, hjName_1) : hasName,\ hjName_1 : HighJumpName,$
$(new_1, city_1) : takesPlace,\ city_1 : CityName\}$

$\Delta_2 = \{new_1 : SportsCompetition,$
$(new_1, hjName_1) : hasName,\ hjName_1 : SportsName,$
$(new_1, city_1) : takesPlace,\ city_1 : CityName\}$
$\}$

Note that the second set of atoms in this example (see second element of Θs in page 55) produces the following explanation which is inconsistent w.r.t. Σ

$\Delta = \{new_1 : PoleVaultCompetition,$
$(new_1, hjName_1) : hasName,\ hjName_1 : PoleVaultName,$
$(new_1, city_1) : takesPlace,\ city_1 : CityName\}$

The universal restriction over the role $hasName$, where the range should be an instance of type $PoleVaultName$, enforces $hjName_1$ to be of type $PoleVaultName$ which produces an inconsistency due to disjointness axioms in the Tbox

$$\begin{aligned} PoleVaultCompetition &\sqsubseteq SportsCompetition \\ &\quad \sqcap \forall hasPart.PoleVaultRound \\ &\quad \sqcap \forall hasName.PoleVaultName \\ HighJumpName &\sqsubseteq SportsName \sqcap \neg PoleVaultName \end{aligned}$$

A Δ that is inconsistent w.r.t. Σ is discarded. In this way, reasoning services and Σ are used to restrict the number of explanations. This helps in keeping the *consistency constraint* introduced before (see page 50). Moreover, this example shows that disjointness axioms in the Tbox are relevant during the variable substitution process.

Continuing with the explanation of the abduction algorithm, line 5 shows how to obtain preferred explanations. The set of explanations in Δ_s is transformed into a poset and all explanations with a score lower than the highest score found in the poset are discarded. The function S computes a score ($S := S_c - S_s$) that meets the criteria of simplicity and consilience following the definitions described in Section 2.5.2. S_c represents the number of assertions in Δ that are entailed in $\mathcal{O} \cup A_x$, and therefore are not hypothesized. S_s is the number of hypothesized assertions in Δ. The score formula can be paraphrased as follows, *the less hypothesized assertions an explanation contains (simplicity, i.e., S_s) and the more bona fide assertions a theory (Δ) explains (consilience, i.e., S_c), the higher its preference score will be.* In this example, the following scores are obtained

$$S_c(\Delta_1) = 2,\ S_s(\Delta_1) = 3,\ S(\Delta_1) = -1$$
$$S_c(\Delta_2) = 2,\ S_s(\Delta_2) = 3,\ S(\Delta_2) = -1$$

In this example, both explanations have the highest score such that the set Δs_2 has more than one explanation Δ. The following steps, from line 6 until line 15, reduces the size of each Δ in Δs_2 to obtain minimum explanations, i.e., containing only assertions that are hypothesized, such that $\{\alpha \in \Delta \mid \{() \mid \alpha\}$ is false w.r.t. $\mathcal{O}\}$. Continuing with the example, Δs_2 is as follows

$$\Delta s_2 = \{$$
$$\Delta_1 = \{new_1 : HighJumpCompetition,$$
$$(new_1, hjName_1) : hasName,\ (new_1, city_1) : takesPlace\}$$

$$\Delta_2 = \{new_1 : SportsCompetition,$$
$$(new_1, hjName_1) : hasName,\ (new_1, city_1) : takesPlace\}$$
$$\}$$

Finally, the Δs in Δs_2 are compared for entailment, such that the most specific Δ(s) are preferred. The following entailment relation exists between Δ_1 and Δ_2: $\Sigma \cup \Delta_1 \models \Delta_2$. The abductive retrieval inference service returns Δ_1 as the most preferred explanation, since it provides more information than Δ_2.

return:
$$\Delta_1 = \{new_1 : HighJumpCompetition,$$
$$(new_1, hjName_1) : hasName,\ (new_1, city_1) : takesPlace\}$$

With the previous example, we explained the Abox abduction algorithm. In the following section we are now able to explain the DLI process which relies on Abox abduction.

2.5.4 The Interpretation Process

DLI is a process that iterates between abductive and deductive reasoning in order to explain a set of observations extracted from media by SLI.

DLI can be adapted to the application needs in two different ways. First, it can be adapted to determine which observations require an explanation, called *fiats* and represented with Γ_2, and which observations are believed to be true by default, called *bona fide* and represented with Γ_1. Thus, the set of observations (Γ) provided by SLI processes is divided into two sets Γ_1 and Γ_2. Second, DLI can be adapted to determine which criteria for hypothesis selection should be applied. These two aspects are represented with two configuration functions, called *trustiness* (Ψ) and *preference* (Ω) respectively. The configuration of trustiness (Ψ) represents a "contract" that is specified through the set of rules $\mathcal{R}$. $\mathcal{R}$ defines the space of possible explanations. In other words, the space of abducibles. As described in Section 3.2, on ontology and rule design patterns, the design of the ontology and rules should be grounded on media content, and, according to the notion of interpretation inspired by [NM06, NM08, MN08], fiat observations are those assertions that represent configurations of objects.

To recall, in [NM06, NM08, MN08], aggregates are used to express the properties and constraints which make a particular set of objects worth being recognized as a whole. In this way aggregates are representational units that help to explain object configurations (relations between the objects) w.r.t. a domain of knowledge. For example, for the object configuration shown in Figure 2.20 (see page 51) the keyword *table top* can be used as an explanation, such that the configuration "makes sense" w.r.t. to the domain of table settings. Thus, that a fork is on one side of a plate and a knife on the other side of the same plate is explained given that they are part of a *table top*.

Provided this notion of aggregates, the expected configuration of trustiness is to consider role assertions as fiat assertions. Concept assertions, on the other hand are considered as bona fide assertions in this specific application.

Nevertheless, there can be specific reasons to consider concept assertions as fiats as well. This is influenced by the type of media content that is to be interpreted. For example, a textual content with an official-informative character such as news reports, clinical reports, etc., one assumes that if the name of a person is explicit in the text (observed), then there is sufficient evidence to explain the existence of a person and there is no need to explain any relations with other observations, such that a rule that applies in a backward-chaining way can be defined that explains the person name with two assertions, e.g., the instance of a person and a role "has name" that relates the instance of a person, with the instance of the person name observed. In this way, trustiness (Ψ)

is a function that considers every observation $\gamma \in \Gamma_2$ as fiat provided a rule head whose predicate matches the predicate of $\gamma \in \Gamma_2$.

Algorithm 2 The Deep-Level Interpretation (DLI) Algorithm

1: **function** interpret($\mathcal{O}, \mathcal{R}, A_x, \Gamma_0, \Omega, \Psi$)
2: Δs:= {}
3: $\Gamma_2 := \Psi(\Gamma_0, \mathcal{R})$
4: $\Gamma_1 := \Gamma_0 \setminus \Gamma_2$
5: $A_x := A_x \cup \Gamma_1$
6: **while** $\Gamma_2 \neq \emptyset$ **do**
7: $\gamma := select(\Gamma_2, \Omega)$
8: $A_x := A_x \cup \{\gamma\}$
9: $\Gamma_2 := \Gamma_2 \setminus \{\gamma\}$
10: Δs := compute_explanations($\mathcal{O}, \mathcal{R}, A_x, \gamma, \Gamma_2, \Omega$)
11: **for all** $\Delta \in \Delta$s **do**
12: $\Gamma_2 := \Gamma_2 \cup$ apply($\mathcal{O}, \mathcal{R}, \Delta \cup A_x$)
13: **if** $\mathcal{O} \cup \Gamma_2 \cup \Delta \cup A_x \nvDash \bot$ **then**
14: interpret($\mathcal{O}, \mathcal{R}, \Delta \cup A_x, \Gamma_2 \cup \Psi(\Delta, \mathcal{R}), \Omega, \Psi$)
15: **end if**
16: **end for**
17: **end while**
18: **return** A_x

The preference function (Ω), as introduced before in Section 2.5.3, configures the abduction algorithm in two respects. First, it is used in the variable substitution process, where priorities over substitution candidates are established. Second, it is used to determine which constraints should be fulfilled to select preferred explanations. In this case the recommended constraints introduced in Section 2.5.2 (see page 2.5.2) can be manipulated. Having introduced both configuration functions (Ψ and Ω), the interpretation algorithm (see Algorithm 2) is now explained together with its application in the running example.

The main function, named *interpret*, is called for the first time whenever there are SLI results for a specific media object d_x, such that there is an Abox, here called Γ_0, containing the results of SLI. The function *interpret* gets as input the following elements: a domain ontology $\mathcal{O}$ (see Appendix A), a set of rules $\mathcal{R}$ (see Appendix B), an empty Abox A_x which is associated to the media object d_x being interpreted, an Abox Γ_0 containing the observations from d_x obtained by SLI, a preference function Ω, and the trustiness function Ψ.

For the step-by-step example consider the assertions in Figure 2.17 (page 40) and Figure 2.19 (page 41) as content for Γ_0. The first step is to divide Γ_0 into bona fide and fiat observations (lines 3 and 4). For this, trustiness Ψ is applied over the observations Γ_0, such that Γ_2 is composed of all assertions in Γ_0 whose predicate matches a predicate

from a rule's head in $\mathcal{R}$. Γ_1 contains all the assertions in Γ_0 minus Γ_2. Afterwards, Γ_1 is added to the Abox A_x (see line 5). The following example shows the division of Γ_0.

$\Gamma_1 = \{$ $date_1$:*StartDate*, ($date_1$, "13 August2002"):*hasValue*,
$country_1$:*CountryName*, ($country_1$, "Russia"):*hasValue*,
$hjName_1$:*HighJumpName*, ($hjName_1$, "high jump"):*hasValue*,
$perf_2$:*Performance*, ($perf_2$, "2.26"):*hasValue*,
$country_2$:*CountryName*, ($country_2$, "Finland"):*hasValue*,
$city_1$:*CityName*, ($city_1$, "Helsinki"):*hasValue*,
$pName_1$:*PersonName*, ($pName_1$, "Jaroslav Rybakov"):*hasValue*,
$perf_1$:*Performance*, ($perf_1$, "2.29"):*hasValue*,
$pName_2$:*PersonName*, ($pName_2$, "Oskari Fronensis"):*hasValue*,
$rank_1$:*Ranking*, ($rank_1$, "won"):*hasValue*,
$rank_2$:*Ranking*, ($rank_2$, "silver"):*hasValue*,
($pName_1$, $perf_1$):*personNameToPerformance*,
($pName_2$, $perf_2$):*personNameToPerformance*,
($hjName_1$, $perf_1$):*sportsNameToPerformance*,
($hjName_1$, $perf_2$):*sportsNameToPerformance*$\}$

$A_x = A_x \cup \Gamma_1$

$\gamma_1 = (hjName_1, city_1)$:*sportsNameToCityName*
$\gamma_2 = (pName_1, country_1)$:*personNameToCountryName*
$\gamma_3 = (pName_2, country_2)$:*personNameToCountryName*
$\gamma_4 = (hjName_1, date_1)$:*sportsNameToDate*

$\Gamma_2 = \{\gamma_1, \gamma_2, \gamma_3, \gamma_4\}$

The aim of the DLI process is to explain each $\gamma \in \Gamma_2$. Notice that before applying the function *compute_explanations* for a given γ, the assertion γ is transferred from the set Γ_2 to A_x. Hence, the interpretation process terminates when there are no more fiats to explain ($\Gamma_2 = \emptyset$).

The set of explanations Δs, for a given γ, obtained after applying the abduction function (see *compute_explanations* in line 10 and algorithm 1 in page 54) can be empty given the absence of applicable rules in $\mathcal{R}$ that produce a consistent explanation. Moreover, if explanations are found, then every $\Delta \in \Delta$s also complies with the recommended constraints given that Ω is used.

Once explanations for a given γ are obtained (Δs $\neq \emptyset$), then deductive and abductive reasoning is applied to each $\Delta \in \Delta$s. Deductive reasoning is executed in the function

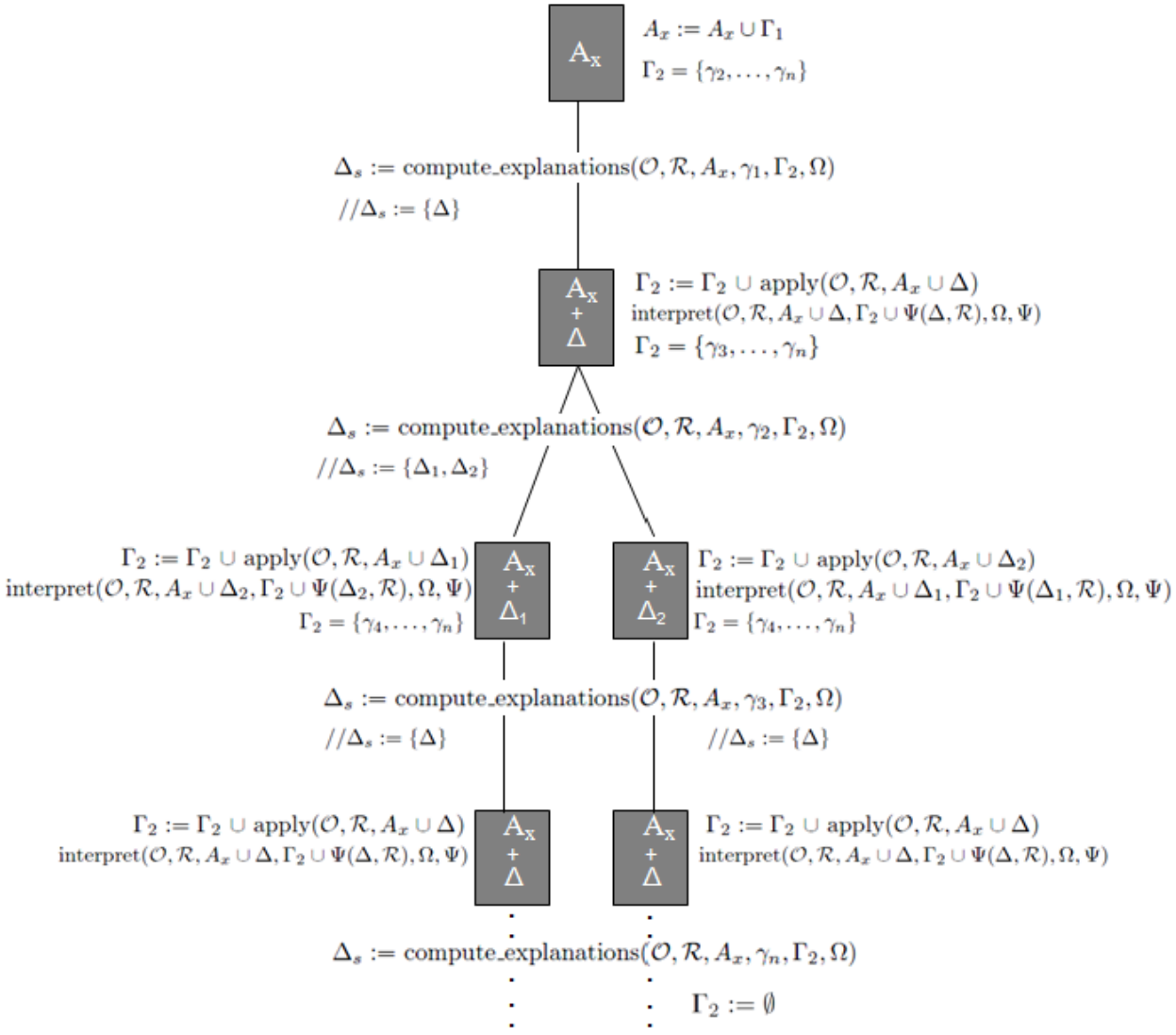


Figure 2.22: Graphical representation of a run of the interpretation algorithm.

apply, in order to add new assertions to the set of fiat assertions (Γ_2) provided the new assertions from Δ. Abductive reasoning is applied in the function *compute_explanations* (see Algorithm 1, page 54) for every recursive call of the function *interpret*. A graphic representation of this algorithm is presented in Figure 2.22.

As described in Section 3.2, the design of the set $\mathcal{R}$ distinguishes[8] between so called *abduction rules* (see page 89) and *deduction rules* (see page 92), such that only deduction rules are applied in a forward-chained way with the function *apply*.

If more than one explanation is found, a branch for each $\Delta \in \Delta s$ is created, resulting in a set of Aboxes. Each Abox is composed of $A_x = A_x \cup \Delta$. Furthermore, a different Γ_2 is

[8]In RacerPro it is possible to use options to distinguish between rules that apply in a forward-chaining and in a backward-chaining way. The option ":forward-rule-p nil" specifies that a rule applies in a backward-chaining way, and the command ":backward-rule-p nil" specifies that a rule applies in a forward-chaining way.

expected provided the different Δ_s after the function *apply* is called on the corresponding $A_x \cup \Delta$. Afterwards, a new recursive call to the function *interpret* is done. This process continues until Γ_2 is empty in each branch. To continue with the interpretation example, consider γ_1

$$\gamma_1 := (hjName_1, city_1) : sportsNameToCityName$$

for which the call to function *compute_explanations* results in the following

$$\Delta_s := \{\Delta_1 = \{new_1 : HighJumpCompetition,$$
$$(new_1, hjName_1) : hasName,\ hjName_1 : HighJumpName,$$
$$(new_1, city_1) : takesPlace,\ city_1 : CityName\}\}$$

(already introduced in Section 2.5.3 on abduction). This explanation is added to the Abox A_x. Applying the rules with the function *apply* to the newly extended Abox, results into two new assertions such that Γ_2 looks as follows:

$$\gamma_2 = (pName_1, country_1) : personNameToCountryName$$
$$\gamma_3 = (pName_2, country_2) : personNameToCountryName$$
$$\gamma_4 = (hjName_1, date_1) : sportsNameToDate$$
$$\gamma_5 = (new_1, perf_1) : sportsCompetitionToPerformance$$
$$\gamma_6 = (new_1, perf_2) : sportsCompetitionToPerformance$$

$$\Gamma_2 = \{\gamma_2, \gamma_3, \gamma_4, \gamma_5, \gamma_6\}$$

A new call to the function *interpret* is done to explain a second $\gamma \in \Gamma_2$, here called γ_2. According to the set $\mathcal{R}$ (see Appendix B) only one rule is applicable suth that the call to the function *compute_explanations* produces the following explanation:

$$\Delta_s = \{\Delta_1 = \{new_2 : Person,$$
$$(new_2, pName_1) : hasName,\ pName_1 : PersonName,$$
$$(new_2, country_1) : hasNationality,\ country_1 : CountryName\}\}$$

After adding this explanation to the Abox, applying the rules, with function *apply*, creates a new fiat assertion; $\gamma_7 = (new_2, perf_1) : personToPerformance$. A new call to *interpret* is done and now γ_3 is explained:

$$\Delta_s = \{\Delta_1 = \{new_3 : Person,$$
$$(new_3, pName_2) : hasName,\ pName_2 : PersonName,$$
$$(new_3, country_2) : hasNationality,\ country_2 : CountryName\}\}$$

Notice that due to number restrictions on the role $hasName$ in the description of the concept $Person$ (see Appendix A, page 145), an instance of person can have at most one role $hasName$. Thus, without these restrictions, the abduction algorithm with the default configuration (Ω) would have used the assertions of the Abox such as $new_2 : Person$ to explain γ_3 without having to hypothesize a new individual. After adding this explanation to the Abox, applying the rules creates a new fiat assertion; $\gamma_8 = (new_3, perf_2) : personToPerformance$. In the next iteration γ_4 (see page 62) is explained

$$\Delta_s = \{\Delta_1 = \{new_1 : HighJumpCompetition,$$
$$(new_1, hjName_1) : hasName,\ hjName_1 : HighJumpName,$$
$$(new_1, date_1) : hasDate,\ date_1 : Date\}\}$$

Note that existing assertions from the Abox are used, thus according to Ω the set individuals that already exist are prioritized over the set of new individuals (new_i) during substitution (see line 4 of Algorithm 1, page 54). The same process continues for γ_5, γ_6, γ_7 and γ_8 until Γ_2 is empty. The result is an Abox containing both SLI (Γ_1 and Γ_2) as well as DLI results as follows

$$A_x = \Gamma_1 \cup \Gamma_2 \cup$$
$$\{new_1 : HighJumpCompetition, (new_1, hjName_1) : hasName,$$
$$(new_1, city_1) : takesPlace, (new_1, date_1) : hasDate,$$
$$new_2 : Person, (new_2, pName_1) : hasName,$$
$$(new_2, country_1) : hasNationality,$$
$$new_3 : Person, (new_3, pName_2) : hasName,$$
$$(new_3, country_2) : hasNationality,$$
$$new_4 : HighJumpRound, (new_1, new_4) : hasPart,$$
$$(new_4, new_5) : hasPart, new_5 : HighJump,$$
$$(new_5, perf_1) : hasPerformance, new_5 : SportsTrial,$$
$$(new_5, new_2) : hasParticipant, (new_5, perf_1) : hasPerformance,$$
$$new_6 : SportsTrial, (new_6, new_3) : hasParticipant,$$
$$(new_6, perf_2) : hasPerformance, new_3 : Athlete,$$
$$(new_4, new_6) : hasPart, new_6 : HighJump\}$$

A graphical display of the Abox extracted by *interpretation* is presented in Figure 2.23 (see page 64). In this figure it can be observed that individual new_3 is not only an instance of $Person$, but also an instance of $HighJumper$. This is implicit by the range restriction over the role $hasParticipant$ in the definition of a high jump trial in the Tbox.

The result of applying DLI to a text document is an Abox (or a set of Aboxes depending on the number of interpretations found, see the branches in Figure 2.22 on page 61), hereafter called *interpretation Abox*. This means that for every document object that is analyzed (d_x), a corresponding interpretation Abox A_x (or set of Aboxes $A_{xi}, i = \{1, \ldots, n\}$

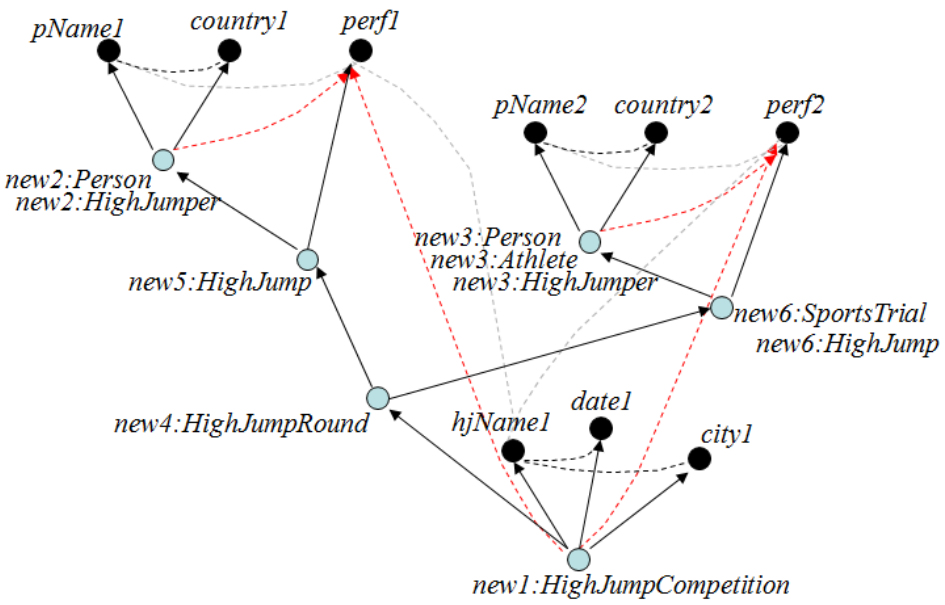


Figure 2.23: Graphical representation of DLI's results for the text in Figure 2.8 (page 28).

) is found, which contains results from SLI and DLI processes. In order to associate the interpretation Abox A_x with the corresponding document object d_x, a content management system is required which provides for unique identifiers. In this way, provided a unique identifier for a document object, a structure is required that associates the unique identifier with the corresponding interpretation Abox. Ideally, such a structure has also the form of Abox assertions (see Figure 2.24), with the advantage that a system that queries over interpretation Aboxes, can use the same query language to obtain the unique identifier of the related document object for its later retrieval. Moreover, in order to associate specific SLI results with media segments, reference annotations are used. As described in Section 2.4.2 (see page 33), reference annotations are stored in a relational structure associating them to the corresponding segments of data by spans that identify start and end offsets. Furthermore, classifying media segments w.r.t. *content structural aspects* is advantageous for further processing for example, to support multimedia fusion (as is described in section 2.6). The research of content structural aspects, e.g., MPEG-7, is beyond this work, but the work proposed by Dasiopoulou et al. in [DDS+07] is used in the DLI framework. More specifically, the MCO (Multimedia Content Ontology)[9] ontology is used to represent the content structural aspects of a media object and relate them to SLI and DLI results such that reference annotations are created.

[9]Downloadable from: http://www.boemie.org/ontologies

The MCO ontology models knowledge about multimedia content structure, which encompasses the representation of:

- Different types of media, e.g., text, image, web pages, etc.
- Logical relations among types of media, e.g., a web page contains text and two captioned images.
- Decomposition of media into segments, e.g., text is decomposed into phrases, which in turn are decomposed into words and characters.
- Relations between media segments and domain semantics, e.g., segments of a text with SLI and DLI results.

In this way, as Figure 2.24 graphically shows, reference annotations have the form of Abox assertions such that a so called *content structure Abox* is created for a media object and associated to the corresponding interpretation Abox. In contrast to SLI which is necessarily a media-dependent process, DLI is a media-independent process that can be used for the interpretation of text and image as long as the input meets the requirements represented in Figure 2.21 (see page 52). In this way, DLI is convenient for the interpretation of multimedia document objects, where typically different types of media complement each other to give a richer source of information for a given domain of interest. With the use of a content structure Abox, consistent w.r.t. the MCO ontology, multimedia fusion can be formalized as a logic-based process as well. In the following section we show how content structure information is useful to enable the process of multimedia fusion.

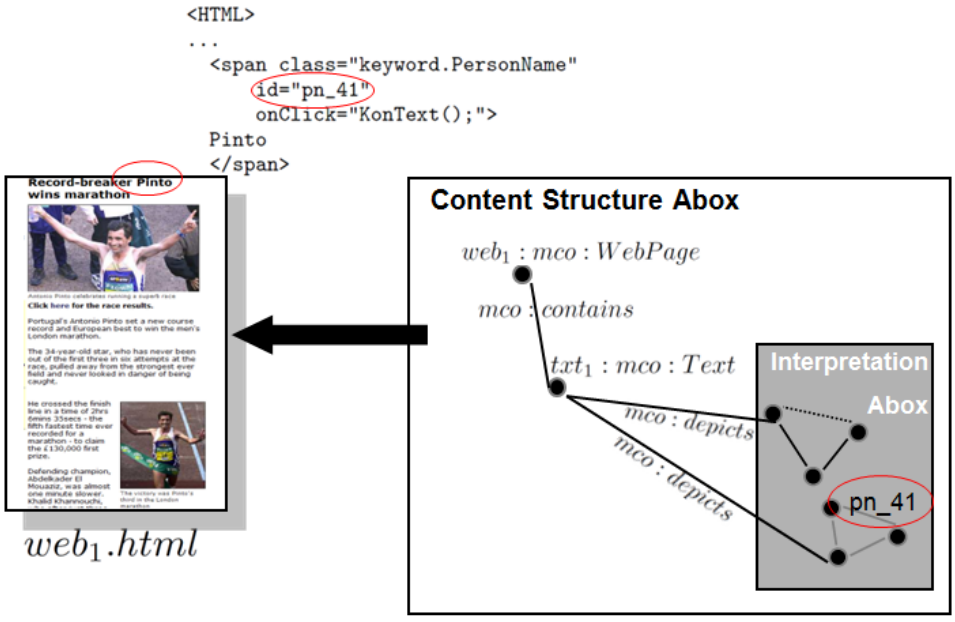


Figure 2.24: DLI results in relation with the document object.

2.6 Logic-based Multimedia Fusion

A multimedia document conveys information through different types of media such as text, image, video or audio. Since the objective of a multimedia object is to illustrate the domain of interest in a richer way than a single modality would possibly do, it is expected that the information conveyed by each media, in a multimedia document, is complementary and therefore related to each other. In this context, the purpose of multimedia fusion is to find interrelations between the content of different types of media and make them explicit through relational structures, more specifically, through logical same-as assertions. For this purpose the results of DLI are useful. The purpose of this section is to highlight the necessity of DLI to support multimedia fusion by arguing that DLI results provide for the necessary abstract level of information which puts observations from different media in a common context that facilitates fusion. In this way fusion is applied on the basis of DLI results and not on observations directly.

In the DLI framework presented here, the information extracted from each media object, by modality-specific SLI processes, can be interpreted w.r.t. a domain-specific background knowledge by means of the DLI process. In this context, the so-called *multimedia interpretation Abox* contains SLI and DLI results for each type of media found in a document, as well as information about the document's content structure. Figure 2.25 gives an example of a multimedia interpretation Abox for a web page. The web page contains a captioned image, such that there is an instance of *WebPage* that is related to an instance of *CaptionedImage* through a role called *contains*. The captioned image is composed of an image and a caption, such that an instance for each of these elements is also found in the Abox. To relate instances of content structure concepts, with instances of domain concepts (DLI instances), the role *depicts* is used. Multimedia fusion provides the following advantages:

- supports information gain of media specific interpretation results, and
- supports ambiguity resolution of media specific interpretation results.

In the following two sections each of these advantages is described to continue in Section 2.6.3 with the description of the fusion algorithm proposed in this work.

2.6.1 Information Gain

Information gain from media specific interpretation results is possible most of the time due to the fact that different types of media contain information of a different nature. For example, facts can be obtained from textual content such as person names, dates, age, etc., and it is unlikely that these facts can be obtained from visual content, with the

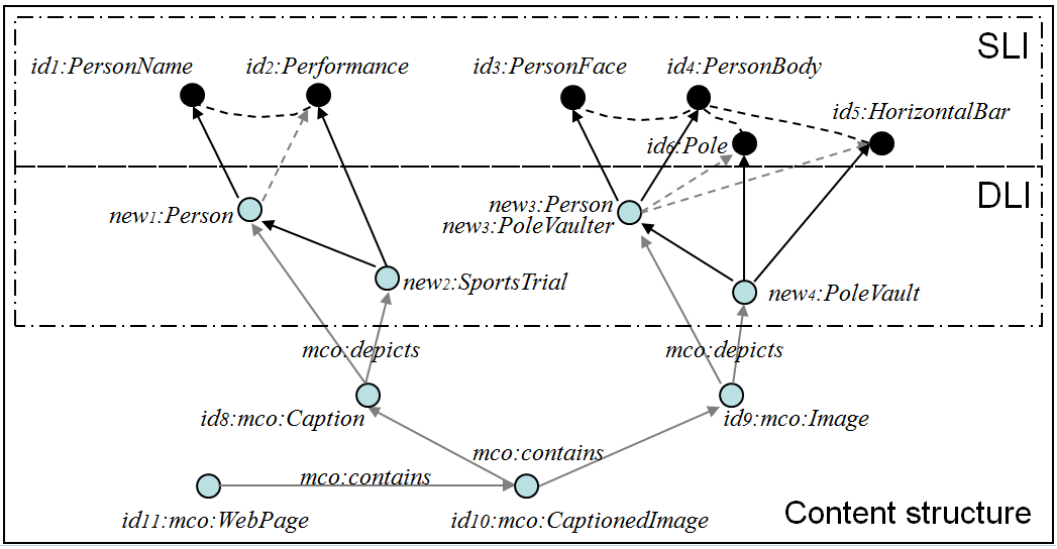


Figure 2.25: Multimedia interpretation Abox.

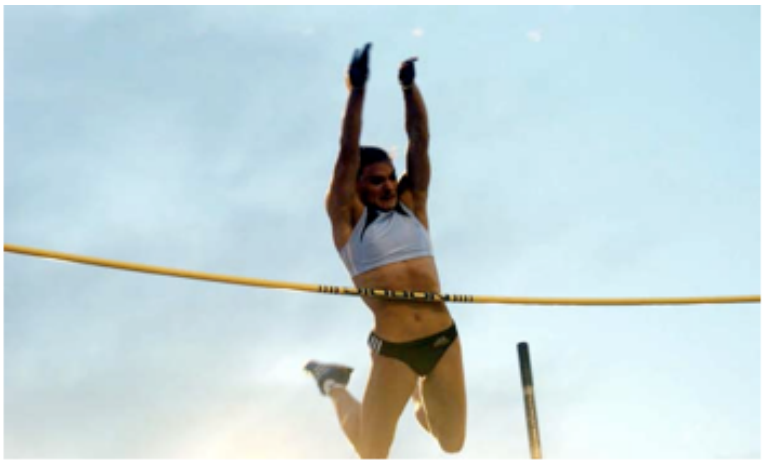


Figure 2.26: Image interpretation contributes to text interpretation.

exception of OCR (Optical Character Recognition) being applied. Extracting a person name from an image would require highly advanced mechanisms involving for example, face recognition. On the other hand, visual content depicts objects that might not be described in the text. From the eyes of a non-expert in athletics, the text in Figure 2.26 gives facts about a person's name and a measure that is assumed to be a performance measure due to the textual context. But still it is not possible to know for sure what is the exact sport in which that performance can be obtained. By observing the image and the captioned text, it is clear that the performance is related to a pole vault trial, and from the text the name of the person in the image can be obtained. Thus, both types of information are complementary, and when considering them separately they provide only limited information.

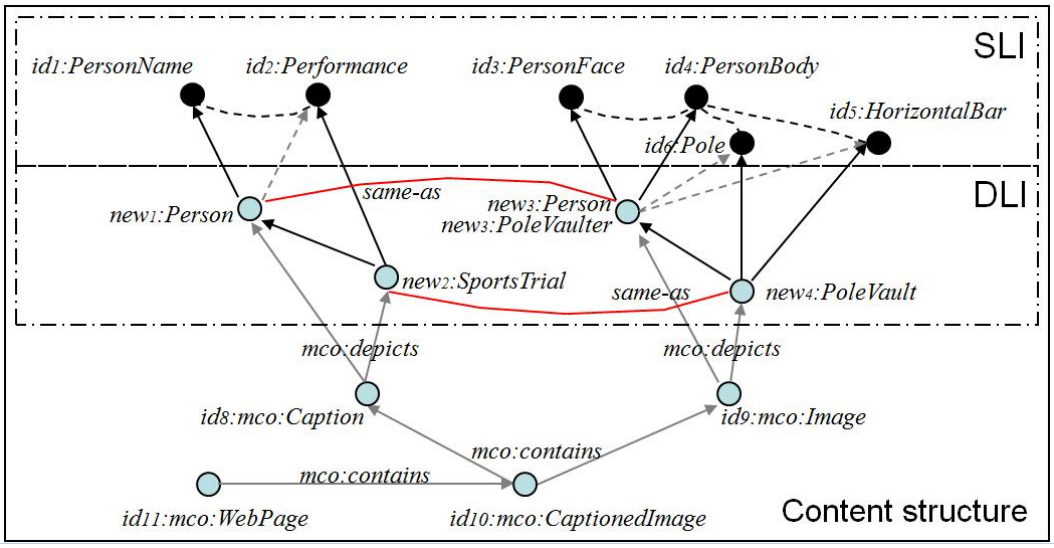


Figure 2.27: Graphical representation of a fused interpretation Abox.

Finding the correlations between the two types of media objects is trivial for a human being, but for a logic-based system, only symbolic representations are used. As illustrated in Figure 2.25, the results of SLI provide for individuals of concepts which are disjoint according to the Tbox in Appendix A (see page 145). Thus, individuals extracted from different media have a different nature. For example, facts, such as person names, are obtained from text content, and objects, such as person's body, are obtained from image content. In the Tbox the concepts $PersonName$ and $PersonBody$ are disjoint. In this way, *same-as* relations between individuals of disjoint concepts make an Abox inconsistent w.r.t. to the Tbox. But from the perspective of DLI results, individuals of more abstract concepts are obtained, for example, individuals of $Person$. Thus, an individual of $Person$ extracted from image can be in a *same-as* assertion with an individual of $Person$ extracted from text. In this way, finding correlations between different types of media is facilitated by using DLI results since they provide for the necessary abstract level of information, placing observations of a different nature into the same context.

To make the correspondence between the elements of the content descriptions of both modalities explicit, logical *same-as* assertions are used. In terms of DLs, same-as assertions express the fact that two instances represent the same individual in the domain. Thus, from the previous example, the Abox represented in Figure 2.25 (see page 67) is extended with the following assertions

(new_2, new_4):*same-as*, (new_3, new_1):*same-as*

to express fusion results in the so-called *fused interpretation Abox* (see Figure 2.27).

As a result, same-as assertions between instances of DLC concepts contribute to information gain. Thus, as can be observed in Figure 2.27, individual new_1 becomes more specific given its fusion with individual new_3, which is an instance of the concept $PoleVaulter$.

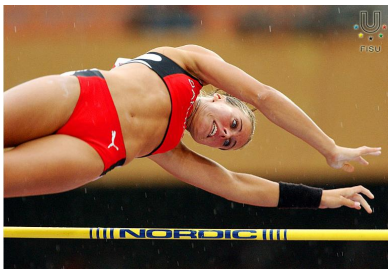


Figure 2.28: Similar observations for different events.

Similarly, individual new_2 becomes more specific given its fusion with new_4. In this way, more specific information means information gain.

2.6.2 Ambiguity Resolution

Very often more than one interpretation is found for a specific media object. This results in a state of ambiguity which is often produced due to poor observations. A clear example for this situation is found in the image modality. Images in multimedia documents are commonly reduced to the most relevant observations since space is usually limited. This has the consequence that various images of the same domain show the same objects although they show different abstract events. For example, in Figure 2.28 (see page 69), both images show a similar set of observations, e.g., a person and a horizontal bar. While it is obvious (for the eye of an expert in athletics) that due to body positions, the image on the left shows a pole vault trial while the image on the right a high jump trial. For an image analysis system that cannot differentiate between body positions, the observations are limited only to objects and spatial relations between them such as the following

$$\Gamma = \{b_1 : PersonBody,\ f_1 : PersonFace,\ hb_1 : HorizontalBar,\\ (b_1, f_1) : adjacent,\ (b_1, hb_1) : isNear\}$$

Applying DLI over the previous Γ, using the set of rules in Figure 2.30 (page 71) and the Tbox in Appendix A (see page 145), produces two consistent interpretations for each one of the images[10]:

$$\Delta_1 = \{new_9 : Person,\ (new_9, b_1) : hasPart,\ (new_9, f_1) : hasPart,\\ new_{10} : HighJump,\ (new_{10}, new_9) : hasParticipant,\ (new_{10}, hb_1) : hasPart\}$$

$$\Delta_2 = \{new_9 : Person,\ (new_9, b_1) : hasPart,\ (new_9, f_1) : hasPart,\\ new_{10} : PoleVault,\ (new_{10}, new_9) : hasParticipant,\ (new_{10}, hb_1) : hasPart\}$$

[10]Since DLI produces the same set of interpretations for both images, only one set is presented for the example.

13 August 2002 - Helsinki. Russia's newly crowned European champion Jaroslav Rybakov won the high jump with 2.29 m. Oskari Fronensis from Finland cleared 2.26 and won silver.

Figure 2.29: Complementary types of content.

Note that due to limitations on the number of atoms that the head of a rule can have (to only one atom can be used in the head), the interpretation process has to iterate over the observations to be explained (fiats). This has the consequence that for those aggregates where their parts, represent the subset of the parts of another aggregate, the DLI process will create an explanation for each of the aggregates. In the previous example, this situation applies to the aggregates *HighJump* and *PoleVault* (see concept definitions in Appendix A, page 145). The only difference between the parts of both aggregates is the concept *Pole*. For this reason, interpreting a high jump image will always provide (given the rules in Figure 2.28, page 69) two explanations. In consequence two multimedia interpretation Aboxes are produced, one which explains the web page as containing an image which depicts a pole vault trial and one which explains the web page as containing an image which depicts a high jump trial. In this example, if the image about the pole vault event would show also a pole, the DLI process provides one explanation, namely that the image depicts a pole vault trial.

In this work fusion is proposed to solve such cases of ambiguity. For example, consider the image in Figure 2.29 (see page 70) which contains complementary information from text and image content. The interpretation of the textual part was presented in Section 2.5 and its graphical representation is found in Figure 2.23 (see page 64). As an interpretation for the image consider the set Δ_s previously presented (see page 69). There exist two multimedia interpretation Aboxes, each one containing one of the image interpretations above (see Δ_1 and Δ_2 in page 69).

adjacent(X, Y)	←	*Person*(Z),
		hasPart(Z, X), *Body*(X),
		hasPart(Z, Y), *Face*(Y).
isOverlapping(X, Y)	←	*Person*(X),
		hasPart(X, Z),
		isOverlapping(Z, Y).
isOverlapping(X, Y)	←	*PoleVault*(Z),
		hasParticipant(Z, X), *PoleVaulter*(X),
		hasPart(Z, Y), *Pole*(Y).
isOverlapping(X, Y)	←	*PoleVault*(Z),
		hasParticipant(Z, X), *PoleVaulter*(X),
		hasPart(Z, Y), *HorizontalBar*(Y).
isOverlapping(X, Y)	←	*HighJump*(Z),
		hasParticipant(Z, X), *HighJumper*(X),
		hasPart(Z, Y), *HorizontalBar*(Y).
isNear(X, Y)	←	*Person*(X),
		hasPart(X, Z),
		isNear(Z, Y).
isNear(X, Y)	←	*ShotPut*(Z),
		hasParticipant(Z, X), *ShotPutter*(X),
		hasPart(Z, Y), *Shot*(Y).
isNear(X, Y)	←	*PoleVault*(Z),
		hasParticipant(Z, X), *PoleVaulter*(X),
		hasPart(Z, Y), *Pole*(Y).
isNear(X, Y)	←	*PoleVault*(Z),
		hasParticipant(Z, X), *PoleVaulter*(X),
		hasPart(Z, Y), *HorizontalBar*(Y).
isNear(X, Y)	←	*HighJump*(Z),
		hasParticipant(Z, X), textitHighJumper(X),
		hasPart(Z, Y), *HorizontalBar*(Y).
isNear(X, Y)	←	*ShotPut*(Z),
		hasParticipant(Z, X), *ShotPutter*(X),
		hasPart(Z, Y), *StopBoard*(Y).

Figure 2.30: Set of rules R for image interpretation.

As previously explained, the objective of fusion is to express the correlations between the content of both media objects through same-as assertions. For this reason it is necessary to identify which domain instances belong to which media objects, such that information about the content structure of a multimedia document is required. For this example the following same-as assertions are added to each of the multimedia interpretation Aboxes.

A_{x1} =text interpretation Abox (see Figure 2.23 in page 64) $\cup$
image interpretation Abox (see $\Gamma \cup \Delta_1$ in page 69) $\cup$
$\{id_{12}$:*mco:Caption*, id_{13}:*mco:Image*, id_{14}:*mco:CaptionedImage*,
id_{15}:*mco:WebPage*, (id_{15}, id_{14}):*mco:contains*,
(id_{14}, id_{12}):*mco:contains*, (id_{14}, id_{13}):*mco:contains*,
(id_{13}, new_9):*mco:depicts*, (id_{13}, new_{10}):*mco:depicts*,
(id_{12}, new_1):*mco:depicts*, (id_{12}, new_2):*mco:depicts*,
(id_{12}, new_3):*mco:depicts*, (id_{12}, new_4):*mco:depicts*,
(id_{12}, new_5):*mco:depicts*, (id_{12}, new_6):*mco:depicts*$\}$ $\cup$
$\{(new_9, new_2)$:*same-as*, (new_9, new_3):*same-as*,
(new_5, new_{10}):*same-as*, (new_6, new_{10}):*same-as*$\}$

A_{x2} =text interpretation Abox (see Figure 2.23 in page 64) $\cup$
image interpretation Abox (see $\Gamma \cup \Delta_2$ in page 69) $\cup$
$\{id_{12}$:*mco:Caption*, id_{13}:*mco:Image*, id_{14}:*mco:CaptionedImage*,
id_{15}:*mco:WebPage*, (id_{15}, id_{14}):*mco:contains*,
(id_{14}, id_{12}):*mco:contains*, (id_{14}, id_{13}):*mco:contains*,
(id_{13}, new_9):*mco:depicts*, (id_{13}, new_{10}):*mco:depicts*,
(id_{12}, new_1):*mco:depicts*, (id_{12}, new_2):*mco:depicts*,
(id_{12}, new_3):*mco:depicts*, (id_{12}, new_4):*mco:depicts*,
(id_{12}, new_5):*mco:depicts*, (id_{12}, new_6):*mco:depicts*$\}$ $\cup$
$\{(new_9, new_2)$:*same-as*, (new_9, new_3):*same-as*$\}$

For the first multimedia interpretation Abox (see A_{x1} above), same-as assertions were found that keep A_{x1} consistent w.r.t. the Tbox in $\mathcal{O}$. For the second Abox A_{x2}, less number of same-as assertions where found that comply with consistency requirements. In this case, adding a same-as assertion that makes the Abox inconsistent w.r.t. $\mathcal{T}$, is used as indicator for ambiguity resolution where the Abox with less number of same-as assertions is discarded. As a result the image in Figure 2.29 (see page 70) is represented as one depicting a high jump trial. Notice that deciding which of the persons in the text correspond to the person in the image remains unsolved.

After describing the advantages of fusion, the following section describes the fusion algorithm proposed in this work.

2.6.3 The Multimedia Fusion Algorithm

The goal of multimedia fusion is to find concept instances within DLI results, obtained from different media objects, where such instances refer to the same individual in the real world and consequently add same-as statements between them (see Figure 2.27 on page 68 for an example).

Algorithm 3 The Fusion Algorithm

1: **function** multimedia_fusion($\mathcal{O}, \mathcal{R}, \{A_{x1}, \ldots, A_{xn}\}$)
2: fused_Aboxes := {}
3: **for all** $A_x \in \{A_{x1}, \ldots, A_{xn}\}$ **do**
4: sameAs := forward_chaining($\mathcal{O}, \mathcal{R}, A_x$)
//sameAs := $\{(i,j)$:same-as $\mid i,j \in A_x, i \neq j\}$
5: **if** $\mathcal{O} \cup A_x \cup$ sameAs $\not\models \bot$ **then**
6: fused_Aboxes := fused_Aboxes $\cup \{A_x \cup$ sameAs$\}$
7: **end if**
8: **end for**
9: preferred_Aboxes := $sup(poset(\text{fused\_Aboxes}, \lambda(x,y) \bullet S(x) < S(y)))$

10: **function** S(A_x)
11: $S := \sharp\{(i,j)$:same-as $\in A_x\}$

12: **return** preferred_Aboxes

This can be formalized as a logic based process (see Algorithm 3) that gets as input a knowledge base composed of an ontology ($\mathcal{O}$), a set of fusion rules ($\mathcal{R}$), and a set of multimedia interpretation Aboxes ($\{A_{x1}, \ldots, A_{xn}\}$). The ontology part contains descriptions for both, domain specific knowledge, e.g., AEO ontology, and knowledge about document's content structure, e.g., MCO ontology. The set $\{A_{x1}, \ldots, A_{xn}\}$ represents the different interpretations for one multimedia document. Each Abox $A_x \in \{A_{x1}, \ldots, A_{xn}\}$ is a multimedia interpretation Abox (an example can be seen in Figure 2.25, page 67), that contains SLI and DLI results for each media object in the multimedia document, as well as assertions that describe the content structure of the multimedia document. The set of fusion rules $\mathcal{R}$ (see Figure 2.32, page 77) provides the mechanism to extend each multimedia interpretation Abox with same-as assertions. To explain the algorithm step-by-step consider the following input:

$\mathcal{O}$ = the domain ontology (AEO) (see Appendix A, page 145) and the content structure ontology (MCO) (see axioms in Figure 2.31, page 75).

$\mathcal{R}$ = the set of fusion rules in Figure 2.32 (page 77).

A set of Aboxes $\{A_{x1}, A_{x2}\}$

The Aboxes contain the following assertions:

A_{x1} = text interpretation Abox (see Figure 2.23 in page 64) ∪
image interpretation Abox (see $\Gamma \cup \Delta_1$ in page 69) ∪
{id_{12}:*mco:Caption*, id_{13}:*mco:Image*, id_{14}:*mco:CaptionedImage*,
id_{15}:*mco:WebPage*, (id_{15}, id_{14}):*mco:contains*,
(id_{14}, id_{12}):*mco:contains*, (id_{14}, id_{13}):*mco:contains*,
(id_{13}, new_9):*mco:depicts*, (id_{13}, new_{10}):*mco:depicts*,
(id_{12}, new_1):*mco:depicts*, (id_{12}, new_2):*mco:depicts*,
(id_{12}, new_3):*mco:depicts*, (id_{12}, new_4):*mco:depicts*,
(id_{12}, new_5):*mco:depicts*, (id_{12}, new_6):*mco:depicts*}

A_{x2} = text interpretation Abox (see Figure 2.23 in page 64) ∪
image interpretation Abox (see $\Gamma \cup \Delta_2$ in page 69) ∪
{id_{12}:*mco:Caption*, id_{13}:*mco:Image*, id_{14}:*mco:CaptionedImage*,
id_{15}:*mco:WebPage*, (id_{15}, id_{14}):*mco:contains*,
(id_{14}, id_{12}):*mco:contains*, (id_{14}, id_{13}):*mco:contains*,
(id_{13}, new_9):*mco:depicts*, (id_{13}, new_{10}):*mco:depicts*,
(id_{12}, new_1):*mco:depicts*, (id_{12}, new_2):*mco:depicts*,
(id_{12}, new_3):*mco:depicts*, (id_{12}, new_4):*mco:depicts*,
(id_{12}, new_5):*mco:depicts*, (id_{12}, new_6):*mco:depicts*}

The fusion process iterates over the set of multimedia interpretation Aboxes (see line 3 in Algorithm 3, page 73), and for each of them the set of fusion rules are applied in a forward chaining way (see line 4). The reason for applying fusion rules in a forward-chained way is because during fusion, we want to avoid the creation of hypotheses about domain related concepts or roles. Thus, in fusion we want to obtain the antecedents from the previously extracted DLI results and from them extract same-as assertions as consequence. In this way, the resulting fused interpretation Abox (see an example in Figure 2.27, page 68) adds only same-as assertions and no hypotheses about DLC or SLC concepts. The resulting same-as assertions are captured in the set *sameAs* (see line 4 in Algorithm 3).

To continue with the example, consider for the first iteration of the fusion algorithm the multimedia interpretation Abox A_{x1}. Note that from the set of fusion rules $\mathcal{R}$ only the first one is applicable, given that A_{x1} describes a content structure of a web page which contains only a captioned image. For the other two rules to apply, information about a text instance should exist in A_{x1}. The following set of same-as assertions is obtained

$A_x = A_{x1}$
sameAs = {(new_9, new_2):*same-as*, (new_9, new_3):*same-as*,
(new_5, new_{10}):*same-as*, (new_6, new_{10}):*same-as*}

$MultimediaContentItem$	$\sqsubseteq$	$\forall hasMediaDecomposition.MultimediaSegment,$
		$\sqcap \exists hasURL.string$
$WebPage$	$\sqsubseteq$	$MultimediaContentItem$
		$\sqcap \forall hasMediaDecomposition.HTMLSegment$
$CaptionedImage$	$\sqsubseteq$	$MultimediaContentItem$
		$\sqcap \exists hasLogicalDecomposition.Caption,$
		$\sqcap \exists hasLogicalDecomposition.Image$
$Image$	$\sqsubseteq$	$MultimediaContentItem$
		$\sqcap \forall hasMediaDecomposition.StillRegion$
$Caption$	$\sqsubseteq$	$MultimediaContentItem$
		$\sqcap \forall hasMediaDecomposition.TextSegment$
$HTMLSegment$	$\sqsubseteq$	$SpaceSegment$
		$\sqcap \forall hasSegmentLocator.SpaceLocator,$
		$\sqcap \exists hasSegmentLocator.SpaceLocator$
$StillRegion$	$\sqsubseteq$	$SpaceSegment$
		$\sqcap \forall hasSegmentDecomposition.StillRegion,$
		$\sqcap \forall hasSegmentLocator.VisualLocator$
$TextSegment$	$\sqsubseteq$	$SpaceSegment$
		$\sqcap \forall hasSegmentDecomposition.TextSegment,$
		$\sqcap \forall hasSegmentLocator.TextualLocator$
$SpaceLocator$	$\sqsubseteq$	$SegmentLocator$
		$\sqcap \exists_{\leq 1}\ hasURL.string$
$TextualLocator$	$\sqsubseteq$	$SpaceLocator$
		$\sqcap \exists_{\leq 1}\ hasStartOffset.string$
		$\sqcap \exists_{\leq 1}\ hasEndOffset.string$

Figure 2.31: An excerpt of the MCO ontology.

Note that the set *sameAs* keeps the Abox consistent. Some same-as assertions are in fact redundant provided top-down information flow. For example the assertion (new_5, new_{10})-*:same-as* makes individual new_2 and new_9 to be synonyms given the universal restriction and number restrictions on the role $hasParticipant$ in the definition of the concept $HighJump$

$$\begin{aligned} HighJump \quad \sqsubseteq \quad & Jumping \sqcap \forall hasParticipant.HighJumper \\ & \sqcap \exists_{\leq 1} hasParticipant.\top \\ & \sqcap \exists_{\leq 1} hasPart.HorizontalBar \\ & \sqcap \neg PoleVault \sqcap \neg ShotPut \end{aligned}$$

For the second iteration over A_{x2} the following same-as assertions are obtained

$A_x = A_{x2}$
sameAs= $\{(new_9, new_2)$:*same-as*, (new_9, new_3):*same-as*$\}$

This set keeps A_{x2} consistent given that according to the Tbox, there is no disjointness axiom between the concepts $PoleVaulter$ and $HighJumper$ since in the athletics domain an athlete, specially those participating in a decathlon competition, can be both a pole vaulter and a high jumper. On the other hand side, relating individual $new_5 : HighJump$ or $new_6 : HighJump$ with individual $new_{10} : PoleVault$ in a same-as relationship is inconsistent due to disjointness axioms in the Tbox.

From the Aboxes in the set $fused\_Aboxes$ (see line 9 in Algorithm 3, page 73) an ordered set called *preferred_Aboxes* is obtained containing the Aboxes with the highest number of same-as assertions. In the example, the function $multimedia\_fusion$ returns one fused interpretation Abox (A_{x1}), describing a web page that contains an image that depicts a high jump trial.

From the examples provided in this section, it can be observed that multimedia fusion is facilitated if more abstract information is represented through Abox assertions. Thus, the information extracted (SLI results) from different media have a different nature, and the results of DLI provide the required level of abstraction that puts observations in a common context (DLC concepts) that facilitate logic-based fusion.

same-as(W, Z)	$\leftarrow$	$CaptionedImage(M)$,
		$contains(M, X)$,
		$contains(M, Y)$,
		$Image(X)$,
		$Caption(Y)$,
		$depicts(X, W)$,
		$depicts(Y, Z)$
same-as(W, Z)	$\leftarrow$	$WebPage(M)$,
		$contains(M, X)$,
		$contains(M, Y)$,
		$Text(X)$,
		$CaptionedImage(Y)$,
		$contains(Y, I)$,
		$depicts(X, W)$,
		$depicts(I, Z)$,
		$SportsTrial(W)$,
		$SportsTrial(Z)$,
		$hasParticipant(W, O)$,
		$hasParticipant(Z, R)$,
		$hasName(O, E)$,
		$hasName(R, F)$,
		$hasNameValue(E) = hasNameValue(F)$
same-as(W, Z)	$\leftarrow$	$WebPage(M)$,
		$contains(M, X)$,
		$contains(M, Y)$,
		$Text(X)$,
		$CaptionedImage(Y)$,
		$contains(Y, I)$,
		$depicts(X, W)$,
		$depicts(I, Z)$,
		$Person(W)$,
		$Person(Z)$,
		$hasName(W, E)$,
		$hasName(Z, F)$,
		$hasNameValue(E) = hasNameValue(F)$

Figure 2.32: Fusion rules.

2.7 Paving the way from SLI to DLI

In order to support content-based services, such as content-based media retrieval, content descriptions are required. By studying the characteristics of DLs, and by investigating how DLs define semantics, we conclude that in order to represent content semantics, DL-based content descriptions are advantageous as follows:

- DL-based content descriptions provide a basis for content annotation that is beyond manually provided metadata. DLs provide to content descriptions the formal and expressive syntax and semantics that allow describing media content in a more abstract and richer way.

- The abstract character of DL-based content descriptions allows retrieving media through the definition of queries that use abstract terms, which are more meaningful in human language (see Figure 2.4, page 21) and that are based on content semantics. Thus, such queries enable retrieval processes that are different from classical IR processes, given the use of content semantics.

- The usage of domain ontologies that are built with DLs, allows applications (that implement content-based services) to exploit logic reasoning in order to check the consistency of content descriptions w.r.t. the domain ontology.

Content-based services are desirable, but they can only be developed if the media objects from the data sources being accessed have the corresponding content descriptions. Provided the high amount of media being produced, a process that automatically creates the necessary DL-based content descriptions is required and, the deep-level interpretation (DLI) process proposed in this chapter fulfills this need.

By describing the framework of the DLI process (see Figure 2.7), we defined the input requirements of DLI and the requirements set on SLI processes. By describing the characteristics of the SLI process for text content, we argued that state-of-the-art technology is able to fulfill DLI's input requirements and conclude the following. If an automatic process for the creation of content descriptions with a more abstract character is required, the DLI process proposed here is advantageous because:

- The input requirements of DLI, which are content descriptions with a superficial character (see example provided in Section 2.4), allow SLI processes, more specifically shallow-processing techniques, to deal with vast amounts of information. This prevents SLI processes from implementing more expensive tasks that require more complex grammars in order to extract descriptions with an abstract character which is found implicit in the content.

- The usage of DL-based domain ontologies $\mathcal{O}$ and domain rules $\mathcal{R}$ as background knowledge for the DLI process, allows for a declarative representation of extraction requirements which are domain-dependent.

- DLI uses the SLI input and the background knowledge to exploit reasoning services to extract further content descriptions with a more abstract character than SLI processes do.

- The abstract character of the content descriptions extracted by DLI allow for multimedia fusion. Thus, given that descriptions obtained from image and text content are different in nature (e.g., objects are obtained from images and facts from text) and therefore corresponding semantics are logically disjoint, abstract descriptions provide a common context, through aggregate concepts, that allows for logic-based fusion.

The step-by-step examples for the interpretation of text content provided in this chapter allowed to illustrate how interpretation is necessarily an iterative processes that requires different reasoning services. It is necessarily iterative because content descriptions are usually incomplete, hence only some elements of the domain are present, called observations. Therefore, an interpretation process starts on the basis of observations that should be explained in order to understand their purpose w.r.t. domain knowledge. Abduction is the process of explanatory reasoning applied in situations with incomplete information. Thus, interpretation is a process that necessarily requires abductive reasoning in order to extract elements of the domain that help to explain the existence of the observations w.r.t. domain of knowledge. The DLI process proposed here is based on abductive reasoning. Different to other works in abductive reasoning which formalize Abox abduction [EKS06] with DLs, the work presented in section 2.5.1 shows how to compute explanations and provides also means for hypothesis selection. The DLI process has been evaluated in the work of A. Kaya in [Kay11] showing that the quality of the interpretation results are of high-quality. Differently to the work of Hobbs et al. [HSAM93], the work presented here exploits reasoning in order to check the consistency of the interpretation results w.r.t. domain ontologies. Moreover, hypothesis selection is based on domain knowledge and is not dependent on the syntactic characteristics of the sentence, as done in [HSAM93]. Of course, the work presented in [HSAM93], which is heavily based on linguistic knowledge, is suitable for applications such as language translation which require deep analysis. The DLI process presented here is useful for its application on media annotation, given that it is based on shallow-processing techniques for SLI. By using shallow processing techniques a process of interpretation can be built that deals with high amounts of data. With the step-by-step interpretation examples provided and

the description of the SLI and DLI processes we can conclude that the results of shallow processing are good enough as input for DLI, and that DLI provides practical means to achieve the automatic extraction of content descriptions that represent more abstract content semantics.

DLI is a knowledge intensive process, consequently the engineering of the required background knowledge is an important task. In the following chapter we describe how a domain ontology and rules should be built, such that DLI is able to produce useful results. Moreover we describe what we define here as knowledge management and how DLI can contribute to support KM.

Chapter 3

Knowledge Management

Provided the high quantities of media being constantly produced, it is comprehensible that the content of media varies over time and that the background knowledge used for the interpretation of media requires necessary extensions to allow the representation of new content descriptions that have not being considered in old or recently produced media. Thus, background knowledge as well as the media content should be continuously analyzed to see if interpretation requirements are met. This involves various tasks, for example, search for gaps of content segments without surface-level annotations, search for results of SLI that lack a deep-level interpretation. These tasks indicate that the design of background knowledge that supports SLI and DLI processes is highly dependent on media content. Moreover, it is unrealistic that these tasks are manually executed. Thus, automatic means should be provided. In this work we define KM as a process that supports these tasks. In this application context, we define knowledge management as follows:

> Knowledge Management provides for systematic means that support engineers in the task of modeling knowledge represented through domain ontologies and rules.

In the following section the concept of grounded ontology design is described. It uses interpretation results as feedback about missing background knowledge to guide the knowledge engineer in the task of designing such that a systematic approach of ontology design can be created. Afterwards, Section 3.2 describes specific ontology and rule design patterns which support the DLI process. The objective is to describe generic patterns (applicable to any domain) that allow DLI to provide for useful results. For example, patterns that support information gain by allowing for downward information flow are described.

3.1 Grounded Ontology Design

Before starting the task of designing a knowledge base, two main questions should be answered, viz. what problem should be solved (application-centered point of view, e.g., text annotation, text translation, etc.) and what is the domain of interest (content-centered point of view, e.g., athletics, medicine, etc.). Providing an answer to these questions is central to the task of ontology design which can be understood as *"what to include in the background knowledge in order to solve a problem"*. To cope with this task, this section proposes the notion of **grounded ontology design**, which means that for every element that is part of the background knowledge base, i.e., the ontology, there is a concrete counterpart (the ground) in the application domain, such that the existence of each element of the ontology is justified. In this case the application domain is text annotation, such that the ground is found within the text content. While the need of a content-centered and application-centered process for ontology design seams natural and is considered in other contexts such as database design, this is not always the case in the context of ontology design in general, and more specifically in the context of DL-based ontologies. Thus, with the exception of some ontologies such as the one described in [JW04], there is a tendency towards designing domain ontologies based only on the knowledge of the domain expert without considering the type of problem the ontology will be used to solve.

By having an application-centered view, design decisions, in terms of DL-language constructors, can be made to identify the expressivity that an ontology requires. Thus, the expressivity of an ontology affects the inference strategy that is applied to solve a specific problem. Moreover, the expressivity directly influences the complexity of the reasoning algorithms that are necessary to support inference mechanisms, and in consequence the performance of the application that uses an ontology is also affected. Performance is a relevant aspect, for this reason one of the main research efforts of the DL scientific community focuses on the exploration of optimized reasoning algorithms [Hor03] for different levels of complexity that have given rise to a classification of DL languages according to their expressivity.

This work proposes grounded ontology design as a necessary process to produce ontologies useful for specific application needs, in this case, for its application on text annotation with deep-semantics. With a content-centered view of design, the knowledge engineer can concentrate only on the modeling of required elements as determined by media content and in this way avoid either an exhaustive or a generic design of the domain. Furthermore, with an application-centered view the ontology engineer is able to justify the commitment to some expressivity level — given performance consequences affected by reasoning algorithms.

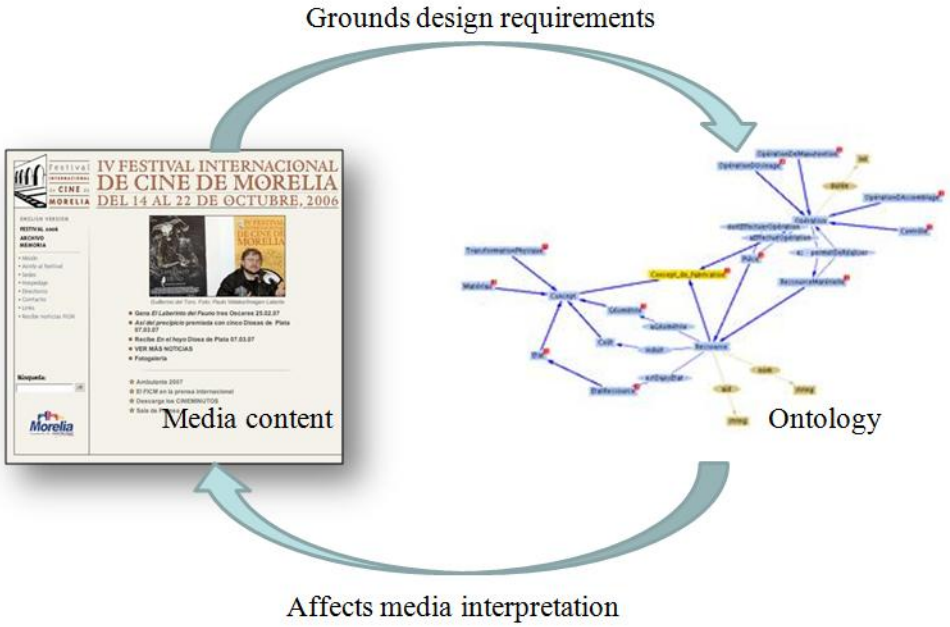


Figure 3.1: Grounded ontology design: a content- and application-centered process

In this work, the application-context is media interpretation, more specifically text interpretation, while the domain of interest can vary. Thus, the information need will determine the domain of interest. Since the domain of interest can be broad and described in different ways, it is necessary to decide which aspects of the domain are relevant. This decision problem is solved, when considering a content-centered point of view, because the media content will determine the aspects of the domain that should be modeled. For example, if the media content conveys information about film festivals, then modeling other aspects of the film domain such as cinematography, film production, etc., is surplus. Thus, as Figure 3.1 shows, there is a bilateral influence between the media content and the elements modeled in the ontology, i.e., the media content is the ground used to design the ontology and the final design of the ontology determines how the content will be interpreted. Therefore, in the context of media interpretation, grounded ontology design means that the choice of certain terminological elements, e.g., atomic concepts, atomic roles, as well as complex concept descriptions, GCIs and constructors, should be influenced by both media content and the requirements imposed by DLI.

To complement this concept, the following section describes ontology and rule design patterns that result from this view of grounded ontology design, which has helped to support strategies for ontology evolution. Thus, the work presented by Castano et al. in [CEF+09, CEF+07] describes how with the use of DLI in combination with design patterns, a systematic approach to ontology evolution can be paved. Castano et al., identified four different situations (called evolution patterns) that trigger ontology evolution based on DLI results, e.g., absence of explanations for observations, ambiguous explana-

tions. According to the identified evolution pattern a specific process of evolution called population and enrichment should be performed, and is articulated into a set of activities for implementing the required changes in the ontology, where the design patterns in the following Section 3.2 are considered. In this way, the process of grounded ontology design represents a cycle as illustrated in Figure 3.1 in which, starting from the results of media interpretation, design requirements are identified (grounded on media content) in order to evolve the ontology with the required axioms.

3.2 Ontology and Rule Design Patterns

Relevant work has been done in the development of methodologies for the construction of ontologies, an extensive summary of such methodologies can be found in [JW04]. In this context, ontology patterns play an important role, and different design patterns have been proposed with a focus on specific aspects. For example, the work in OWL[1] focuses on expressivity issues under a specific language, the work in [GCMS03] focuses on generic aspects common to different domains of knowledge so called conceptual patterns that help to construct foundational ontologies (also called upper ontologies). Upper ontologies can be further refined according to the knowledge of domain experts, e.g., the upper ontology DOLCE [GCMS03] designed. Other methodologies such as the one followed in the KACTUS project [BLC96] focus on knowledge reuse such that existing ontologies are restructured according to application needs.

The work of Gangemi in [Gan05] describes so called content-oriented patterns which aim at solving design problems for domain-related classes and properties. In [Gan05], the design of an ontology starts with the definition of so called Generic Use Case (GUC), which are expressions about recurrent issues in a specific domain. For example: Who does what, when and where, which objects take part in a certain event, what are the parts of something, what is an object made of, what is the place of something, etc. GUCs are similar to competency questions, also proposed by other authors in [UK95, GF94], that will guide the engineer in understanding the design needs. Afterwards, formal patterns (so-called CODePs) encode the corresponding GUC, by using fragments of an upper ontology, such that a pattern is axiomatized according to the extracted fragment and its later specialized according to domain requirements, to become a so called core pattern. The work in [Gan05] proposes a frame for the documentation of the patterns, with slot-value pairs to include textual descriptions of the GUC, its specialization (domain of interest), the logic language used, reference ontologies (the upper ontologies from which it was extracted), etc.

[1] See OWL in http://www.w3.org/TR/2004/REC-owl-features-20040210/

The patterns presented here are based on the concept of grounded ontology design. While these patterns do not exclude the usage of competency questions, as the ones described in [Gan05], their focus is on the specific constructs required to design aggregate concepts and rules to support the interpretation of text and image content. Thus, the patterns presented here are application-centered, more specifically, focused on the support of the DLI process and content-centered, i.e., the patterns consider media content as ground for the definition of complex concepts. Naturally, in order to apply the proposed patterns, domain knowledge is required, such that the use of competency questions can be auxiliary. These patterns describe precise axioms, and an order of application is suggested to further facilitate the design task. To illustrate the patterns, examples of DL axioms and rules will be given for a specific domain.

To begin with the description of the patterns presented here, the first steps to follow should be to answer the previously mentioned questions about domain of interest and problem to solve. As an example consider the domain of *film festivals*. The problem is to interpret web pages that contain film-festival news for their annotation with content descriptions. Note that for this section a domain different to the running example (athletics domain) is considered. This is done on purpose to show that the patterns are generic and applicable to different domains of interest. An excerpt of typical news in the film-festivals domain can be seen in Figure 2.6 (see page 24). In the rest of this section the patterns for ontology and rule design are described in an ordered sequence. The sequence suggests the order in which the patterns can be used in order to facilitate the task of knowledge base design. Note that the examples provided in this section are also running examples, such that each succeeding example builds on the axioms of previous examples.

Pattern 1. Initial Ontology. As shown in Figure 2.21 (see page 52), media interpretation is considered as a two stage process that consists of SLI and DLI. The terminological part of the knowledge base, composed of Tbox ($\mathcal{T}$) and a set of rules ($\mathcal{R}$), should include the terminology necessary to express both SLI and DLI results. SLI results are instances of so-called surface-level concepts (SLC) and role assertions of so-called surface-level roles (SLR). DLI results are instances of so-called deep-level concepts (DLC) and role assertions of so called deep-level roles (DLR). Therefore, an ontology used for the DLI framework initially contains two generic atomic concepts and atomic roles to represent the results of each interpretation stage.

Definition Initial Ontology. *Let $\mathcal{S}$ be a signature in an ontology $\mathcal{O}$, SLR and DLR role names, SLC and DLC concept names in $\mathcal{S}$. The initial ontology contains the following elements*

$$\begin{aligned} Signature: & \\ & CN = \{SLC, DLC\} \\ & RN = \{SLR, DLR\} \\ Tbox: & \\ & SLC \sqsubseteq \neg DLC \end{aligned}$$

where SLC and DLC are disjoint concept names that stand for "surface-level concept" and "deep-level concept" respectively. DLC is a generic concept used to model specific aggregates. SLC is a generic concept used to model observations resulting from SLI. SLR stands for "surface-level role" and is used as a generic role to model configurations between observations. DLR stands for "deep-level role" and is used as a generic role to model relations between an aggregate and it parts.

Pattern 2. Surface-Level Terms. The next step is to identify representative media of the domain of interest and determine the information needs that should be fulfilled. This helps in order to determine the set of atomic concept descriptions and atomic role descriptions that will shape the set of surface-level terms and that SLI uses to denote observations and relations between observations.

Definition Surface-Level Terms. *Let $\mathcal{S}$ be a signature in an ontology $\mathcal{O}$, O a concept name and OR a role name in $\mathcal{S}$. The following terminological axioms are added to the Tbox $\mathcal{T}$ in $\mathcal{O}$*

$$\begin{aligned} O_i &\sqsubseteq SLC \\ O_1 \sqcap ... \sqcap O_n &\sqsubseteq \bot \\ OR_i &\sqsubseteq SLR \\ \exists OR_i.\top &\sqsubseteq O_i \\ \top &\sqsubseteq \forall OR_i.O_i \end{aligned}$$

where for all $i \in \{1, \ldots, n\}$, O stands for "observation" and O_i for a specific SLC. In the set of observations, the elements $O_s \in \mathcal{T}$ are disjoint from each other. OR stands for "observed relation" and OR_i for a specific SLR. The set of observed relations OR_s are domain and range restricted to observations O_i.

Example 1. Considering the text in Figure 2.6 (see page 24) the following terminology can be used for surface-level terms:

$$
\begin{aligned}
CityName &\sqsubseteq SLC \\
FilmFestivalName &\sqsubseteq SLC \\
FilmName &\sqsubseteq SLC \\
Date &\sqsubseteq SLC \\
Duration &\sqsubseteq SLC \\
DirectorName &\sqsubseteq SLC \\
ActorName &\sqsubseteq SLC \\
PersonName &\sqsubseteq SLC \\
FilmFestivalName \sqcap CityName \sqcap & \\
FilmName \sqcap Date \sqcap Duration &\sqsubseteq \bot \\
filmFestivalNameToCityName &\sqsubseteq SLR \\
filmFestivalNameToFilmName &\sqsubseteq SLR \\
\exists filmFestivalNameToCityName.\top &\sqsubseteq FilmFestivalName \\
\top &\sqsubseteq \forall filmFestivalNameToCityName.CityName \\
\exists filmFestivalNameToFilmName.\top &\sqsubseteq FilmFestivalName \\
\top &\sqsubseteq \forall filmFestivalNameToFilmName. \\
& \quad FilmName
\end{aligned}
$$

The definition of specific surface-level roles (OR_i) is useful for the performance of the DLI process, since defining more specific names for SLR restricts the number of applicable rules during abduction (see Section 2.5.1). The names provided to OR_i roles are descriptive about the existence of domain and range restrictions, e.g., the surface-level role $filmFestivalNameToCityName$ is domain restricted to objects of type $FilmFestivalName$ and range restricted to objects of type $CityName$. Disjointness axioms between observations are necessary to avoid the computation of inconsistent explanations w.r.t. the domain knowledge during the variable substitution process of abduction.

Pattern 3. Aggregate. As defined in [NM06], aggregates are conceptual units consisting of parts tied together to form a concept that satisfies certain constraints, in other words, to represent object configurations, which are used as building blocks for the interpretation process. As previously described in Section 2.5.1, the notion of aggregates and the notion of perception as abduction have been used together in this work to support DLI. In this way, observations are perceptions (obtained from SLI processes) about object configurations, which can be justified or explained through aggregates. Thus, aggregates describe more abstract concepts (e.g., events) that are not directly observable from the surface of the content but that are obvious, for a domain expert, given object configurations. Object configurations are media-dependent, i.e., in an image, spatial configuration for objects are found, and in text, linguistic configurations for words can be found. For

example, the text "Abel Ferrara's Mary" in Figure 2.6 is understood by a domain expert as a film called Mary having Abel Ferrara as director, even though this information is not explicit in the text. The interpretation of the text is therefore influenced by background knowledge and observations. In the example above the possessive case gives a linguistic configuration between Abel Ferrara and Mary, which constitute the parts of a film, where the film is a DLC concept (aggregate). Aggregates are modeled for the DLI framework through GCIs as in the following definition.

Definition Aggregate. *Let $\mathcal{T}$ be a Tbox in $\mathcal{O}$, C_i are concept descriptions that model aggregates in $\mathcal{T}$. Aggregates are defined as follows*

$$\begin{aligned} C_i &\sqsubseteq DLC \sqcap \exists R_i.O_i \\ C_1 \sqcap ... \sqcap C_n &\sqsubseteq \bot \\ R_i &\sqsubseteq DLR \\ O_i &\sqsubseteq SLC \end{aligned}$$

where for all $i \in \{1, .., n\}$, C_i are specific DLC concepts being disjoint from each other. R_i are specific DLR which are existentially quantified and O_i are observations.

Example 2. This example is a continuation of Example 1. The following axioms are added to the terminology showing four aggregates, *FilmFestival*, *Film*, *FilmDirector* and *Actor* as complex concepts composed of various parts.

$$\begin{aligned} FilmFestival &\sqsubseteq DLC \\ &\quad \sqcap \exists hasFestivalName.FilmFestivalName \\ &\quad \sqcap \exists takesplaceIn.CityName \\ &\quad \sqcap \exists hasDate.Date \\ Film &\sqsubseteq DLC \\ &\quad \sqcap \exists hasFilmName.FilmName \\ &\quad \sqcap \exists hasDuration.Duration \\ FilmDirector &\sqsubseteq DLC \\ &\quad \sqcap \exists directs.FilmName \\ &\quad \sqcap \exists hasName.DirectorName \\ Actor &\sqsubseteq DLC \\ &\quad \sqcap \exists actsIn.FilmName \\ &\quad \sqcap \exists hasName.ActorName \\ FilmFestival \sqcap Film \sqcap FilmDirector \sqcap Actor &\sqsubseteq \bot \\ hasFestivalName &\sqsubseteq DLR \\ takesplaceIn &\sqsubseteq DLR \\ hasDate &\sqsubseteq DLR \\ hasFilmName &\sqsubseteq DLR \\ hasFilmDirector &\sqsubseteq DLR \\ hasActor &\sqsubseteq DLR \\ hasDuration &\sqsubseteq DLR \end{aligned}$$

Note that aggregates are modeled through necessary conditions. Necessary conditions are used because the ontology engineer might not be able to define exhaustively an aggregate according to media content. It is impractical to read a media corpus in detail in order to identify all relevant parts of an aggregate. Even through a detailed analysis of a media corpus, the corpus is usually incomplete w.r.t. conceptualization of a domain knowledge, such that only some parts of an aggregate might be found within the content. Moreover, SLI can fail to extract some observations, such that often is not possible to use deduction to identify the existence of an aggregate provided the observations. Existential quantification is used during the process of hypothesis selection in Abox abduction to make sure each possible explanation Δ is consistent with respect to the Tbox such that the formula $\Sigma \cup \Delta \models \gamma$ holds.

Pattern 4. Abduction Rule. Abduction rules define the space of possible explanations, also called *abducibles*. This type of rules complies with the definition in page 18 about rules. Rules are applied in a backward-chaining way. The objective is to find out what is missing in the prerequisites of a rule, and to make necessary hypotheses, such that the consequences of rules can follow. In the context of media interpretation, abduction is used to hypothesize aggregates (and corresponding role assertions), that explain the configuration of the observations (SLR assertions) or the observations alone (SLC instances). The hypotheses are checked for consistency w.r.t. the Tbox.

As highlighted before, abduction rules follow the definition on page 18, with the additional restriction that the number of atoms in the body is restricted. A rule body is composed of the following atoms: one unary atom corresponding to the aggregate concept, two binary atoms to express relations between the aggregate and the variables in the head, and two unary atoms that use the variables in the head. (An explanation to this restriction is provided after Example 3). The head of an abduction rule contains either a binary atom, which corresponds to the configuration of observations, or an unary atom, which corresponds to an observation. For every observation, or configuration of observations (relation between the parts of an aggregate), that is expected to be obtained from SLI of media, an abduction rule is defined.

Definition Abduction Rule. *Let $\mathcal{R}$ be a set of rules. An abduction rule r in $\mathcal{R}$ can have one of the following forms*

$$
\begin{array}{lcl}
 & (1) & \\
OR_i(X,Y) & \leftarrow & C_i(Z), \\
 & & R_i(Z,X), \\
 & & O_i(X), \\
 & & R_j(Z,Y), \\
 & & O_j(Y)
\end{array}
\qquad\qquad
\begin{array}{lcl}
 & (2) & \\
O_i(X) & \leftarrow & C_i(Z), \\
 & & R_i(Z,X)
\end{array}
$$

where $i \in \{1, .., n\}$ and rule (1) is used to explain configurations of observations and rule (2) is used to explain observations. C_i is a specific DLC concept found in the ontology. R_i is a specific DLR role and O_i is a specific SLC concept.

Example 3. In this example an abduction rule that explains the configuration $filmFestivalNameToCityName$ between a $FilmFestivalName$ instance and a $CityName$ instance is presented:

$$\begin{aligned} filmFestivalNameToCityName(X,Y) \leftarrow \quad & FilmFestival(Z), \\ & hasFilmFestivalName(Z,X), \\ & FilmFestivalName(X), \\ & takesplaceIn(Z,Y) \\ & CityName(Y) \end{aligned}$$

In the film festivals domain, it is of interest to obtain the place in which a specific event takes place. In this way, the role *filmFestivalNameToCityName* captures the relation between an instance of *FilmFestivalName* and an instance of *CityName* that should be obtained as a result of SLI and that should be explained by DLI.

To explain why the number of atoms in the body of an abduction rule is restricted, consider the following example. If rules such as the following were allowed:

$$\begin{aligned} near(X,Y) \leftarrow \quad & HighJump(Z), \\ & hasParticipant(Z,X), \\ & HighJumper(X), \\ & hasPart(Z,Y), \\ & HorizontalBar(Y) \end{aligned} \qquad \begin{aligned} near(X,Y) \leftarrow \quad & PoleVault(Z), \\ & hasParticipant(Z,X), \\ & PoleVaulter(X), \\ & hasPart(Z,Y), \\ & HorizontalBar(Y), \\ & hasPart(Z,W), \\ & Pole(W) \end{aligned}$$

during the abduction process, applicable rules would produce explanations that are always discarded due to simplicity constraints when compared against competing explanations. Consider the image on the left side of Figure 2.28 (see page 69) and the results of SLI to be the following $\{(i,j) : near,\ i : Person,\ j : HorizontalBar\}$, to explain the role assertion, following explanations are expected:

$$\Delta_1 = \{new_1 : HighJump,\ (new_1, i) : hasParticipant,\ i : HighJumper,\ (new_1, j) : hasPart\}$$

$$\Delta_2 = \{new_1 : PoleVault,\ (new_1, i) : hasParticipant,\ i : PoleVaulter,\ (new_1, j) : hasPart,\ (new_1, new_2) : hasPart,\ new_2 : Pole, \}$$

Explanation Δ_2 hypothesizes two additional assertions and one individual. Comparing them w.r.t. the simplicity criterion, then Δ_2 is discarded and Δ_1 is preferred. Thus, increasing the number of atoms in the body can increase the number of hypothesized assertions and consequently decrease the preference score w.r.t. simplicity, such that the less number of hypothesized assertions an explanation contains, the higher its preference score becomes. In the previous example, the second explanation is discarded before the ambiguity could be solved through fusion. In this way, the design of rules without this restriction, causes the DLI process to execute less iterations and discard competing explanations too early, thus preventing DLI to solve ambiguities during fusion. Furthermore, the rule does not allow to ensure that the elements of the aggregate are related to each other according to SLI result, such that they can be considered as part of the same aggregate instance.

Pattern 5. Related Aggregates. Once aggregates and abduction rules have been defined, the next step is to model relations between aggregates.

Definition Related Aggregates. *Let $\mathcal{T}$ be a Tbox in $\mathcal{O}$, C_i and D_i are DLC concepts to model aggregates in a relation R_i in $\mathcal{T}$. The compositional hierarchy of aggregates can be defined as:*

$$\begin{aligned} C_i &\sqsubseteq DLC \sqcap \exists R_i.D_i \\ D_i &\sqsubseteq DLC \\ R_i &\sqsubseteq DLR \end{aligned}$$

where $i \in \{1, .., n\}$, and R_i is a specific DLR role.

Example 4. In the following example, instances of the deep-level concept *FilmFestival* are enforced to be in a relation *presents* with instances of the deep-level concept *Film*.

$$\begin{aligned} FilmFestival &\sqsubseteq DLC \sqcap \\ &\quad \exists hasFestivalName.FilmFestivalName \sqcap \\ &\quad \exists takesplaceIn.CityName \sqcap \\ &\quad \exists hasStartDate.StartDate \sqcap \\ &\quad \exists presents.Film \\ presents &\sqsubseteq DLR \end{aligned}$$

Note that a relation between aggregates can be defined due to domain knowledge but also due to observed relations between parts of different aggregates.

For every observed relation (OR_i) between parts of different aggregates, a deduction rule is defined.

Pattern 6. Deduction Rule. In cases where relations between parts of different aggregates (OR_i) exist, a set of rules to be applied in a forward-chaining way should be modeled. The objective is to explicitly relate an aggregate with the part of another aggregate explicitly. This follows the idea that *whatever applies to the parts applies also to the whole*. This rule pattern is easier to understand in the image modality. For example, consider Figure 2.29 (see page 70) and the following SLR assertion $near(body_1, bar_1)$, a deduction rule can be defined expressing that if the body of a person is near a horizontal bar, then the person is also near that horizontal bar. These rules are called deduction rules, and they are used to make explicit the information that follows from hypotheses obtained through abduction. The application of a deduction rule results in a new role assertion which, in turn, might be further explained through another abduction rule.

Definition Deduction Rule. *Let $\mathcal{R}$ be a set of rules.* (1) *is a deduction rule and* (2) *is an abduction rule in $\mathcal{R}$:*

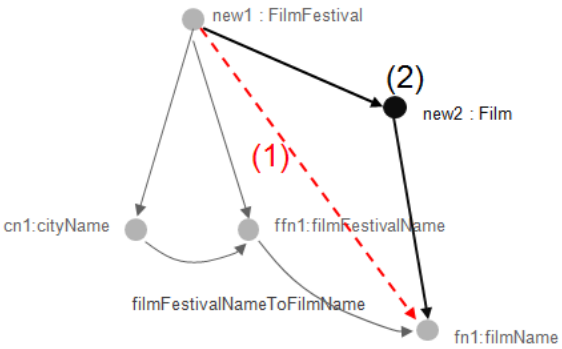
$$\begin{array}{lcl@{\qquad}lcl} & (1) & & & (2) & \\ S_i(X,Y) & \leftarrow & C_i(X), & S_i(X,Y) & \leftarrow & D_i(Z), \\ & & R_i(X,Z), & & & R_i(Z,X), \\ & & O_i(Z) & & & C_i(X) \\ & & OR_i(Z,Y) & & & R_j(Z,Y), \\ & & & & & O_j(Y) \end{array}$$

where $i \in \{1,..,n\}$ and (1) is a rule applied in a forward-chained way. C_i is the name of a specific DLC concept, R_i is a specific DLR role. OR_i is a specific SLR role that relates two instances of the observations O_i and O_j. (2) is a rule applied in a backward-chained way to explain the relation S_i. S_i is a specific DLR role that results from applying rule (1). D_i is a DLC concept which has as parts C_i and O_j.

Example 5. In the following example *filmFestivalNameToFilmName* expresses a configuration observed between two instances that are parts of two different aggregates, e.g., *FilmFestival* and *Film* (see the grey colored elements of the graphic in page 93). (1) is a deduction rule that is applied in a forward-chaining way such that a *filmFestivalToFilmName* relation between the instance of *new1:FilmFestival* and the instance of *fn1:FilmName* follows (see red arrow of the graphic in page 93). (2) is an abduction rule that is applied in a backward-chained way to explain the *filmFestivalToFilmName* relation and hypothesize an instance of the concept $Film$ related to the existing instance of the concept $FilmFestival$ and has individual $fn1$ as part.

(1) Deduction rule
$filmFestivalToFilmName(X, Y) \leftarrow$
$FilmFestival(X)$,
$hasFilmFestivalName(X, Z)$,
$FilmFestivalName(Z)$,
$filmFestivalNameToFilmName(Z, Y)$

(2) Abduction rule
$filmFestivalToFilmName(X, Y) \leftarrow$
$FilmFestival(Z)$,
$presents(Z, X)$,
$Film(X)$,
$hasFilmName(Z, Y)$,
$FilmName(Y)$



Pattern 7. Aggregate Specialization. DLC concepts that allow to represent aggregates can also be part of a taxonomy. The definition of more specific DLC concepts has the purpose of exploiting more specific observations resulting from SLI to support the abduction process in creating more specific explanations. These explanations can be compared against others to support the *informativeness* constraint.

For example, in the context of image interpretation, if SLI is able to recognize an object as a *ball* and more specifically as a *football*, then for DLI it is possible to interpret this object as part of a *football game* and not as part of a generic event such as a *game*. Thus, more specific observations means that more specific deep-level information can be extracted and checked for consistency against the definitions in the Tbox. A more specific aggregate is defined as follows.

Definition Aggregate Specialization. *Let $\mathcal{T}$ be a Tbox in $\mathcal{O}$, C_i and C'_i are specific DLC concepts that form a taxonomy in $\mathcal{T}$, such that C_i subsumes C'_i as follows*

$$\begin{aligned} C_i &\sqsubseteq DLC \sqcap \exists R_i.O_i \sqcap \exists R_j.D_i \\ O_i &\sqsubseteq SLC \\ D_i &\sqsubseteq DLC \\ R_i &\sqsubseteq DLR \end{aligned}$$

$$\begin{aligned} C'_i &\sqsubseteq C_i \sqcap \forall R_i.O'_i \sqcap \forall R_j.D'_i \\ O'_i &\sqsubseteq O_i \\ D'_i &\sqsubseteq D_i \end{aligned}$$

where $i \in \{1, .., n\}$, the parts of C'_i are also more specific, such that O'_i is more specific than O_i and D'_i is more specific than D_i.

Example 6. For the text in Figure 2.6 (see page 24) consider that SLI is capable of extracting a more specific film festival name out of a string segment, such that this example shows a specialization of *FilmFestival* given the capability of SLI to extract instances of *MoreliaFilmFestivalName*. Note that a *FeatureFilm* is also defined as an aggregate, which is characterized by a specific *Duration*. According to the domain of film festivals a feature film can not have a duration lower than 40 minutes. Moreover in the Morelia film festival only feature films are presented.

$$
\begin{aligned}
MoreliaFilmFestival &\sqsubseteq FilmFestival \\
&\quad \sqcap \forall hasFestivalName.MoreliaFilmFestivalName \\
&\quad \sqcap \forall takesplaceIn.Morelia \\
&\quad \sqcap \forall presents.FeatureFilm \\
FeatureFilm &\sqsubseteq Film \\
&\quad \sqcap \forall hasDuration.FeatureFilmDuration \\
MoreliaFilmFestivalName &\sqsubseteq FilmFestivalName \\
Morelia &\sqsubseteq CityName \\
FeatureFilm &\sqsubseteq Film \\
FeatureFilmDuration &\sqsubseteq Duration
\end{aligned}
$$

In this example is possible to note that during ontology design, for the support of DLI processes, the combination of pattern application (based on grounded ontology design) and domain knowledge for the definition of aggregates is advantageous as follows. Provided the knowledge of the ontology engineer about the corpus content and the capabilities of SLI processes, the ontology engineer knows if certain information needs are fulfilled e.g., the extraction of more specific observations such as *MoreliaFilmFestivalName*. Provided such knowledge, the ontology engineer can follow this pattern to create aggregate specializations and use universal restrictions to impose downward information flow to other parts of the aggregate, as long as such restrictions are coherent w.r.t. the domain knowledge. Given specific observations and downward information flow, it is possible to use reasoning to extract further information that otherwise is not possible to extracted by SLI processes, e.g., SLI not being able to extract observations about specific types of duration. In this way, reasoning on DLI results allows information gain by making SLI results more specific. The next step is to ensure the use of more specific observations to create explanations, for this purpose Pattern 8. is suggested.

Pattern 8. Specific Abduction Rule. This pattern ensures that during abduction, more specific explanations are created in case more specific observations are extracted. In the context of media interpretation, the computation of more specific explanations results in higher abstraction which is desired.

Definition Specific Abduction Rule. *An abduction rule (2) produces a more specific explanation than another rule (1), iff they have the following forms:*

$$\begin{array}{lcl} & (1) & \\ OR_i(X,Y) & \leftarrow & C_i(Z), \\ & & R_i(Z,X), \\ & & O_i(X), \\ & & R_j(Z,Y), \\ & & O_i(Y) \end{array} \qquad \begin{array}{lcl} & (2) & \\ OR_i(X,Y) & \leftarrow & C'_i(Z), \\ & & R_i(Z,X), \\ & & O'_i(X) \\ & & R_j(Z,Y), \\ & & O_i(Y) \end{array}$$

C'_i are aggregate concepts and C'_i is more specific than C_i ($C'_i \sqsubseteq C_i$) in the ontology. O'_i is a specific observation that motivates the creation of a more specific rule (2), such that $O'_i \sqsubseteq O_i$. R_i is a specific SLR.

Example 7. The rule shown in this example produces a more specific explanation than the rule in Example 3, given that an instance of *MoreliaFilmFestivalName* has been extracted.

$$\begin{array}{lcl} filmFestivalNameToCityName(X,Y) & \leftarrow & MoreliaFilmFestival(Z), \\ & & hasFilmFestivalName(Z,X), \\ & & MoreliaFilmFestivalName(X), \\ & & takesplaceIn(Z,Y), \\ & & CityName(Y), \end{array}$$

As discussed in Section 2.5, if two explanations have the same preference score, then the most specific one is the preferred explanation following the informativeness constraint (see page 50). Assume that the observations contain an instance of a *MoreliaFilmFestivalName* in the *filmFestivalNameToCityName* relation with a *CityName* instance. Both rules in Example 7 and Example 3 are applicable. The resulting explanations have the same preference score but, the explanation generated by the rule in Example 7 is more specific and is preferred. However, if the observations contain a *FilmFestivalName*, then the explanation generated by the rule in Example 3 is preferred because it has a higher preference score. In fact, the explanation generated by the rule in Example 7 hypothesizes the *FilmFestivalName* to be a *MoreliaFilmFestivalName*, which is penalized with a decrease in its preference score.

Pattern 9. $\leq$ Number Restrictions. This pattern ensures that during abduction, a new instance for a DLC is hypothesized, whenever there exists already a instance of a DLC that fulfills the number restrictions. For example, consider the text interpretation Abox from Figure 2.23 (see page 64). Provided the definition of the DLC concept *Person*

$$\begin{array}{lcl} Person & \sqsubseteq & DLC \sqcap \exists_{\leq_1} hasName.\top \sqcap \forall hasName.PersonName \\ & & \sqcap \exists_{\leq_1} hasNationality.\top \sqcap \forall hasNationality.CountryName \\ & & \sqcap \exists_{\leq_1} hasPart.PersonBody \\ & & \sqcap \exists_{\leq_1} hasPart.PersonFace \end{array}$$

and two instances of person names $pName1 : PersonName$ and $pName2 : PersonName$ corresponding to two different segments in the text, then two different individuals of type $Person$ are hypothesized and related to each of the names through a role $hasName$. Thus, according to the definition of the concept $Person$ at most one role $hasName$ should exist related to an individual of type $Person$.

$\leq$ Number Restrictions. *Let $\mathcal{T}$ be a Tbox in $\mathcal{O}$, C is a specific DLC concepts and R is a specific DLR*

$$C \quad \sqsubseteq \quad DLC \sqcap \exists_{\leq_n} R.\top$$

where n represents the cardinality of the numer restriction depending on the domain of interest.

Example 8. The concept $MoreliaFilmFestival$ defined previously (see page 3.2) takes place in only one city. Therefore a number restriction on the role $takesplaceIn$ is added such that the following description is obtained

$$\begin{aligned} MoreliaFilmFestival \quad \sqsubseteq \quad & FilmFestival \\ & \sqcap \forall hasFestivalName.MoreliaFilmFestivalName \\ & \sqcap \forall takesplaceIn.Morelia \\ & \sqcap \exists_{\leq_1} takesplaceIn.\top \\ & \sqcap \forall presents.FeatureFilm \end{aligned}$$

3.3 DLI for Knowledge Management Services

Knowledge management is defined in this work as the systematic means to support engineers in knowledge modeling tasks. The work presented in this chapter contributes to knowledge management by proposing, on the one hand side, a concept of grounded ontology design, and on the other hand side, ontology and rule design patterns.

Grounded ontology design provides a basis for the development of applications, as done by [CEF+09, CEF+07], that support the evolution of an ontology in a semiautomatic way. The concept of grounded ontology design represents a cycle. In a first iteration, the task of ontology design takes place as a task that is grounded on media content in order to identify requirements on terminological elements. Once a "starting" ontology is produced, the next step is the execution of media interpretation. By using the results of interpretation, new design requirements can be identified such that a new cycle starts with the design of further axioms that evolve an ontology. The work of Castano et al. [CEF+09, CEF+07] shows that grounded ontology design is helpful to support the creation of tools that support the semi automation of ontology design processes. In their work, the identification of design requirements, based on the results of interpretation processes, is achieved by searching

for segments of content that are left without interpretation, and by searching for SLI results that have no relation to DLI results, or have ambiguous interpretations. The results of these search tasks allow a system to alert knowledge engineers such that the required extensions to the ontology are undertaken. From the description of the concept of grounded ontology design in Section 3.1, it is possible to recognize that, in a context of continuous media production, the representation of new content semantics are required over time, and a cyclic process of design is required that is grounded on media content. The concept of grounded ontology design fulfills this need.

The ontology and rule design patterns presented here are based on the concept of grounded ontology design. Different to other existing patterns, which are focused on issues such as language expressivity, specific domain knowledge [Gan05],[Woh00] or the reuse of upper ontologies [BLC96] or the definition of competency questions [GF94], the patterns proposed here are useful to support media interpretation with DLI. The use of competency questions as proposed in [GF94] can be adopted to complement the patterns presented here as a starting point before the definition of axioms. Different to these patterns, the ones proposed here are formal, i.e., specific axioms are proposed, and specially designed to serve the requirements of the DLI framework.
In this work, the following thesis is stated:

> **Thesis:** *An ontology built with the design patterns proposed here (therefore that comply with the concept of grounded ontology design) is useful not only for media interpretation, but it is also useful for other applications that deal with media, such as content management tasks. Formal semantics ensures that the way formulas are written does not matter.*

DLI contributes to KM in guiding the task of designing an ontology. The following chapter describes the usefulness of content descriptions to support CM. Since many content management tasks use content descriptions for the retrieval of media, reasoning services w.r.t. a domain ontology are used.

In this way, by describing content management tasks that exploit content descriptions and domain ontologies, we want to show that the previous thesis can be stated.

Chapter 4

Content Management

Content Management (CM) is widely described [Nak01],[SM03] as a compound of strategies, methods and tools which aim to support the management of unstructured content in tasks such as content creation, edition, storage, versioning and publication. These tasks are supported by computer-based tools and methods implemented in so-called Content Management Systems (CMSs). The main application scenario of CMSs has been the maintainance of large scale corporate websites and intranets, which have been used as interfaces of ISs within companies. As described in Chapter 2, the integration of structured information and media to interfaces of ISs, has influenced the expectations of end users. Thus, users expect that classical GUI-based interaction scenarios in application programs are seamlessly extended with media shown in situation-specific ways. Hence, new means of content management are required, namely the management of content on the basis of content semantics.

In [Rob], Robertson describes further requirements which are imposed on CM according to the business goals of an organization:

1. Increase flexibility of the website interface such that it can quickly adapt to match new products and services.

2. Improve customer experience such that the website interface is easy to use.

3. Support marketing since websites have become an important place for business marketing.

4. Provide for powerful searching and browsing techniques to cope with information overload.

In this chapter, we argue that to fulfill these requirements imposed on CM, content-based services should be developed that exploit content descriptions which have the characteristics proposed in Chapter 2. To prove this, the following section describes specific

content-based services exploiting content descriptions that can be developed to fulfill the requirements above.

With the description of specific content-based services, we will see that the tasks the services need to execute can be summarized in: search tasks to find related media, content-based retrieval of media, and the invocation of application programs bases on content semantics. Content Management is defined in this work as follows.

> *Content Management means the capacity to search, retrieve and use media based on content semantics.*

4.1 Content-based Services to Support CM

Relevant work has been done that can contribute to cope with the requirements set on CM today (see previous section). Although they were not developed specifically to support CM, they are mentioned here because they inspired the content-based services presented in the following subsections. Therefore, we start this section by shortly describing these works and by explaining how they can fulfill the requirements on CM. Afterwards, we will be able to describe the specific content-based services in the following subsections.

The work in [McK91] provides an example of how interfaces can quickly adapt in order to provide for the services required by an end-user, thus, a scenario that is useful compatible with Goal 1 (see above). In [McK91] the Common Lisp Interface Manager (CLIM) is described which uses so-called presentation types. Presentation types are an extension of the Common Lisp Object System (CLOS). Presentation types in CLIM are associated to graphical output and are useful to determine if a certain graphical operation is applicable when an object of a certain type is requested provided certain input. Inheritance mechanisms of object-oriented systems are exploited on top of presentation types in order to determine applicability of a service.

Inspired by the work in [McK91], and motivated by the demands on the quick adaptation of a web-based interface to match new services, Section 4.1.3 describes a scenario called *Dynamic identification of applicable services*. This scenario explains how the services offered by a web-based interface are adapted to the content semantics of the media being rendered. It is highlighted that, provided the existance of annotations of media content expressed with content descriptions, reasoning services can be exploited to dynamically determine, from a set of predefined services, which services are applicable. Applicable services are used to determine the items that a common graphical object, e.g., context menus, should make accessible to the end-user.

Regarding Goal 2 (see page 99), various services can be thought of to improve the end-user experience of a corporate websites, but one that is currently highly demanded

are location-aware services. Various tools have been developed to support location-aware services, for example, Google Earth, Yahoo Maps, OpenStreetMap. The success of these tools has proved how appealing it is for end-users to search for media on the basis of geographic references. But current location-aware services work on the basis of manually provided metadata, e.g., geographic reference names. This chapter describes a scenario called *geography-aware information navigation*, which uses content descriptions to solve the following: it uses DLI results to solve ambiguities during geotagging, it allows to geotag not only text that contains geographic reference names in its surface, but also to geotag image content, thanks to the abstract character of DLI and fusion results.

Another scenario proposed here is called *content activation*. It proposes alternative means of interaction besides common graphical objects found in standard web-based interfaces. This is done by activating specific segments of the content to facilitate further interaction. The purpose is that annotations related to specific segments of content are used to determine the services that can be offered and that are suitable w.r.t. the content semantics of the active content. Thus, this scenario requires content management that exploits content semantics and more precise information about content structure, e.g., specific content segments.

Content activation is now starting to find acceptance in the market mainly for advertisement purposes on top of text, so-called *in-text advertisement*, which is a scenario that can be used to cope with Goal 3 (see page 99). Examples of this scenario can be seen in web pages from HP, Microsoft, NEC, Softonic and others. There are various companies (e.g., VibrantMedia[1], Kontera[2] to name a few) that provide services for the implementation of active textual content, focused on advertisement. Different from their services that rely on the use of surface-level information, the scenario on content activation presented in Section 4.1.2 also uses deep-level information in order to provide for more specific services (including advertisement services).

Regarding Goal 4 (see page 99), the research done in the context of the Semantic Web is relevant, since at the heart of the Semantic Web is the objective of providing machine-processable descriptions of web content in order to support search tasks. An example of Semantic Web-related work is the browser plug-in tool called ClearForestGnosis[3]. With it, words on a web page are annotated and activated for interaction purposes. Context menus are used for interaction that offer services to transfer the annotations, in the form of strings, to other web portals, e.g., news agencies, wikipedia and search engines, such that the transferred string can be used to find further information on other portals. This process is described as *in-depth browsing*. But, the work presented here argues that in-

[1] http://www.vibrantmedia.com

[2] http://www.kontera.com

[3] Available at: https://addons.mozilla.org/en-US/firefox/addon/3999.

depth browsing is only achieved if reasoning on top of surface- and deep-level information is applied. Annotations about surface- and deep-level information help in reducing the semantic gap between the information need and the media content.

As can be observed from previous descriptions, the scenarios presented in the following section are inspired by current commercial interest in creating semantics-driven applications providing for situation-specific and location-aware scenarios. A description of the required architecture to implement these scenarios is provided in Chapter 5.

The following subsections describe situation-specific and location-aware scenarios called geography-aware information navigation, content activation and dynamic identification of applicable services. The scenarios will be described using screenshots of an application, called BOEMIE Semantic Browser (BSB), that implements these three scenarios. The application was developed according to the architecture introduced in Chapter 5.

4.1.1 Geography-Aware Information Navigation

Digital maps have nowadays become geographic databases that provide a common platform for various types of information with geographical references, e.g., museums, shops, offices, facilities, etc. Moreover, different types of digital content, e.g., audio, video, text documents, images, etc., are constantly being produced and are even publicly available on the web. As a result, various applications have been developed that use those geographic databases together with publicly available content to provide for geography-aware information navigation. Examples of such applications are Google Earth, Yahoo Maps, OpenStreetMap, etc., which are used by a variety of people for retrieving various pieces of information linked to a certain location. Examples of users are tourists interested in the various attractions of a city, as well as business people interested in commercial events happening in a city, etc. The wide acceptance of these applications proves how appealing it is for end-users to use geographical references on a map as anchors for browsing multimedia content, a scenario called here as *geography-aware information navigation*.

But this kind of applications demand geographic tagging. *Geographic tagging*, also called *geotagging*, refers to the process of attaching geographic information to digital content. Geographic information can be a *geocode* (longitude and latitude coordinates) or any *geographical reference* that can be characterized with a geocode, for example, the name of a place, a point of interest, a street name, etc. Geographic references are the basis to obtain coordinates from a geographic database. With the help of a geocode it is possible to associate the corresponding media with a specific part of a map.

Manual geotagging is a tedious and expensive activity. Therefore, some systems such as Google Earth, Flickr, etc., have to depend on manual annotation from hobby end-users. Hobby end-users can upload their own content and either attach it to a digital map or tag

their content with names that can be automatically characterized with a geocode later on, such that it can be accessed from other users of the system by specifying a related geolocation. This is a very attractive scenario for service providers since the expenses for manually annotating hundreds of multimedia content are very high. Thus, much of the content in these applications is limited to personal content of hobby end-users. This is called a hobby scenario where only a few content items are manually annotated per end-user. While hobby scenarios are plausible for on-line communities or for personal use, automatic geotagging is of special interest for professional scenarios in which information providers cannot rely on hobby end-users to annotate their content, and therefore are forced to pay for manual annotation work. Here the scenario is called professional, where information providers

- are owners of huge amounts of information and wish to provide geography-aware information navigation, e.g., news agencies,
- want to avoid manual annotation costs, and
- want to keep their rights over their information.

The automation of geographic tagging has been facilitated for visual modalities given the advances in Global Position System (GPS) present in many cellular devices, location-aware cameras and video recorders, etc., which, together with other context information such as time stamps, can be used by applications to organize the content, for example, to follow the path of a city tour. However, GPS information is not always precise, it just provides an approximation, i.e., geotags register the location of the photographer at the time of shooting and not the position of the object in the picture, such that for long distance shooting, GPS information is an approximation. But GPS technology is not appropriate for the geotagging of textual content. Thus, different to still pictures, in which usually their content is created on-site, i.e., the photographer is in the same place as the place shown in the picture, this is not the case for textual content.

Geotagging of textual documents consists in the attachment of geocodes to segments of text that results from the interpretation of its content. In the easiest case, the geocode is attached to a name entity, e.g., country name, monument name, lake name, etc. In a more complex case, the geocode is attached to a paragraph or to the whole document as a result of interpreting the text beyond pure named entity recognition. Thus, geotagging of text documents required for a scenario of geography-aware information navigation, needs more than a geographic reference recognized in the content. It requires a context that helps to determine how relevant the geographic reference is within the content, in order to be considered as basis for geolocalization. Sometimes a geographic reference is used to name a person, a nationality, the name of a novel, etc., such that, if used as

basis for the geolocalization of a textual document, geotagging results in lower precision. This section describes a study about the feasibility of using deep-level information in the form of content descriptions to support the automation of geographic tagging such that applications offering geography-aware information navigation services can be developed with minimum effort.

DLI for the Geotagging of Media

Figure 4.1 (see page 105) shows a screenshot of an application that implements geography-aware information navigation. The image shows the interface elements (bubble-icons) that are used to indicate that specific points of interest on the map have some multimedia content items available for retrieval. In the application, the problem of media geotagging is solved by exploiting the results from the three levels of media interpretation (see Figure 1.2, page 4) with the following steps:

1. SLI of text is used for the NE recognition of geographic references.
2. DLI results are used as context to determine the relevance of geographic references (previously identified) within the domain of interest and determine whether the geographic reference is to be considered or not as basis for geolocalization (how this is measured is explained in the following lines).
3. SLI and DLI results are obtained from the image content that is surrounded by the previously analyzed text.
4. The DLI results of text, containing geographic references, are fused with DLI results from image to obtain geotagging of image content.

These steps follow the general principle that provided the difficulty of translating low-level features from image into content descriptions, the geotagging of image content is based on the geographic references found in the surrounding text and the result of fusing image and text content. As explained above, in Section 2.6, DLI results are required for multimedia fusion.

Geolocalization means, to classify a media under a specific location. The basis to solve the geolocalization process is to identify the geographic references in textual content. With geographic references we mean the following:

- Geopolitical areas, for example, cities, countries, districts, states, towns, villages, etc. and
- Points of interest (POI) such as stadium names, sports POIs, etc.

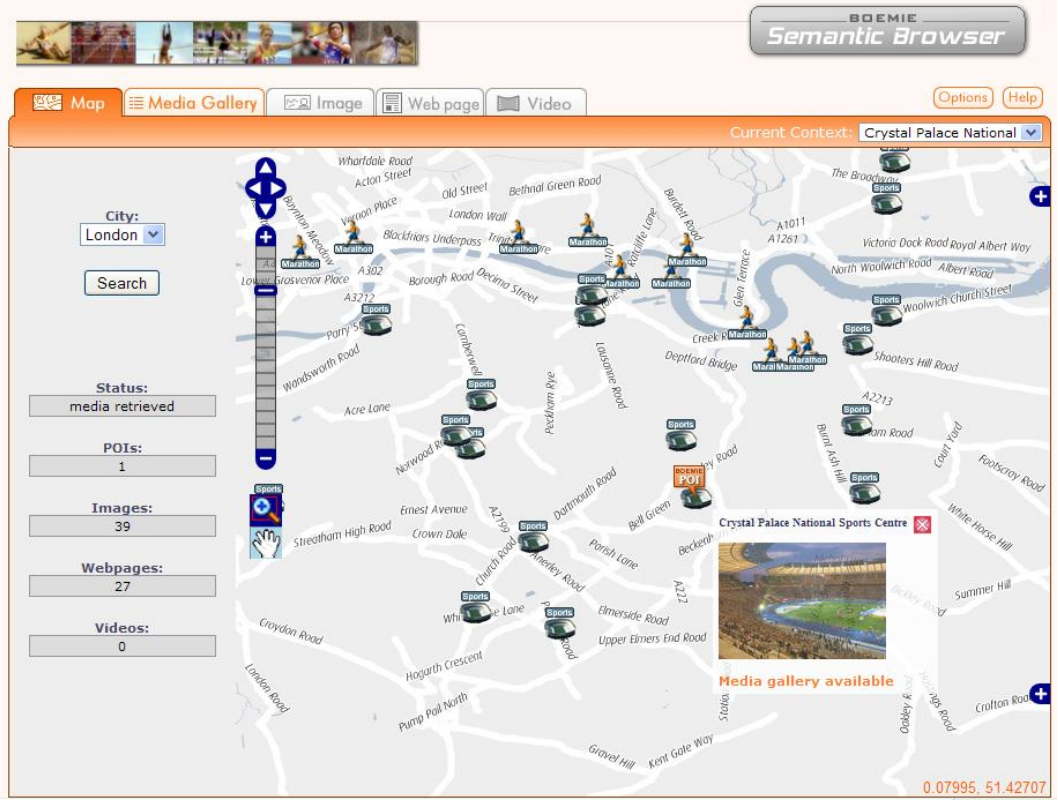


Figure 4.1: Geography-aware information navigation in the BSB.

Named entity recognition (NER) from textual content is an NLP technique that easily achieved by current NLP applications. On the other hand side, in the context of image analysis, translating low-level features to geographic references is hard to achieve. For this reason, as surveys on content-based image retrieval [SWS+00] and visual analysis techniques [PBG+07] suggest, the text that surrounds an image should be exploited to support the extraction of semantics from image content. This also applies for geographic tagging, such that geographic information in surrounding text is used for the geographic tagging of image content. In this work, the SLI and DLI results of text that surrounds image content are used as context for the geotagging of image content. The results of SLI from text that contain geographic references are the basis to search for coordinates on geographic databases. Currently, a great number of providers of geographic databases exist, such that geocodes can be obtained for common information e.g., address, city, states, or even for more specialized domains such as athletics points of interest, touristic points of interest, etc. The Getty Thesaurus of Geographic Names (TGN)[4] is a good example for a database available for education and research. TGN provides geographic names associated to various information such as coordinates, names in different languages, variant names, type (geopolitical, physical), etc. Other databases of the kind are GEOnet[5]

[4]See TNG in http://www.getty.edu/research/conducting_research/vocabularies/tgn/about.html
[5]See NGA GEOnet Names Server in http://earth-info.nga.mil/gns/html/

Monday, 18 May 2009

Vili sets 20.69m Oceania Shot Put record in Rio

Rio de Janeiro, Brazil - New Zealander Valerie Vili, the 2007 World and 2008 Olympic champion at Shot Put, produced the best performance of the "2009 Grande Prêmio Rio/Caixa de Atletismo", held at the "Cidade Maravilhosa" on Sunday 17 May, more precisely at the "Estádio João Havelange", also known as "Engenhão".

The **2009 Grande Prêmio Rio/Caixa de Atletismo** is one of a select group of Area meetings at which points can be acquired by athletes to qualify for the **IAAF / VTB Bank World Athletics Final,** to be held on **12-13 September in Thessaloniki, Greece.**

Vili set a new Oceania record with a put of 20.69m, which is also the best mark in the world in 2009. Vili had the previous 2009 lead with 20.25 (Waitakere 20 February).

Belarus' Natallia Mikhnevich, the Olympic silver medallist was also second in Rio with 19.48 in her first outdoor competition of 2008. Cuban Misleydis González was third with 18.93 (12cm of her SB), while Cubans Mailín Vargas (4th with 18.68) and Yaniuvis López (6th with 18.23), and Trinidad & Tobago's Cleopatra Borel-Brown (4th with a SB of 18.52) were all over the 18m barrier.

Vili opened with 19.41, followed by 20.38, 20.69, foul, 19.69, and 19.89, in a very solid series. Her previous Area record was 20.54, set when winning the 2007 World title in Osaka. The 20.69 performance moves her up to the 39th position in the all-time lists, and also makes the 24-year-old New Zealander the 6th performer of the new millennium.

Maggi and Menéndez regain control

Brazilian Maurren Higa Maggi, who in Beijing 2008 became the first South American female to win an Olympic athletics title, took the Long Jump with a 6.85m leap (0.5 wind), to avenge her third place from the Doha GP.

Keila Costa made it 1-2 for the Brazilians when capturing the second place with a SB of 6.78/0.3. The performance qualified the 26-year-old to the 2008 World Championships in Berlin.

Valerie Vili puts Oceania record in Rio (Ismar Ingber/CBAt)

Figure 4.2: A web page about athletics events.

and GNIS[6]. However, as previously highlighted, identifying geographical references from a document is not enough for a precise geolocalization of a document or, in other words, its classification under a geographic reference.

Classification of documents is still an open challenge that has been approached by machine learning techniques, resulting in a research area called Named Entity Classification (NEC) where the number of occurrences of a given NE is used to determine the document's classification. Similar procedures can be also applied for geographic classification.

In this work, the results of DLI contribute as a complement to current NEC procedures. Thus, relational structures resulting from DLI are used as a context to determine the relevance of a geographic reference and therefore support the geolocalization of documents. This can be illustrated through an example. Consider the web page in Figure 4.2. In different sections of the text, geopolitical references are found, for example from the first paragraph, the following SLI results are obtained:

pn_1*:PersonName,* (pn_1, 'Valerie Vili')*:hasValue,*
p_1*:Performance,* (p_1, '20.69m')*:hasValue,*
sn_1*:SportsName,* (sn_1, 'Shot Put')*:hasValue,*
c_1*:CityName,* (c_1, 'Rio de Janeiro')*:hasValue,*

[6]See GNIS in http://geonames.usgs.gov/pls/gnispublic/f?p=139:1:1752191558295648

c_2:*CountryName,* (c_2, 'Brazil'):*hasValue,*
c_3:*CountryName,* (c_3, 'New Zealand'):*hasValue,*
se_1:*EventName,* (se_1, '2009 Grande Premio Rio/Caixa de Atletismo'):*hasValue,*
s_1:*StadiumName,* (s_1, 'Estadio Joao Havelange'):*hasValue,*
(pn_1, c_3):*personNameToCountryName,* (pn_1, p_1):*personNameToPerformance*
(sn_1, s_1):*sportsNameToStadiumName,* (se_1, sn_1):*eventNameToSportsName*
(se_1, c_2):*eventNameToCountryName,* (se_1, c_1):*eventNameToCityName*
(sn_1, p_1):*sportsNameToPerformance*

And from previous SLI results, the following DLI results are obtained considering a background knowledge based composed of the ontology in Appendix A and the rules in Appendix B.

new_1:*Person,* ($new_1, pn1$):*hasName,* (new_1, c_3):*hasNationality,*
(new_1, p_1):*personToPerformance,* new_2:*SportsTrial,* (new_2, new_1):*hasParticipant,*
(new_2, p_1):*hasPerformance,* new_1:*Athlete,*
new_3:*SportsCompetition,* (new_3, sn_1):*hasName,* (new_3, s_1):*takesPlaceInSportsPOI,*
new_4:*SportsEvent,*(new_4, se_1):*hasName,* (new_4, c_2):*takesPlaceIn,*
(new_4, c_1):*takesPlaceIn,* (new_4, sn_1):*sportsEventToSportsName,*
(new_4, new_3):*hasPart,* (new_3, p_1):*sportsCompetitionToPerformance,*
(new_3, new_5):*hasPart,* new_5:*SportRound,* (new_5, new_2):*hasPart,*
web_1:*WebPage,* txt_1:*Text,* (web_1, txt_1):*contains,*
(txt_1, new_1):*depicts,* (txt_1, new_2):*depicts,*
(txt_1, new_3):*depicts,* (txt_1, new_4):*depicts,* (txt_1, new_5):*depicts*

From this paragraph, five aggregate instances are obtained. A person ($new_1 : Person$), a sports trial ($new_2 : SportsTrial$), a sports competition ($new_3 : SportsCompetition$), a sports event ($new_4 : SportsEvent$) and a sports round ($new_5 : SportsRound$). In the second paragraph, another athletics event is described, namely the 'IAAF / VTB Bank World Athletics Final' taking place in Thessaloniki. From the second paragraph similar SLI and DLI results as the ones above can be obtained. Finally, in the fourth paragraph, various country names are found, producing SLI and DLI results similar to the following:

c_4 :*CountryName,* (c_4, "Belarus"):*hasValue,*
pn_2 :*PersonName,* (pn_2, "Natallia Mikhnevich"):*hasValue,*
(pn_2, c_4):*personNameToCountryName,*
new_6:*Person,* (new_5, pn_2):*hasName,* (new_5, c_4):*hasNationality*
(txt_1, new_6):*depicts*

As can be seen from the previous examples, in the athletics domain, some geographic references represent the nationalities of athletes and others represent the place where an athletics event took place. In order to decide which of those geographical references should be used as basis to geotag the web page, the domain of interest should be considered as

follows. In the athletics domain, various aggregates can be extracted as a result of DLI, e.g., athletics events, sport competitions, sport rounds, sport trials, athletes, persons, etc. From all those aggregates, the *SportsEvent* aggregate is the most relevant one, since it is composed of all the other aggregates. As the terminology in Appendix A shows, a sports event is composed of competitions, competitions are composed of rounds, rounds of trials and trials of athletes. Thus, the *SportsEvent* aggregate is the top most aggregate composed of all the other ones. In this approach, the geographic reference that is part of the most relevant aggregate in the domain of interest, is the one that determines the geolocalization of the content. For the current example the web page in Figure 4.2 can be classified under three different locations, namely 'Rio de Janeiro', 'Estadio Joao Havelange' and 'Thessaloniki'. Note that the name of the stadium provides a more precise geocode than a city name, such that the position of the stadium can be highlighted on the map through an icon. For this reason, if there exists a second geographic reference that is in a *contains* relationship with the most relevant reference, e.g., a city contains a stadium, then the second reference is used since it increases the precision of the geolocalization. This works as long as the ontology is built according to the patterns explained in Section 3.2.

This classification strategy is useful also for the geolocalization of images found in multimedia objects, e.g., a web page, with the use of fusion. As previously described in Section 2.6, fusion uses the results of DLI from different modalities. DLI results provide the necessary abstract information that place observations of a different nature (disjoint w.r.t. the domain of interest) into the same context represented through aggregate instances. In this way, fusion finds aggregates extracted from different modalities that represent the same entity in the real world. To continue with the example and show the advantages of fusion for the geotagging of image content, consider the DLI results of the captioned image in Figure 4.2 (see page 106). From the caption of the image the following is obtained:

pn_2*:PersonName,* (pn_2, "Valerie Vili")*:hasValue,*
c_4*:CityName,* (c_4, "Rio")*:hasValue,*
c_5*:Continent,* (c_5, "Oceania")*:hasValue,*
new_6*:Person,* (new_6, pn_2)*:hasName,*
ci_1*:CaptionedImage,* c_1*:Caption,*
(web_1, ci_1)*:contains,* (ci_1, c_1)*:contains,*
(c_1, new_6)*:depicts*

From the image, the following DLI results are obtained:

pb_1*:PersonBody,* pf_1*:PersonFace,* s_1*:Shot,*
(pb_1, pf_1)*:adjacent,* (pb_1, s_1)*:isNear,*
new_7*:Person,* (new_7, pb_1)*:hasPart,* (new_7, pf_1)*:hasPart,*

(new_7, s_1):*isNear,* new_8:*ShotPut,* (new_8, new_7):*hasParticipant,*
new_7:*ShotPutter,* (new_8, s_1):*hasPart,*
i_1:*Image,* (ci_1, i_1):*contains,*
(i_1, new_7):*depicts,* (i_1, new_8):*depicts*

From applying the fusion process (see rules in Figure 2.32, page 77) on the instances describing the caption and the image content, the assertion (new_6, new_7) :same-as, is obtained, and, finally from applying fusion on the instances of the describing the captioned image and the text content, (new_6, new_1) :same-as, is obtained. Note that the first same-as assertion makes the DLI results from caption more precise, since it converts the individual new_6 into an instance of a more specific concept, namely a *ShotPutter*. Moreover, from the caption two geographic references are identified. But, given that no role assertion involving any of them exists, a domain specific context for each of them is not identified (see DLI results above). This makes geolocalization ambiguous, since both references could be used for the geolocalization of the image. In this way, only until fusion between the captioned image and the text is accomplished ((new_6, new_1) :same-as), a geographic reference for the image is found. To be more specific, individual new_7 (from the image content) is first fused with individual new_6 (from the caption content) and later fused with individual new_1 (from the textual content), such that at the end new_7 is the instance of a participant of a sports trial in a competition in 'Estadio Joao Havelange', which is part of an event taking place in 'Rio'. In this way the image is geolocalized in 'Rio' and, to be more precise, in the stadium 'Estadio Joao Havelange'.

Thus, the results of fusion involving geographic references are useful to extract complementary information, which, if considered isolated, would not give enough hints to recognize the geographic location of modality specific content. Finally, provided the geographic reference, a standard map service can be used to obtain the coordinates necessary to associate the media object with a specific coordinate on a map.

Using geographic ontologies for information navigation

As described in the previous sections, ontologies are used as an infrastructure for the annotation of content. In the case of geographic tagging, a geographic ontology can be used. For this application, the GIO [DEDT07] ontology is used. In this ontology only thematic aspects of geography are defined, such as geopolitical areas, man-made points of interest (such as harbors, marinas, stadiums, tunnels). The GIO ontology is used to query for content with a geographic reference, and during query processing, subsumption reasoning is exploited. This process can be illustrated with an example. As previously introduced, names for geographical references such as museums, shops, venues, offices, facilities, etc., can be found in media content, as in the following sentence:

"Beijing's Bird's Nest stadium opens with IAAF Race Walking Challenge competition".

Bird's Nest is the name of a geographical reference, i.e., a stadium, and "Beijing" is also a geographical reference, i.e., a city. These references are annotated with the corresponding terminology of the geographic ontology. In this way, according to the excerpt of the geographic ontology in Figure 4.3, the information is annotated in the form of Abox assertions as shown in Figure 4.4. It is described that a "Bird's Nest" is the name of a stadium that is located in the city of "Beijing" .

$$
\begin{aligned}
\mathit{GeographicObject} &\sqsubseteq \mathit{Thing} \\
&\quad \sqcap \exists \mathit{isLocatedIn}.\mathit{Location} \\
&\quad \sqcap \exists \mathit{belongsToTimeZone}.\mathit{TimeZone} \\
&\quad \sqcap \exists \mathit{directionalRelation}.\mathit{GeographicObject} \\
&\quad \sqcap \exists \mathit{topologicalRelation}.\mathit{GeographicObject} \\
&\quad \ldots \\
\mathit{POI} &\sqsubseteq \mathit{GeographicObject} \\
\mathit{SportsPOI} &\sqsubseteq \mathit{POI} \\
\mathit{StadiumName} &\sqsubseteq \mathit{SportsPOI} \\
&\quad \sqcap \exists \mathit{hasStadiumNameNameValue}.\mathit{string} \\
\mathit{GeopoliticalArea} &\sqsubseteq \mathit{GeographicObject} \\
\mathit{CityName} &\sqsubseteq \mathit{GeopoliticalArea} \\
&\quad \sqcap \exists \mathit{hasCityNameNameValue}.\mathit{string}
\end{aligned}
$$

Figure 4.3: An excerpt of the geographic ontology.

According to the ontology a *StadiumName* is a type of "sports point of interest" (*SportsPOI*), and a sports-point-of-interest is a specific type of point of interest (*POI*). Therefore when querying for all points of interest in Beijing, the $StadiumName_1$ is retrieved.

```
<gio:StadiumName rdf:ID="StadiumName_1">
   <gio:hasStadiumNameNameValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
Birds Nest
   </gio:hasStadiumNameNameValue>
   <gio:isLocatedIn rdf:resource=#CityName_1>
</gio:StadiumName>

<gio:CityName rdf:ID="CityName_1">
   <gio:hasCityNameNameValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
Beijing
   </gio:hasCityNameNameValue>
</gio:CityName>
```

Figure 4.4: Geographic annotations.

4.1.2 Content Activation

The previous section has motivated that interacting with specific points of interest on a map for information navigation is appealing and well accepted by end-users. In a similar way, specific segments of media content can be used as anchors for further interaction, a scenario named here as *content activation*. To achieve this, information about the structure of media content, obtained from SLI processes, is exploited. The objective is to use the media annotations and related information about segment descriptions, e.g., start- and end-offset of a word in a text, polygon information of an object in an image, to make media content active to support interaction. The annotations associated with a segment can be used for the following purposes:

- Providing for information about the segment's related content semantics, e.g., objects in images or words in text.
- To retrieve related media, where DLI results help as a basis to define a specific IR query.

Suppose that the image in Figure 4.5 has been accessed through the image gallery of a sportsnews portal, and segments of it are activated such that users can further interact with the content. The interaction with the content could activate a context menu which offers different services such as:

- Suggest related media, e.g., further images of the same person participating in high jump trials.
- Advertisements, e.g., showing the price, model and producer of the sport shoes used by the athlete.

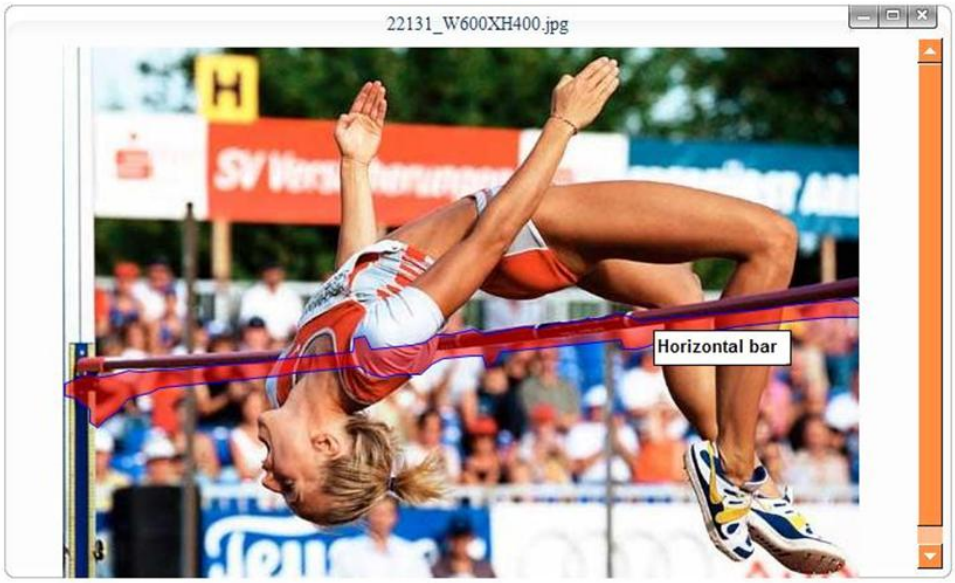


Figure 4.5: Activation of content given image SLI results.

The work that is necessary to obtain information about content structure, i.e., segment descriptions, such as polygon coordinates in an image or character position in a text is demanding if done manually. Moreover, the association of structure descriptions with content descriptions is required. For this reason SLI results are useful since, in the process of SLI necessary structure information is obtained. As described before, interaction with active media content is now starting to find acceptance in the market, mainly for advertisement purposes on top of text, so-called *in-text advertisement* (see Figure 4.6).

With the usage of DLI and fusion results, situation-specific services can be activated per segment for further interaction, a scenario which is explained in the following section.

4.1.3 Dynamic Identification of Applicable Services

Currently, web-based interfaces present a variety of graphical objects to support the interaction of the end user with the application. Graphical objects such as menus, drop-down lists, buttons, check boxes, etc., commonly provide a static set of items. Such items invoke services to execute the corresponding code that implements the application's functionality. Thus, it is through those items that the interface provides access to the application's functionality. In the context of semantics-driven applications, the items presented by the application do not only reflect the application's purpose, but also the domain semantics of the media content being managed by the application. For example, currently, text editors allow users to insert different types of objects to help illustrate the text, e.g., pictures, clipart, shapes, graphics, etc. To allow this, a menu is commonly offered where each menu-item corresponds to the type of object to insert. If a text editor uses content descriptions, its functionality can be adapted to offer content-related services. In this way if

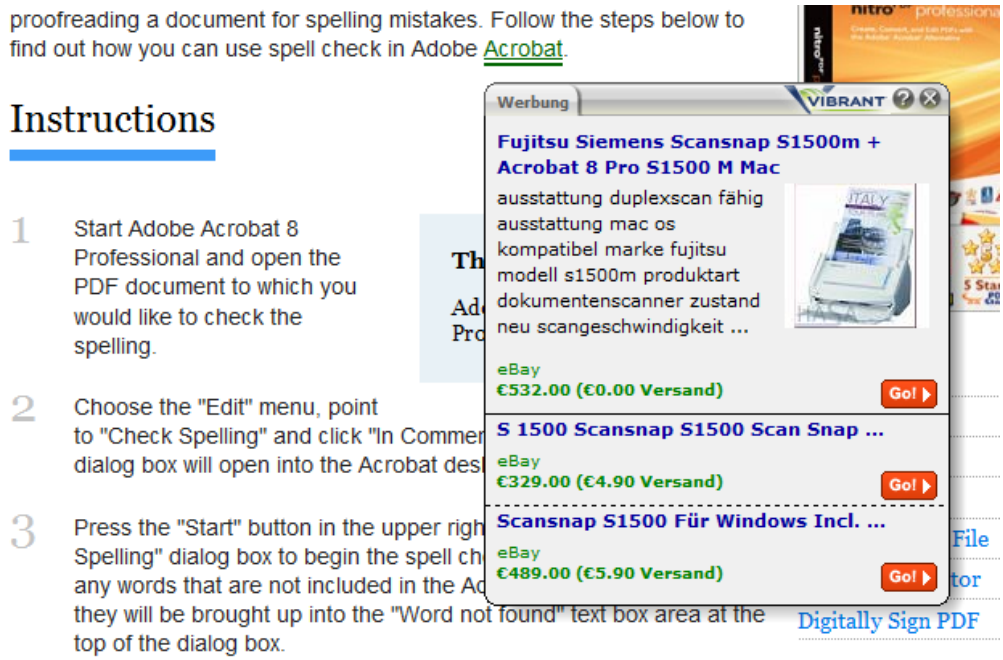


Figure 4.6: In-text advertisement.

the text content is related to the athletics domain and the user decides to insert an object, e.g., an image, then domain related objects are suggested, e.g., an image of an athletics event. If the domain of interest changes, then correspondingly the application services adapt to the domain. Thus, in the context of semantics-driven applications, services can be parameterized with information about the content semantics of media.

The objective of this section is to describe how the content descriptions obtained from SLI, DLI and fusion processes can be used to dynamically determine the association of content-based services (from a predefined set) with a graphical element of the interface, e.g., a context menu. To explain the process of association, we will use examples of the services implemented in the BSB application. The BSB offers browsing functionality over athletics news. To support this functionality a *search-by-association* approach is used. As described in [SWS+00], in search-by-association, at the beginning users have no specific aim other than finding interesting media. Search-by-association often implies iterative refinement of the search by using the similarity of the content with which the search started. This browsing approach is described later on in detail in Chapter 5.

In the BSB, text from web pages as well as images contain active content which is used by end users to interact with the application in order to access context menus. Each item of such context menus (see Figure 4.7) corresponds to a service (from a predefined set of services), which executes the corresponding code that implements the functionality of the application. The available services are dynamically assigned to a context menu according to the annotations related to the active content.

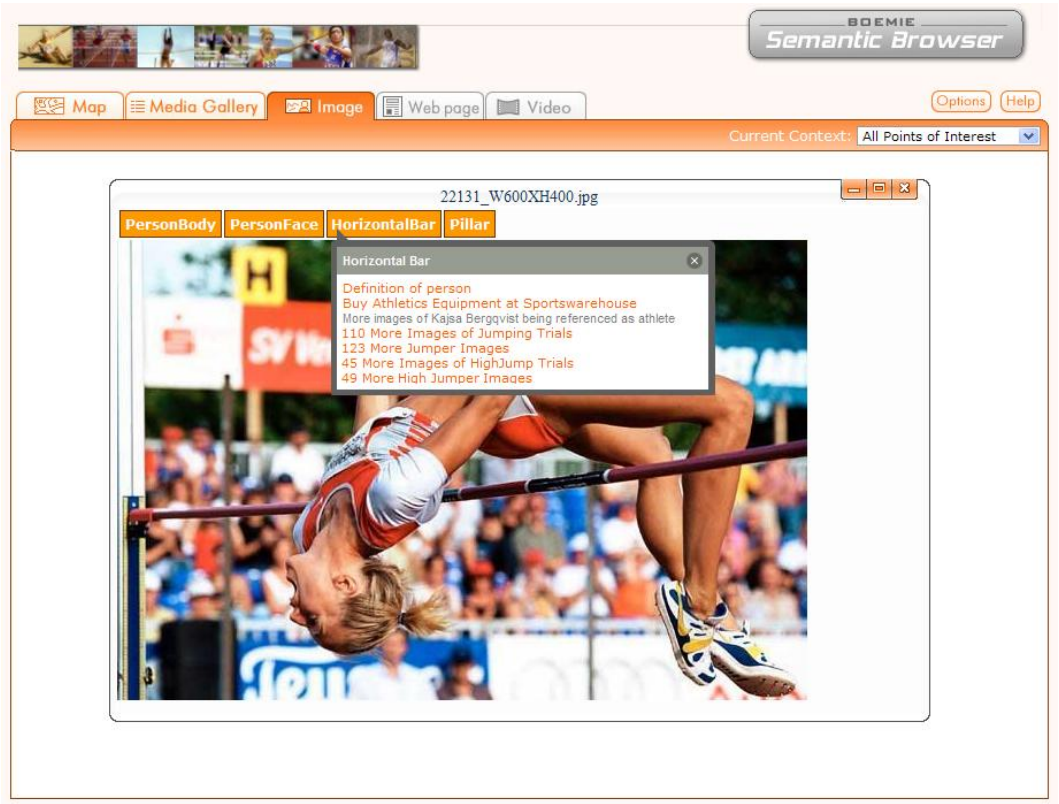


Figure 4.7: Context menus for active content in the BOEMIE Semantic Browser.

As Figure 4.8 shows (see page 115), the active content are such segments of media content that have been annotated. In this example, the highlighted segment in the image corresponds to an instance of *HorizontalBar*. Thus, segments are directly associated with SLI results, given that segments belong to explicit observations within media content. From SLI results a system can have access to DLI and fusion results by querying the corresponding Abox.

For example, Figure 4.9 shows a segment from text with related interpretation results. As shown, the highlighted segment corresponds to an instance of type *PersonName*. While this is obtained from SLI results, by observing the segment's surrounding text, it should be clear that DLI results should explain that the name is related to an instance of *Person* through a role *hasName*. Moreover, the instance of type *Person* is also of type *Athlete*, more specifically, a *JavelinThrower*. With the use of reasoning services, on top of interpretation results, top-down information flow can be obtained from DLI results that make SLI results more specific. In this example, the highlighted segment in Figure 4.9 is associated, implicitly, with an instance of *JavelinThrowerName* provided an axiom in the TBox such as $JavelinThrower \sqsubseteq \forall hasName.JavelinThrowerName$. In this way DLI results contribute in making SLI results more precise.

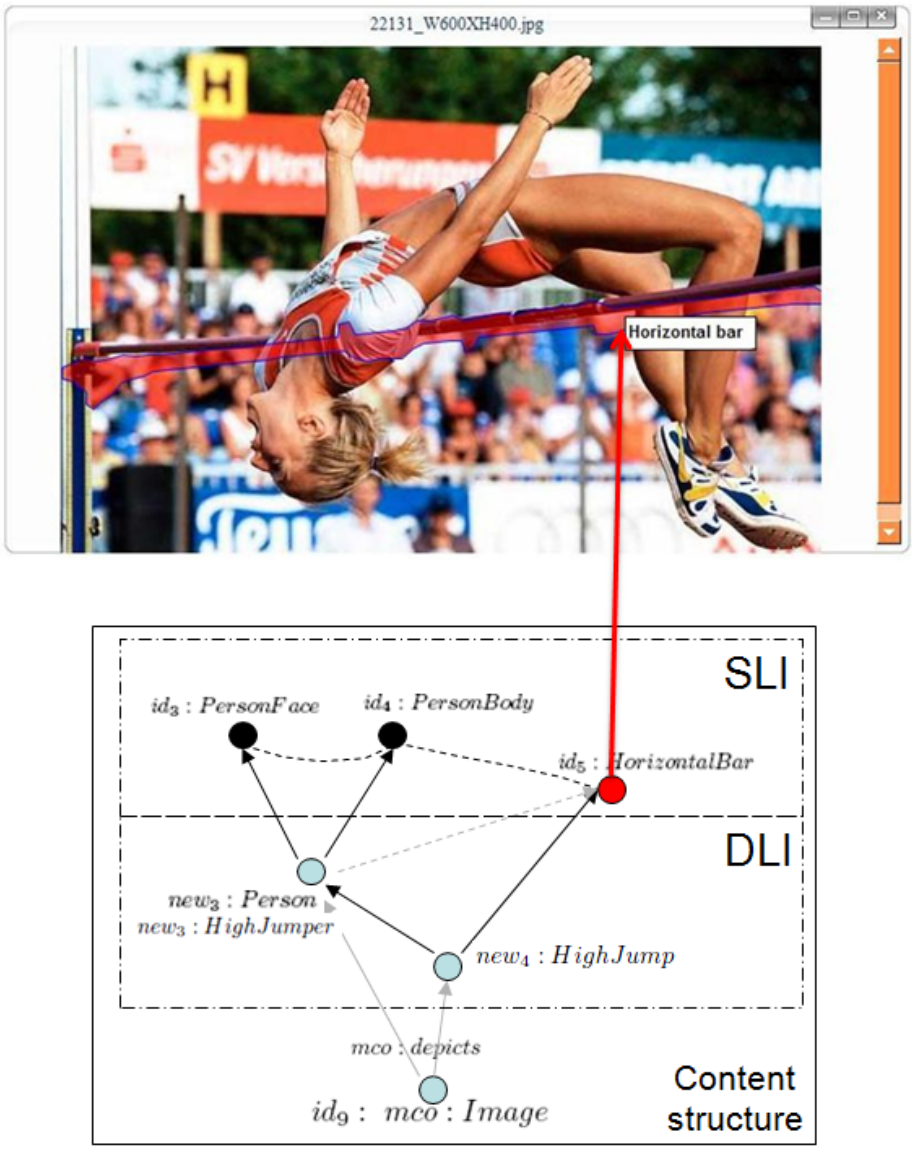


Figure 4.8: SLI and DLI results of an image.

The dynamic composition of context menu-items is determined by the content semantics related to the active content through annotations. The annotations are used for the identification of so-called applicable services. As Figure 4.7 shows, a context menu for the horizontal bar is accessed containing the following items:

1. Buy Athletics Equipment at Sportswarehouse
2. Definition of person
3. More images of Kajsa Bergqvist being referenced as athlete
4. More images of jumping trials
5. More jumper images

6. More images of high jump trials

7. More high jumper images

Each item corresponds to an applicable service. The applicability of a service is determined by the parameters that the service requires as input. Thus, input parameters are typed with terms from the domain ontology, e.g., the AEO ontology. Given the SLI results related to the segment in Figure 4.8, more specifically $id_5 : HorizontalBar$, different applicable services are discovered and shown as items in the context menu. The first item calls a service that requires an argument of type *AthleticsEquipment*. Thus, the first item is activated by reasoning on SLI results since according to the AEO ontology (see Appendix A in page 145) *HorizontalBar* is an *AthleticsEquipment*. The second item relates to a service that requires as input an argument of type *Person*, thus this service is applicable due to DLI results which involve an instance of type *Person* (see Figure 4.8).

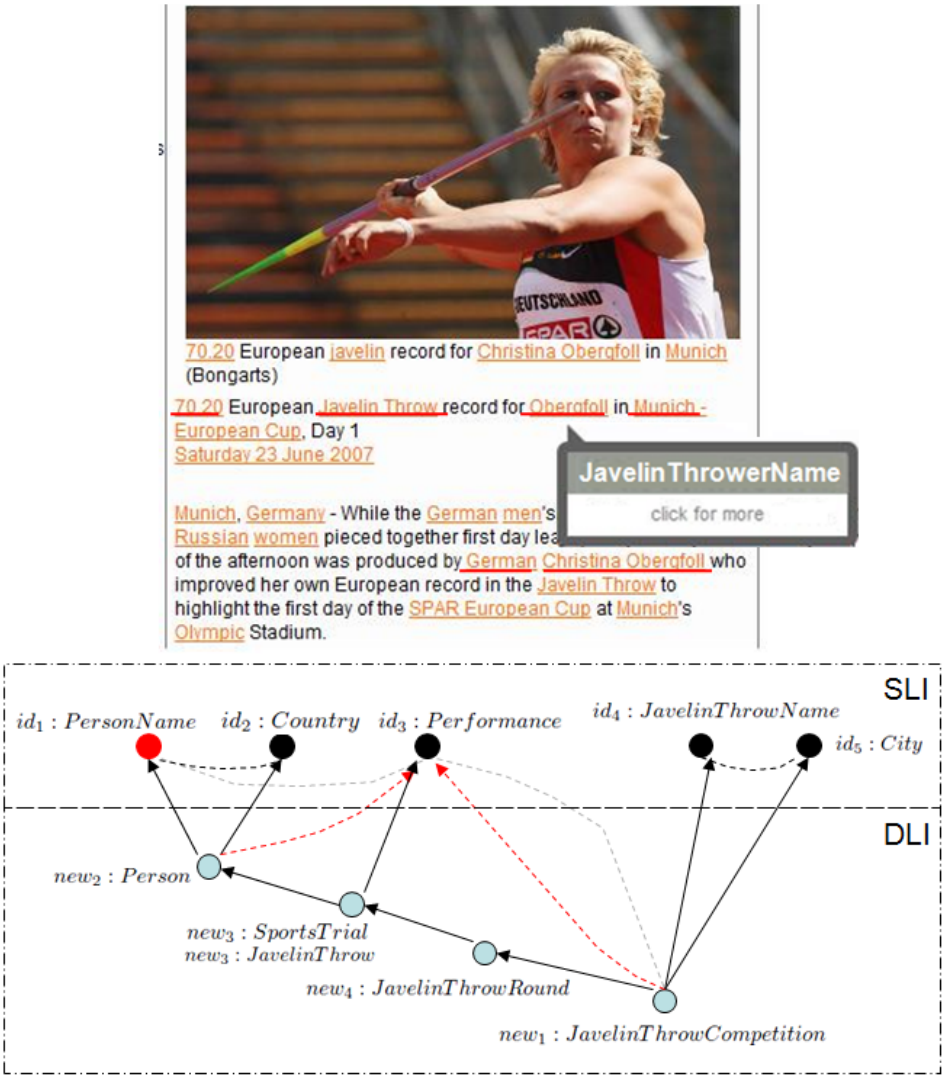


Figure 4.9: SLI and DLI results of text.

The third item relates to a service that requires an instance of type *Athlete* in relation to an instance of type *PersonName* with a specific value. In this case the name *Kajsa Bergqvist* comes from the fusion of individual new_3 with another instance coming from text (fusion results are omitted in Figure 4.8). The fourth item relates to a service whose argument is of type *Jumping*, thus it is applicable by reasoning on DLI results, since individual new_4 is of type *HighJump* and according to the AEO ontology *HighJump* is a specific type of a *Jumping* trial. The following three items require arguments of type *Jumper*, *HighJump* and *HighJumper* which are applicable by reasoning on the individuals new_3 and new_4 of Figure 4.8. In this way, the applicability of a service is determined by the annotations related to the active content previously accessed. A more specific description of how service applicability is implemented will be given in Chapter 5.

The general principle behind this scenario is that provided a set of services that use ontology terms to type their input parameters, as well as content descriptions associated to specific segments of content, reasoning can be used to find the association between services and media, and correspondingly configure the items of graphical elements in an interface that adapts to media content.

In the BSB application, the content-based services provided are primarily to support *web navigation*, *target search* and *activation of tools*. The first is useful for placing semantically related content that can be reached through a hypertext link. This functionality is suitable for *advertisement* purposes. The second is useful for browsing purposes to be more specific as a *target search*, thus the system suggests semantically related content, e.g., image, text or video, that can be reached through a query over a triple store repository. For example, the target search can be another image of the same object which the user has activated. Finally, the third service shows that semantics can be used as arguments to *access the application's internal methods*, for example, for the activation of the map tool. In the BSB, context menus are accessed through a click on the segments that indicate active objects of an image (see Figure 4.7) or active words in the text of a web page. Note that the relevance of this scenario relies on the dynamic composition of items and not on the specific graphical objects used. Thus, the same principle can be used to support other types of graphical objects, for example, to influence the content of side bars typically used for advertisement, items of drop-down lists, etc.

To explain the implementation of the scenarios described in this section in more detail, the following Section 5 describes the required architecture and the implementation of the BSB.

4.2 DLI for Content Management Services

Content management is defined in this work as the capacity to search, retrieve and use media based on content semantics. The main application scenarios of content management has been in the maintainance of corporate websites used as interfaces for ISs. These types of interfaces have imposed various requirements on content management. In this chapter we showed that these requirements are fulfilled if content descriptions are made available.

With the usage of content descriptions, content-based services can be built that support the quick adaptation of interfaces to match new services and new information, extend interaction scenarios to media content, and provide better search and browsing techniques. To prove this, specific services were described, and compared with similar ones available in the market. In this way, we showed how the DLI process and the usage of content descriptions allow to provide for better services than available today. Therefore contributing to CM.

In the following chapter an architecture is presented that shows how the content-based services presented here can be realized and, in this way, shows that the contributions of this work to CM are applicable in reality.

Chapter 5

A Software Architecture for Content-driven Web Applications

In this chapter a generic architecture is described to highlight the elements that an application requires in order to support situation-specific and location-aware services based on content descriptions. Figure 5.1 illustrates the architecture for the client and server side. On the client side a web application can use state-of-the-art web technologies, such as, AJAX technologies to support the communication with the server in order to access and retrieve multimedia as well as map information. On the server side, three modules are required to support content-based services such as the ones described in the previous chapter. To support location-aware services, digital maps and geography databases are required. For the implementation of the BSB, for example the *TeleAtlas*[1] Web Map Research Platform (WMRP) is used. From this platform three services are exploited:

1. **Geocoder** which provides geocodes (latitude and longitude coordinates) for the corresponding geographic reference, e.g., street name, city, country, etc.

2. **Web Feature Service** (WFS) which provides geocodes for particular points of interest in a city, e.g., sports, cultural, business point of interest, etc.

3. **Web Map Service** (WMS) which provides maps with specific layers of information, e.g., street names, hybrid layer, etc., in a specific area described by a bounding box (lower left corner coordinates, upper right corner coordinates).

To support content activation, services are required that use the segment descriptions obtained by SLI processes, from image and text analysis, to create hyperlinks on the segments of media content. To support the application's performance, these services should be used as offline preprocessing services. The media that has been preprocessed

[1] http://www.teleatlas.com

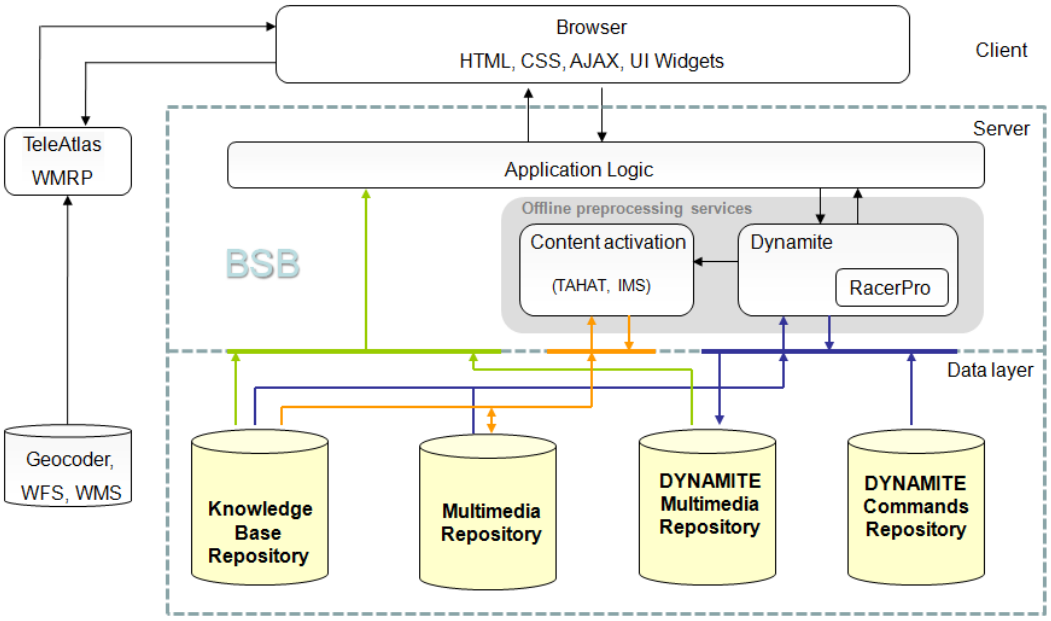


Figure 5.1: Overall architecture to support content-based applications.

by these services is stored in a specific repository (see "Multimedia repository" in Figure 5.1). In the BSB the following modules are used:

1. A module for the construction of hyperlinks on image content, which uses HTML image maps. The resulting files is hereafter called IMS (Image Map Service) file.

2. A module for the creation of hyperlinks on text content of web pages. This module uses a specific tool called TAHAT (Text Analysis HTML Annotation Tool) developed by [Pal09]. TAHAT uses the Ellogon open-source text engineering platform[2]. The resulting file is hereafter called TAHAT file.

To support the dynamic composition of services the following modules are used:

1. **DYNAMIc InTEractive web pages** (DYNAMITE) is a generator of Javascript code that shows menus at runtime. The script code extends the IMS and TAHAT files with the required script code. It works as an offline preprocessing service. The resulting file is hereafter called DYNAMITE file.

2. **RacerPro** which is the DL reasoner used by DYNAMITE to identify the applicable services that correspond to items of context menus.

[2]www.ellogon.org

For the data layer the following repositories are required:

1. A multimedia repository, which contains the media objects to be interpreted by SLI, DLI and fusion processes.

2. A knowledge base repository, containing ontologies and Aboxes about content descriptions for each media object found in the multimedia repository. The required ontologies are domain ontologies, e.g., *Athletics Event Ontology* (AEO)[DEDT07], a geographic ontology, e.g., *Geographic Information Ontology* (GIO), and an ontology to address the structural aspects of media objects, e.g., *Multimedia Content Ontology* (MCO).

3. A DYNAMITE Multimedia Repository containing DYNAMITE files. Queries are executed over this repository for the retrieval of media objects.

4. A DYNAMITE Service Repository which contains service definitions from which an application can look-up for applicable services to compose the items of context menus.

The following three sections will describe in more detail the implementation of each of the scenarios described in Chapter 4 according to the architecture presented here.

5.1 Implementation of Geography-Aware Information Navigation

In Figure 5.2, the interaction between the different modules used to support location-aware services is provided. The end user starts the interaction by using the map tool. Here, the user enters a city name to retrieve its map. The application uses three data services provided by the TeleAtlas (WMRP). First, the city name is sent as a parameter to the TeleAtlas geocoder in order to retrieve the corresponding geocode. The request is encoded in GML as the following example shows:

```
<?xml version="1.0" encoding="UTF-8"?>
<GeocodeRequest xmlns="http://www.opengis.net/xls"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:gml="http://www.opengis.net/gml"
        xsi:schemaLocation="http ://www.opengis.net/xls
        http://schemas.opengis.net/ols/1.1.0/LocationUtilityService.xsd">
<Address countryCode="GB">
<StreetAddress>
<Street> </Street>
</StreetAddress>
```

```
<Place type="Municipality">LONDON </Place>
</Address>
</GeocodeRequest>
```

The geocode which is composed of a longitude-latitude coordinate pair indicates some central point of the requested city. With this geocode the application can compute a bounding box (BBox) such that coordinates for lower-left corner and upper-right corner coordinates of the map can be specified. The BBox is required by both the WFS (Web Feature Service) and the WMS (Web Map Service). From the WFS the application can obtain a list of points of interest (POIs) and corresponding geocodes that are found within the BBox for a specific type of layer, e.g., tourism, business, athletics, etc. From the WMS the application obtains the specified map layers corresponding to the BBox. Examples of map layers are the street name layer, the satellite layer, etc.

Once the list of POIs and respective coordinates are obtained a call is send to the server side of the BSB in order to retrieve a list of POIs that can be found within the Ontology Repository. This means, that a list of POIs that have been extracted from multimedia content is obtained. Afterwards, a process starts to find out if any of the POIs obtained from the Ontology Repository are found also in the list given by the WFS. If so, then the list of common POIs and the list of respective geocodes is returned to the map tool on the client side. With this information a set of icons called BOEMIE POIs (see Figure 4.1) can be shown on the map. Finally, the Media Gallery section of the interface is activated and provides an overview of all the multimedia content that has been found either related to the city or to a specific POI of the city.

5.2 Implementation of Content Activation

Content activation is implemented by two services which provide for semantic annotation of the content with domain semantics and also for the activation of the corresponding segments within the content. As Figure 5.1 (see page 120) shows, these services operate in an offline mode to process the content found in the multimedia repository. Both services receive two elements as input, the media object such as an image or a web page and the associated Abox containing content descriptions. With these two elements the services can produce the annotated version of the media object. The annotations consist of individual IDs found in the input Abox, the IDs are encoded with HTML technology with the corresponding segments offsets for objects of an image or words in a text. These services also encode a set of gesture handling events to trigger scripts that support the highlighting of segments as Figure 4.5 (see page 112) shows.

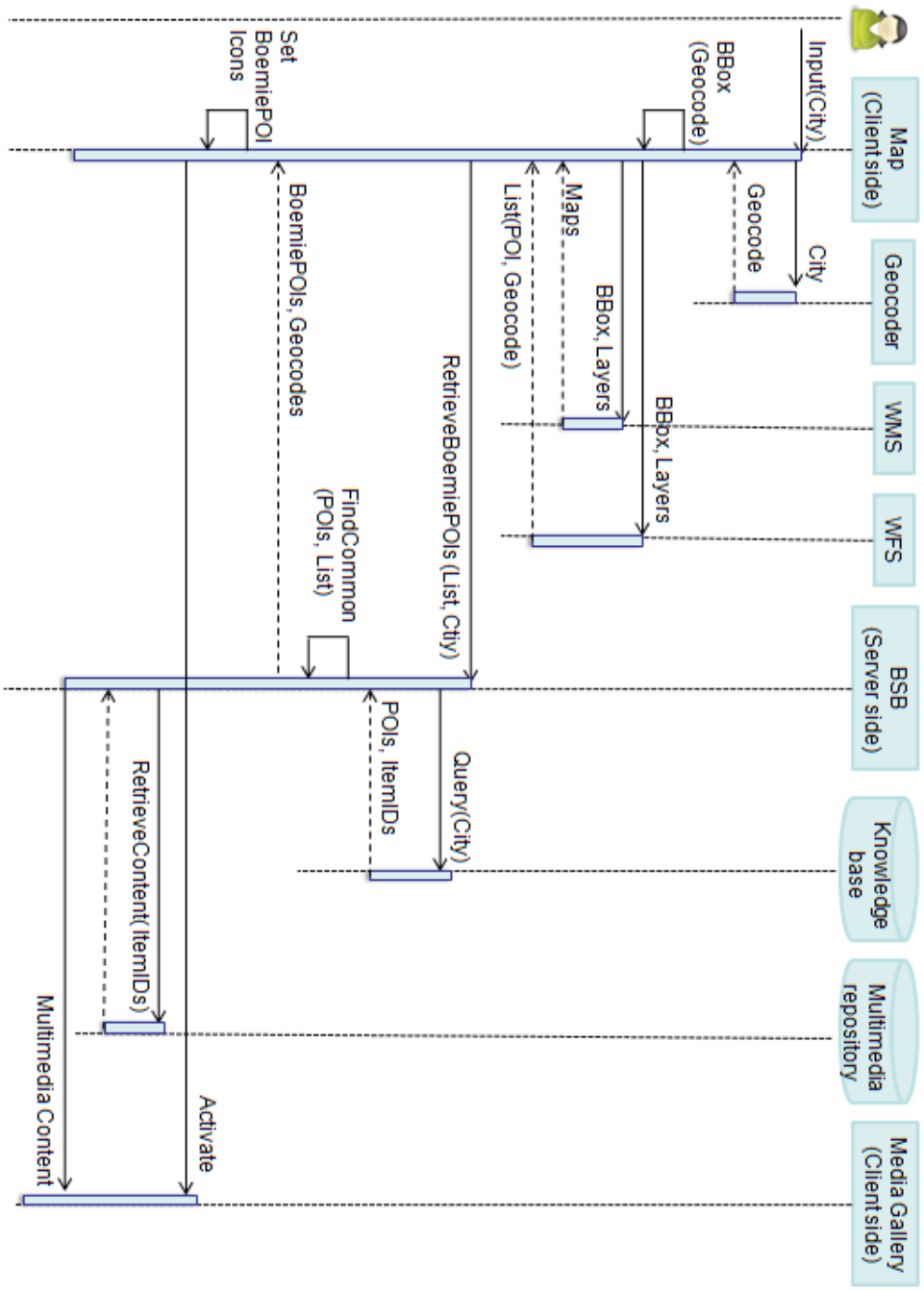


Figure 5.2: Application logic for geography-aware information navigation.

5.3 Implementation of Dynamic Identification of Applicable Services

This scenario is implemented in a process called DYNAMITE (DYNAMIc inTEractive web pages). The output of this process, the so-called DYNAMITE file is first described to understand its purpose and characteristics. Two algorithms serve the DYNAMITE process to solve the problem of discovering applicable services, but before explaining them a description is necessary about the elements that compose a service definition.

- **DYNAMITE File**

To support the performance requirements of this scenario, the application retrieves so-called DYNAMITE files, which are web pages for which context menus have been precomputed for each segment of active content. Figure 5.3 shows the steps that the DYNAMITE process executes in order to produce such web pages. As input it requires the annotated version of a content item, which can either be an IMS or a TAHAT file, as well as the related Abox containing SLI results, hereafter called SLI-Abox. The IMS/TAHAT file is parsed in order to find all the semantic annotations. Semantic annotations are expressed through individual names originally found in the corresponding Abox. The Abox is loaded to the reasoner RacerPro. In this way, for each individual name found in the IMS/TAHAT file a search process can start which exploits reasoning techniques in order to find the semantic context of the corresponding individual. As previously shown in Figure 4.9 (see page 116), the semantic context of an individual is represented by the Abox graph composed of DLI and fusion results that are adjacent and indirectly related to the individual name. By obtaining the semantic context of each individual it is possible to identify the services that are applicable from a set of service definitions. Both, the search algorithm and the identification of applicable services are at the core of DYNAMITE and both will be explained later in this section.

The DYNAMITE file extends the IMS/TAHAT file with script code that defines the gesture handling events that should be used to trigger the context menu with corresponding content-based services. In the following lines an excerpt from a DYNAMITE file is presented to show the parameters encoded for one active segment of an image.

```
<!-- MENUCONTENT PersonBody_113019383
|Analysis|PersonBody|Interpretation|PersonName]Kajsa Bergqvist MENUCONTENT -->
<AREA SHAPE=POLY COORDS="45,167,52,159,60,153,70,153,80,152,90,148,100,.......,
79,351,79,342,82,332,91,325,89,315,87,305,82,295,77,285,72,275,69,265,69,255,66,
245,64,235,62,225,59,215,57,205,54,196,50,186,48,176" id="PersonBody_113019383"
href="#" target="_self" alt="PersonBody" entry="PersonBody"
onmouseover="ImageInfoMenu('PersonBody_113019383');"
onmouseout="ImageInfoMenuOff('PersonBody_113019383');"
```

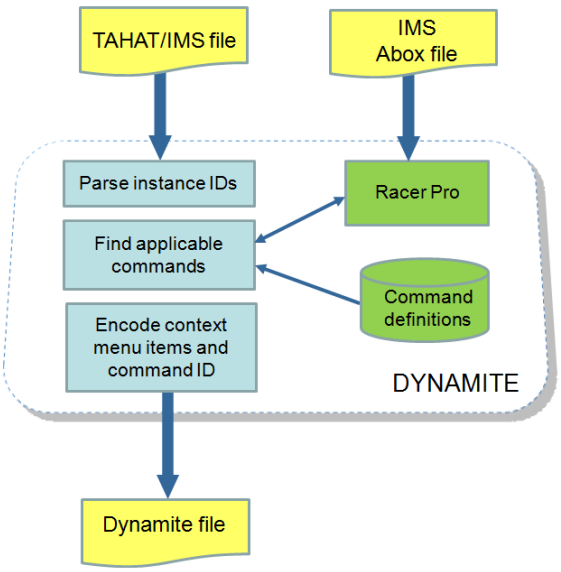


Figure 5.3: The offline process DYNAMITE (DYNAMIc inTEractive web pages).

```
onclick="ShowMenu('
WebNavigation::3:Sponsors for skin care#
SPARQLQ:(PersonName(hasPersonNameValue,Kajsa Bergqvist)):
57:More full body images of Kajsa Bergqvist#',this);"/>
```

As can be seen, the event *onclick* triggers a script function for which some parameters are also encoded. The event is related to an active segment which is part of an image map annotated with the individual name *PersonBody_113019383*. Two services have been found applicable for the semantic annotation. Correspondingly, two items are found in the context menu with the legends *'Sponsors for skin care'* and *'More full body images of Kajsa Bergqvist'*. Moreover, the applicable service ids and some parameters are also encoded. The first item *'Sponsors for skin care'*, corresponds to a service which requires an argument of type *PersonBody*. The second item, corresponds to a service that requires a *PersonName* and a value as arguments. Thus, this presupposes that the semantic context of the individual name *PersonBody_113019383* is indirectly related to an individual of *PersonName* given an individual of *Person* and its fusion with another instance coming from non-visual content such that a person's name can be obtained. In the following section a description of all this parameters is provided.

- **Service Definitions**

Service definitions contain all the necessary information required by the application server in order to executed them at a later time as requested by end users. These services are of course predefined and the corresponding programming code should be developed by software engineers. The service definitions follow a specific XML Schema. In the following

lines the first two service definitions correspond to the ones applicable for *PersonBody* as introduced in the example of the previous section.

- **Service Id:** 3
  Legend: Sponsors for skin care
  SemanticFootprint: {(Z)| *PersonBody*(Z)}
  Type: WebNavigation
  Code: http://www.coppertone.com/coppertone/index.jsp

- **Service Id:** 57
  Legend: More full body images of %Value1%
  SemanticFootprint: {(Z)| *Person*(Z), *hasPersonName*(Z,Y), *PersonName*(Y)
  hasPart(Z,X), *PersonBody*(X)}
  Arguments: (*PersonName* (*hasPersonNameValue*, %Value1%))
  Type: SparqlQ
  Code:

```
PREFIX aeo: <http://repository.boemie.org/ontology_repository_tbox/aeo.owl#>
PREFIX mco: <http://repository.boemie.org/ontology_repository_tbox/mco.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT DISTINCT ?u WHERE {
?x rdf:type mco:WebPage .
?x mco:hasURL ?u .
?x mco:contains ?y .
?y rdf:type mco:Image .
?y mco:depicts ?z .
?z rdf:type aeo:Person
?z aeo:hasPart ?w
?w rdf:type aeo:PersonBody
?z rdf:same-as ?o
?o aeo:hasPersonName ?n
?n aeo:hasPersonNameValue '%Value1%'^^xsd:string . }
```

- **Service Id:** 10
  Legend: Show %Value1% on a map
  SemanticFootprint: {(Z)| *CityName*(Z)}
  Arguments: (*CityName* (*hasCityNameNameValue*, %Value1%))
  Type: GIS
  Code:

```
<GeocodeRequest xmlns="http://www.opengis.net/xls"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:gml="http://www.opengis.net/gml"
xsi:schemaLocation="http ://www.opengis.net/xls
http://schemas.opengis.net/ols/1.1.0/LocationUtilityService.xsd">
```

```
<Address>
<Place type="Municipality">%Value1%</Place>
</Address>
</GeocodeRequest>
```

- **Service Id:** 306
  Legend: Other %Value1% jumpers from %Value2%
  SemanticFootprint: {(Z)| *Jumper*(Z), *hasNationality*(Z,Y), *CountryName*(Y), *hasGender*(Z,X), *Gender*(X)}
  Arguments:(*Gender* (*hasGenderValue*, %Value1%)), (*CountryName* (*hasCountryNameNameValue*, %Value2%))
  Type: SparqlQ
  Code:

```
PREFIX aeo: <http://repository.boemie.org/ontology_repository_tbox/aeo.owl#>
PREFIX mco: <http://repository.boemie.org/ontology_repository_tbox/mco.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT DISTINCT ?u WHERE {
?x rdf:type mco:WebPage .
?x mco:hasURL ?u .
?x mco:contains ?y .
?y rdf:type mco:Text .
?y mco:depicts ?z .
?z rdf:type Jumper.
?z aeo:hasNationality ?n.
?n aeo:hasCountryNameNameValue '%Value2%'^^xsd:string . }
?z aeo:hasGender ?g.
?g aeo:hasGenderValue '%Value1%'^^xsd:string . }
```

In this way, a service is defined first by a **service identifier**. This ID is encoded within the DYNAMITE file in association with the legend used for a context menu item. Whenever a context menu item is used, the corresponding service ID is send to the BSB server in order to find the necessary code for execution. Second, the **legend** element is used to indicate an item in a context menu. Whenever required, a variable is included (%Value1%) to indicate its substitution with one of the service's arguments. Third, the **semantic footprint** is used to assess the applicability of a service. It takes the form of a query representing the semantics that are required for a service to be executed. Fourth, the service's **arguments** involve told datatype fillers which are specially required whenever the semantic context involves SLI intances from textual content. Fifth, the **type** is used to know how to manage the information within the code element. Three different types are defined called *WebNavigation*, *SparqlQ* and *GIS*. *WebNavigation* indicates that a hypertext link is expected within the code element and it is used to reference external

sources into a new browser window. This was found useful to support advertisement services or to support IR within other portals, e.g., Wikipedia. *SparqlQ* indicates that a sparql query is expected within the code element that will be executed over the knowledge base. This is useful to suport browsing as *target search* within the multimedia repository. Finally, *GIS* indicates that GML code is expected within the code element to send the required parameters for the Map Tool to be activated. Finally, the **code** element includes relevant part of the code that is also used as parameter handled by the corresponding server methods to support the service functionality.

- **Dynamite core algorithms**

To find applicable services for a specific segment, two elements are relevant. First, the semantic footprint of a service and second the semantic context of the active segment. The applicability of a service is determined by exploiting subsumption reasoning between these two elements. This is done by comparing the semantic context against the semantic footprint of all known services.

Whenever a footprint subsumes the context (as later explained, this is defined by query subsumption), then the corresponding service is considered applicable. Thus, the more complex the semantic context is, the more restrictive and therefore fewer semantic footprints can be found that subsume it. This suggests that in order to increase the number of applicable services, various semantic contexts should be considered that vary in complexity. For a better illustration consider Figure 5.4, which shows a directed graph corresponding to the semantic context for the individual $perf_1$. Dark colored vertices and dashed arcs correspond to SLI results. These vertices represent concept assertions that are directly associated to an active segment. Note that all SLI results in this graph, with the exception of $pName_3$ and $country_3$ share the same domain semantic context. Blue colored vertices and dark arcs correspond to abduction results and red colored edges correspond to fusion results. Gray colored arcs and vertices represent the structural aspects of the content item, in this example they correspond to a web page which contains text and a captioned image. While the structural aspects are also part of the Abox, they are not considered part of the domain semantic context for a SLI instance. The following lines give an example of possible semantic contexts for the instance $perf_1$, starting with the less restrictive one which contains only the individual name corresponding to the semantic annotation of a segment, until more complex ones, containing various vertices and arcs of the semantic context.

1. $perf_1$

2. (new_5, $perf_1$):$hasPerformance$

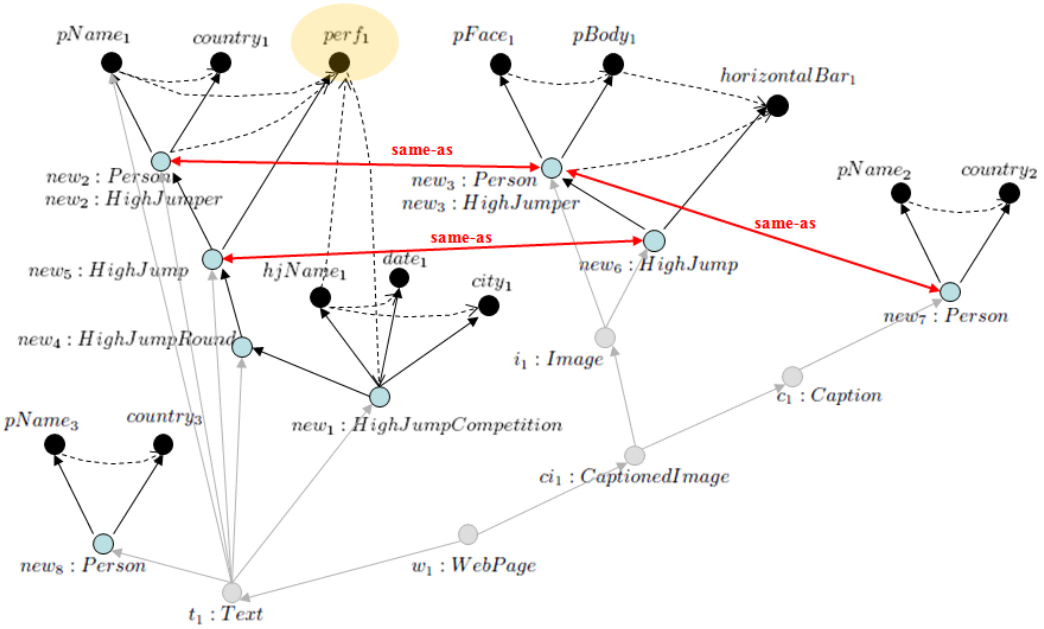


Figure 5.4: Semantic context for the individual name $perf_1$.

3. (new_5, $perf_1$):*hasPerformance*, (new_5, new_2):*hasParticipant*,
   (new_2, $pName_1$):*hasPersonName*

4. (new_5, $perf_1$):*hasPerformance*, (new_5, $horizontalBar_1$):*hasPart*

5. (new_5, $perf_1$):*hasPerformance*, (new_4, new_5):*hasPart*,
   (new_1, new_4):*hasSportsRound*, (new_1, $hjName_1$):*hasSportsName*

In order to compare a semantic context with a footprint for subsumption, the semantic context is transformed into a query. In this way both, the footprint and the semantic context are compared w.r.t. query subsumption [Rac07], which will be explained later. In order to identify applicable services, two algorithms are at the core of the DYNAMITE process, these are called *Semantics Explorer* and *Service Discovery*. Both of them are generic and can be applied to different domain ontologies.

Semantics Explorer Algorithm

The Semantics Explorer algorithm searches within the fused interpretation Abox (see a graphical example in Figure 2.27, page 68), which is a directed graph, to identify the semantic context of an individual obtained from SLI results, hereafter called *origin*. It is called origin, since is the starting point of the search algorithm, from the origin DLI and fusion results can be searched for. Note that the origin is the individual used for content annotation such that is directly associated to an active segment in media content. Given an Abox with fused interpretation results, this algorithm finds every vertex and arc adjacent or connected to the origin, excluding those graph elements which are part

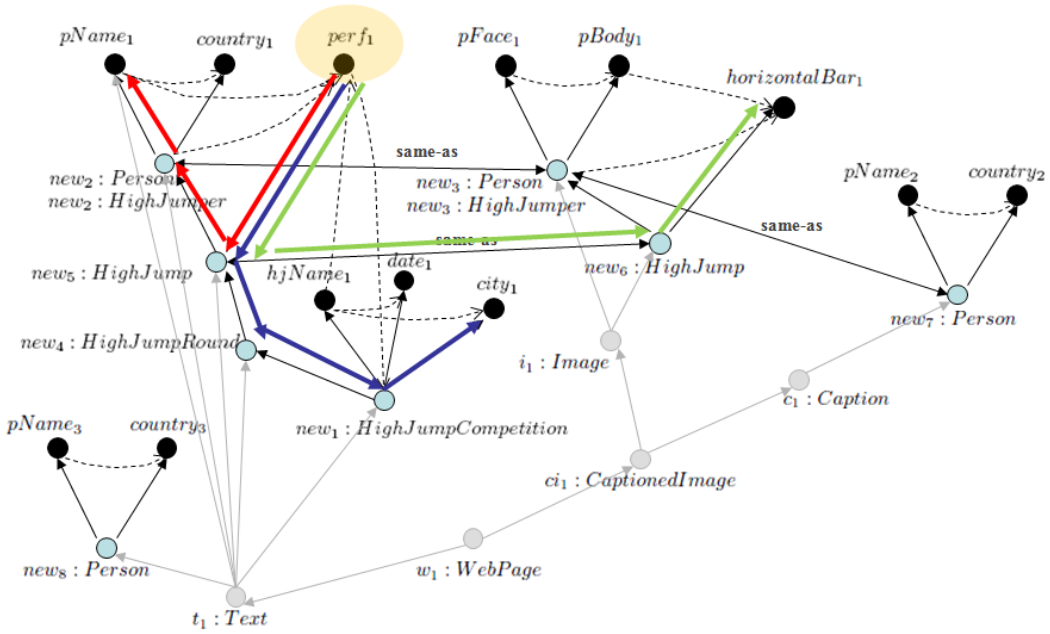


Figure 5.5: Examples of simple paths

of the content structure. The output is a set $\mathcal{K}$ of directed graphs $\kappa \in \mathcal{K}$ where each κ represents a semantic context for the *origin*. A $\kappa \in \mathcal{K}$ can be of the following types:

- **Simple path.** A simple path is represented by a directed graph composed of a sequence of vertices such that from each vertex there is an arc to the next vertex in the sequence. The arcs in the path correspond to role assertions within an Abox. A path starts from the vertex representing the origin and ends with a leaf vertex. Leaf vertices are individuals which are instances of the concept *SLC*. Figure 5.5 shows three examples of a simple path (see highlighted arrows).

- **Vertex.** A vertex is represented by a concept assertion which corresponds to the origin or to an aggregate, where the aggregate is adjacent to the origin or connected to the origin. Aggregates are individuals which are instances of the concept *DLC*.

- **Tree.** A tree is represented by the neighborhood of a vertex, where the vertex corresponds to an aggregate and the aggregate is adjacent to the origin or connected to the origin. Moreover, the neighborhood considers only outgoing arcs where the adjacent vertices are individuals which are instances of the concept *SLC*. Figure 5.6 shows three examples of a tree.

Each κ of the set $\mathcal{K}$ is finally transformed into a query resulting in a set $\mathcal{Q}$ of queries $q \in \mathcal{Q}$, which will be used as input for the Service Discovery algorithm to find applicable services.

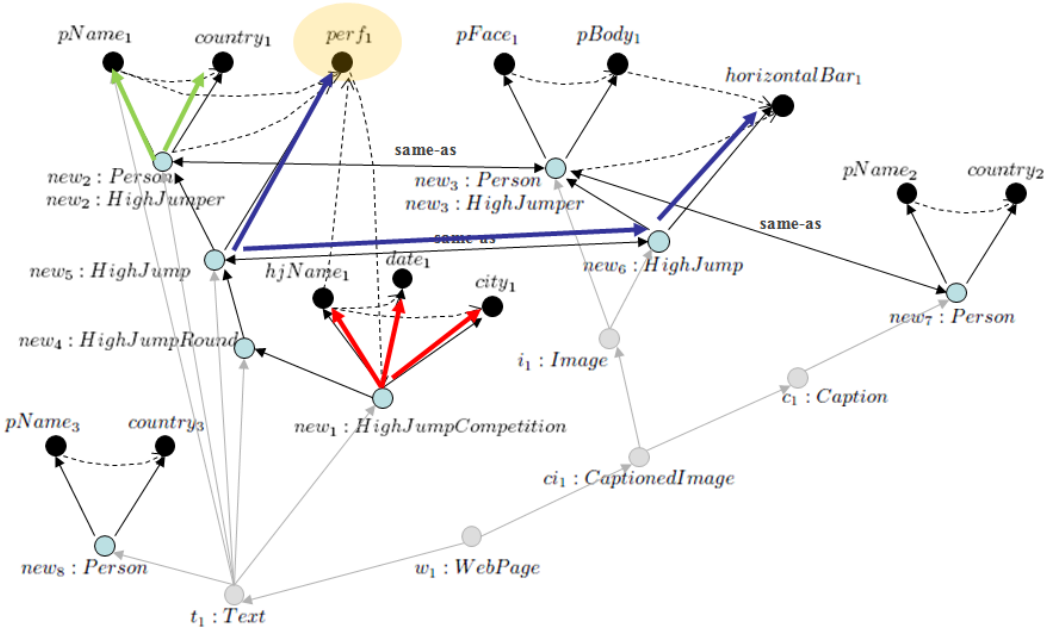


Figure 5.6: Examples of trees

Service Discovery Algorithm

Having transformed the set of semantic contexts ($\mathcal{K}$) for an individual name into a set of queries ($\mathcal{Q}$), the two necessary elements for service discovery are ready to be compared for subsumption. For this purpose the so-called **Qbox** is a main element used in this process. The Qbox is the query repository service provided by the reasoner RacerPro. It can be seen as a taxonomy of queries in which for each query its most specific subsumers and most general subsumees are computed [Rac07]. The main purpose of a Qbox is to maintain a *hierarchical cache* to improve the performance of query answering. Thus, the cached information found in the Qbox can help for optimization purposes such that future queries can totally or partially profit from the cached information thereby reducing computational effort. In this way when executing a query, the superset and subset caches corresponding to the parents and children of the query within the hierarchy can be retrieved. Similarly, when an equivalent query is found the cached answer set can be returned. For the purpose of service discovery only query classification is exploited, this means that a Qbox is constructed containing the semantic footprints of all known services by using a Tbox and an empty Abox. Thus, the interest is not to cache the tuples of previous queries, but only to exploit query subsumption. The following lines show a Qbox as depicted in the shell of the reasoner RacerPro, that classifies various semantic footprints.

```
[18] ? (show-qbox-for-abox)
;;;
;;; QBOX FOR racer-dummy-substrate FOR ABOX D:/SofiaOficina/BOEMIE_TS/Tbox/aeo.owl
;;;
;;;  0:master-top-query
;;;   |---13:query-183
;;;   |   |___0:master-bottom-query
;;;   |---12:query-134
;;;   |   |___0:master-bottom-query
;;;   |---11:query-125 = (SUBQUERY-3-OF-query-134)
;;;   |   |___0:master-bottom-query
;;;   |---10:query-115
;;;   |   |___4:query-10 = (SUBQUERY-1-OF-query-183)
;;;   |       |___3:query-5
;;;   |           |___0:master-bottom-query
;;;   |---2:query-2
;;;   |   |___0:master-bottom-query
;;;   |---1:query-1 = (SUBQUERY-5-OF-query-33)
;;;   |   |___0:master-bottom-query
;;;   |---5:query-17
;;;   |   |___6:query-24
;;;   |       |___0:master-bottom-query
;;;   |---7:query-33
;;;   |   |___0:master-bottom-query
;;;   |---8:query-75
;;;   |   |___0:master-bottom-query
;;;   |___9:query-93
;;;       |___0:master-bottom-query

[18] > :see-output-on-stdout
```

The numbers at the beginning of each line register the order in which the queries have been added to the Qbox. Afterwards, a query id is assigned, e.g., *query-125*, and the parents and children for each query are graphically depicted. Some of the queries are followed by a remark such as *11:query-125 = (SUBQUERY-3-OF-query-134)*, which means that the query with id *query-125* is equal to the third atom of the query with id *query-134*. This makes *query-125* a subquery of query *query-134*. This can be more clearly seen when extending the Qbox with query bodies as following lines show.

```
[19] ?  (show-qbox-for-abox  D:/SofiaOficina/BOEMIE_TS/Tbox/aeo.owl t)
;;;
;;; QBOX FOR racer-dummy-substrate FOR ABOX D:/SofiaOficina/BOEMIE_TS/Tbox/aeo.owl
;;;
;;;  master-top-query
;;;   |---(and (?z Jumper) (?z ?y hasNationality) (?y CountryName) (?z ?x hasGender)
```

```
;;;   |   |    (?x Gender))
;;;   |   |___master-bottom-query
;;;   |---(and (?z SportsCompetition) (?z ?y takesplaceInCityName) (?y CityName))
;;;   |   |___master-bottom-query
;;;   |---(?z CityName)
;;;   |   |___master-bottom-query
;;;   |---(?z Person)
;;;   |   |___(?z Jumper)
;;;   |       |___(?z HighJumper)
;;;   |           |___master-bottom-query
;;;   |---(?z AthleticsEquipment)
;;;   |   |___master-bottom-query
;;;   |---(?z PersonBody)
;;;   |   |___master-bottom-query
;;;   |---(?z Jumping)
;;;   |   |___(?z HighJump)
;;;   |       |___master-bottom-query
;;;   |---(and (?z Person) (?z ?y hasPersonName) (?y PersonName)
;;;   |   |    (?z ?x hasPart) (?x PersonBody))
;;;   |   |___master-bottom-query
;;;   |---(and (?z Athlete) (?z ?y hasPersonName) (?y PersonName))
;;;   |   |___master-bottom-query
;;;   |___(and (?z PoleVaulter) (?z ?y hasNationality) (?y CountryName))
;;;       |___master-bottom-query

[19] > :see-output-on-stdout
```

As can be seen *(?z CityName)* is a subquery of *(and (?z SportsCompetition) (?z ?y takesplaceInCityName) (?y CityName))* given the third query atom. Having a Qbox of semantic footprints, the objective of the service discovery algorithm is to classify each semantic context $q \in \mathcal{Q}$ within the Qbox of semantic footprints and in that way obtain its query equivalents and parents that will determine service applicability. To observe the process consider the following semantic contexts. They are part of $\mathcal{K}$ for which applicable services can be found.

$\kappa_{16} := \{(new_1,\ city_1)$:*takesplaceInCityName,* $(city_1,$ "Athens"$)$:*hasCityNameNameValue*$\}$,

$\kappa_{20} := \{new_5\}, \kappa_{24} := \{new_2\}$,

$\kappa_{30} := \{(new_2,\ pName_1)$:*hasPersonName*,

$(pName_1,$ "Yelena Slesarenko"$)$:*hasPersonNameValue*$\}$,

$\kappa_{32} := \{(new_2,\ pName_1)$:*hasPersonName,*

$(pName_1,$ "Yelena Slesarenko"$)$:*hasPersonNameValue,* $(new_2,\ pBody_1)$:*hasPart*$\}$

The algorithm Service Discovery gets as input the corresponding queries $\mathcal{Q}$ for $\mathcal{K}$, an xml file containing service definitions, the domain ontology and the *Qbox* of all semantic footprints found in the file of service definitions. In the following lines the set of queries $\mathcal{Q}$ for the previous extract from $\mathcal{K}$ is provided.

q_{16} := {(Z)|*HighJumpCompetition*(Z), *takesplaceInCityName*(Z,Y), *CityName*(Y)},
d_{16} := (*CityName*(*hasCityNameNameValue*, "Athens"))
q_{20} := {(Z)|*HighJump*(Z)}
q_{24} := {(Z)|*HighJumper*(Z)}
q_{30} := {(Z)|*HighJumper*(Z), *hasPersonName*(Z,Y), *PersonName*(Y)},
d_{30} := (*PersonName*(*hasPersonNameValue*, "Yelena Slesarenko"))
q_{32} := {(Z)|*HighJumper*(Z), *hasPersonName*(Z,Y), *PersonName*(Y),
hasPart(Z,X), *PersonBody*(X)},
d_{32} := (*PersonName*(*hasPersonNameValue*, "Yelena Slesarenko"))}

Each $q \in \mathcal{Q}$ is added to the Qbox to obtain a set of equivalent and a set of parent queries. The elements of these two sets contain query ids that correspond to semantic footprints of services. Thus, identifying those footprints means identifying the corresponding applicable services. For example, if q_{16} is added to the Qbox previously described, the following is obtained:

equivalents: ∅
parents: {p_1 =*query-134*}

In this example one applicable service is found for the semantic context q_{16}. In this way, context menus can be configured for every active segment in the media object.

5.4 Semantics-driven Applications: DLI-based KM & CM Evaluated

This chapter showed that domain ontologies built on the basis of the design patterns presented in Section 3.2 can also be used by other applications, such as the BOEMIE Semantic Browser (BSB). It was described how reasoning services are exploited to support various scenarios, such that to support the dynamic identification of applicable services, entailment reasoning is exploited. For this, the Qbox functionality of the reasoner RacerPro [3] is used. Thus, provided the semantics from the segment of a content item, in the form of a query, and a Qbox containing the semantic footprints of all known services, query subsumption is exploited in order to obtain equivalent and parent queries and in that way determine applicability (see Section 5.3). Services are defined to support three types of functionality. First, web navigation is supported for advertisement and IR within other portals, e.g., Wikipedia. Second, *search-by-association* as a way of browsing multimedia content. For this the BSB suggests semantically related content, e.g., image, text or video, by using the semantics of the activated content to construct a suitable query that can retrieve the corresponding multimedia content. Finally, the third type of functionality shows that semantics can be used as parameters to the system's internal methods. This functionality is implemented for the Map Tool in which a geographic reference recognized in media-content, is transfered to the Map Tool to start a new search.

The BSB was evaluated through testing sessions with end users and presentations to companies within the frame of the BOEMIE project. Thus, the evaluation was made by third parties and not by the author of this work, to ensure a neutral evaluation result, which is relevant when a qualitative evaluation is executed. The result of this evaluation showed, as described in [PMR09], that using BSB as a showcase for the BOEMIE project contributed to the research in software engineering by proposing a flexible way of developing application interfaces which can dynamically incorporate a set of situation-specific services to be applied in the semantic context. It supports new ideas in practical informatics by providing various exploitation scenarios where semantics-based information extraction and retrieval play an important role.

[3] http://www.racer-systems.com/

Chapter 6

Conclusion and Future Work

The vast amount of media being produced in an organization has created the need of their integration into Information Systems (ISs). Thus, interfaces for ISs are being produced that integrate both structured information and media. An example of such interfaces are corporate websites. CMSs are therefore highly demanded to meet the integration requirements of media to ISs. Development efforts on CMSs have until now being focused on storage, edition, versioning and publication of content, where the content is referenced by means of format or editorial metadata. But, the expectations of users on ISs, as well as the business goals of organizations, are challenging the capabilities of current CMSs. Users expect that that classical GUI-based interaction scenarios done with unstructured information in application programs are seamlessly extended with media shown in situation-specific ways, such that interfaces of ISs provide a tight mode of interaction between interface elements and media. On the other hand, business goals of organizations demand more valuable services integrated to ISs, such as interfaces that quickly adapt to match new products and services, support for marketing, powerful searching and browsing techniques, etc. The fulfillment of these expectations is hindered by the lack of better means for the structuring of content that go beyond format and editorial metadata. Thus, a new requirement is set on CMSs, namely the ability to manage content on the basis of content semantics. Moreover, content-based services should be developed that exploit content semantics to fulfill the specific expectations of users and organizations (see above). In this context various challenges are set:

1. To find the suitable means for representing content semantics with content descriptions, including useful knowledge representation patterns.

2. To design a process that automates the creation of content descriptions.

3. To design a knowledge management strategy that keeps track of knowledge modeling needs w.r.t. new incoming media objects, to cope with the continues amount of media production.

4. To design an architecture that facilitates the development of content-based services.

This work proposed solutions for all these challenges. With respect to the representation of content semantics with content descriptions, this work proposes the use of DL-based knowledge bases. It is argued that given the nature of media content, which contain incomplete information, it is hard to use conventional means, e.g., relational databases, for the structuring of media. DL-based knowledge bases, on the other side, are suitable. They are composed of domain ontologies and a set of Abox files that describe the content of a specific media object. With the use of Abox files, a structure can be associated to a media object that describes its content in detail and is consistent w.r.t. a domain ontology. DLs provide formal syntax and semantics to the elements of the knowledge base. DLs are advantageous because systems called reasoners exist that allow using the knowledge bases to obtain implicit information and even hypothesize new information. These are relevant tasks when dealing with the incomplete nature of media content.

The DLI process proposed here supports the automatic extraction of content descriptions from media, more specifically, a text interpretation process was presented. The advantage of the DLI process is that it is able to extract more abstract semantics from the results of shallow processing techniques for text interpretation. Thus, DLI imposes fewer requirements on surface-level interpretation (SLI), and in this way, allowing the interpretation of large scale corpora. DLI is designed as an abduction based process. Different from related work in abduction reasoning, the abduction process proposed here, uses DL-based ontologies as means to ensure consistency of results w.r.t. a domain knowledge. DLI is a declarative process that exploits reasoning over domain ontologies and rules.

Patterns for ontology and rule design are described in this work. They provide knowledge engineers a guide to design a background knowledge base that is useful to support the DLI process. These patterns describe the construction of specific axioms to allow, for example for downward information flow. With the use of downward information flow, DLI prevents SLI processes to become less generic. Thus, with the use of reasoning over domain ontologies, the results of SLI can be enriched with domain specific semantics that were otherwise absent.

A knowledge management strategy has been conceptualized in this work, called grounded ontology design. This concept has served for the construction of tools that pave the way towards the automation of ontology evolution. Thus, considering the vast amounts of information being created, it is necessary to cope with new interpretation requirements, such that available domain ontologies and rules should be extended accordingly to deal with new content semantics. The concept of grounded ontology design proposes a cyclic strategy that uses the results of interpretation to provide hints to a knowledge engineer on design requirements.

This work described a set of specific content-based services. They helped to exemplify the type of services that can be develop in order to fulfill the expectations of users and organizations when dealing with media in interfaces of ISs. The described services show how content descriptions can be exploited as well as the usefulness of having content descriptions describing more abstract semantics. To show that these services can be realized, an architecture was presented that shows the elements required for the realization of these specific content-services. Furthermore, by presenting the architecture, we showed that both theses presented here (see page 2.3 and page 3.3) can be maintained.

Future Work

Although the application of DLI, in the context of the BOEMIE Project[1], probed to be useful and convenient for its application on large-scale corpora, due to minimum requirements, e.g., shallow-text processing on the SLI layer, performance can still be improved. A strategy to improve DLI is described in this section, as well as about various practical applications that could profit from the results of DLI.

Enhancing the Performance of DLI

As described in Section 2.5, the process of DLI is an iterative one. An iteration starts with abductive reasoning (through backward-chaining rules) and finishes with deductive reasoning (through forward-chaining rules). In abductive reasoning, fiat assertions, also called observations, are tried to be explained through new individuals, called aggregates. In deductive reasoning, given the results of abductive reasoning, new assertions are deduced and classified as new fiat assertions. A new iteration starts in order to explain the fiat assertions deduced in the previous iteration. The iterations continue until there are no more fiat assertions to explain.

[1] www.boemie.org

To improve the performance of DLI w.r.t. time and space, changes in the definition of rules are proposed. Thus, it is proposed that the head of backward-chaining rules is extended to contain more than one atom, such that rules, as the ones shown in Figure 6.1, could be defined. In these rules, the body consists of atoms which contain variables

$$
\begin{aligned}
Body(W),\\
adjacent(W,V),\\
Face(V),\\
isOverlapping(W,Y),\\
HorizontalBar(Y) \quad &\leftarrow \quad HighJump(Z),\\
&\quad\ hasPart(Z,Y),\\
&\quad\ hasParticipant(Z,X),\\
&\quad\ Person(X),\\
&\quad\ hasPart(X,W),\\
&\quad\ hasPart(X,V).
\end{aligned}
$$

$$
\begin{aligned}
Body(W),\\
adjacent(W,V),\\
Face(V),\\
isNear(W,U),\\
Pole(U),\\
isOverlapping(W,Y),\\
HorizontalBar(Y) \quad &\leftarrow \quad PoleVault(Z),\\
&\quad\ hasPart(Z,Y),\\
&\quad\ hasPart(Z,U),\\
&\quad\ hasParticipant(Z,X),\\
&\quad\ Person(X),\\
&\quad\ hasPart(X,W),\\
&\quad\ hasPart(X,V).
\end{aligned}
$$

Figure 6.1: More than one atom in the head of backward-chaining rules.

that are also mentioned in atoms of the head. Moreover, the atoms in the head should contain at least one variable that relates all head atoms together in a direct or indirect way. For example, in the rules of Figure 6.1 the variable W relates all the atoms in the head.

With these changes, processing time can be reduced given that the number of iterations during interpretation is reduced. This is possible given the following two aspects. First, by having more than one atom in the head of a rule that represents a set of surface-level relations, it is possible to:

1. avoid ambiguities that are solved in later iteration cycles,

2. avoid ambiguities that are produced given the order in which surface-level relations are explained.

Second, by including more than one aggregate concept on the body of a rule and corresponding relations to the parts, the number of iterations are reduced since it is not required any more to hypothesize one of the aggregates first, e.g., Person, in order to extract a second one, e.g. PoleVault.

By considering this, performance is enhanced not only in time by reducing the number of iterations but also in space, since avoiding ambiguities means less number of explanation Aboxes as Section 2.5 illustrates with Figure 2.22.

DLI for other Practical Applications

Feedback to Improve NLP

The results of DLI can be used as feedback to improve the results of NL analysis specially to support ambiguity resolution. To illustrate this, an example is provided for the problem of preposition (PP) attachment, which is helpful to support grammar checkers. The work of Lee and Knutson in [LK08] studies the problem of *preposition generation* which aims to determine the best preposition to use given its context in an input sentence. They focus on the study of different kinds of features, e.g., lexical, syntactic, POS, and their contribution to solve this problem. In their work it is explained that a preposition can be easily predicted if its lexical head in the sentence is identified as well as the prepositional complement. The following example is provided *"Pierre Vinken joined the board — a non executive director."*, where *"director"* represents the prepositional complement and *"joined"* the lexical head. For this example, knowing that PP is attached to *"joined"* as lexical head rather than *"board"* clearly helps to predict the preposition *"as"*. Moreover, besides identifying the lexical head and the complement of the preposition it is explained that distinguishing the kind of relation viz. argument or adjunct character, that exists between these two elements is of outmost importance to determine the suitable preposition. In the context of this work, the results of DLI can be useful as a complement to the linguistic knowledge required in [LK08]. Thus, DLI can support with the use of domain knowledge, where the domain knowledge can be engineered to serve specific grammatical rules of a language. For example, consider the usage of prepositions for locations as in the following cases

1) Fire destroys garage — Aveneu C.
2) Our house is — 323 Third Street.

Choosing between *at, in* or *on*, can be solved by knowing that the prepositional complement is an address or only a street name. An address is understood as a fixed point and therefore "at" should be generated as preposition for sentence 2). While several common uses exist for the preposition "on" in relation with a street, in the case of sentence 1), "garage" is in relation with the street name but the specific point of the street is not specified, such that "on" should be generated as preposition for sentence 1).

$$
\begin{aligned}
FixedPoint &\sqsubseteq Location \\
Street &\sqsubseteq Location \\
Aveneu &\sqsubseteq Street \\
Address &\sqsubseteq FixedPoint \sqcap \exists hasStreetName.Street \\
&\quad \sqcap \exists hasNumber.Number \\
&\quad \sqcap \exists hasPostalCode.PostalCode
\end{aligned}
$$

$$
\begin{aligned}
numberToStreet\,(X,Y) \leftarrow\ & Address(Z), \\
& hasNumber(Z,X), \\
& Number(X), \\
& hasStreetName(Z,Y),
\end{aligned}
$$

Figure 6.2: DLI's background knowledge.

$$
\begin{aligned}
stName_1 &: StreetName \\
(stName_1, \text{“Third Street”}) &: hasValue \\
num_1 &: Number \\
(num_1, \text{“323”}) &: hasValue \\
(num_1,\ stName_1) &: numberToStreetName \\
avenue_1 &: Aveneu \\
(avenue_1, \text{“Aveneu C.”}) &: hasValue
\end{aligned}
$$

Figure 6.3: Abox containing the results of SLI.

In this way, provided the background knowledge for DLI in Figure 6.2 and the results of SLI in Figure 6.3, the string "323 Third Street" could be interpreted as an aggregate of type *Address* and therefore as a *FixedPoint*, while "Aveneu C." is a generic *Location*.

Editing Support

Practical applications that can directly profit from the advances in the field of NLP are text editors. Some text editors already provide convenient services that support writers during revising and editing tasks. Thus, services such as automatic hyphenation, spell checking, grammar checking and style checking are currently provided. But as stated in [MP08], "word processors offer few functions, if any, for handling text on the same cognitive level as the author". Their work focuses on the use of high-level linguistic terms to support authors in the revision and edition of textual documents. High-level linguistic terms are elements such as conjunctions, verbs in a specific mode, sentences without verbs, etc., that allow for conciseness, expression, grammar, etc. They provide use cases for such linguistic elements that allow for *highlighting* and *editing actions*. For example, a text editor that is able to highlight all conjunctions of the text and/or all verbs, gives the writer a quick overview of these constructions in order to analyze his diction. An example

for editing actions is to allow for the transpositions of conjuncts in one step instead of the typical copy-paste steps required in current editors thus saving various editions steps.

While the point of view in [MP08] to support text edition is only linguistic, results of DLI can also be of great use to support the edition of texts, to be more precise, in the enrichment of text by exploiting dynamic suggestion of content. In this way the editor can receive suggestions, for example, for related literature references, image content for illustration purposes, etc. This follows the scenario of semantics-driven applications already described in Section 4.1.3.

Web Service Discovery Support

The BOEMIE Semantic Browser (BSB) showcases the dynamic discovery of applicable services as described in Section 4.1.3. In this scenario reasoning on DLI results is exploited to dynamically determine suitable services that can be offered as items on a graphical object, e.g., context menu. In this way, semantically related functionality is offered according to the active content. The BSB uses a registry of service definitions, each service has a semantic footprint as a way of specifying the semantics required for the activation of the service. If the semantics of the active content are subsumed by the footprint of a service, then the corresponding service is enabled and offered through a context menu. This scenario could be extended in relation with web service technologies to support service discovery. To make this possible, the web service's interface should be semantically marked-up and registered to a UDDI service where a semantic-driven application such as the BSB could look for applicable services. This scenario could profit from advances in the research of semantic web services [MSH01, RKL$^+$05] which deal with all aspects inherent to web services such as discovery, execution, composition and interoperation.

Appendix A

The Athletics Events Ontology

Signature $\mathcal{S}$. As explained in Section 2.1, the design of an ontology starts by choosing elementary descriptions grouped in the so called signature. For the AEO ontology, the athletics domain is of interest and the signature contains the following:

$CN =$ $\{SportsEvent,\ SportsCompetition,\ SportsRound,\ SportsTrial,$
$Jumping,\ HighJumpCompetition,\ HighJumpRound,\ HighJump,$
$PoleVaultCompetition,\ PoleVaultRound,\ PoleVault,\ Person,$
$Athlete,\ Jumper,\ HighJumper,\ PoleVaulter,\ ShotePuter,$
$OrganismPart,\ PersonBody,\ PersonFace, AthleticsEquipment,$
$HorizontalBar,\ Ball,\ Pole,\ StopBoard,\ Performance,\ Ranking,$
$Date,\ StartDate,\ Name,\ SportsName,\ EventName,\ CountryName,$
$HighJumpName,\ PoleVaultName,\ StadiumName,\ RoundName,$
$PersonName,\ CityName,\ DLC,\ SLC\}$

$RN =$ $\{takesPlaceIn,\ hasDate,\ hasName,\ hasPart,\ takesPlaceInSportsPOI,$
$hasParticipant,\ hasPerformance,\ hasRanking,\ hasNationality,$
$personNameToPerformance,\ personNameToCountryName,$
$performanceToRanking,\ sportsNameToPerformance,\ sportsNameToDate,$
$sportsNameToCityName,\ adjacent,\ near,\ hasPart,\ hasParticipant,$
$hasNationality,\ SLR,\ DLR\}$

$IN =$ $\{\ \}$

Notice that the set of individual names is empty. As explained in Section 2.1, the Abox part of an ontology $\mathcal{A} \in \mathcal{O}$ contains only relevant individuals which make sense to share. For this work, more specifically w.r.t. the deep-level interpretation (DLI) process, $\mathcal{A}$ is empty. In the DLI process only the Aboxes related to a media object are used.

Tbox $\mathcal{T}$. The following inclusion axioms describe the semantics of aggregate objects.

$$
\begin{array}{rcl}
DLC & \sqsubseteq & \neg SLC \\
SportsEvent & \sqsubseteq & DLC \sqcap \exists_{\leq 1} takesPlaceIn.CountryName \\
 & & \sqcap \exists_{\leq 1} takesPlaceIn.CityName \sqcap \exists hasDate.Date \\
 & & \sqcap \exists hasName.\top \\
 & & \sqcap \exists hasPart.SportsCompetition \\
 & & \sqcap \neg SportsCompetition \sqcap \neg SportsRound \\
 & & \sqcap \neg SportsTrial \sqcap \neg Person \\
SportsCompetition & \sqsubseteq & DLC \sqcap \exists hasName.\top \\
 & & \sqcap \exists hasDate.Date \sqcap \exists takesPlaceIn.CityName \\
 & & \sqcap \exists takesPlaceInSportsPOI.StadiumName \\
 & & \sqcap \exists hasPart.SportsRound \\
 & & \sqcap \neg SportsRound \sqcap \neg SportsTrial \sqcap \neg Person \\
HighJumpCompetition & \sqsubseteq & SportsCompetition \\
 & & \sqcap \forall hasPart.HighJumpRound \\
 & & \sqcap \forall hasName.HighJumpName \\
 & & \sqcap \neg PoleVaultCompetition \\
PoleVaultCompetition & \sqsubseteq & SportsCompetition \\
 & & \sqcap \forall hasPart.PoleVaultRound \\
 & & \sqcap \forall hasName.PoleVaultName \\
SportsRound & \sqsubseteq & DLC \sqcap \exists hasName.RoundName \sqcap \exists hasDate.Date \\
 & & \sqcap \exists hasPart.SportsTrial \\
 & & \sqcap \neg SportsTrial \sqcap \neg Person \\
HighJumpRound & \sqsubseteq & SportsRound \sqcap \forall hasPart.HighJump \sqcap \neg PoleVaultRound \\
PoleVaultRound & \sqsubseteq & SportsRound \sqcap \forall hasPart.PoleVault \\
SportsTrial & \sqsubseteq & DLC \sqcap \exists_{\leq 1} hasParticipant.\top \\
 & & \sqcap \exists_{\leq 1} hasPerformance.\top \\
 & & \sqcap \exists_{\leq 1} hasRanking.\top \sqcap \neg Person \\
Jumping & \sqsubseteq & SportsTrial \\
HighJump & \sqsubseteq & Jumping \sqcap \forall hasParticipant.HighJumper \\
 & & \sqcap \exists_{\leq 1} hasParticipant.\top \\
 & & \sqcap \exists_{\leq 1} hasPart.HorizontalBar \sqcap \neg PoleVault \sqcap \neg ShotPut \\
PoleVault & \sqsubseteq & Jumping \sqcap \forall hasParticipant.PoleVaulter \\
 & & \sqcap \exists_{\leq 1} hasParticipant.\top \sqcap \exists_{\leq 1} hasPart.Pole \\
 & & \sqcap \exists_{\leq 1} hasPart.HorizontalBar \sqcap \neg ShotPut \\
ShotPut & \sqsubseteq & SportsTrial \sqcap \forall hasParticipant.ShotPuter \\
 & & \sqcap \exists_{\leq 1} hasParticipant.\top \\
 & & \sqcap \exists_{\leq 1} hasPart.StopBoard \sqcap \exists_{\leq 1} hasPart.Ball \\
Person & \sqsubseteq & DLC \sqcap \exists_{\leq 1} hasName.\top \sqcap \forall hasName.PersonName \\
 & & \sqcap \exists_{\leq 1} hasNationality.\top \sqcap \forall hasNationality.CountryName \\
 & & \sqcap \exists_{\leq 1} hasPart.PersonBody \\
 & & \sqcap \exists_{\leq 1} hasPart.PersonFace \\
Athlete & \sqsubseteq & Person \\
Jumper & \sqsubseteq & Athlete \\
HighJumper & \sqsubseteq & Jumper \\
PoleVaulter & \sqsubseteq & Jumper \\
ShotPutter & \sqsubseteq & Athlete
\end{array}
$$

In the following axioms we describe subsumption relations, disjointness requirements, and domaind and range restrictions.

$$
\begin{aligned}
OrganismPart &\sqsubseteq SLC \sqcap \neg AthleticsEquipment \sqcap \neg Date \\
&\quad \sqcap \neg Name \sqcap \neg Performance \sqcap \neg Ranking \\
PersonBody &\sqsubseteq OrganismPart \sqcap \neg PersonFace \\
PersonFace &\sqsubseteq OrganismPart \\
AthleticsEquipment &\sqsubseteq SLC \sqcap \neg Date \sqcap \neg Name \\
&\quad \sqcap \neg Performance \sqcap \neg Ranking \\
Pole &\sqsubseteq AthleticsEquipment \sqcap \neg HorizontalBar \\
&\quad \sqcap \neg Ball \sqcap \neg StopBoard \\
HorizontalBar &\sqsubseteq AthleticsEquipment \sqcap \neg Ball \sqcap \neg StopBoard \\
Ball &\sqsubseteq AthleticsEquipment \\
StopBoard &\sqsubseteq AthleticsEquipment \sqcap \neg Ball \\
Date &\sqsubseteq SLC \sqcap \neg Name \sqcap \neg Performance \sqcap \neg Ranking \\
StartDate &\sqsubseteq Date \\
Name &\sqsubseteq SLC \sqcap \neg Performance \sqcap \neg Ranking \\
EventName &\sqsubseteq Name \sqcap \neg SportsName \\
SportsName &\sqsubseteq Name \\
HighJumpName &\sqsubseteq SportsName \sqcap \neg PoleVaultName \\
PoleVaultName &\sqsubseteq SportsName \\
Performance &\sqsubseteq SLC \sqcap \neg Ranking \\
Ranking &\sqsubseteq SLC \\
personNameToPerformance &\sqsubseteq SLR \\
personNameToCountryName &\sqsubseteq SLR \\
performanceToRanking &\sqsubseteq SLR \\
sportsNameToPerformance &\sqsubseteq SLR \\
sportsNameToDate &\sqsubseteq SLR \\
sportsNameToCityName &\sqsubseteq SLR \\
adjacent &\sqsubseteq SLR \\
near &\sqsubseteq SLR \\
hasPart &\sqsubseteq DLR \\
hasParticipant &\sqsubseteq DLR \\
hasNationality &\sqsubseteq DLR \\
\exists personNameToPerformance.\top &\sqsubseteq PersonName \\
\top &\sqsubseteq \forall personNameToPerformance.Performance \\
PersonName &\sqcap (\leq_1 personNameToPerformance.Performance) \\
\exists personNameToCountryName.\top &\sqsubseteq PersonName \\
\top &\sqsubseteq \forall personNameToCountryName.CountryName \\
PersonName &\sqcap (\leq_1 personNameToCountryName.CountryName) \\
\exists performanceToRanking.\top &\sqsubseteq Performance \\
\top &\sqsubseteq \forall performanceToRanking.Ranking \\
Performance &\sqcap (\leq_1 performanceToRanking.Ranking) \\
\exists sportsNameToPerformance.\top &\sqsubseteq SportsName \\
\top &\sqsubseteq \forall sportsNameToPerformance.Performance \\
\exists sportsNameToDate.\top &\sqsubseteq SportsName \\
\top &\sqsubseteq \forall sportsNameToDate.Date \\
\exists hasParticipant.\top &\sqsubseteq Athlete \\
\exists hasPerformance.\top &\sqsubseteq Performance \\
\exists hasRanking.\top &\sqsubseteq Ranking \\
\exists hasNationality.\top &\sqsubseteq CountryName \\
\exists sportsNameToCityName.\top &\sqsubseteq SportsName \\
\top &\sqsubseteq \forall sportsNameToCityName.CityName
\end{aligned}
$$

Appendix B

Rules

$$
\begin{aligned}
personNameToCountryName(X,Y) \leftarrow\ & Person(Z),\\
& hasName(Z,X), PersonName(X),\\
& hasNationality(Z,Y), CountryName(Y).\\
personToPerformance(X,Y) \leftarrow\ & Person(X),\\
& hasName(X,Z), PersonName(Z),\\
& personNameToPerformance(Z,Y).\\
personToPerformance(X,Y) \leftarrow\ & SportsTrial(Z),\\
& hasParticipant(Z,X), Athlete(X),\\
& hasPerformance(Z,Y), Performance(Y).\\
personToRanking(X,Y) \leftarrow\ & Person(X),\\
& hasName(X,Z),\\
& PersonName(Z),\\
& personNameToRanking(Z,Y).\\
personToRanking(X,Y) \leftarrow\ & SportsTrial(Z),\\
& hasParticipant(Z,X), Athlete(X),\\
& hasRanking(Z,Y), Ranking(Y).\\
sportsNameToCityName(X,Y) \leftarrow\ & HighJumpCompetition(Z),\\
& hasName(Z,X), HighJumpName(X),\\
& takesPlace(Z,Y), CityName(Y).\\
sportsNameToCityName(X,Y) \leftarrow\ & PoleVaultCompetition(Z),\\
& hasName(Z,X), PoleVaultName(X),\\
& takesPlace(Z,Y), CityName(Y).\\
sportsNameToCityName(X,Y) \leftarrow\ & SportsCompetition(Z),\\
& hasName(Z,X), SportsName(X),\\
& takesPlace(Z,Y), CityName(Y).\\
sportsNameToDate(X,Y) \leftarrow\ & HighJumpCompetition(Z),\\
& hasName(Z,X), HighJumpName(X),\\
& hasDate(Z,Y), Date(Y).\\
sportsNameToDate(X,Y) \leftarrow\ & SportsCompetition(Z),\\
& hasName(Z,X), SportsName(X),\\
& hasDate(Z,Y), Date(Y).
\end{aligned}
$$

$$sportsNameToStadiumName(X,Y) \leftarrow SportsCompetition(Z), hasName(Z,X), SportsName(X), takesPlaceInSportsPOI(Z,Y), StadiumName(Y).$$

$$sportsNameToDate(X,Y) \leftarrow PoleVaultCompetition(Z), hasName(Z,X), PoleVaultName(X), hasDate(Z,Y), Date(Y).$$

$$eventNameToCityName(X,Y) \leftarrow SportsEvent(Z), hasName(Z,X), EventName(X), takesPlaceIn(Z,Y), CityName(Y).$$

$$eventNameToCountryName(X,Y) \leftarrow SportsEvent(Z), hasName(Z,X), EventName(X), takesPlaceIn(Z,Y), CountryName(Y).$$

$$eventNameToDate(X,Y) \leftarrow SportsEvent(Z), hasName(Z,X), EventName(X), hasDate(Z,Y), StartDate(Y).$$

$$sportsEventToSportsName(X,Z) \leftarrow SportsEvent(X), hasName(X,Y), EventName(Y), sportsEventNameToSportsName(Y,Z).$$

$$sportsEventToSportsName(X,Y) \leftarrow SportsEvent(X), hasPart(X,Y), SportsCompetition(Y), hasName(Y,Z), SportsName(Z).$$

$$sportsCompetitionToPerformance(X,Y) \leftarrow SportsCompetition(X), hasName(X,Z), SportsName(Z), sportsNameToPerformance(Z,Y).$$

$$sportsCompetitionToPerformance(X,Y) \leftarrow HighJumpCompetition(Z), hasPart(Z,X), HighJumpRound(X), hasPart(X,W), HighJump(W) hasPerformance(W,Y), Performance(Y).$$

$$sportsCompetitionToPerformance(X,Y) \leftarrow PoleVaultCompetition(Z), hasPart(Z,X), PoleVaultRound(X), hasPart(X,W), PoleVault(W) hasPerformance(W,Y), Performance(Y).$$

List of Acronyms

Abox Assertional box

AI Artificial Intelligence

App Application

AR Aggregate Role

ASCII American Standard Code of Information Interchange

BSB BOEMIE Semantic Browser

CM Content Management

CMSs Content Management Systems

CN Concept Names

CDOeP Conceptual (or Content) Ontology Design Pattern

DLC Deep-Level Concept

DLI Deep-Level Interpretation

DLR Deep-Level Role

DLs Description Logics

DNLP Deep Natural Language Processing

DSSs Decision Support Systems

FN Feature Names

FOL First-Order Logic

GATE General Architecture for Text Engineering

GCIs Generalized Concept Inclusions

GML Geography Markup Language

GUC Generic Use Case

IE Information Extraction

IN Individual Names

IMS Image Map Service

ISs Information Systems

JPEG Joint Photographic Experts Group

KBSs Knowledge-Based Systems

KM Knowledge Management

NEs Named Entities

NER Named Entity Recognition

NLP Natural Language Processing

nRQL new Racer Query Language

O Observation

OR Observed Relation

OWA Open World Assumption

OWL Web Ontology Language

POI Point of Interest

RN Role Names

SLC Surface-Level Concept

SLI Surface-Level Interpretation

SLR Surface-Level Role

TAHAT Text Analysis HTML Annotation Tool

Tbox Terminological box

UDDI Universal Description Discovery and Integration

WFS Web Feature Service

WMS Web Map Service

WMRP Web Map Research Platform

Bibliography

[ACG+05] A. Acciarri, D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, M. Palmieri, and R. Rosati. QUONTO: QUerying ONTOlogies . In *Proc. of AAAI05 - Volume 4*, pages 1670–1671. AAAI Press, 2005.

[All95] J. Allen. *Natural Language Understanding*. Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA, 2nd edition, 1995.

[BLC96] A. Bernaras, I. Laresgoiti, and J. M. Corera. Building and reusing ontologies for electrical network applications. In W. Wahlster, editor, *In Proc. of ECAI*, pages 298–302. John Wiley and Sons, Chichester, 1996.

[BMSW97] D. M. Bikel, S. Miller, R. Schwartz, and R. Weischedel. Nymble: High-performance learning name-finder. In *Proc. of ANLC '97*, pages 194–201, Stroudsburg, PA, USA, 1997. Association for Computational Linguistics.

[BN03] F. Baader and W. Nutt. Basic description logics. In F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors, *The Description Logic Handbook: Theory, Implementation and Applications*, pages 43–95. Cambridge University Press, 2003.

[CEF+07] S. Castano, S. Espinosa, A. Ferrara, V. Karkaletsis, A. Kaya, S. Melzer, R. Möller, and G. Petasis. Ontology dynamics with multimedia information: The BOEMIE evolution methodology. In *Proc. of IWOD-07*, pages 41–54, 2007.

[CEF+09] S. Castano, S. Espinosa, A. Ferrara, V. Karkaletsis, A. Kaya, R. Möller, S. Montanelli, G. Petasis, and M. Wessel. Multimedia interpretation for dynamic ontology evolution. In *Journal of Logic and Computation*, number 19 in 5, pages 859–897. Oxford University Press, 2009.

[CMB+09] H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan, C. Ursu, M. Dimitrov, M. Dowman, N. Aswani, I. Roberts, Y. Li, A. Shafirin, and A. Funk. Developing Language Processing Components with GATE Version 5 (a User Guide). http://gate.ac.uk/sale/tao/tao.pdf, 2009.

[Cun02] H. Cunningham. GATE, A General Architecture for Text Engineering. *Computers and the Humanities*, 36:223–254, 2002.

[Cun05] H. Cunningham. Information Extraction, Automatic. *Encyclopedia of Language and Linguistics, 2nd Edition*, 2005.

[CV01] P. H. Carstensen and L. Vogelsang. Design of web-based information systems - New challenges for systems development? In *Proc. of ECIS*, pages 536–547, 2001.

[DDS+07] S. Dasiopoulou, K. Dalakleidi, G. Stoilos, V. Tzouvaras, and Y. Kompatsiaris. Multimedia content and descriptor ontologies - final version. Technical report, 2007. BOEMIE Deliverable D3.8.

[DEDT07] K. Dalakleidi, C. Evangelou, S. Dasiopoulou, and V. Tzouvaras. Domain ontologies - Version 2, 2007. BOEMIE Deliverable D3.5.

[EKM11] S. Espinosa, A. Kaya, and R. Möller. Logical formalization of multimedia interpretation. In G. Tsatsaronis G. Paliouras, C. D. Spyropoulos, editor, *Knowledge-Driven Multimedia Information Extraction and Ontology Evolution*, number 6050 in LNAI, pages 110–133. Springer, 2011.

[EKS06] C. Elsenbroich, O. Kutz, and U. Sattler. A Case for Abductive Reasoning over Ontologies. In *Proc. OWL: Experiences and Directions Workshop*, pages 1–10, 2006.

[Gal08] A. Galochkin. A Strategy to support Information Extraction from Natural Language at production time. Master's thesis, Hamburg University of Technology, 2008.

[Gan05] A. Gangemi. Ontology design patterns for semantic web content. In *Proc. of ISWC*, pages 262–276. Springer LNCS Series, 2005.

[GCMS03] N. Guarino, A. Oltramari C. Masolo, and L. Schneider. DOLCE : a Descriptive Ontology for Linguistic and Cognitive Engineering, 2003. WonderWeb Deliverable D17.

[GF94] M. Gruninger and M. S. Fox. The role of competency questions in enterprise engineering. In *Proc. of IFIP WG5.7 Workshop on Benchmarking - Theory and Practice*, 1994.

[GHLS07] B. Glimm, I. Horrocks, C. Lutz, and U. Sattler. Conjunctive Query Answering for the Description Logic $\mathcal{SHIQ}$. In *Proc. of IJCAI-07*. AAAI Press, 2007.

[GMN+09] O. Gries, R. Möller, A. Nafissi, K. Sokolski, and M. Rosenfeld. Casam domain ontology. Technical report, Hamburg University of Technology, 2009.

[Hat07] D. Hatch. Data management 2.0: Making sense of unstructured data. Technical report, Aberdeen Group, 2007.

[HM01] V. Haarslev and R. Möller. RACER System Description. In *Proc. IJCAR-01*, volume 2083 of *LNAI*, pages 701–705. Springer-Verlag, 2001.

[HMW04] V. Haarslev, R. Möller, and M. Wessel. Querying the semantic web with Racer + nRQL. In *Proc. of ADL'04, Ulm, Germany, September 24*, 2004.

[Hor03] I. Horrocks. Implementation and optimisation techniques. In F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors, *The Description Logic Handbook: Theory, Implementation and Applications*, pages 313 – 358. Cambridge University Press, 2003.

[HSAM93] J. R. Hobbs, M. Stickel, D. Appelt, and P. Martin. Interpretation as abduction. volume 63, pages 69–142, Essex, UK, October 1993. Elsevier Science Publishers Ltd.

[HSME88] J. Hobbs, M. Stickel, P. Martin, and D. Edwards. Interpretation as abduction. In *Proc. of ACL'88*, pages 95–103, Stroudsburg, PA, USA, 1988. Association for Computational Linguistics.

[HSTT00] I. Horrocks, U. Sattler, S. Tessaris, and S. Tobies. How to decide Query Containment under Constraints using a Description Logic. In *LPAR-00*, volume 1955 of *LNCS*, pages 326–343. Springer-Verlag, 2000.

[JW04] L. Jain and X. Wu. The most outstanding ontologies. In A. Gomez-Perez, M. Fernandez-Lopez, and O. Corcho, editors, *Ontological Engineering: Advanced Information and Knowledge Processing*. Springer, 2004.

[Kay11] A. Kaya. *A Logic-Based Approach to Multimedia Interpretation*. PhD thesis, Hamburg University of Technology, Hamburg, Germany, 2011.

[KD02] A. Kakas and M. Denecker. Abduction in logic programming. In A. Kakas and F. Sadri, editors, *Computational Logic: Logic Programming and Beyond. Part I*, LNAI, pages 402–436. Springer, 2002.

[KKT92] A. C. Kakas, R. A. Kowalski, and F. Toni. Abductive logic programming. *Journal of Logic and Computation*, 2(6):719–770, 1992.

[KM91] A. C. Kakas and P. Mancarella. Knowledge assimilation and abduction. In *Workshop on Truth Maintenance Systems*, pages 54–70, London, UK, 1991. Springer-Verlag.

[LK08] J. Lee and O. Knutsson. The role of PP attachment in preposition generation. In A. Gelbukh, editor, *Proc. of CICLing-2008*, number 4919 in LNCS, pages 643–654. Springer, 2008.

[Luk07] T. Lukasiewicz. Probabilistic description logic programs. In *Proc. of ECSQARU*, volume 45, pages 288 – 307, 2007. Journal of Approximate Reasoning.

[McK91] S. McKay. CLIM: the Common Lisp Interface Manager. *Commun. ACM*, 34(9):58–59, 1991.

[MGM98] A. Mikheev, C. Grover, and M. Moens. Description of the LTG system used for MUC-7. In *Proc. of MUC-7*, 1998.

[MMG99] A. Mikheev, M. Moens, and C. Grover. Named entity recognition without gazetteers. In *Proc. of ACL'99*, pages 1–8, Stroudsburg, PA, USA, 1999. Association for Computational Linguistics.

[MN08] R. Möller and B. Neumann. Ontology-based Reasoning Techniques for Multimedia Interpretation and Retrieval. In *Semantic Multimedia and Ontologies: Theory and Applications*, pages 55–98. Springer, 2008.

[MP08] C. Mahlow and M. Piotrowski. Linguistic support for revising and editing. In A. Gelbukh, editor, *Proc. of CICLing-2008*, number 4919 in LNCS, pages 631–642. Springer, 2008.

[MRS08] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 1st ed. edition, 2008.

[MSH01] S. A. McIlraith, T. C. Son, and H.Zeng. Semantic web services. *IEEE Intelligent Systems*, 16(2):46–53, 2001.

[Nak01] R. Nakano. *Web Content Management: A Collaborative Approach*. Addison Wesley, 1st edition, 2001.

[NB03] D. Nardi and R.J. Brachman. An introduction to description logics. In F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors, *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.

[NM06] B. Neumann and R. Möller. On Scene Interpretation with Description Logics. In H.I. Christensen and H.-H. Nagel, editors, *Cognitive Vision Systems: Sampling the Spectrum of Approaches*, number 3948 in LNCS, pages 247–278. Springer, 2006.

[NM08] B. Neumann and R. Möller. On scene interpretation with description logics. volume 26, pages 82–101. Butterworth-Heinemann, January 2008.

[Pal09] G. Paliouras. Boemie - final report. Technical report, July 2009. BOEMIE Deliverable.

[PBG+07] I. Pratikakis, A. Bolovinou, B. Gatos, M. Anthimopoulos, S. Perantonis, C. Seibert, S. Eickeler, and T. Merten. Semantics extraction from visual content tools - the state-of-the-art. Technical report, 2007. BOEMIE Deliverable D2.2.

[PKG+02] G. Petasis, V. Karkaletsis, G.Paliouras, I. Androutsopoulos, and C. D. Spyropoulos. Ellogon: A new text engineering platform. In *Proc. LREC 2002, Las Palmas, Canary Islands*, pages 72–78, 2002.

[PKM+07a] S. Espinosa Peraldi, A. Kaya, S. Melzer, R. Möller, and M. Wessel. Multimedia Interpretation as Abduction. In *Proc. of DL-2007*, 2007.

[PKM+07b] S. Espinosa Peraldi, A. Kaya, S. Melzer, R. Möller, and M. Wessel. Towards a media interpretation framework for the semantic web. In *In Proc. of WI'07*, 2007.

[PKM09a] S. Espinosa Peraldi, A. Kaya, and R. Möller. The BOEMIE Semantic Browser: An application exploiting rich semantic metadata. In *Proc. of AST*, LNI, 2009. Gesellschaft für Informatik, Bonn, INFORMATIK 2009.

[PKM09b] S. Espinosa Peraldi, A. Kaya, and R. Möller. Formalizing multimedia interpretation based on abduction over description logic aboxes. In B. Cuenca Grau, I. Horrocks, B. Motik, and U. Sattler, editors, *Description Logics*, volume 477 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009.

[PKMM08] S. Espinosa Peraldi, A. Kaya, S. Melzer, and R. Möller. On ontology based abduction for text interpretation. In A. Gelbukh, editor, *Proc. of CICLing-2008*, number 4919 in LNCS, pages 194–205. Springer, 2008.

[PMR09] S. Espinosa Peraldi, R. Möller, and M. Rogowsky. Final evaluation report. Technical report, Hamburg University Of Technology, May 2009. BOEMIE Deliverable D5.9.

[Rac07] Racer Systems. RacerPro User's Guide Version 1.9.2. Technical report, 2007.

[RKL+05] D. Roman, U. Keller, H. Lausen, J. de Bruijn, R. Lara, M. Stollberg, A. Polleres, C. Feier, C. Bussler, and D. Fensel. Web service modeling ontology. *Applied Ontology*, 1(1):77–106, 2005.

[RN03a] S. J. Russell and P. Norvig. First-order logic. In S. J. Russell and P. Norvig, editors, *Artificial intelligence a Modern Approach*, pages 240–271. Prentice Hall, 2003.

[RN03b] S. J. Russell and P. Norvig. Practical natural language processing. In S. J. Russell and P. Norvig, editors, *Artificial intelligence a Modern Approach*, pages 691–721. Prentice Hall, 2003.

[Rob] J. Robertson. What are the goals of a CMS? Technical report, Step Two Designs.

[SGC09] G. Simoes, H. Galhardas, and L. Coheur. Information extraction tasks: a survey. In *Proc. of INForum 2009*, 2009.

[Sha05] M. Shanahan. Perception as Abduction: Turning Sensor Data Into Meaningful Representation. *Cognitive Science*, 29(1):103–134, 2005.

[SM03] H. A. Smith and J. D. McKeen. Developments in practice VIII: Enterprise content management. *In Journal of Communications of the Association for Information Systems*, 11(41):647 – 659, 2003.

[Sod97] S. G. Soderland. *Learning Text Analysis Rules For Domain-Specific Natural Language Processing.* PhD thesis, University of Massachusetts, 1997.

[Spi09] N. Spivack. Search 3.0: Present, personal, precise. In *Proc. of the 8th International Semantic Web Conference (ISWC 2009)*, number 5823 in LNCS, pages 1003–1004. Springer, 2009.

[SWS+00] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:1349–1380, 2000.

[Tha78] P. R. Thagard. The best explanation: Criteria for theory choice. *The Journal of Philosophy*, 75(2):76–92, 1978.

[UK95] M. Uschold and M. King. Towards a methodology for building ontologies. In *Workshop on Basic Ontological Issues in Knowledge Sharing, held in conjunction with IJCAI-95*, 1995.

[Wir95] U. Wirth. Abduktion und ihre Anwendungen. *Zeitschrift für Semiotik*, pages 405–424, 1995.

[WM06] M. Wessel and R. Möller. A Flexible DL-based Architecture for Deductive Information Systems. In G. Sutcliffe, R. Schmidt, and S. Schulz, editors, *Proc. IJCAR-06 workshop of ESCoR*, pages 92–111, 2006.

[Woh00] P. Wohed. Conceptual patterns for reuse in information systems analysis. In *Wangler (Eds.), CAiSE 2000, LNCS 1789*, pages 157–175. Springer, 2000.

Curriculum Vitae

Personal Information

28.10.1978	born in Morelia, Mexico

Education

09/1993 - 07/1996	High School, Instituto Latino de Morelia. Morelia, Mexico
09/1996 - 10/2001	Bachelor in Computer Science, Instituto Tecnológico de Morelia. Morelia, Mexico
09/2002 - 07/2005	MSc. Information and Media Technologies, TU Hamburg-Harburg. Hamburg, Germany
since 03/2006	PhD. candidate at Institute of Software Technology Systems, TU Hamburg-Harburg. Hamburg, Germany

Practical Experience

11/2001 - 03/2002	Software development, Interlinea S.A. de C.V. Uruapan, Mexico
03/2003 - 05/2003	System analyst, ELCH Project, TU Hamburg-Harburg. Hamburg, Germany
12/2003 - 01/2003	System analyst, ELCH Project, TU Hamburg-Harburg. Hamburg, Germany
08/2003 - 11/2003	Internship SAP/ABAP Consultant, Passport Business Engineering GmbH, Hamburg, Germany
02/2004 - 11/2004	Working student, E-Learning development, Airbus Deutschland GmbH. Hamburg, Germany
09/2005 - 02/2006	Working student, E-Learning development, Airbus Deutschland GmbH. Hamburg, Germany
03/2006 - 05/2009	Research Assistant, EU project BOEMIE TU Hamburg-Harburg. Hamburg, Germany