Università degli Studi di Bari
Facoltà di Scienze
Dipartimento di Informatica

Similarity-based Learning Methods for the Semantic Web

Claudia d'Amato

Supervisor:
Prof. Floriana Esposito

Coordinator:
Prof. Annamaria Fanelli

Acknowledgement

At the end of the time spent for reaching this PhD I would like to thanks many persons that have had important roles in this experience. First of all I would thank God for gave me the strength of leaving a most sure way that was my work in the company and start this very uncertain and unstable way, that is the research, at least in Italy, but also for gave me the strength to arrive here and overcome all the difficulties found. Tank you God, for all the times that you take my hand and make my difficulties less hard.

I like to Domenico, for having accepted and shared my decision of starting the way of research, even if it makes very hard our life together. Thank you for your patience and for your capacity to share everything with me, thank you because you put me and my happiness first of all, also first of you. I hope that I am able to do at least sometimes the same.

I want to tank my parents. I know very well that you did not agree with my choice of starting this PhD, that you imagined for me a different life. Anyway, with your silence you shared this time with me and perhaps today you are also just a little bit proud of this little results.

Thank you to my supervisor, Floriana Esposito, that made possible to change the direction of my life with this PhD, that grant me all the opportunities for growing in this time and also, I hope, for believing in me.

Thank you to Nicola Fanizzi, the person that, most of any other one, has taught me the basis of this work, has shared every idea with me and have made possible to arrive here.

I like to thank Steffen Staab for gave me the possibility to spend some moths of my PhD in his Laboratory. This time had been very precious for me. I learnt a lot, not only for the work but also for my life. Thank you for the possibility to share my little knowledge with you, thank you for had treated me as one of your group, thank you for had felt me at home. Thank you also to all the guys of the ISWeb group for the way in which you welcome me, I cannot forget every time spent with you.

I like to remember here also the persons with which I spent most of nice time, even if not for work. I like to remember Annalisa, we spent together our breakfast every morning, changing the color of every day life. I like to remember Mara, Nico, Stefano and Geni that besides of Nicola have been the friends of the lunch. I like to remember Luigi, Ignazio and Mimmo friends of the laboratory with which had been possible to laugh also when the right thing to do was cry.

In the last, I cannot forget the friends that shared with me all my life: Claudia, Giusy, Luciana, Mariangela, Rossella (strictly lexicographic order) and also the most recent friend but very very important for me: Roberto and Mariangela.

To all this person I say THANKS, this goal would not have been possible without YOU.

Preface

Ontological knowledge plays a key role for the interoperability in the Semantic Web perspective. Nowadays, standard ontology markup languages are supported by well-founded semantics of Description Logics (DLs) together with a series of available automated reasoning services. However, several tasks in an ontology life-cycle, such as their construction and/or integration are still almost entirely delegated to knowledge engineers.

Most of the research in the Semantic Web field focuses on methods based on deductive reasoning. Inductive reasoning and knowledge discovery have received less attention. Nevertheless, inductive reasoning may assist several important tasks that are likely to be supported by knowledge-based systems, such as clustering, classification, revision, mapping and alignment. Even retrieval may be regarded from both a deductive perspective and from an opposite one, through approximate reasoning. Similarity measures play an important role in these tasks as well as in information integration. Moreover, in the Semantic Web context, the construction of the knowledge bases and their population should be supported by automated inductive inference services.

The induction of structural knowledge is not new in machine learning, especially in the context of *concept formation*, where clusters of similar objects are aggregated in hierarchies according to heuristic criteria or similarity measures. Almost all of these methods apply to zero-order representations while, ontologies are expressed by means of fragments of first-order logic. Yet, the problem of the induction of structural knowledge turns out to be hard in first-order logic or equivalent representations. So far, the automated induction of knowledge bases expressed in DLs representations has not been investigated in depth. While, really, knowledge-intensive methods could be of great help for the Semantic Web.

In this perspective, this thesis introduces a set of novel (dis-)similarity measures applied to expressive DLs knowledge representations. They are able to assess (dis-)similarity value between (complex) concepts, individuals and between (complex) concept and individual asserted in an ontology. Measures applicable to ontological knowledge need to be founded on the underlying semantics of the considered elements, rather than on their syntactic structure. Differently from the previous works that are mainly syntactic-driven, the measures defined in this thesis are semantic-based and so are able to exploit the semantics of the considered element.

This thesis also focuses on the definition and application of inductive inference learning methods to the Semantic Web context. Specifically, instance-based learning methods applicable to ontological knowledge are presented with the goal of improving many different tasks such as: concept retrieval and query answering, ontology-population and completion, service discovery and ranking. Specifically, by combining an instance-based (analogical) approach with a notion of semantic (dis-)similarity measure, the completion of ontologies as well as the ontology population could be performed through the inductive reasoning. In turn, this may trigger other related services such as learning and/or revision of faulty knowledge bases. Furthermore, based on the same principles, concept retrieval and query answering could also be performed. Moreover, by the use of (conceptual) clustering methods based on (dis-)similarity measures, the efficiency of the service discovery process can be improved by matching a request against the intensional cluster descriptions rather than against to all available services. Hence, the services returned by this process could be ranked on the ground of their similarity with respect to the request.

The results of this research constitute a new approach to the Semantic Web issues. I hope that they will be the starting point of a wide line of research in the future.

Bari, January 2007

Claudia d'Amato

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This dissertation focuses on the definition of new similarity and dissimilarity measures applicable to a relational setting, particularly ontological knowledge. Hence it is shown how it is possible to set up methods that, using such measures, perform supervised and unsupervised learning on ontological knowledge. Specifically, known instance based and clustering methods and algorithms are extended in this field in order to cope with the expressiveness of ontological representation.

The proposed measures and methods are used to solve some typical problems such as completions of an incomplete knowledge base by the use of *Analogy Reasoning*, improvement of the efficiency of the resources discovery task (with particular attention to the service discovery process) by the use of clustering methods, and improvement of the effectiveness of the ranking process of the retrieved resources. All these aspects, together with practical motivation and implications of the choices will be explained throughout this work.

1.1 Semantic Web

Semantic Web is the new vision of the Web whose main goal is to make Web contents not only human readable but also machine readable and processable. In the most famous paper introducing the Semantic Web (SW), Tim Berners-Lee and his co-authors explain this goal: "Most of the Web content today is designed for humans to read, not for computer programs to manipulate meaningfully. Computers can adeptly parse Web pages for layout and routine processing here a header, there a link to another page but in general, computers have no reliable way to process the semantics: this is the home page of the Hartman and Strauss Physio Clinic, this link goes to Dr. Hartman's curriculum vitae." [23].

The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation. The essential property of the World Wide Web is its universality. The power of a hypertext link is that "anything can link to anything." Web technology, therefore, must not discriminate among different types of information. On the contrary, information varies along many axes. One of these is the difference between information produced primarily for human consumption and that produced mainly for machines.

The Semantic Web brings structure to the meaningful content of Web pages, creating an environment where software agents, roaming from page to page, can readily carry out sophisticated tasks for users. So, for example, such an agent coming to the Web page of the Department of Computer Science will know not only that the page has keywords such as "Department, courses, students, library" but also that Dr. Smith works at this Department on Mondays, Wednesdays and Fridays and that the script takes a date range in yyyy-mm-dd format and returns appointment times. These semantics were encoded into the Web page using off-the-shelf software for writing Semantic Web pages along with resources listed on the University's site.

Like the Internet, the Semantic Web has to be as decentralized as possible. Decentralization requires compromises: the Web had to throw away the ideal of total consistency of all of its interconnections, ushering in the infamous message "Error 404: Not Found" but allowing unchecked exponential growth. Traditional knowledge-representation systems typically have been centralized, requiring everyone to share exactly the same definition of common concepts such as "parent" or "vehicle." But central control is stifling, and increasing the size and scope of such a system rapidly becomes unmanageable. On the contrary, in Semantic Web context, paradoxes and unanswerable questions are accepted as the price that must be paid to achieve versatility.

However decentralization raises an important issue: different identifiers could be used for the same concept, consequently it is necessary to have a way to know that different identifiers mean or refer to the same thing. For example, considered two databases that refer to *car* and *automobile* respectively, a program that wants to compare or combine information across the two DBs has to know that these two terms are being used to mean the same thing. Ideally, the programs must have a way to discover such common meanings for whatever DBs it encounters. A solution to this problem is provided by the use of collections of information called *ontologies*. In philosophy, an ontology is a theory about the nature of existence, of what types of things exist; Ontology as a discipline studies such theories. Artificial-intelligence and Web researchers have co-opted the term for their own jargon. For them an ontology is a formal definition of the semantics of the resources and their relationships.

1.1.1 Ontology: Meaning, Usage and Representation

Ontologies have shown to be the right answer to the Semantic Web vision, by providing a formal conceptualization of a domain that is shared and reused across domains, tasks and group of people [87]. Their role is to make semantics explicit. Particularly, ontologies describe domain theories for the explicit representation of the semantics of the data. The semantic structuring achieved by ontologies differs from the superficial composition and formatting of information (as data) offered by relational and XML[1] databases. Within a database context, virtually, all of the semantic content has to be captured in the application logic. On the contrary, ontologies are able to provide an objective specification of domain information, by representing a consensual agreement on the concepts and relationships characterizing the way knowledge in that domain is expressed. The result is a common domain of discourse available on the Web, that can be interpreted further by inference rules and application logic. Note that ontologies put no constraints on publishing (possibly contradictory) information on the web, only on its (possible) interpretations.

Ontologies may vary not only in their content, but also in their structure and implementation. Building an ontology means different things to different practitioners. Indeed an ontology could be used for describing simple lexicons or controlled vocabularies to categorically organized thesauri and taxonomies where terms are given distinguishing properties, to full-blown ontologies where these properties can define new concepts and where concepts have named relationships. Ontologies also differ with respect to the scope and purpose of their content. The most prominent distinction is between the domain ontologies describing specific fields of endeavor like medicine, and upper level ontologies describing the basic concepts and relationships invoked when information about any domain is expressed in natural language. The synergy among ontologies (exploitable by a vertical application) springs from the cross-referencing between upper level ontologies and various domain ontologies.

Building an ontology means to distinguish between two different structural components: the *Terminological* component and the *Assertional* component. The terminological component is roughly analogous to what it is known as the schema for a relational database or XML document. It defines the terms and structure of the ontology's area of interest. The assertional component populates the ontology further with instances or individuals that manifest that terminological definition. To build an ontology, a number of possible language could be used, including general logic programming languages like Prolog[2]. However in the last few years, the *Web Ontology Language*[3] (OWL) has become the de-facto standard for the knowledge

[1]eXtensible Mark-up Language (XML). See http://www.w3.org/XML/

[2]PROLOG (PROgramming in LOGic) is a programming language based on logic paradigm.

[3]http://www.w3.org/2004/OWL/

representation in the SW. It is based on a logic thought to be especially computable, known as Description Logics (DLs) [8]. It is a fragment of First Order Logic.

OWL is a language for defining and instantiating Web Ontologies. An OWL ontology may include descriptions of *Classes* (that are mainly concepts of the analyzed domain), *properties* (that are relations among defined concepts) and their *instances*. Given such an ontology, the OWL formal semantics (based on DLs) specifies how to derive its logical consequences, i.e. facts not literally reported in the ontology, but *entailed* from it. For this reason a number of reasoners have been implemented, based on such semantics. Some examples are FaCT [104], RACER[96], PELLET[184].

The OWL language provides three increasingly expressive sublanguages;

- *OWL Lite* supports the classification hierarchy inference and simple constraint features, i.e. cardinality constraints on properties where only cardinality values of 0 and 1 are permitted.
- *OWL DL* supports maximum expressiveness without provoking the losing of computational completeness (all entailments are guaranteed to be computed) and decidability (all computation will finish in finite time) of the related inference services. OWL DL includes all OWL language constructs with restrictions such as type separation (a class can not also be an individual or a property, a property can not also be an individual or a class). OWL DL is so called due to its correspondence with Description Logics.
- *OWL Full* is meant for having maximum expressiveness and syntactic freedom (of RDF[4]) with no computational guarantees. For instance, in OWL Full a class can be treated simultaneously as a collection of individuals and as an individual in its own right. OWL Full allows an ontology to augment the meaning of the pre-defined (RDF or OWL) vocabulary. Nevertheless, it is unlikely that reasoner will be able to support every feature of OWL Full.

Each of these sublanguages is an extension of its simpler predecessor, both in what can be legally expressed and in what can be validly concluded. So:

- Every legal OWL Lite ontology is a legal OWL DL ontology
- Every legal OWL DL ontology is a legal OWL Full ontology
- Every valid OWL Lite conclusion is a valid OWL DL ontology
- Every valid OWL DL conclusion is a valid OWL Full ontology

[4]http://www.w3.org/RDF/

Reasoners for OWL Lite have desirable computational properties. Reasoners for OWL DL, while dealing with a decidable sublanguage, are subject to higher worst-case complexity.

The effective use of ontologies requires not only a well-defined and designed ontology language, but also support from efficient reasoning tools. Reasoning is important both to ensure the quality of an ontology, and to exploit the rich structure of ontologies. Specifically, reasoning can be employed in different phases of the ontology life-cycle. During the ontology design, it can be used to test whether concepts are non-contradictory and to derive implied relations. It may be used when an ontology is deployed. As well as, exploiting the pre-computed concept hierarchy (linked by a *is-a* relation), reasoning can be used to determine the consistency of facts stated in annotations, or to infer relationships between annotations and ontology classes. Moreover, when searching web pages annotated with terms from an ontology, it may be useful to consider not only exact matches, but also matches with respect to more general or more specific terms - where the latter choice depends on the context. In the deployment phase, the requirements on the efficiency of reasoning are much more stronger than in the design phase.

All the presented tasks are solved by the use of deductive reasoning. However there is a number of open problems that could be solve or at least partially solved by the use of inductive reasoning[5]. Specifically, all the tasks that require certain and correct results need to employ deductive reasoning. Some examples are: computing class hierarchy, checking ontology consistency. Moreover these tasks do not require greater generality of the conclusions w.r.t. the premises. On the contrary tasks such as: ontology learning, ontology population by assertions, ontology evaluation, ontology mapping and alignment require inductive inferences as they need of higher general conclusion with respect to the premises and can be obtained as inference result from a set of available existing examples. Indeed, by the use of inductive learning, and specifically learning from example, it could be possible to achieve new and more general knowledge than such examples represent. However nowadays, most of these tasks are are manually made. So it is necessary to have tools able to execute these tasks at least in a semi-automatic way. Unfortunately not many efforts have been invested in the direction of inductive and uncertain reasoning. On the contrary, I believe that inductive reasoning applied on top of deductive reasoning could be the right direction in order to have new methods able to make human-centered time consuming tasks less heavy. In Sec. 1.3 basics about inductive reasoning are presented, while in Chapt. 5, definitions and applications of inductive learning methods to the semantic web domain are shown.

[5]See Sect. 1.3 for more details about inductive and deductive reasoning.

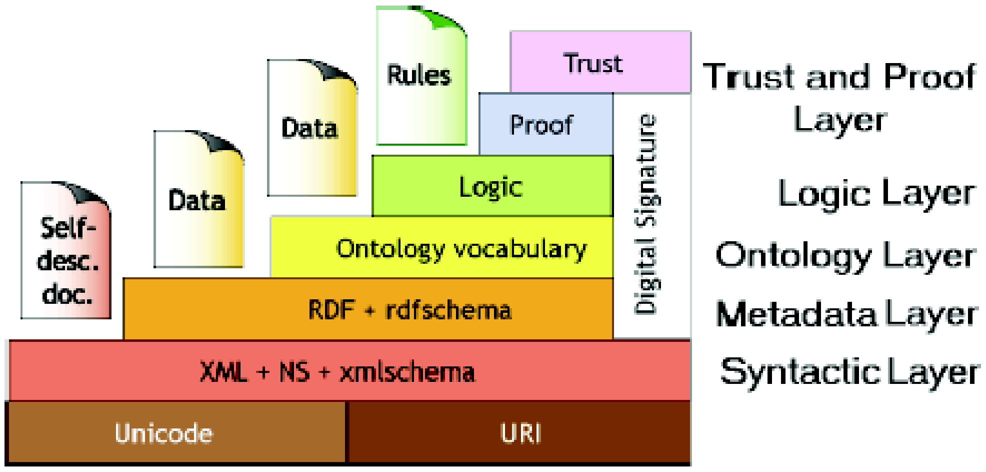

Figure 1.1: The Semantic Web Stack

1.1.2 The Semantic Web Architecture

The Semantic Web goal: *"a shared understanding of the meaning of descriptions of digital content, both by the human and by machine to machine"*, as proposed by *Tim Berners-Lee* was not presented simply as an abstract idea. It was presented jointly with a concrete solution schema for reaching such goal. Indeed in [22] the SW Architecture (see Fig.1.1) was shown. It consists of several layers arranged as the level/protocols stack in computer networks (namely ISO/OSI stack protocols). Every layer presents a set of duties and a set of technologies for coping with them. Since the overall objective is to share information at a higher level than in the current Web, every technology, in its layer, will have a representational duty (in the sense of Knowledge Representation - KR) and technologies that are underlining the KR scope. In the following, each layer is presented, starting from the lowest level of representational capabilities towards the highest one.

- *Syntactic Layer*: the Web is based on the philosophy of a navigable space, by the use of a mapping from resource ids (URI)[6] to resources. This layer has the task of representing documents as the abstraction of a unity of information that can contain subunits in a hierarchical fashion. A document is represented by a tree rooted in the documents itself, whose children are the subparts of the document (and, recursively, each part can has children that are subtrees). The main technology of this layer is the XML that proposes a textual portable representation of documents in order to grant syntactic interoperability.

[6]Uniform Resource Identifier (URI): a standard means of addressing resources on the Web. See http://www.w3.org/Addressing/ for how to assign URIs to resources for preserving uniqueness.

- *Metadata Layer*: in order to make the current Web machine processable, preexisting Web content has been enriched with meta-information representing their meaning. The Metadata layer is responsible of such representation. This layer has as its main entity the meta-datum, that is some information about some information. Here comes the first actual step beyond usual Web. Metadata can be about Web pages, abstract resources, namely about everything that can be identified by a URI. The main technology used in this layer is Resource Description Framework (RDF), a formalism for writing metadata, namely for the description of resources and their types. It allows for *reification* and it can be seen as a first order language with only unary and binary predicates, existential quantifier and conjunction. RDF relies on the syntactic portability of the Syntactic Layer because it can be (among the other solutions) serialized into XML to be exchanged across the Web.

- *Ontology Layer*: the layers analyzed so far guarantee that Web resources are uniquely identified, are endowed with a portable representation and are machine processable, but they are not able to add semantics to them. On the contrary, one of the main goal of the SW is to make resources machine processable and "understandable". As seen, the right answer to this need is given by the use of ontologies. The function of the ontology layer is to add a shareable and common semantics to the existing meta-data, without conveying any information about how to use concepts and relationships (defined in the ontologies). Indeed, recalling Gruber's definition *"ontologies are formal specification of a shared conceptualization"* [94], ontologies represent the necessary framework for granting a common understanding among applications. Ontologies play the role of metadata vocabularies where annotators can draw and reuse existing metadata terminologies, add new ones, and where applications consuming metadata can find a reference for the intended meanings.

- *Logic Layer*: this is the point at which assertions from around the Web can be used to derive new knowledge. One of the challenges of the SW is to provide a language that expresses both data and rules for reasoning about the data and that allows rules from any existing KR system to be exported onto the web. For data representation, OWL has been defined, whose theoretical counterpart is DLs that have reasoning operators making possible to reason about the data. However, some problems remain for the definition of a language for rule definition. Currently the Semantic Web Rule Language[7] (SWRL) initiative is exploring the possibility of integrating rules and ontologies. The idea is to extend the set of OWL axioms to include rules (expressed as Horn-like clauses). This enables rules to be combined with an OWL knowledge base.

[7]W3C Member Submission http://www.w3.org/Submission/ 2004/SUBM-SWRL-20040521/

Unfortunately, there are some negative results [103] concerning the undecidability of consistency check (see Sect. 2.3) in full fledged SWRL. This layer nowadays represents the frontier in the SW field.

- *Trust and Proof Layer*: In his vision, Berners-Lee highlights the importance of digital signatures. Although public key cryptography has been around for some time, it has not really taken off. Berners-Lee observed that one contributing factor to this, was that it was too coarsely-grained, with the choice of either trusted or not trusted. An infrastructure must be in place where a party can be trusted within a particular domain. Once such granularity is possible, digital signatures can be used to establish the provenance not only of data but of ontologies and of deductions. Nevertheless other important aspects of the Web have to be considered, particularly its heterogeneity and decentralization. These are the main strengths of the Web, but they are also responsible for one of its main issue which is the availability of contradictory information. For the same reason such contradictions could be a problem also in SW. Moreover, SW relies (since DLs do) on Open World Assumption that, roughly speaking, says that if something is missing then everything about the default part can be assumed. Therefore, it is very likely that SW application will find themselves reasoning in presence of inconsistent knowledge bases. Such applications should, then, make decisions on which of the contrasting parts they should privilege in order to get rid of the inconsistencies. So, considering the proofs that each party provides for its conclusion and in accordance with Berners-Lee's vision: "Systems would digitally sign and export these proofs for other systems to use and possibly incorporate into the Semantic Web", one strategy could be the examination of the proofs and consequently the strongest proofs will be considered as the mean for choosing the most reliable sources of information.

The presented brief summary gives an idea of the complexity of the Semantic Web and of the large number of open problems that need to be solved. Every layer of the presented architecture presents something that needs of improvement. My choice concern the ontology layer. Particularly ontology learning and ontology population are nowadays crucial issues in order to automatize (or semi-automatize) tasks that today require a costly and human massive intervention. One of the problems investigated in this thesis, and not previously analyzed in the literature, is how to enrich and populate (with individuals) existing ontology in a semi-automatic way as well as how to improve the retrieval[8] of a known concept and/or of a new query concept built from the considered ontology, by the use of inductive learning methods. Details about this will be illustrated in Sect. 5.1.1

[8]See Sect. 2.3 for the notion of retrieval.

1.2 Semantic Web Services

In the last few years the Web encountered two revolutionary changes which aim to transform it from a static document collection to an intelligent and dynamic data integration environment, where resources are shared and information is generated on demand. The first is the Semantic Web vision and technology (see Sect. 1.1). The second one is the Web service technology[9], mainly characterized by allowing uniform access via Web standards to software components residing on various distributed platforms and written in different programming languages. As a result, software components providing a variety of functionalities (ranging from currency conversion to flight booking or book buying) are now accessible via a set of Web standards and protocols.

1.2.1 From Web Services to Semantic Web Services

A *Web service* can be defined as any piece of software that makes itself available over the Internet and uses a standardized XML messaging system. Web services are emerging as a popular standard characterized by being platform-independent, self-describing, reusable and finalized to sharing data and functionality among loosely-coupled, heterogeneous systems. They also allow to build complex applications faster and with less programming effort. The term "service" is often used in different ways [2, 161]. It sometimes refers to an abstract business interaction between two parties, and other times it refers to a computational entity with a Web service interface. Here it is used in the second sense.

A Web service is identified by a URI and it can be accessed through the Internet by means of its exposed interface. The interface of a Web service describes the operations the service performs, the type of messages to be exchanged during the interaction with the service and the physical location of ports where information will be exchanged. The Web service model includes three component roles: the service requesters which invoke services, the service providers which offer services and respond to the requests, and the registries where services can be published. The common usage scenario for Web services consists of three phases: publishing, finding and binding services (see Fig.1.2). A service provider publishes the description of the service it offers to a service registry. Such description, or advertisement, contains information about the service provider (e.g., company name, address and contact information), about the service itself (e.g., name and category) and the URL of its interface definition. When a service requester needs for a service, he finds the desired service either by expressing a query, or directly browsing the registry. Next, the

[9]www.w3.org/2002/ws/

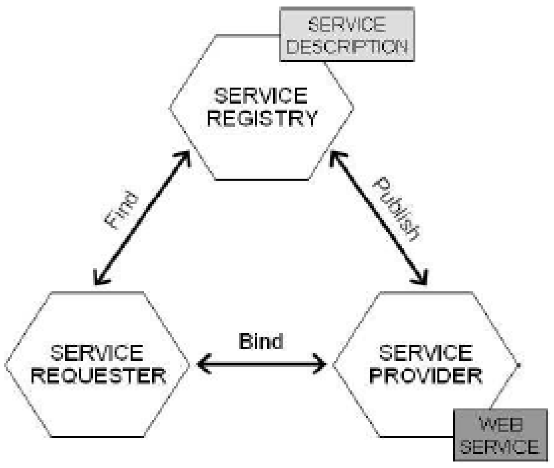

Figure 1.2: Web service usage scenario

service requester (i.e., the developer who is building a new application) interprets the meaning of the interface description of the discovered service (by exploiting variables names, comments and additional documentation) and he binds such service within the application he is developing. When a service requester binds a service, he includes an invocation of such service within the application he is deploying. The requester invokes the discovered service by means of the Web service communication protocol.

In this scenario, crucial issues and tasks are:

- modeling service description
- service discovery
- service composition and/or orchestration

Modeling service descriptions is an important task, indeed it is the only way in order to "understand" whether a certain service is adequate with respect to a certain need or not. The current standard XML-based language for describing Web services is WSDL[10] (Web Service Description Language). A WSDL description details the operations that the service offers in terms of input and output messages, specification about how to invoke the service and one or more network locations or connection points where a service can be accessed. So, mainly it provides a syntactic-level description of service functionalities, without specifying any formal definitions of their meaning.

[10]http://www.w3.org/TR/wsdl

Linked to modeling service descriptions are the *service discovery* task and the *service composition* task. *Web service discovery* (also known as *matchmaking*) is the process of (searching and) localizing available services that are able to satisfy client requirements. *Web service composition* is the process of providing value-added services through the composition of basic Web services (possibly offered by different providers). These tasks, in the first time manually made by humans, nowadays result to be time consuming, due to the increasing amount of available Web services. Indeed the main reason of the attention to modeling service descriptions is the need of automating such processes.

The current infrastructure for Web service discovery is typically human-oriented, in the style of yellow-pages. It features syntax-based searches which take into account keywords and taxonomies. Therefore, given one or more keywords, descriptions containing the input keywords are returned as searched results. This kind of search can often give poor performance, because it can fail to retrieve services described by synonyms (as well as singular/plural) of the searched string. Consequently, the requester must select the service which satisfies his requirements by manually filtering the returned services. Moreover, the syntax-based search severely affects the process of dynamically composing Web services, that involves: automatically discovering (and selecting) services, ensuring semantic and data-type compatibility and automatically invoking services.

The automation of the Web Service discovery would require service descriptions including machine-interpretable semantic information. The automation of service invocation would require service descriptions exposing service behavior, specifying the invocation order of the service operations. Unfortunately, available WSDL interfaces provide neither semantics information to describe the service functionality nor behavioral information to describe the service interaction behavior. They provide only descriptions at syntactic-level, making it difficult for requester and provider to interpret or represent non-trivial statement such as the meaning of inputs and outputs or applicable constraints. Therefore, the current standards for Web service is not really able to support neither automated service discovery nor (static and dynamic) service composition. This limitation may be overcome by providing a rich set of semantic annotations that augment the service descriptions. Indeed this is the direction on which researchers are focussing their efforts.

Particularly, two main problems there are currently analyzing: 1) how to model complex service interactions and 2) how to enrich service descriptions with semantic information. To address the problem of modeling complex service interactions, various choreography and orchestration languages have been proposed, such as BPEL [3], WSCI [5] and WS-CDL [117]. Such languages rely on standard WSDL and describe a service as a workflow of operations. They allow to specify the execution order of the operations, the operations that may be executed concurrently as well as the alternative execution path (if the workflow language provides conditional operators). Hence, by means of choreography/orchestration languages, simple (and complex) services can be composed together in order to model a new complex service. Particularly, such languages address the problem of statically composing Web services, but they do not solve the problem of automatic (or dynamic) Web service composition issue. Indeed, the selection of services that have to be orchestrated and their composition have to be manually performed by a computer-expert user.

To address the problem of how to enrich service descriptions with semantic information about their functionalities and meaning, researchers have augmented Web service with a semantic description of the functionality [140, 160], according to the SW vision (see Sect. 1.1). The result of the combination of the SW and Web services research domains for service description has been ***Semantic Web Services*** (SWS). A SWS includes a machine-understandable description of its capabilities, defined by referring shared knowledge contained in one or more ontologies. A SWS is defined though a service ontology, which enables machine interoperability of its capabilities and integration with domain knowledge. The availability of machine-understandable semantic descriptions of Web services is a must for automatizing their discovery, composition and execution across heterogeneous users and domains.

With SWSs, semantics information (i.e. concepts contextualizing the domain of the functionality that a service describes) are added to the provided service descriptions. The same could be done in the case of a service request, so the discovery process, rather then use a keyword-based search, could assess all the functionality connected to the semantic information specified with the request and hence search among the provided functionality that better satisfy the needs of the request.

1.2.2 The Semantic Web Services Infrastructure

Semantic Web Service infrastructure can be characterized along three orthogonal dimensions (see Fig. 1.3): *usage activities, architecture and service ontology* [38]. These dimensions relate to the requirements for SWS at business, physical and conceptual levels. *Usage activities* define the functional requirements that a framework for SWSs ought to support. The *architecture* of SWS defines the components needed for accomplishing these activities. The *service ontology* aggregates all concept models related to the description of a SWS, and constitutes the knowledge-level model of the information describing and supporting the usage of the service.

From the **usage activities perspective**, SWSs are seen as objects within an execution scenario of a business application. The activities required for running an application using SWS include: publishing, discovery, selection, composition, invocation, deployment and ontology management. They are described in the following.

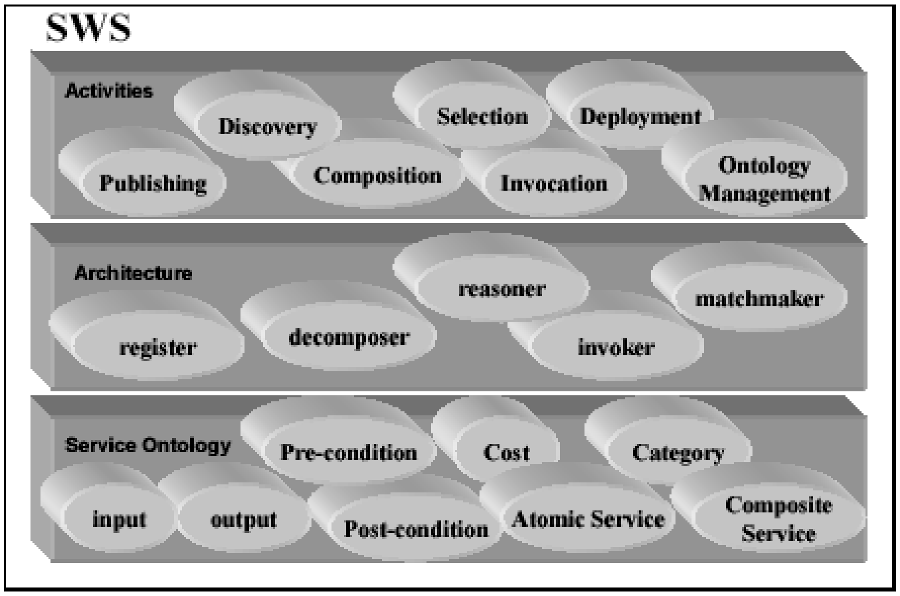

Figure 1.3: Semantic Web Services infrastructure dimensions.

Publishing and advertising SWSs will allow agents or applications to discover services based on its goals and capabilities. A semantic registry is used for registering instances of the service ontology for individual services. The service ontology distinguishes between information used for the matching during discovery and information used during the service invocation. In addition, domain knowledge should also be published or linked to the service ontology.

Service discovery consists of semantic matching between the description of a service request and the one of a published service. Queries involving the service name, input, output, preconditions and other attributes can be constructed and used for searching in the semantic registry. The matching can also be done at the level of tasks or goals to be achieved, followed by a selection of services which solve the task. The degree of match is based on some criteria, an example is the inheritance relationship of types. For instance, an input of type Professor of a provided service can be said to match an input of type Academic of a requested service.

A selection of services is required if there is more than one service matching the request. Non-functional attributes such as cost or quality can be used for choosing one service. In a more specialized or agent-based type of interaction, a negotiation process can be started between a requester and a provider. In general, a broker would check that the preconditions of tasks and services are satisfied and proves that the services postconditions and effects imply goal accomplishment. An explanation of the decision making process should also be provided.

Composition and choreography allow SWSs to be defined in terms of other simpler services. A workflow expressing the composition of atomic services can be defined in terms of the service ontology, by using appropriate control constructs. This description would be grounded on a syntactic description like BEPL4WS [3]. Dynamic composition is also being considered as an approach, during service request, in which the atomic services required to answer a request are located and composed on the fly. This requires an invoker which matches the output of atomic services against the input of the requested service.

The invocation of SWSs involves a number of steps, once the required inputs have been provided by the service requester. First, the service and domain ontologies associated with the service must be instantiated. Second, the inputs must be validated against the ontology types. Finally the service can be invoked or a workflow executed through the grounding provided.

The deployment of a Web service by a provider is independent of the publishing of its semantic description, since the same Web service can serve multiple purposes. But, the SWS infrastructure can provide a facility for the instant deployment of code for a given semantic description.

The management of service ontologies is fundamental for the SWS. It guarantees that semantic descriptions are created, accessed and reused within the SW.

From the **architecture perspective** (Fig. 1.3), SWSs are defined by a set of components realizing the activities illustrated above jointly with underlying security and trust mechanisms. The components include: a *register*, a *reasoner*, a *matchmaker*, a *decomposer* and an *invoker*. The reasoner is used during all activities and provides the reasoning support for interpreting the semantic descriptions and queries. The register provides the mechanisms for publishing and locating services in a semantic registry as well as functionalities for creating and editing service descriptions. The matchmaker mediates between the requester and the register, during the discovery and selection of services. The decomposer executes the composition model of composed services. The invoker mediates between requester and provider or between decomposer and provider when invoking services.

The **service ontology** is another dimension under which it is possible to define SWSs (Fig. 1.3). It represents the capabilities of a service and the restrictions applied to its use. The service ontology integrates, at the knowledge-level, the information which has been defined by Web services standards with related domain knowledge. This include: functional capabilities such as input, output, pre and post-conditions; non-functional capabilities such as category, cost and quality of service; provider related information, such as company name and address; task or goal-related information; and domain knowledge, defining, for instance, the type of the input of the service. This information can be divided in several ontologies.

Anyway, the service ontology used for describing SWSs will rely on the expressivity and inference power of the underlying ontology language supported by the SW.

Currently, there are three main approaches for semantically describing services. The first approach aims at adding semantics to WSDL. In this direction there are different initiatives and projects, one of the most important is WSDL-S[11], a language designed to augment the expressiveness of WSDL. The second approach aims at the development of new solutions to the problem of augmenting semantics to Web Services. It uses ontologies (and languages) specifically developed to describe services. Research efforts that belong to this approach include initiatives as OWL-S [41], WSMO [61] and SWSO [60]. They propose new standards to compile service descriptions exposing semantic information. OWL-S, for instance, is an agent oriented approach to SWSs, providing an ontology for describing Web Service capabilities. Moreover, they provide formalisms to specify the behavior of services and offer control constructs, in the style of the common programming languages, to model arbitrarily complex internal service workflows. The third approach is based on DLs as formal languages for expressing rich service descriptions. Efforts in this direction have been done in [129, 88, 195, 159, 196, 93]. The main reason of the attention to such an approach is that DLs are closely connected to OWL, hence, the use of DLs ensures compatibility with existing ontology standards. Furthermore, the formal semantics of DLs allows precise definition of the semantics of service descriptions. Moreover, discovery algorithms can be formally defined in terms of well-known DL inferences.

One of the aspects treated in this thesis focuses on the third approach. By the use of DLs for describing semantic services, the notion of *Constraint Hardness* is introduced. Such constraints allow to define optional and mandatory aspects of a service request that can be usefully exploited, during the matching process, for better satisfying a request. Moreover, a ranking procedure for matched services is proposed. It ranks services w.r.t. their similarity to the request and their capability to satisfy both optional and mandatory aspects of the request. This part of the work will be presented in detail in Sect. 5.2.3. Furthermore, clustering algorithms to cluster DL-based service descriptions are set up. They are exploited to improve the efficiency of the service discovery task. Details about clustering methods will be presented in the next section while the usage of such methods for improving the service discovery will be present in Sect. 5.2.2.

[11]http://www.w3.org/Submission/WSDL-S/

1.3 Inductive Learning for the Semantic Web

The availability of inference services in the Semantic Web context is fundamental for performing several tasks such as the consistency check of an ontology, the construction of a concept taxonomy, the concept retrieval etc. (see Sect. 1.1 and Chapt. 2 for more details).

Currently, the main approach used for performing inferences is deductive reasoning. In traditional Aristotelian logic, deductive reasoning is defined as the inference in which the (logically derived) conclusion is of no greater generality than the premises. Other logic theories define deductive reasoning as the inference in which the conclusion is just as certain as the premises. The conclusion of a deductive inference is necessitated by the premises: the premises cannot be true while the conclusion is false. Such characteristics of deductive reasoning are the reason of its usage in the SW. Indeed computing class hierarchy as well as checking ontology consistency require certain and correct results and do not need of high general conclusions with respect to the premises.

Conversely, tasks such as ontology learning, ontology population by assertions, ontology evaluation, ontology mapping and alignment require inferences that are able to return higher general conclusions with respect to the premises. To this end, inductive learning methods, based on inductive reasoning, could be effectively used. Indeed, inductive reasoning generates conclusions that are of greater generality than the premises, even if, differently from the deductive reasoning, such conclusions have less certainty than the premises. Specifically, in contrast to the deduction, the starting premises of the induction are specific (typically facts or examples) rather than general axioms. The goal of the inference is to formulate plausible general assertions explaining the given facts and that are able to predict new facts. Namely, inductive reasoning attempt to derive a complete and correct description of a given phenomenon or part of it[12].

The application of inductive learning methods in the SW context has received less attention. Some initial results in this direction have been proposed in [119, 43, 73], where the task of learning knowledge bases from examples is mainly focussed. This thesis shows that inductive learning methods and uncertain reasoning can be effectively used to outperform several tasks such as: ontology population, concept retrieval and query answering, resource retrieval and ranking. Indeed, most of this tasks are currently manually made (i.e. ontology population) or are not very efficient with the growth of available resources (resource retrieval and ranking). The

[12]Note that, of the two aspects of inductive inference: the generation of plausible hypothesis and their validation (the establishment of their truth status), only the first one is of primary interest to inductive learning research, because it is assumed that the generated hypothesis are judged by human experts and tested by known methods of deductive inference and statistics.

application of inductive reasoning on top of deductive reasoning results could be the right direction for having new and most efficient methods able to also make human-centered time consuming tasks less heavy.

Inductive learning methods are part of a most large discipline that is Machine Learning (ML), whose goal is to construct computer programs able to *learn*, namely able to improve their performance (w.r.t. a performance measure) in executing a task, on the ground of a certain experience. The learning process in ML is generally reduced to learn a target function able to perform the chosen task.

Inductive learning methods are made up of two different approaches, distinguished on the ground of the amount of inference they perform:

Learning from examples: (also called **supervised learning**) is a special case of inductive learning. Given a set of examples and counterexamples of a concept, the learner induces a general concept description that describes all of positive examples and none of the counterexamples.

Learning from observation: (also called **unsupervised learning**) is a very general form of inductive learning that includes discovery systems, theory-formation tasks, creation of classification criteria to form taxonomic hierarchies, without benefit of an external teacher. The learner is not provided with a set of instances of a particular concept, nor it is given access to an oracle that can classify internally-generated instances as positive or negative instances of any given concept. Furthermore, rather than focus on a single concept at a time, the observation may span several concepts that need to be acquired.

In learning from example, the observational statements are characterizations of some objects pre-classified by a teacher into one or more classes (concepts). The induced hypothesis can be viewed as a concept recognition rule (rather the target function) such that if an object satisfies this rule then it represents the given concept. Mainly, learning from example is used to solve classification problems.

In unsupervised learning the goal is to determine a general description characterizing a collection of observations. Thus, in contrast to learning from example that produces descriptions for classifying objects into classes on the basis of the object properties, unsupervised learning produces descriptions specifying properties of objects belonging to a certain class.

In the following these two different kinds of learning will be analyzed in more details. Moreover their extension for applying them to the SW context will be briefly illustrated.

1.3.1 Instance Based Learning Methods

Learning by example (or supervised learning) is supported by two different kinds of methods [1]:

- learning methods that construct a general, explicit description of the target function when training examples are provided. Algorithms that implement this learning method are called *eager learning algorithms* because they compile inputs into an intensional concept description (e.g. represented by a rule set, decision tree, neural network[13]). Responses to the information requests are given using this a priori induced description that is retained for future requests;
- *instance-based learning methods* that simply store the training examples. The generalization beyond these examples is postponed until a new instance must be classified. Each time a new query instance is encountered, its relationship to the previously stored examples is examined in order to assign a target function value for the new instance. Algorithms that implement this learning method are called *lazy learning algorithms.* They exhibit three characteristics that distinguish them from the previous ones: 1) they defer processing of their inputs until an information request is received; 2) they simply store inputs for future use; replies to the information requests are given by combining stored (e.g. training) data. 3) they discard the constructed answer and any intermediate results. This type of learning algorithms are also referred to as *memory-based* learning algorithms.

Lazy learning algorithms have the option of selecting a different hypothesis or local approximation for the target function for each of query instance. On the contrary, eager learning algorithms, building a generalized hypothesis at training time, must commit to a single hypothesis, that covers the entire training instance space, the classification of any new query instance. This makes such last methods more limited and make possible supposing that lazy learning algorithms could result more accurate. Anyway this reliability is payed with an increase of the computational cost at classification time with respect to eager learning algorithms.

Instance-based learning includes nearest neighbor algorithms and locally weighted regression algorithms (see [150] for more details) that often assume instances to be represented as points in an Euclidean space, and case-based reasoning algorithms [150] that use more complex, symbolic representations for instances. This thesis focus on nearest neighbor algorithms. The most important algorithm falling in this category is the k-NEAREST NEIGHBOR algorithm [47, 70, 150] that is a *classifier* that inputs a query instance x_q and outputs a prediction for its class.

[13]See [150, 143] for more details about the different kinds of representations.

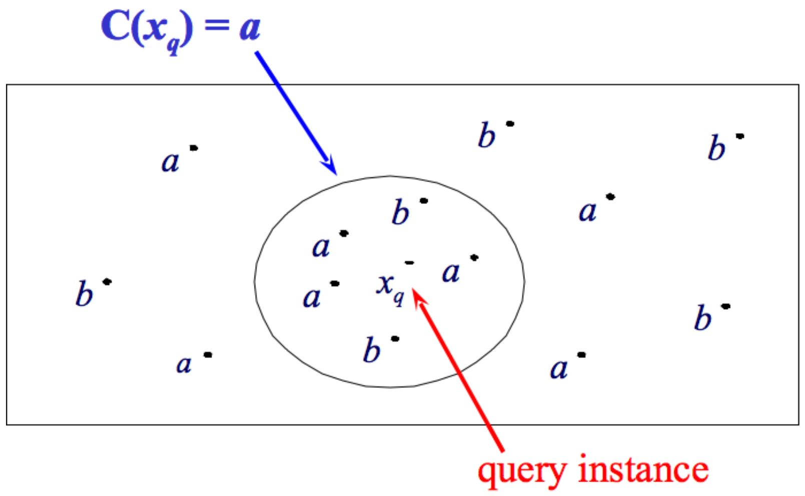

Figure 1.4: Classification of a query instance x_q by means of the 5-NEAREST NEIGHBOR algorithm. It assigns the class a to x_q.

In the classical setting, the k-NEAREST NEIGHBOR algorithm assumes each training instance $x = \{x_1, x_2, \ldots, x_{|F|}\}$ defined by a feature set F and corresponding to a point in multidimensional space. Each training instance is known to belong to a class c_j, member of a set of classes C. The learning phase consists in simply storing the training data. When a new query instance x_q is encountered, its classification is performed. Firstly, a set (namely k) of nearest neighbors training examples to x_q is retrieved from the memory by the use of a distance/dissimilarity function (typically the Euclidean distance is used). Once that the neighbors have been selected, the algorithm assigns to x_q the most common class value among the k nearest training examples (for more details about the k-NEAREST NEIGHBOR algorithm see Appendix A). An explanation of how the algorithm works is given in Fig. 1.4 in which it is assumed that instances are points in a two-dimensional space, the number of the neighboring elements to select is $k = 5$, the measure used is the euclidean distance, and the set of the classes is $C = \{a, b\}$.

As the k-NEAREST NEIGHBOR algorithm is a lazy learning algorithm, it never forms an explicit general hypothesis regarding the target function. It simply computes the classification of each new query instance by the use of a majority voting criterion. The inductive bias of this algorithm corresponds to an assumption that the classification of an instance x_q will be most similar to the classification of the other instances that are nearby w.r.t. the chosen dissimilarity/distance function. The crucial aspects of the k-NEAREST NEIGHBOR algorithm are: the value of the parameter k, determining the number of similar training instances to chose and the effectiveness of the dissimilarity measure used for selecting the training instances.

The accuracy of the algorithm strictly depends from them.

This thesis investigates the application of instance-based learning methods in the Semantic Web context that is a relational setting. Particularly in Sect. 5.1.1 an extension of the k-NEAREST NEIGHBOR algorithm applied to ontological knowledge is presented. The application of such algorithm to expressive knowledge bases such as ontologies is not straightforward at all. Indeed, k-NEAREST NEIGHBOR algorithm, in its classical setting, is applied to propositional knowledge bases, consequently described by very low expressive representation languages. The usage of a more expressive representation language requires the availability of new similarity measures able to exploit such expressive power. For this reason a set of semantic similarity and dissimilarity measures for Description Logics have been defined in Chapt. 4. Moreover, in order to cope with ontological knowledge, two main problems have to be taken into account: non-disjointness of the classes (rather a multi-class classification problem) and the Open World Assumption[14] that is typically made in the Semantic Web. On the contrary, in the classical k-NEAREST NEIGHBOR setting, classification is performed with classes assumed to be disjoint and an implicit Closed World Assumption (all that is not explicitly stated is false) is made.

Some effort in applying nearest neighbors methods to more expressive knowledge representation (namely more rich symbolic representations) and/or relational setting have been made in the literature [4, 190, 7, 202, 72]. Anyway, to the best knowledge of the writer, there is no developed work defining nearest neighbor methods that can be applied to ontological representations.

The developed relational k-NEAREST NEIGHBOR algorithm (see Sect. 5.1.1) classifies individuals of an ontology with respect to the concepts therein or with respect to new complex concept definitions built (on the fly) from the reference ontology. It can be effectively used for improving several different tasks such as: making the ontology population a less time consuming task, improving the retrieval[15] of a concept defined in an ontology or of a new query concept. Indeed, as will be experimentally shown in Sect. 5.1.1, classifying instances in an ontology by analogy w.r.t. the neighbors makes possible to enable the induction of new knowledge (new assertions non logically derivable). Such new assertions can be used as suggestions to the knowledge engineer that has only to validate them rather than manually create assertions. Moreover these assertions augment the amount of the information returned by the concept retrieval inference service computed only in a deductive way.

[14]See Sect. 1.1.2 for the meaning of *Open World Assumption*.

[15]See Sect. 2.3.2 for a formal definition of retrieval which is a DL inference.

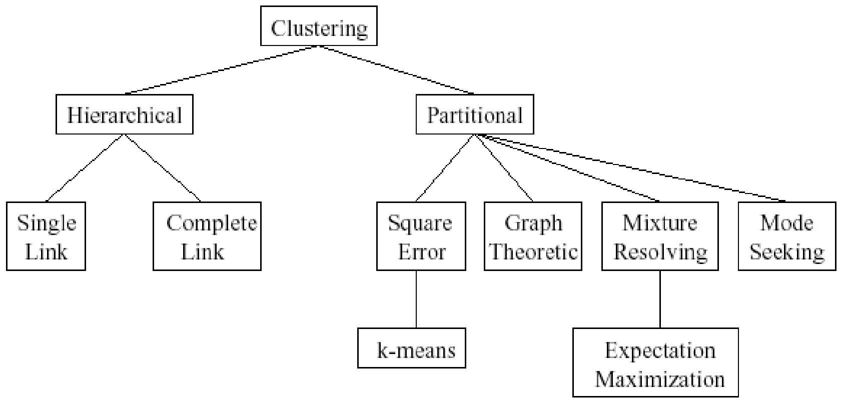

Figure 1.5: A taxonomy of clustering approaches

1.3.2 Cluster Analysis

Cluster Analysis is the discipline having as a main goal the study and setting of different methods finalized to the organization of a collection of unlabeled pattern (usually represented as a feature vector) into meaningful clusters based on a similarity criterion. Other terms synonymous with cluster analysis include: *unsupervised learning* [107], *numerical taxonomy* [186], *learning by observation* [142]. With the growing of the results in cluster analysis another goal has been focused, that is to set up clustering methods able to generate intensional (possibly human readable) description of the obtained clusters. This methods have been denominated *Conceptual Clustering methods*. In the sequel they will be analyzed separately.

Clustering Methods

Given a data collection, clustering algorithms give as output a set of meaningful clusters obtained by the use of a similarity criterion. Clusters (classes) are collections of objects whose intra-cluster similarity is high and inter-cluster similarity is low, namely, pattern within a valid cluster are more similar to each other than they are to a pattern belonging to a different cluster.

Different approaches for clustering data have been formalized. In Fig. 1.5 a taxonomy of the main clustering methods (based on the discussion in [107]) is shown. At the top level there is a distinction between hierarchical and partitional approaches. Hierarchical methods produce a nested series of partitions based on similarity criteria for merging or splitting clusters. The obtained clustering structure is usually called *dendrogram*. On the contrary, partitional methods identify the partition that optimizes a criterion function defined either locally (on a subset of the patterns) or globally (defined over all of the patterns). So a single data partition is obtained rather than a clustering structure. Partitional methods have advantages in applications involving large data sets for which the construction of a dendrogram is computationally prohibitive. On the other side, a problem accompanying the use of a partitional method is the choice of the number of desired output clusters. Moreover, partitional algorithms typically run multiple times with different starting states, in order to find the best configuration.

The simplest and most commonly used partitional clustering algorithm is the *k-means* algorithm [141] based on the use of the squared error criterion function which is the most frequently used. Other approaches belonging to the class of partitional clustering methods are: those based on graph theoretic (grounded on the construction of the *minimal spanning tree*[16]); and the mixture resolving and mode seeking based algorithms (whose goal is to find the parameters for a supposed distribution of the patterns[17]). With regard to hierarchical clustering approach, particularly important are: single link [186] and complete link [120] algorithms, as most of the hierarchical clustering algorithms are variants of such algorithms (besides of the minimum-variance algorithm [204, 153]).

Hierarchical algorithms are more versatile than partitional algorithms even if last ones are less expensive and so suitable for clustering large data sets. An analysis of the main algorithms for each category [108] showed that the single-link algorithm is suitable to perform clustering on data sets containing non-isotropic clusters including well-separated clusters, chain-like clusters, and concentric clusters. On the contrary, the k-means algorithm works well only on data sets having isotropic clusters, furthermore it strictly depends from the choice of the initial k seeds for starting the clustering process. Hybrid algorithms have been developed [154] in order to exploit the good features of both categories.

This thesis focuses on the single and complete link algorithms and their application to the SW context (more specifically to SWS), in order to improve the efficiency of the resource (service) discovery process. This choice has been motivated by the fact that the single and complete link algorithms are considered suitable for clustering non-large amount of data with very few information available [108].

The Single-Link and the Complete Link are hierarchical agglomerative clustering algorithms, generally applied to feature vector representations. They are called agglomerative because they start the clustering process by considering every pattern in a distinct cluster. Successively they merges clusters together on the ground of an adopted similarity function, until a stopping criterion is satisfied. They differ in the way the similarity between a pair of clusters is characterized. In the single-link algo-

[16]See [208] for more details.
[17]See [108] for more details.

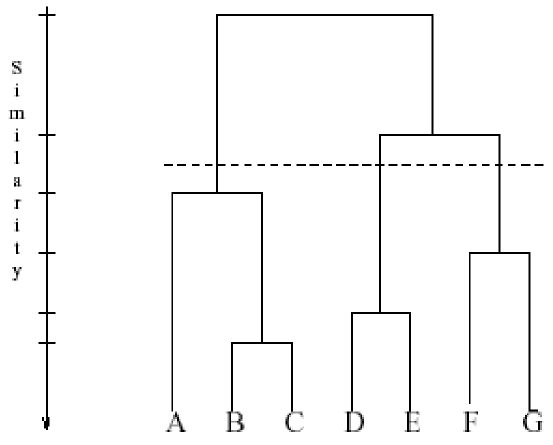

Figure 1.6: The dendrogram obtained using the single-link algorithm

rithm, the distance between two clusters is the *minimum* of the distances between all pairs of patterns drawn from the two clusters (one pattern from the first cluster, the other from the second). Conversely, in the complete-link algorithm, the distance between two clusters is the *maximum* of all pairwise distances between patterns in the two clusters. In both cases, two clusters are merged to form a larger cluster on the ground of the minimum distance criteria (for more details about single-link and complete-link algorithms see Appendix B). As these algorithms make use of the hierarchical methodology, their return a *dendrogram* as output that represents the nested grouping of patterns and the similarity levels at which grouping changes. An example of dendrogram, obtained from the single link algorithm, is shown in Fig. 1.6. A, B, C, E, F and G are labels of the corresponding patterns. The dendrogram can be broken at different levels to yield different clustering of the data.

A dendrogram can be effectively used in order to perform the resource discovery. Indeed, rather than match every available resource to find the requested one, clustered resources, represented by a dendrogram, can be explored by following the dendrogram paths that satisfy the match condition. However, in order to do this, intensional cluster descriptions could be useful. On the contrary, complete link and single link algorithms return only clustered data without any intensional descriptions. To reach this goal conceptual clustering methods are necessary.

Conceptual Clustering Methods

The problem of determining the *meaning* of the obtained clusters is simply left to the researcher in the presented algorithms. This represents a significance disadvantage. Indeed, a researcher analyzing data may be typically more interested in creating clusters that, besides of being mathematically well defined, also have a meaningful conceptual interpretation.

The difficulties in determining intensional cluster definitions are mainly due to the used measures that are typically numerical measures of object *similarity* that takes into account all pattern features but are *context-free*, namely they do not take into account the "environment" surrounding the objects of which similarity have to be measured, useful for characterizing objects configurations. Consequently, the resulting clusters (classes) do not necessarily have a simple conceptual description and may be difficult to interpret them. Moreover, such an approach could not be fully meaningful in cases in which only some attributes are relevant for describing object similarity.

First efforts for generating intensional cluster descriptions have been made by Diday et al. in [63]. They define a compact cluster description in terms of clusters prototypes or in terms of representative patterns as the centroid. However, this kind of descriptions do not take into account the context or concepts useful to characterize objects configurations. Such limitations have been overcome by ***conceptual clustering*** in which a configuration of objects forms a class only if it is describable by a concept from a predefined concept class. The main works introducing conceptual clustering are [144, 145] and [146] in which CLUSTER/2, that is one of the most famous conceptual clustering system, has been presented.

Conceptual clustering is defined as the process of constructing a concept network characterizing a collection of objects, with nodes marked by concepts describing object classes and links marked by the relationship between the classes. Conceptual clustering methods are distinguished from ordinary data clustering by generating a concept description for each generated class. In conceptual clustering it is not only the inherent structure of the data that drives the cluster formation, but also the description language available to the learner. Thus, a statistically strong grouping in the data may fail to be extracted by the learner if the prevailing concept description language is incapable of describing that particular regularity.

There is another important aspect solved by the conceptual clustering methods that is the use of similarity measures grounded on: 1) the characteristics of the considered objects; 2) the surrounding context; 3) the concepts representing existing cluster descriptions. This is because, the treatment of the context is not the unique aspect that has to be taken into account for obtaining meaningful cluster descriptions. Indeed, also a *context-sensitive* measure[18] as that proposed in [92] is

[18]The measure is the reciprocal of the *mutual distance*. To determine the mutual distance of an object A to B, objects in the collections are ranked w.r.t. their distances (generally the Euclidean distance) to A, and then w.r.t. their distances to B. The mutual distance from A to B is the sum of the rank of A w.r.t. B, and the rank of B w.r.t. A.

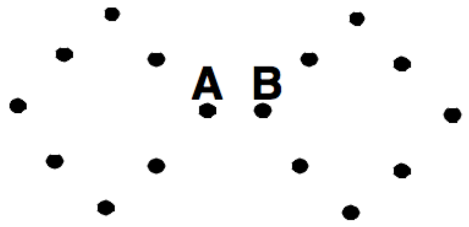

Figure 1.7: An example of Conceptual Clustering

not sufficient for reaching this goal. The reason is that such measure is *concept-free*, namely it does not depend from any external concepts which might be useful to characterize object configurations. Consequently methods that use these kinds of measures may be fundamentally unable to capture the "Gestalt properties" of object clusters, namely properties that characterize a cluster as a whole and that are not derivable from properties of individual entities. In order to detect such properties, the clustering method must be equipped with the ability to recognize configurations of objects that correspond to certain "concepts". To illustrate this aspect, a famous problem of clustering point is shown in Fig. 1.7. If a human analyze this figure he/she will typically describe the observed points as "arranged in two diamonds". Thus, the point A and B, although closer to each other than the other points, will be placed in separate clusters. The human solution involves partitioning the points not on the basis of pairwise distance, but on the basis of *concept membership*. This is the underlying idea of conceptual clustering. Indeed, from the viewpoint of conceptual clustering, the "similarity" between two point A and B depends not only on those points and surrounding points (E), but also on a set of concepts C which are available for describing A and B together. In [144] such measure is called *conceptual cohesiveness*[19] and it is expressed as a function:

$$\text{ConceptualCohesiveness}(A, B) = f(A, B, E, C)$$

where A and B are two considered objects, E is the neighborhood of A and B and C is the set of concepts available for describing data to cluster.

Hence, besides of the generation of intensional cluster descriptions, the other major difference of conceptual clustering methods w.r.t. numerical taxonomy methods lies in the extension of the notion of similarity measure between objects into a more general notion of "conceptual cohesiveness" that takes into account not only the properties of individual objects, but also their relationship to other objects and their relationship to some predetermined concepts characterizing object collections.

This Thesis focus on the realization of conceptual clustering methods applied to ontological knowledge in order to improve the resource discovery process. The application of the illustrated clustering methods to expressive data descriptions requires

[19](See [144, 146] for more details about conceptual cohesiveness measure.

non trivial efforts. First of all, suitable measures, able to exploit the expressiveness of the language have to be considered and most complex descriptions have to be managed. In Sect. 5.2.2 three clustering algorithms for ontological knowledge will be presented. The first one is a conceptual clustering algorithm grounded on the agglomerative hierarchical approach. It iteratively computes the similarity between objects, by the use of the similarity measures proposed in Chap. 4, and merge the two most similar objects in one cluster, hence, by the use of DL non-standard inferences (see Sect. 2.4) an intensional description of the obtained cluster is defined and the objects in the cluster are discarded. The process stop when all available objects are in a unique cluster. The other two developed methods are an extended version of the single-link and complete link algorithms able to cope with ontological knowledge. For this reason, similarity/dissimilarity measures for ontological knowledge (presented in Chap. 4) are used. The stop criterion is given by collecting all elements in a unique cluster. Once that the dendrogram of the clustering is obtained, an intensional cluster description at each clustering level is generated, by the use of DL non-standard inferences.

Clustering algorithms are often used to speed up the resource retrieval process in presence of very large data sets or to increase the efficiency of the decision making task. In [174], a large collection of documents is clustered and each of the clusters is represented using its centroid. In order to retrieve documents relevant to a query, the query is matched with the cluster centroids rather than with all the documents. In [69] clustering methods are used to perform indexing of large data collection and so improve the efficiency of the decision making task.

This thesis propose a way to improve the service discovery process of semantic web services (see Sect. 1.2) by the use of the realized clustering algorithm for ontological knowledge. Service discovery is the process responsible for searching and finding an available service able to satisfy a request among a set of provided services. Nowadays this process is performed by matching the request against all available provided services. Obviously, the efficiency of the discovery task degrades with the increasing of the amount of available services. The idea, that will be presented in Sect. 5.2.2, consists in firstly clustering the provided services jointly with the intensional cluster descriptions of every node of the obtained dendrogram. Hence, the matching procedure could be performed w.r.t. the intensional description of the nodes in the dendrogram rather than w.r.t. all available service descriptions. In this way, in the best case, it is possible to reduce the number of matching from linear (in the number of available services) to logarithmic. All the implication of the case will be analyzed in Sect. 5.2.2.

1.4 Objectives of the Dissertation

The Semantic Web represents the new vision of the current web. It is a Web of enriched machine-readable content (called metadata), in which the semantics of web-resources is encoded in a machine-interpretable form. The SW goal is to make resources, available on the Web, not only human-readable but also machine-readable and machine-interoperable. The realization of the SW vision requires interdisciplinary research involving: Knowledge Representation, Databases, Software Engineering, Natural Language Processing, Machine Learning (ML) besides of business aspects and applications. Nevertheless, as Rudi Studer highlighted during the last edition of the International Semantic Web Conference[20], ML has a great potential w.r.t. the necessity of the Semantic Web research unfortunately not yet really exploited.

A goal of this thesis is to apply ML methods to the Semantic Web and Semantic Web Services fields in order to improve reasoning procedures, induce new knowledge not logically derivable and improve the efficiency and effectiveness of tasks such as ontology population and service discovery and ranking. Nevertheless, as seen in Sect. 1.3, most of ML methods need of (dis-)similarity measures able to determine the (dis-)similarity value among the considered resources. Unfortunately, assessment of similarity values among complex concept descriptions (as those in the ontologies), is a field that has not been deeply investigated (as also asserted by Borgida et al. in [29]). Hence, in order to reach the goal, a set of (dis-)similarity measures for ontological settings is defined.

The considered knowledge representation language are Description Logics (DL), which represent the theoretical foundation of the OWL Language, that is the standard ontology representation language. Hence, the set of defined (dis-)similarity measures are formalized with respect to such representation languages and most of all $\mathcal{ALC}$ logic, even if measures for $\mathcal{ALN}$ logic and language independent measures are also defined.

Such measures applicable to ontological knowledge may be used to define new and/or modified ML methods for ontological representations. This thesis investigates and set up various learning methods and algorithms. The first learning algorithm defined is an instance-based classifier able to classify individuals w.r.t. concepts asserted in an ontology. The realization of such algorithm requires to solve important problems: how to cope with the Open World Assumption typically made in the SW, and how to cope with the non-disjointness of the classes (concepts) with respect to the classification is performed. The solutions to these problems besides

[20]see http://iswc2006.semanticweb.org/program/keynote_studer.php for the slides of the presentation (ISWC 2006)

of the algorithm definitions are illustrated in Sect. 5.1.1.

The realized classifier may be used to enforce the computation of the concept retrieval[21]. Indeed, asking for the instances of an existing concept in an ontology, it returns a set of instances induced from the Knowledge base, besides of the set of instances that are asserted in the knowledge base or that can be derived. It performs a form of *approximate reasoning*[22] grounded on induction, retrieving concept assertions, even in presence of inconsistent knowledge bases. As argued by Hitzler et al. in [100], approximate reasoning techniques are fundamental for obtaining scalable systems, even if the price to be paid could be unsoundness or incompleteness (or both); anyway this is done in a controlled and well-understood manner that allows to assess the quality of the induction made by the approximate reasoner.

On the ground of the way in which the classifier performs the concept retrieval, it may be used also to improve the query concept tasks. Indeed, in the same way, the classifier could be used for determining the extension of a new query concept, defined in terms of existing concept in the considered ontology. Furthermore, the induced assertions could be employed to make the ontology population task less time consuming. Indeed, they can be provided as suggestion to the knowledge engineer that has simply to validate the induced knowledge rather than manually write all the assertions.

Such a classifier has been realized in a double version: extending the k-nearest neighbor algorithm to work on ontological knowledge and by integrating a defined relational kernel function for ontological knowledge in an existing Support Vector Machine available on the Web.

In this thesis, conceptual clustering algorithms applicable to ontological knowledge are also presented (see Sect. 5.2.2). Specifically, extensions of the complete-link and single-link algorithms applied to DLs knowledge bases are formalized. The main characteristic of these formalizations is the construction of intensional definitions for the obtained clusters by the use of DL non-standard inferences. Moreover, another conceptual clustering algorithm is formalized. It is an iterative agglomerative hierarchical clustering algorithm: at each step the two most similar elements are found and merged in a unique cluster. Then an intentional cluster description is generated and the elements belonging to the cluster are discarded. The process stops when all elements are merged in a cluster. All three algorithms return a tree as output of the clustering process that can be used for improving retrieving and indexing semantically annotated resources. Specifically, this thesis focuses the improvement of the service discovery task exploiting clustering methods for ontological knowledge.

[21]See Sect. 2.3.2 for more details about concept retrieval.

[22]Some example of approximate reasoning are: non-monotonic reasoning, paraconsistent reasoning and resolution-based reasoning. See [100] for more details.

Once that provided services able to satisfy a request are discovered, a desirable situation for the requester would be to have a rank of the selected services based on a fixed criterion. Currently most of the methods simply return a flat list of discovered services. Following the DLs-based framework for describing services proposed by Grimm et al. in [93], a service description (particularly for the service requests) characterized by the distinction between *Hard Constraints* i.e. constraints that have necessarily to be satisfied, HC for short and *Soft Constraints* i.e. constraints that can be optionally satisfied, SC for short is presented in this thesis (see Sect. 5.2.1). Hence, a ranking procedure is proposed. It ranks the most semantically similar services to the request and satisfying both HC and SC in the highest positions, while services less semantically similar to the request and/or satisfying only HC are ranked in the lowest positions (see Sect. 5.2.3). In this way the requester can find the proper service in a more efficient and effective way and also the negotiation process could result less complex.

1.5 Chapter Summaries

This PhD Thesis aims at providing a set of semantic (dis-)similarity measures applicable to ontological knowledge expressed in Description Logic and to formalize some learning methods to be applied in the Semantic Web and Semantic Web Services contexts. It is organized as follow.

Chapter 1 provides an overview the Semantic Web and Semantic Web Services, defining what currently is intended with these expressions and the issues related to them. Hence, the learning problem in Machine Learning is presented as well as an overview of the main learning methods. The applications of learning methods to the Semantic Web and Semantic Web Services are briefly presented.

In Chapter 2 the reference language for the knowledge representation is presented. Particularly, Description Logics are analyzed jointly with the reasoner services they offer.

Chapter 3 reviews previous works on (dis-)similarity measures assessment in propositional and relational setting. Specifically, the main approaches for determining (dis-)similarity functions are analyzed. They include: measures based on geometrical models, path distance measures, feature matching measures, Information Content based measures, measures based on alignment and transformational models, relational kernel functions, and first measure proposals for DLs. Hence, the reasons of the necessity of defining new measures for assessing similarity between elements expressed in Description Logics are briefly explained.

In Chapter 4 a set of semantic (dis-)similarity measures for concept descriptions and individuals expressed in Description Logics and particularly in $\mathcal{ALC}$ and $\mathcal{ALN}$ logics are presented. Moreover, some language independent measures are also defined.

Chapter 5 presents similarity based learning methods applied to the Semantic Web and Semantic Web Services contexts. Specifically, some classifiers for individuals with respect to classes (concepts) defined in an ontology or with respect to new concepts built on the fly are presented. Two different classifier are presented. One is developed as a modified version of the k-nearest neighbor algorithm applied to ontological knowledge. Another is obtained by integrating a relational kernel function with an existing Support Vector Machine, available on the Web. Moreover clustering methods are presented in order to improve the efficiency of the Semantic Web Service discovery. Hence, a ranking procedure for ranking services selected by the discovery process is proposed. It ranks provided services on the base of constraints hardness and similarity criterion with respect to the request.

Chapter 6 presents conclusions and further research directions. It discusses the main contributions and drawbacks of the presented work and proposes lines for future research.

Chapter 2

Description Logics

Research in the field of knowledge representation and reasoning is usually focused on methods for providing high-level descriptions of the world that can be effectively used to build intelligent applications. In this context, "intelligent" refers to the ability of a system to find implicit consequences of its explicitly represented knowledge. Such systems are therefore known as knowledge-based systems.

Description Logics (DLs) is the name for a family of knowledge representation (KR) formalisms. By the use of such formalisms, the knowledge referring to an application domain (the "world") is described by first defining the relevant concepts of the domain (its terminology), and then by specifying object properties and concept assertions occurring in the domain (the world description).

Description Logics descend from the so-called "structured inheritance network representations" [32, 33], which were introduced to overcome the ambiguities of the early semantic networks [187, 34] and frames [149], and which were first realized in the system KL-One [35]. The following ideas, first put forward in Brachmanś work on structured inheritance networks [32], have largely shaped the subsequent development of the DLs:

- The basic syntactic building blocks are atomic concepts (unary predicates), atomic roles (binary predicates) and individuals (constants)
- The expressive power of the language is restricted in that it uses a rather small set of (epistemologically adequate) constructors for building complex concepts and roles
- Implicit knowledge about concepts and individuals can be inferred automatically by the use of inference procedures.
- Adoption of the *open-world assumption*

The semantics of concepts is defined in terms of the set-theoretic interpretation. A concept is interpreted as a set of individuals while roles are interpreted as sets of pairs of individuals. The domain of interpretation can be chosen arbitrarily, and it may be infinite.

DLs distinguish features are the formal, logic-based semantics, the emphasis on reasoning as a central service, and the *open-world assumption*. Particularly, reasoning is the mechanism that allows to infer implicit knowledge from the knowledge that is explicitly contained in the knowledge base. DLs support inference patterns that occur in many applications of intelligent information processing systems, and which are also used by humans to check and understand the world. An example of inference is the classification of concepts and individuals. The classification of concepts determines subconcept/superconcept relationships (called subsumption relationships in DL) between the concepts of a given terminology. Hence, a terminology can be structured in the form of a subsumption hierarchy. The classification of individuals (or objects) determines whether a given individual is an instance of a certain concept (i.e. whether this instance relationship is implied by the description of the individual and the definition of the concept). It provides useful information on the properties of an individual.

In the remainder of this chapter an overview of syntax and semantics for the whole family of DLs will be given. Moreover the main standard and non-standard DL inference services will be presented. In this chapter, just the necessary results for gaining the minimal awareness of DL basic theory are reported; a thorough exploration of 20 years of DLs research can be found in [8]. In order to avoid the citation of the same reference many times, for this chapter the following convention will hold: all definitions, propositions and results have been drawn from [8] where not differently indicated.

2.1 Knowledge Representation in DL

The realization of knowledge systems involves two primary aspects. The first consists in providing a precise characterization of a knowledge base. This involves precisely characterizing the type of knowledge to be specified to the system as well as clearly defining the reasoning services the system has to provide. The second aspect consists in providing a rich development environment where the user can benefit from different services that can make his/her interaction with the system more effective.

As regard building a knowledge base, it is important to distinguish between: the *intentional knowledge* that is the general knowledge about the problem domain, and the *extensional knowledge*, which is specific of a particular problem. So, a DL knowledge base is comprised by: a *"TBox"* and an *"ABox"*. The TBox contains intensional knowledge in the form of a terminology (hence the term "TBox", even if "taxonomy" could be also used). It is built through declarations that describe general properties of concepts. It mainly represents the vocabulary of an application domain. The ABox contains extensional knowledge, also called assertional knowledge (hence the term "ABox"), namely knowledge that is specific to the individuals of the domain of discourse. Intensional knowledge is usually thought not to change, to be "timeless" in a way, while extensional knowledge is usually thought to be contingent, or dependent on a single set of circumstances, and therefore subject to occasional or even constant change.

In the following the notion of TBox and ABox are formalized. A TBox contains elementary descriptions that are *atomic concepts* and *atomic roles*. Complex descriptions can be built from them, inductively, by the use of *concept constructors*. Such complex concept descriptions can be named by the use of a symbolic name; it represents an abbreviation of the complex concept description. The basic form of a declaration in a TBox is a concept *definition*, that is, the definition of a new concept in terms of other previously defined concepts. Such definitions are called *terminological axioms*; they makes statements about how concepts and roles are related to each other. For example, a woman can be defined as a female person by writing the following declaration: $\mathsf{Woman} \equiv \mathsf{Person} \sqcap \mathsf{Female}$. Such a declaration is usually interpreted as a logical equivalence, which amounts to providing both sufficient and necessary conditions for classifying an individual as a woman. This kind of axioms are also called *equality* axioms. A terminology containing only equality axioms is called a *definitorial* terminology.

In a TBox, a symbolic name is defined no more than once. Given a TBox $\mathcal{T}$, the atomic concepts occurring in $\mathcal{T}$ can be divided into two sets, the *name symbols* $\mathcal{N}_\mathcal{T}$ that occur on the left-hand side of axioms and the *base symbols* $\mathcal{B}_\mathcal{T}$ that occur only on the right-hand side of axioms. Name symbols are often called *defined concepts* and base symbols *primitive concepts*[1].

Equality axioms are much stronger than the ones used in other kinds of knowledge representation which typically impose only necessary conditions. These kind of axioms are called *inclusion* axioms. They are used when the knowledge engineer is not able to precisely define a concept. So, for instance the concept Woman could be defined, by an inclusion axiom, as: $\mathsf{Woman} \sqsubseteq \mathsf{Person}$. An inclusion axiom (i.e. an inclusion whose left-hand side is atomic) is also called *specialization*.

A set of axioms is called a *generalized terminology* if the left-hand side of

[1]Note that some papers use the notion of "primitive concept" with a different meaning; e.g. it is used as synonym of what here is called atomic concept or it is used to denote the (atomic) left-hand side of concept inclusions.

each axiom is an atomic concept and for every atomic concept there is at most one axiom where it occurs on the left-had side. Hence, a generalized terminology may contain both equality and inclusion axioms. It is possible to transform a generalized terminology into a *definitorial* terminology (see [8] for more details). Moreover a TBox is called *cyclic* if it contains at least one *cycle*, namely if there exists at least on axioms in which an atomic concept uses itself in its definition (see [8] for more details), otherwise a TBox is called *acyclic*. In this dissertation, only acyclic TBoxes will be considered, where not specified differently.

The ABox represents the *world description*. In the ABox, it is described a specific state of affairs of an application domain in terms of concepts and roles. Here, individuals are introduced by giving them names, as well as, properties of these individuals are asserted. An ABox contains assertions about individuals, usually called *membership assertions*. For example Woman(MARY) states that the individual MARY is a Woman and represents a *concept assertion*. In the same way an ABox contains *role assertions* such as hasChild(MARY,JOHN) that specifies that MARY has a child which is JOHN.

A DL system not only stores terminologies and assertions, but also offers services to reason about them. Typical reasoning tasks for a terminology are to determine whether a description is *satisfiable* namely non-contradictory and to determine whether one description is more general than another one, namely whether the first *subsumes* the second. The subsumption test allows to organize the concepts of a terminology into a hierarchy, according to their generality. Reasoning tasks for an ABox are: to find out whether the set of assertions is *consistent* and to check whether the assertions in the ABox entail that a particular individual is an *instance* of a given concept description. A concept description can also be conceived as a query, describing a set of objects one is interested in. Thus, with instance tests it is possible to retrieve the individuals that satisfy the query.
Satisfiability checks of descriptions and consistency checks of sets of assertions are useful to determine whether a knowledge base is meaningful or not.

In any application, a Knowledge Representation (KR) system is embedded into a larger environment. Other components interact with the KR component, by querying the knowledge base and by modifying it, that is, by adding and retracting concepts, roles, and assertions. A restricted mechanism to add assertions are rules (which are an extension of the logical core formalism) is still logically interpreted.

Since Description Logics are a KR formalism, as a KR system should always answer user queries in reasonable time, in the same way DL researchers are interested in reasoning procedures that are *decision procedures*, i.e., unlike, e.g., first-order theorem provers, these procedures should always terminate, both for positive and for negative answers. Anyway, the guarantee to have an answer in finite time not imply that the answer is also given in reasonable time. For this reason a lot of research about DLs concerns the investigation of their computational complexity by the use of decidable inference problems.

Decidability and complexity of the inference problems depend on the expressive power of the considered DL. Expressive DLs are likely to have inference problems of high complexity, or they may even be undecidable. Weak DLs (with efficient reasoning procedures) may not be efficiently expressive to represent the important concepts of a given application. Investigating this trade-off between the expressivity of DLs and the complexity of their reasoning problems has been one of the most important issues in DLs research.

2.2 Syntax and Semantics

Description logics are a family of knowledge representation languages, fragment of First Order Logic (FOL). As seen in the previous section, atomic concepts and atomic roles represent elementary descriptions. Complex descriptions can be built from atomic concepts and roles, inductively, by the use of concept constructors. Different description logics are distinguished in the DLs family, by means of the constructors they provide. In the sequel, various languages from the family of $\mathcal{AL}-$*languages* will be discussed, with particular attention to $\mathcal{ALE}$, $\mathcal{ALC}$, $\mathcal{ALN}$ and $\mathcal{ALCN}$ DLs. The language $\mathcal{AL}$ stands for *attributive language*, it has been introduced in [176] as a minimal language that is of practical interest. The other languages of this family are extension of $\mathcal{AL}$.

Whatever DL language is considered, it can be defined in terms of two non empty sets $N_C = \{A, B, ...\}$ and $N_R = \{R, S, ...\}$. Items in N_C are atomic concepts, items in N_R are atomic roles. Complex concept descriptions C, D in $\mathcal{AL}$ are formed according to the syntax and semantics presented in Table 2.1.

Table 2.1: $\mathcal{AL}$ constructors and their meaning.

Name	**Syntax**	**Semantics**
atomic concept	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
top concept	$\top$	$\Delta^{\mathcal{I}}$
bottom concept	$\bot$	$\emptyset$
atomic negation	$\neg A$	$\Delta^{\mathcal{I}} \setminus A^{\mathcal{I}}$
concept conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
limited existential restriction	$\exists R.\top$	$\{a \in \Delta^{\mathcal{I}} \mid \exists b \ (a,b) \in R^{\mathcal{I}}\}$
universal restriction	$\forall R.C$	$\{a \in \Delta^{\mathcal{I}} \mid \forall b \ (a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\}$

Note that, in $\mathcal{AL}$, negation can only be applied to atomic concepts, and only the top concept is allowed in the scope of an existential quantification (existential concept restriction) over a role.

In order to define a formal semantics of $\mathcal{AL}-$concepts, *interpretations* $\mathcal{I}$ have to be considered. An interpretation consists of a non-empty set $\Delta^{\mathcal{I}}$, representing the *domain of the interpretation*, and an *interpretation function*, which assigns to every atomic concept A a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and to every atomic role R a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The interpretation function is then extended to concept descriptions by means of the inductive definitions presented in Table 2.1. Moreover two concepts C and D are defined *equivalent* ($C \equiv D$ in symbols), if and only if $C^{\mathcal{I}} = D^{\mathcal{I}}$. In addition it is possible to extend the interpretation to provide (terminological[2]) TBox semantics. The interpretation of a TBox amounts to interpreting all of its axioms and hence all concepts contained therein. Therefore, an interpretation is a *model* for a TBox $\mathcal{T}$ if it satisfies all of its axioms.

Besides of the giving a semantics to the TBox, a semantics to the ABox is also given, it is an "*open-world semantics*". The semantics to the ABox is given by extending interpretations to individual names. Considered an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, it also maps each individual name a to an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. In general it is assumed that distinct individual names denote distinct objects, therefore this mapping has to respect the *unique name assumption (UNA)*, that is if a, b are distinct names, then $a^{\mathcal{I}} \neq b^{\mathcal{I}}$. The interpretation $\mathcal{I}$ *satisfies* the concept assertion $C(a)$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$, and it *satisfies* the role assertion $R(a, b)$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$; b is called role filler of R. An interpretation satisfies the ABox $\mathcal{A}$ if it satisfies each assertion in $\mathcal{A}$; $\mathcal{I}$ is a *model* of $\mathcal{A}$. Finally, $\mathcal{I}$ satisfies an assertion α or an ABox $\mathcal{A}$ with respect to a TBox $\mathcal{T}$ if in addition to being a model of α or of $\mathcal{A}$, it is a model of $\mathcal{T}$. A model of $\mathcal{A}$ and $\mathcal{T}$ is an abstraction of a concrete world, where the concepts are interpreted as subsets of the domain as required by the TBox and where the membership of the individuals to concepts and their relationships with one another in terms of roles respect the assertions in the ABox. Remembering that a knowledge base $\mathcal{K}$ is defined as $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ where $\mathcal{T}$ is a TBox and $\mathcal{A}$ is an ABox, an interpretation $\mathcal{I}$ is a *model* of $\mathcal{K}$ if it is a model of both $\mathcal{T}$ and $\mathcal{A}$.

More expressive languages than the $\mathcal{AL}$-language can be obtained adding further constructors to $\mathcal{AL}$. Different languages are defined, depending from the constructors they allow. In Table 2.2, the list of the possible constructors and their semantics is reported, while in Table 2.3 the list of the different languages is reported. For every language, the allowed constructors are shown.

By looking at Tab. 2.2, it is important to note that the *full existential quan-*

[2]In case of cyclic TBox, the notion of interpretation make use of a fixpoint semantics; see [8] for more details.

Table 2.2: DL Constructors semantics

Name	Syntax	Semantics
atomic negation	$\neg A,\ A \in N_C$	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
full negation	$\neg C$	$C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
concept conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
concept disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
full existential restriction	$\exists R.C$	$\{a \in \Delta^{\mathcal{I}} \mid \exists b\ (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}$
at most restriction	$\leq nR$	$\{a \in \Delta^{\mathcal{I}} \mid \mid \{b \in \Delta^{\mathcal{I}} \mid (a,b) \in R^{\mathcal{I}}\} \mid \leq n$
at least restriction	$\geq nR$	$\{a \in \Delta^{\mathcal{I}} \mid \mid \{b \in \Delta^{\mathcal{I}} \mid (a,b) \in R^{\mathcal{I}}\} \mid \geq n$
qualified at most restriction	$\leq nR.C$	$\{a \in \Delta^{\mathcal{I}} \mid \mid \{b \in \Delta^{\mathcal{I}} \mid (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\} \mid \leq n$
qualified at least restriction	$\geq nR.C$	$\{a \in \Delta^{\mathcal{I}} \mid \mid \{b \in \Delta^{\mathcal{I}} \mid (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\} \mid \geq n$
one-of	$\{a_1, a_2, ...a_n\}$	$\{a \in \Delta^{\mathcal{I}} \mid a = a_i, 1 \leq i \leq n\}$
has value	$\exists R.\{a\}$	$\{b \in \Delta^{\mathcal{I}} \mid (b, a^{\mathcal{I}}) \in R^{\mathcal{I}}\}$
inverse of	R^-	$\{(a,b) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (b,a) \in R^{\mathcal{I}}\}$

tification (or full existential restriction), indicated by $\exists R.C$ differs from the limited existential quantification, indicated by $\exists R.\top$. Indeed the latter allows only the top concept in the scope of an existential quantification over a role, while the former allows arbitrary concepts to occur in the scope of the existential quantifier. In the same way, it is important to note that *atomic negation*, indicated by $\neg A$, allows to apply negation only to atomic concepts while *full negation*, indicated by $\neg C$, allows the negation of any concept (atomic or defined). Another interesting difference is between *number restriction*, indicated by $\leq n\ R$, $\geq n\ R$, and *qualified number restriction*, indicated by $\leq n\ R.C, \geq n\ R.C$. The semantics of the former requires that the number of instances of R are at least/at most n without any other constraints

Table 2.3: Some $\mathcal{AL}$-languages

DL Name	Constructors
$\mathcal{ALN}$	$\neg A, \sqcap, \forall R.C, \exists R.\top, \leq nR, \geq nR$
$\mathcal{ALE}$	$\neg A, \sqcap, \forall R.C, \exists R.C$
$\mathcal{ALEN}$	$\mathcal{ALE} \cup \{\leq nR, \geq nR\}$
$\mathcal{ALC}$	$\mathcal{ALE} \cup \{\neg C, \sqcup\}$
$\mathcal{ALCN}$	$\mathcal{ALC} \sqcup \{\leq nR, \geq nR\}$
$\mathcal{SHOIN}$	$\mathcal{ALCN} \cup \{R^-\}$, role hierarchies, role transitivity, role symmetry, (inverse) functional properties

while the latter require that the number of instances of R having *fillers* (namely the second element of the instance of a role) that belong to the concept C are at least/at most n.

Among the languages of DL family (see Table 2.3), some DLs are well known languages as they have been thoroughly studied in literature or have been employed in some remarkable application domain. Some examples are: $\mathcal{ALN}$ that is the simplest DL including number restrictions, $\mathcal{ALC}$ that is, in its turn, the simplest allowing for full negation[3] and, increasing in expressivity, $\mathcal{SHOIN}$ [8, 102] that has been employed for defining the OWL language (see Sect. 1.1.1).

2.3 Standard Inference Services

A knowledge representation system based on DLs is able to perform specific kinds of reasoning. Indeed the purpose of a knowledge representation system goes beyond storing concept definitions and assertions. As explained in Sect. 2.1, a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ is made by a set of axioms, and so, like any other set of axioms (such as in first-order predicate logic), $\mathcal{K}$ contains implicit knowledge that can be made explicit through inferences. In the following, DL inferences will be discussed, starting from inferences for concepts and TBoxes, then inferences for ABoxes, and finally for TBoxes and ABoxes together. It will turn out that there is one main inference problem, namely the *consistency check* for ABoxes, to which all other inferences can be reduced.

2.3.1 TBox Reasoning

In the construction of a terminology $\mathcal{T}$, new concepts are defined, likely in terms of concepts already defined. During this process, it is important to find out whether a newly defined concept makes sense or whether it is contradictory. Logically, a concept makes sense if there is an interpretation that satisfies the axioms of $\mathcal{T}$ (that is, if there is a model of $\mathcal{T}$). A concept with this property is said to be *satisfiable* w.r.t. $\mathcal{T}$ and *unsatisfiable* otherwise. Checking satisfiability of concepts is formally defined as:

Definition 2.3.1 (Satisfiability) *A concept C is* satisfiable *w.r.t. $\mathcal{T}$ if there exists a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}}$ is nonempty. In this case it is said also that $\mathcal{I}$ is a model of C.*

[3]As full negation is available, De Morgans laws for disjunctions can be used [176].

Other important inferences for concepts (that can be reduced to (un-)satisfiability) are: *subsumption*, checking *concepts equivalence* and checking *concepts disjointness.* They can be formally defined as follow.

Definition 2.3.2 (Subsumption) *A concept C is subsumed by a concept D w.r.t. $\mathcal{T}$ if $C^{\mathcal{I}} \subset D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$. In this case it is written $C \sqsubseteq_{\mathcal{T}} D$ or $\mathcal{T} \models C \sqsubseteq D$.*

The subsumption test is the most common inference in DL. One of its most useful use cases is building up concepts subsumption hierarchies (taxonomies) in order to organize the concept graph of a TBox for instance in a knowledge base management system.

Definition 2.3.3 (Equivalence) *Two concepts C and D are equivalent w.r.t. $\mathcal{T}$ if $C^{\mathcal{I}} = D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$. In this case it is written $C \equiv_{\mathcal{T}} D$ or $\mathcal{T} \models C \equiv D$.*

Definition 2.3.4 (disjointness) *Two concepts C and D are disjoint w.r.t. $\mathcal{T}$ if $C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset$ for every model $\mathcal{I}$ of $\mathcal{T}$.*

If the TBox $\mathcal{T}$ is clear from the context, the qualification "w.r.t. $\mathcal{T}$" is sometimes dropped. It is also dropped the qualification in the special case where the TBox is empty, in this case it is simply written $\models C \sqsubseteq D$ if C is subsumed by D, and $\models C \equiv D$ if C and D are equivalent.

Among the presented inferences, subsumption plays a key role. Indeed it is sufficient in order to implement and prove the other inferences. This is possible by reducing these inferences to a subsumption problem.

Proposition 2.3.5 (Reduction to Subsumption) *Given two concepts C and D it holds that:*

- *C is unsatisfiable $\Leftrightarrow$ C is subsumed by $\perp$;*
- *C and D are equivalent $\Leftrightarrow$ C is subsumed by D and D is subsumed by C;*
- *C and D are disjoint $\Leftrightarrow$ $C \sqcap D$ is subsumed by $\perp$.*

The statements can be defined also with respect to a TBox.

If the considered description languages, besides of intersection, allow for the use of full negation, it is possible to reduce all inferences to the unsatisfiability problem [185].

Proposition 2.3.6 (Reduction to Unsatisfiability) *Given two concepts C and D it holds that:*

- *C is subsumed by $D \Leftrightarrow C \sqcap \neg D$ is unsatisfiable*
- *C and D are equivalent $\Leftrightarrow$ both $C \sqcap \neg D$ and $\neg C \sqcap D$ are unsatisfiable*
- *C and D are disjoint $\Leftrightarrow C \sqcap D$ is unsatisfiable.*

The statements can be defined also with respect to a TBox.

Hence, the ability to solve the subsuption test is fundamental for solving any other inference. For less expressive DLs the so called structural subsumption algorithms were devised. The basis of such algorithms for deciding subsumption is the syntactical comparison of normal form concepts (see Sect. 2.4 for more details about normalization of a concept). Namely, after turning concepts into normal form, the algorithm returns `true` if and only if the atomic concepts contained in the top level conjunction of the potential subsumer appear also in the subsumee and all concepts in the value restrictions of the subsumers subsume the ones in the corresponding restrictions in the subsumee.

Structural subsumption algorithms fail to be complete treating description languages that allow for disjunction, full negation and full existential concept restriction. For languages including these constructors, the tableau-approach has to be used. It is a decision procedure for implementing subsuption test whose corner stone is given by Prop. 2.3.6. Indeed, in such cases, the reduction of the subsumption to the satisfiability problem gives the way to implement the subsumption decision procedure.

Specifically, Tableau algorithm, that can be proved to be a complete and consistent procedure, prove that $C \sqsubseteq_{\mathcal{T}} D$ by testing whether the concept $D \sqcap \neg C$ is unsatisfiable in every interpretation of the TBox $\mathcal{T}$. So Tableau algorithm, differently from structural subsumption algorithms, decides whether a concept is satisfiable, rather it returns `true`, if and only if for any intepretation of a TBox there can exist a TBox model for the input concept, and gives it out as output. If such model does not exist, it returns false. Operationally, starting from the input concept of which satisfiability has to be verified, the algorithm tries to build a model of it, i.e.: an individual that is an instance of such concept. For more details about tableau and structural subsuption algorithm see [8].

Note that Prop. 2.3.5 and Prop 2.3.6 are given considering an empty TBox, anyway they continue to be valid also in presence of a TBox. This is because, if a TBox $\mathcal{T}$ is acyclic it is always possible to reduce reasoning problems w.r.t. $\mathcal{T}$

to problems w.r.t. an empty TBox. This is possible because $\mathcal{T}$ is equivalent to its *expansion* $\mathcal{T}'$, where $\mathcal{T}'$ is obtained from $\mathcal{T}$ by replacing iteratively each occurrence of a (non-primitive) concept name on the right-hand side of a definition in $\mathcal{T}$ with the concepts that it stands for. Since there is no cycle in the set of definitions, the process eventually stops and ends up with a terminology $\mathcal{T}'$ consisting solely of definitions of the form $A \equiv C'$ where C' contains only base symbols (primitive concept names) and no name symbols (defined concept names). $\mathcal{T}'$ is called the expansion of $\mathcal{T}$ and the size of the expansion can be exponential in the size of the original terminology [156].

Considering the way in which the expansion C' of C is obtained, it is straightforward to understand that both are interpreted in the same way in any model of $\mathcal{T}$, consequently it follows that $C \equiv_{\mathcal{T}} C'$. Hence, C is satisfiable w.r.t. $\mathcal{T}$ iff C' is satisfiable w.r.t. $\mathcal{T}$. However, C' contains no defined names, and thus C' is satisfiable w.r.t. $\mathcal{T}$ iff it is satisfiable. This yields that

- C is satisfiable w.r.t. $\mathcal{T}$ iff C' is satisfiable.

In the same way, if D is another concept, $D \equiv_{\mathcal{T}} D'$. Thus, $C \sqsubseteq_{\mathcal{T}} D$ iff $C' \sqsubseteq_{\mathcal{T}} D'$ and $C \equiv_{\mathcal{T}} D$ iff $C' \equiv_{\mathcal{T}} D'$. Again, since C' and D' contain only base symbols, this implies that:

- $\mathcal{T} \models C \sqsubseteq D$ iff $\models C' \sqsubseteq D'$
- $\mathcal{T} \models C \equiv D$ iff $\models C' \equiv D'$

With similar arguments it is possible to show that:

- C and D are disjoint w.r.t. $\mathcal{T}$ iff C' and D' are disjoint.

Hence it is possible to conclude that, expanding concepts with respect to an acyclic TBox allows one to get rid of the TBox in reasoning problems. The reason of the attention to such aspect is that for developing reasoning procedures it is conceptually easier to abstract from the TBox or, what amounts to the same, to assume that it is empty, even if this require to pay the computational cost of expanding the TBox.

2.3.2 ABox Reasoning

A knowledge representation system based on DLs is able to perform specific kinds of reasoning. Besides of TBox inference services, ABox inference services are also available. They are useful in order to make explicit assertional knowledge implicitly contained in the ABox. Specifically, standard reasoning tasks for ABoxes are:

- ABox consistency check
- Instance checking
- Retrieval

They are analyzed in detail in the following.

ABox consistency check solves the problem of checking if a new assertion (concept or role assertion) in an ABox $\mathcal{A}$ makes $\mathcal{A}$ inconsistent of not w.r.t. a TBox $\mathcal{T}$. This service is important because the representation of the knowledge in the ABox (after a TBox $\mathcal{T}$ has been built and TBox taxonomy and consistency have been checked) has to be consistent with $\mathcal{T}$, because otherwise arbitrarily conclusions can be drawn from it. This means, for instance, that considering a simple TBox $\mathcal{T} = \{\mathsf{Woman} \equiv \mathsf{Person} \sqcap \mathsf{Female}, \mathsf{Man} \equiv \mathsf{Person} \sqcap \neg\mathsf{Female}\}$, if the ABox contains the assertions $\mathsf{Woman(MARY)}$ and $\mathsf{Man(MARY)}$, the system should be able to find out that, together with $\mathcal{T}$, these statements are inconsistent. Formally, by the use of the presented model theoretic semantics it is said that:

Definition 2.3.7 (ABox Consistency (w.r.t. a TBox)) *An ABox $\mathcal{A}$ is consistent with respect to a TBox $\mathcal{T}$ if there exists an interpretation that is a model of both $\mathcal{A}$ and $\mathcal{T}$. It is simply said that $\mathcal{A}$ is consistent if it is consistent with respect to the empty TBox.*

So, using the example just above, the set of assertions $\{\mathsf{Woman(MARY)}, \mathsf{Man(MARY)}\}$ is consistent with respect to the empty TBox because, without any further restrictions on the interpretation of Woman and Man, the two concepts can be interpreted in such a way that they may have a common element. However, the assertions are not consistent with respect to the presented TBox $\mathcal{T}$, since in every model of it, Woman and Man are interpreted as disjoint sets.

Considering Def. 2.3.7, as for TBoxes, also for ABoxes it is simpler to reason on them having an empty TBox rather than a non empty one. In order to obtain this simplification, as for TBoxes, it is possible to expand an ABox. Particularly, it can be defined the *expansion* of an ABox $\mathcal{A}$ with respect to a Tbox $\mathcal{T}$ as the ABox $\mathcal{A}'$ that is obtained from $\mathcal{A}$ by replacing each concept assertion $C(a)$ in $\mathcal{A}$ with the assertion $C'(a)$, where C' is the expansion of C with respect to $\mathcal{T}$[4] (see Sect. 2.3.1). In every model of $\mathcal{T}$, a concept C and its expansion C' are interpreted in the same way (see Sect. 2.3.1 for more details). Therefore, it is possible to assert that $\mathcal{A}'$ is

[4]Note that only concept assertions are expanded because the focus is on description languages that do not provide constructors for role descriptions, hence TBoxes with role definitions (RBoxes) are not considered.

consistent w.r.t. $\mathcal{T}$ iff $\mathcal{A}$ is consistent w.r.t. $\mathcal{T}$. However, since $\mathcal{A}'$ does not contain a name symbol defined in $\mathcal{T}$, it is consistent w.r.t. $\mathcal{T}$ iff it is consistent. Namely:

- $\mathcal{A}$ is consistent w.r.t. $\mathcal{T}$ iff its expansion $\mathcal{A}'$ is consistent

As for the subsumption, ABox consistency is computed by the means of structure-based algorithms for less expressive DLs, while for DLs allowing disjunction, full negation, and full existential concept restriction the Tableau algorithm is used. For more details see [8].

For the sake of simplicity, the definitions of inferences given in the following will be formalized with respect to ABoxes alone, considering that every inference w.r.t. a TBox can be reduced to inference about expansion, provided that the TBox is acyclic.

The **instance check** is the prototypical ABox inference consisting in checking whether an assertion is entailed by an ABox. Indeed this is the inference used in order to allow queries over an ABox concerning concepts, roles and individuals. Formally:

Definition 2.3.8 (Assertion entailment) *An assertion α is entailed by an ABox $\mathcal{A}$ and it is written $\mathcal{A} \models \alpha$, if every interpretation that satisfies $\mathcal{A}$, (i.e. every model of $\mathcal{A}$), also satisfies α.*

If α is a role assertion, the instance check is easy, since the considered description language does not contain constructors to form complex roles. If α is of the form $C(a)$, it is possible to reduce the instance check to the consistency problem for ABoxes thanks to the following connection:

- $\mathcal{A} \models C(a)$ iff $\mathcal{A} \cup \{\neg C(a)\}$ is inconsistent.

Moreover, also reasoning about concepts can be reduced to consistency checking. Indeed as important reasoning problems for concepts can be reduced to deciding whether a concept is (un)satisfiable (see Prop. 2.3.6), in the same way concept satisfiability can be reduced to ABox consistency. This is because of the following connection:

- For every concept C: C is satisfiable iff $C(a)$ is consistent where a is an arbitrary individual name[5].

As a knowledge base is also a way for storing information about individuals, it is likely to require to know, for instance, all individuals that are instances of a given concept description C. This is equivalent to use the description language to formulate queries on the knowledge base. The possibility of making queries to a knowledge base for getting the set of individuals that are instances of a certain concept is called **retrieval problem**. The retrieval problem can be formally defined as:

Definition 2.3.9 (Retrieval Problem) *Given an ABox $\mathcal{A}$ and a concept C, to find all individuals a such that $\mathcal{A} \models C(a)$*

A straightforward algorithm for a retrieval query can be realized by testing for each individual occurring in the ABox whether it is an instance of the concept C.

Note that for any of the presented inference services the respective complexity has been not given. This is because it depends on the particular description language used (i.e. $\mathcal{ALE}$, $\mathcal{ALE}$...). In the next section non-standard inference services for some particular DLs will be presented, hence the corresponding computational complexity will be specified.

2.4 Non-Standard Inference Services

The name *non-standard inference services* refers to a set of reasoning services useful for the development of knowledge bases but they are typically not provided by DL Knowledge Reasoners Systems (KRS). The implementation of such services relies on additional inference techniques that are considered non-standard, because they go beyond the basic reasoning services provided by DL-KRS (presented in Sect. 2.3). Such new inference services can formally be defined as new types of inference problems. Non-standard reasoning services can serve to a variety of purposes. For instance they can be used in order to support tasks such as building and maintaining knowledge bases, or can be used to get information from the represented knowledge.

Among the most useful non-standard inference tasks in DLs there are: the computation of the *least common subsumer*, the *most specific concept*, *matching*

[5]Moreover has been showed in [175] also that ABox consistency can be reduced to concept satisfiability in languages with the "set" and the "fills" constructor, namely languages with *one-of* and *has-value* constructors (see Tab. 2.2). Furthermore, in [68] has been showed that if these constructors are not available in the considered description language, instance check may be harder than the satisfiability and the subsumption problem.

unification of concept descriptions and *concept rewriting*. They will be analyzed in detail in the following, focussing, first of all, on the $\mathcal{ALC}$ logic and secondarily on the $\mathcal{ALE}$ and the $\mathcal{ALN}$ logics, in a way that is functional for this work.

Every non-standard inference has been built to solve a particular problem, in order to improve the development and the maintainability of the knowledge base. Particularly, the standard inferences can be applied, after a new concept has been defined, to find out whether the concept is non-contradictory or whether its place in the taxonomy coincides with the intuition of the knowledge engineer. However, these inferences do not directly support the process of actually defining the new concept. To overcome this problem, the non-standard inference services for computing the *least common subsumer* and for computing the *most specific concept* have been proposed.

If a knowledge base is maintained by different knowledge engineers, it is necessary to have a tool for detecting multiple definitions of the same intuitive concept. Since different knowledge engineers might use different names for the "same" primitive concept, the standard equivalence test may not be adequate to check whether different descriptions refer to the same notion. The non-standard inference service performing *unification of concept descriptions* tackles this problem by allowing to replace concept names by appropriate concept descriptions before testing for equivalence. *Matching* is a special case of unification. It has been used, for instance, for pruning irrelevant parts of large concept descriptions before displaying them to the user. Moreover, the non-standard inference performing *rewriting of concept descriptions* allows one to transform a given concept description C into a "better" description D, which satisfies certain optimality criteria (e.g., small size) and is related (e.g., by equivalence or subsumption) to the original description C. An overview of the state of the art of this field and detailed proofs of several of the results mentioned below can be found in [125].

Approaches for solving the new inference problems are usually based on an appropriate characterization of subsumption, which can be used to obtain a structural subsumption algorithm[6]. At first, the concept descriptions are turned into a certain *normal form*, in which implicit facts have been made explicit. This is because semantically equivalent (yet syntactically different) descriptions can be given for the same concept. Anyway they can be reduced to a canonical form by means of equivalence-preserving rewriting rules [151, 8]. Then, the structure of the normal forms is compared appropriately. This is one of the reasons why most of the results on non-standard inferences are restricted to languages that can be treated by structural subsumption algorithms. In the following the definitions of $\mathcal{ALE}$, $\mathcal{ALC}$ $\mathcal{ALN}$

[6]Note that most of the results on non-standard inferences are restricted to languages that can be treated by structural subsumption algorithms.

normal form concept descriptions (see [16, 151, 8]) are reported (see Tab. 2.2 for their available constructors). First of all, some notations are necessary to access the different parts of a concept description C:

- $\mathsf{prim}(C)$ denotes the set of all (negated) concept names occurring on the top-level conjunction of C;
- if there exists a value restriction of the form $\forall R.C'$ on the top level conjunction of C, then $\mathsf{val}_R(C) := C'$, otherwise $\mathsf{val}_R(C) := \top$;
- $\mathsf{ex}_R(C) := \{C' \mid \text{there exists } \exists R.C' \text{ on the top-level conjunction of } C\}$.

Definition 2.4.1 ($\mathcal{ALE}$ Normal Form Concept Descriptions) *[36] An $\mathcal{ALE}$ concept description C is in $\mathcal{ALE}$ normal form iff*

1. *if $C \equiv \bot$ then $C = \bot$; if $C \equiv \top$ then $C = \top$*
2. *otherwise, C is of the form*

$$C = \prod_{A \in \mathsf{prim}(C)} A \sqcap \prod_{R \in N_R} \left[\forall R.\mathsf{val}_R(C) \sqcap \prod_{E \in \mathsf{ex}_R(C)} \exists R.E \right]$$

An $\mathcal{ALE}$ concept description C is normalized by applying the following equivalence preserving rewriting rules modulo associativity and commutativity of conjunction:

$$\begin{array}{rr} \forall R.E \sqcap \forall R.F \rightarrow \forall R.(E \sqcap F) & \forall R.E \sqcap \exists R.F \rightarrow \forall R.E \sqcap \exists R.(E \sqcap F) \\ \forall R.\top \rightarrow \top & E \sqcap \top \rightarrow E \\ \exists R.\bot \rightarrow \bot & E \sqcap \top \rightarrow \bot \\ A \sqcap \neg A \rightarrow \bot \text{ for each } A \in N_C & \end{array}$$

Note that, due to the second rule in the first line, this normalization may lead to an exponential blow-up of the concept descriptions. Every $\mathcal{ALE}$ concept description can be turned into an equivalent concept description in normal form.

Definition 2.4.2 ($\mathcal{ALC}$ Normal Form Concept Descriptions) *[36] An $\mathcal{ALC}$ concept description C is in $\mathcal{ALC}$ normal form iff*

1. *if $C \equiv \bot$ then $C = \bot$; if $C \equiv \top$ then $C = \top$*
2. *otherwise, C is of the form $C = C_1 \sqcup \cdots \sqcup C_n$ with*

$$C_i = \prod_{A \in \mathsf{prim}(C_i)} A \sqcap \prod_{R \in N_R} \left[\forall R.\mathsf{val}_R(C_i) \sqcap \prod_{E \in \mathsf{ex}_R(C_i)} \exists R.E \right]$$

$C_i \not\equiv \bot$ and every concept description in $\mathsf{ex}_R(C_i)$ and $\mathsf{val}_R(C_i)$ is in $\mathcal{ALC}$ normal form, for all $i = 1, \cdots, n$.

Every $\mathcal{ALC}$-concept descriptions can be turned into an equivalent concept description in $\mathcal{ALC}$-normal form. Unfortunately, this may take exponential time. As for the $\mathcal{ALE}$ concept normalization, also for $\mathcal{ALC}$, concept normalization is obtained by the use of rewriting rules (see [8] for more details).

Definition 2.4.3 ($\mathcal{ALN}$ Normal Form Concept Descriptions) *A concept description C is in $\mathcal{ALN}$ normal form iff*

1. *if $C \equiv \bot$ then $C = \bot$; if $C \equiv \top$ then $C = \top$*
2. *otherwise C is of the form*

$$C = \bigsqcap_{P \in \mathsf{prim}(C)} P \sqcap \bigsqcap_{R \in N_R} (\forall R.C_R \sqcap \geq n.R \sqcap \leq m.R)$$

where $C_R = \mathsf{val}_R(C)$, $n = \mathsf{min}_R(C)$ and $m = \mathsf{max}_R(C)$ and

- $\mathsf{min}_R(C) = \max\{n \in \mathbb{N} \mid C \sqsubseteq (\geq n.R)\}$ *(always a finite number);*
- $\mathsf{max}_R(C) = \min\{n \in \mathbb{N} \mid C \sqsubseteq (\leq n.R)\}$ *(if unlimited then $\mathsf{max}_R(C) = \infty$).*

The complexity of the normalization process is polynomial [8]. About rewriting rules for obtaining an $\mathcal{ALN}$ concept normal form see [151, 8].

In the following all cited non-standard inferences will be analyzed. Particular attention will be posed to determining the *least common subsumer* of concepts and the *most specific concept* of a considered individual, besides, in the context of *concepts rewriting*, a particular attention will be given to the approximation of $\mathcal{ALC}$ concept descriptions to $\mathcal{ALE}$ concept descriptions.

2.4.1 The Least Common Subsumer

Intuitively, the least common subsumer of a given collection of concept descriptions is a concept description that represents the properties that all the elements of the collection have in common. Particularly, it is the most specific concept description (w.r.t. the subsumption relationship) that subsumes the given concept descriptions:

Definition 2.4.4 (Least Common Subsumer) *Let $\mathcal{L}$ be a description language. A concept description E of $\mathcal{L}$ is the* ***least common subsumer (lcs)*** *of the concept descriptions $C_1, \cdots, C_n$ in $\mathcal{L}$ (lcs$(C_1, \cdots, C_n)$ for short) iff it satisfies:*

1. *$C_i \sqsubseteq E$ for all $i = 1, \cdots, n$ and*

2. *E is the least $\mathcal{L}$-concept description satisfying (1), i.e. if E' is an $\mathcal{L}$-concept description satisfying $C_i \sqsubseteq E'$ for all $i = 1, \cdots, n$, then $E \sqsubseteq E'$.*

From the definition 2.4.4 it is straightforward to see that the *lcs* is unique up to equivalence.

It should be noted, however, that, depending on the DL language, the lcs need not always exist. This can happen for two different reasons:

1. there may be several subsumption incomparable minimal concept descriptions satisfying (1) of Def. 2.4.4;

2. there may be an infinite chain of more and more specific descriptions satisfying condition (1).

It is easy to see that, case (1) of not existence of lcs does not occur for DLs allowing for conjunction of descriptions.

The lcs inference has first been introduced by Cohen et al. in [42] as a new inference task useful for different reasons. First, finding the most specific concept that generalizes a set of examples is a common operation in inductive learning, called learning from examples. Moreover, experimental results concerning the learnability of concepts based on computing the lcs have been presented in [43]. Another reason for considering the lcs is to use it as an alternative to concept disjunction. The idea is to replace disjunctions like $C_1 \sqcup \cdots \sqcup C_n$ by $lcs(C_1, \cdots, C_n)$. In [42, 30] this operation is called *knowledge base vivification*. Although, in general, the lcs is not equivalent to the corresponding disjunction, it is the best approximation of the disjunctive concept within the available DL. Using such an approximation is motivated by the fact that, in many cases, adding disjunction would increase the complexity of reasoning. Observe that, if the DL already allows for disjunction, it has $lcs(C1, \ldots, C_n) \equiv C_1 \sqcup \cdots \sqcup C_n$. This means that, for such DLs, the lcs could not be really of interest. Finally, as proposed in [9, 12], the lcs operation can be used to support the "bottom-up" construction of DL knowledge bases. In contrast to the usual "top-down" approach, where the knowledge engineers first define the terminology of the application domain in the TBox and then uses this terminology for describing individuals in the ABox, the "bottom-up" approach proceeds as follows. The knowledge engineer first specifies some "typical" examples of a concept to be defined, using individuals in the ABox. Then, in a second step, these individuals are generalized to their most specific concept (see Sect. 2.4.2). Finally, the knowledge engineers inspects and possibly modifies the concept description obtained in this way.

The methods for computing the least common subsumer are restricted to rather inexpressive description logics, not allowing for disjunction (and thus not allowing for full negation). This is because, as seen above, endowed languages allowing disjunction, the lcs of a collection of concepts is simply their disjunction, and so nothing new can be learned from it, as it represents a very poor generalization of the considered concepts. In contrast, for languages without disjunction, the lcs extracts the "commonalities" of the given collection of concepts. Particularly, the computation of the lcs for these less expressive languages, characterizes , as first step, the subsumption of the concept descriptions (which is typically a structural subsumption), hence the lcs is computed as a new concept containing the structural commonalities of the considered concept collection. In the following, the computation of the lcs for the $\mathcal{ALE}$ description logic will be shown (for more details see [12]). Specifically, it will be shown that the lcs for $\mathcal{ALE}$ concept descriptions always exists and can effectively be computed. The attention will be mostly focused on computing the lcs of two concept description, since the lcs of $n > 2$ concept descriptions can be obtained by the iterated application of the binary lcs operation.

Computing Least Common Subsumer in $\mathcal{ALE}$ logic

Computing the least common subsumer in $\mathcal{ALE}$ requires, as seen, first of all a characterization of $\mathcal{ALE}$ subsumption concepts, hence the lcs is given exploiting the characterization. Specifically, this characterization is given in terms of product of description tree, hence, the notion of $\mathcal{ALE}$-description tree is clarified in the following.

An *$\mathcal{ALE}$-description tree* is a way of representing a normalized $\mathcal{ALE}$-concept by means of a tree structure. An $\mathcal{ALE}$-description tree has the form $\mathcal{G} = (V, E, v_0, l)$, where $\mathcal{G}$ is a tree with root v_0 whose edges $vrw \in E$ are labeled with primitive roles $r \in N_R$ and whose nodes $v \in V$ are labeled with sets $l(v)$ of primitive concepts from N_C. The empty label correspond to the top-concept. Given an $\mathcal{ALE}$ concept C in normal form, it can be translated in an $\mathcal{ALE}$-description tree as follows. The set of all (negated) primitive concepts occurring in the top-level conjunction of C yields the label $l(v_0)$ of the root v_0, and each existential and/or universal restriction i.e. $\exists r_i.C_i$ and/or $\forall r_i.C_i$ in this conjunction yields an r_i-successor (resp. a $\forall r_i$-successor) that is the root of the tree corresponding to C_i. For example, for the $\mathcal{ALE}$ concepts in normal form:

$C := \forall r.\bot \sqcap \exists s.(P \sqcap \exists r.Q)$
$D := (\forall r.(\exists r.P \sqcap \exists r.\neg P)) \sqcap (\exists s.\exists r.Q)$

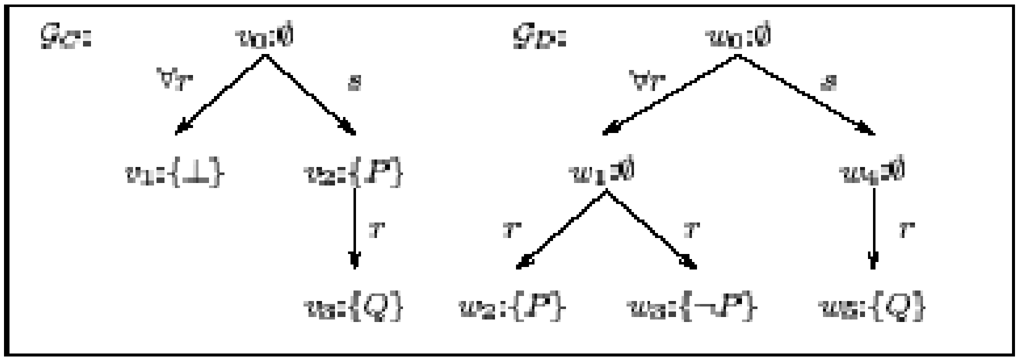

Figure 2.1: $\mathcal{ALE}$-description trees

the corresponding $\mathcal{ALE}$-description trees $\mathcal{G}_C$ and $\mathcal{G}_D$ is depicted in Fig. 2.1.

Note that every $\mathcal{ALE}$ description tree $\mathcal{G} = (V, E, v_0, l)$ can be translated into an $\mathcal{ALE}$ concept description $C_\mathcal{G}$ (in normal form), so there is a $1-1$ correspondence between $\mathcal{ALE}$ concept descriptions and $\mathcal{ALE}$-description trees.

Each $\mathcal{ALE}$-description tree $\mathcal{G} = (V, E, v_0, l)$, obtained obtained as just shown, satisfies the following properties [12]:

- For each node $v \in V$ and each primitive role $r \in N_R$, v has at most one outgoing edge labeled $\forall r$.
- Let $\{vrw, v\forall rw'\} \subseteq E$, and let C denote the $\mathcal{ALE}$-concept description corresponding to the subtree of $\mathcal{G}$ with root w, and C' the one corresponding to the subtree of $\mathcal{G}$ with root w'. Then $C \sqsubseteq C'$.
- Leaves in $\mathcal{G}$ labeled with the empty set cannot be reached via an edge labeled $\forall r$ for some $r \in N_R$, i.e., $C_\mathcal{G}$ does not contain a subconcept of the form $\forall r.\top$
- If the label of a node contains $\bot$, then its label is $\{\bot\}$ and it is a leaf that cannot be reached by an edge with label $r \in N_R$

The meaning of these rules is immediate, thinking of the equivalence preserving rewriting rules for obtaining an $\mathcal{ALE}$-concept in normal form, reported after Def. 2.4.1.

In order to characterize the subsuption of $\mathcal{ALE}$-concepts, the notion of *homomorphism* between $\mathcal{ALE}$-description trees is necessary.

Definition 2.4.5 *A* homomorphism *from an $\mathcal{ALE}$-description tree $\mathcal{H} = (V_H, E_H, w_0, l_H)$ into an $\mathcal{ALE}$-description tree $\mathcal{G} = (V_G, E_G, v_0, l_G)$ is a mapping $\varphi : V_H \to V_G$ such that:*

1. $\varphi(w_0) = v_0$

2. $l_H(v) \subseteq l_G(\varphi(v))$ or $l_G(\varphi(v) = \{\bot\})$ for all $v \in V_H$

3. for all $vrw \in E_H$,
either $\varphi(v)r\varphi(w) \in E_G$, or $\varphi(v) = \varphi(w)$ and $l_G(\varphi(v)) = \{\bot\}$

4. for all $v\forall rw \in E_H$,
either $\varphi(v)\forall r\varphi(w) \in E_G$, or $\varphi(v) = \varphi(w)$ and $l_G(\varphi(v)) = \{\bot\}$.

Considering Fig. 2.1, and mapping w_0 onto v_0; w_1, w_2, and w_3 onto v_1; w_4 onto v_2; and w_5 onto v_3, the conditions of Def. 2.4.5 are satisfied, i.e. this mapping yields a homomorphism from $\mathcal{G}_D$ into $\mathcal{G}_C$.

By the use of the notion of homomorphism between $\mathcal{ALE}$-description trees, the subsumption[7] in $\mathcal{ALE}$ can be characterized in a sound and complete way [12].

Theorem 2.4.6 *Let C, D be two $\mathcal{ALE}$-concept descriptions and $\mathcal{G}_C, \mathcal{G}_D$ the corresponding $\mathcal{ALE}$-descriptions trees. Then $C \sqsubseteq D$ iff there exists a homomorphism from $\mathcal{G}_D$ into $\mathcal{G}_C$*

Note that the presented subsumption is structural. The characterization of the subsumption by homomorphisms allows to determine the lcs by means of the product of $\mathcal{ALE}$-description trees [12].

Definition 2.4.7 (Product of $\mathcal{ALE}$-description trees) *The* product $\mathcal{G} \times \mathcal{H}$ *of two $\mathcal{ALE}$-description trees $\mathcal{G} = (V_G, E_G, v_0, l_G)$ and $\mathcal{H} = (V_H, E_H, w_0, l_H)$ is defined as follows.*

- *If $l_G(v_0) = \{\bot\}$ ($l_H(w_0) = \{\bot\}$), then $\mathcal{G} \times \mathcal{H}$ is defined by replacing each node w in $\mathcal{H}$ (v in $\mathcal{G}$) by (v_0, w) ((v, w_0))*

- *Otherwise $\mathcal{G} \times \mathcal{H}$ is defined by induction on the depth of the trees. Let $\mathcal{G}_v$ denote the subtree of $\mathcal{G}$ with root v. (v_0, w_0) is defined to be the root of $\mathcal{G} \times \mathcal{H}$, labeled with $l_G(v_0) \cap l_H(w_0)$. For each r-successor v of v_0 in $\mathcal{G}$ and w of w_0 in $\mathcal{H}$, an r-successor (v, w) of (v_0, w_0) is obtained in $\mathcal{G} \times \mathcal{H}$, that is the root of the product of $\mathcal{G}(v)$ and $\mathcal{H}(w)$. In the same way a $\forall r$-successor (v, w) of (v_0, w_0) is obtained in $\mathcal{G} \times \mathcal{H}$ if v is the $\forall r$-successor of v_0 in $\mathcal{G}$ and w the one of w_0 in $\mathcal{H}$, and (v, w) is the root of the product of $\mathcal{G}(v)$ and $\mathcal{H}(w)$.*

[7]Subsumption in $\mathcal{ALE}$ is an NP-complete problem [65].

Considering the example in Fig. 2.1, the product $\mathcal{G}_C \times \mathcal{G}_D$ can be obtained from $\mathcal{G}_D$ by replacing w_0 with (v_0, w_0), w_i with (v_1, w_i) for $i = 1, 2, 3$, w_4 with (v_2, w_4), and w_5 with (v_3, w_5).

The lcs in $\mathcal{ALE}$ is characterized as follows [12].

Theorem 2.4.8 *Let C, D be two $\mathcal{ALE}$-concept descriptions and $\mathcal{G}_C, \mathcal{G}_D$ their corresponding $\mathcal{ALE}$-description trees. Then $C_{\mathcal{G}_C \times \mathcal{G}_D}$ is the lcs of C and D.*

Note that the size of the lcs of two $\mathcal{ALE}$-concept descriptions C, D may be exponential in the size of C, D.

The lcs algorithm is formalized below. Let C a normalized $\mathcal{ALE}$-concept description. $\mathsf{names}(C)$ (resp. $\overline{\mathsf{names}}(C)$) denotes the set of (negated) concept names occurring in the top-level conjunction of C, $\mathsf{roles}^{\exists}(C)$ (resp. $\mathsf{roles}^{\forall}(C)$) denotes the set of role names occurring in an existential (value) restriction on the top-level of C, and $\mathsf{restrict}^{\exists}_r(C)$ (resp. $\mathsf{restrict}^{\forall}_r(C)$) denotes the set of all concept descriptions occurring in an existential (value) restriction on the role r in the top-level conjunction of C. Given C, D normalized $\mathcal{ALE}$-concept descriptions, if C (D) is equivalent to $\bot$, then $\mathsf{lcs}_{\mathcal{ALE}}(C, D) = D$ ($\mathsf{lcs}_{\mathcal{ALE}}(C, D) = C$). Otherwise, it has:

$$\begin{aligned}
\mathsf{lcs}_{\mathcal{ALE}}(C, D) \;=\; & \prod_{A \in \mathsf{names}(C) \cap \mathsf{names}(D)} A \sqcap \prod_{\neg B \in \overline{\mathsf{names}}(C) \cap \overline{\mathsf{names}}(D)} \neg B \;\sqcap \\
\sqcap\; & \prod_{r \in \mathsf{roles}^{\exists}(C) \cap \mathsf{roles}^{\exists}(D)} \; \prod_{E \in \mathsf{restrict}^{\exists}_r(C), F \in \mathsf{restrict}^{\exists}_r(D)} \exists r.\mathsf{lcs}_{\mathcal{ALE}}(E, F) \;\sqcap \\
\sqcap\; & \prod_{r \in \mathsf{roles}^{\forall}(C) \cap \mathsf{roles}^{\forall}(D)} \; \prod_{E \in \mathsf{restrict}^{\forall}_r(C), F \in \mathsf{restrict}^{\forall}_r(D)} \forall r.\mathsf{lcs}_{\mathcal{ALE}}(E, F)
\end{aligned}$$

Note the presented way for computing the lcs in $\mathcal{ALE}$ uses a syntactic-based approach and does not consider the TBox to which the concept descriptions refer. In [16], the notion of *good common subsumer* (gcs) is introduced. It is a good approximation of the lcs in $\mathcal{ALE}$ that considers the TBox to which the concepts for computing the lcs refer. Moreover, in [16] it is shown that, in general, the gcs is more specific than the lcs. Furthermore it is proved that the gcs always exist for acyclic definitorial TBoxes. Hence in case of generalized TBoxes (c.f.r. Sect. 2.1), they have to be translated into acyclic definitorial TBoxes for computing the gcs.

2.4.2 The Realization Problem and the Most Specific Concept

The *realization problem* is the dual inference with respect to retrieval (Sect. 2.3.2). Given an individual a and a set of concepts, the realization problems consists in finding the *most specific concepts* (msc for brevity) C from the given concept set, such that $\mathcal{A} \models C(a)$ where $\mathcal{A}$ is the reference ABox. Specifically, the most specific concepts are those that are minimal with respect to the subsumption ordering $\sqsubseteq$. Realization can be used in systems that generate natural language. Indeed if terms are indexed by concepts, a term as precise as possible can be found for an object occurring in a discourse. Moreover, based on the computation of the msc of a set of assertions about individuals, it could be possible to incrementally construct a knowledge base (see [10]). The msc of an individual is formally defined as follow.

Definition 2.4.9 (Most Specific Concept) *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ a knowledge base, a an individual and C a concept. C is the most specific concept of a ($\mathsf{msc}(a) = C$ for short) iff*

1. $\mathcal{A} \models_{\mathcal{T}} C(a)$
2. *For all concepts D, $\mathcal{A} \models_{\mathcal{T}} D(a) \Rightarrow C \sqsubseteq_{\mathcal{T}} D$*

Generalizing, it is possible to define the msc of a set of individuals.

Definition 2.4.10 *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ a knowledge base. A concept description E is the most specific concept (msc) of the individuals $a_1, \cdots, a_n$ defined in an ABox $\mathcal{A}$ ($msc(a_1, \cdots, a_n)$ for short) iff*

1. *$\mathcal{A} \models E(a_i)$ for all $i = 1, \ldots, n$ and*
2. *E is the least concept satisfying (1) i.e., if E' is a concept description satisfying $\mathcal{A} \models E'(a_i)$ for all $i = 1, \cdots, n$ then $E \sqsubseteq E'$*

The process of computing the msc can be split into two subtasks: computing the most specific concept of a single individual, and computing the least common subsumer of a given finite number of concepts. In fact, it is easy to see that $msc(a_1, \cdots, a_n) \equiv lcs(msc(a_1), \cdots, msc(a_n))$.

Similarly to the computation of the lcs, also for the msc computation, the techniques proposed rely on compact representations of concepts using DLs with low expressiveness. This computation are built either following the structural subsumption approach, or though the definition of a well-suited normal form.

It is important to note that the msc needs not to exist in every DL allowing for existential and qualified number restrictions [125]. The most common solution to solve this problem in the literature is to approximate the msc (see [43, 123]). The reason of this choice is that often the existence of the msc is guaranteed by allowing cyclic definitions that, on the other hand, represent a problem for most of the other reasoning tasks. So, the most common strategy is to impose a depth bound in the recursive exploration of relationships among individuals. In [125] Küsters shows that in $\mathcal{EL}$ there is an algorithm that yields to the msc of an individual in polynomial time, provided that it exists. Anyway, it is also proved that the same result is not guaranteed considering more expressive DLs. In the following a set of problems and (partial) solutions for computing msc for more expressive DLs are reported.

Given an ABox $\mathcal{A}$, in order to simplify the notation it is denoted:

- $\mathsf{Ind}\mathcal{A}$ the set of all the individuals in $\mathcal{A}$
- $\mathsf{CA}(a) = \{C \mid C(a) \in \mathcal{A}\}$
- $\mathsf{IR}(a) = \{R \mid R(a,b) \in \mathcal{A}\}$
- $\mathsf{RF}(R,a) = \{b \in \mathsf{Ind}(\mathcal{A}) \mid R(a,b) \in \mathcal{A}\}$
- $\mathsf{RF}^*(R,a) = \{b \in \mathsf{Ind}(\mathcal{A}), R^*(a,b) \in \mathcal{A}$ where R^* is the closure[8] of $R\}$

The idea underlining the msc computation consists in characterize the instance of a concept. For DLs not allowing disjunction such characterization is similar to those of the structural subsumption, by the use of description trees (see Sect. 2.4.1). In this case the notion of *individual graph* is used.

Definition 2.4.11 (Individual Graph) *Let $\mathcal{A}$ be an ABox and $a \in \mathsf{Ind}(\mathcal{A})$. The individual graph of a ($G(a)$ for short) can be defined as the tuple (V, E, l) as follows:*

- *V is the set of vertices of G: $V = \{a\} \cup \mathsf{RF}^*(R,a)$ for any $R \in \mathsf{IR}(a)$*
- *E is the set of edges of G: $E = \{v_1 R v_2 \mid v_1, v_2 \in \mathsf{Ind}(\mathcal{A}), R$ role, such that $R(v_1,v_2) \in \mathcal{A} \ \wedge v_1, v_2 \in \mathsf{RF}^*(R,a)\}$*
- *l is a labeling function that maps each vertex a (individual) with the set $\mathsf{CA}(a)$*

In [125] it is shown that, considering DLs for which structural subsumption is complete, the instance checking (see Sect. 2.3.2) for an individual a w.r.t. a concept C can be computed as a homomorphism between the description tree built for the

[8]see [8] for more details about role closure

concept C (also called concept graph) and the individual graph $G(a)$. Conversely, it is possible to obtain a concept graph starting from an individual graph. The only inconvenience is given by the presence of cycles. A possible solution is to translate every cycle by inserting a new node instead of drawing an edge back on the loop node, and deciding a *depth bound* in exploring cyclic paths. The concept obtained in this way is called C_a^k. It represents the k-bound approximation of $msc(a)$. Using this translation, Baader et al. [11] show the following result:

Theorem 2.4.12 (Most specific Concept in $\mathcal{EL}$ logic) *Let $\mathcal{A}$ be an $\mathcal{EL}$ ABox, $a \in \mathsf{Ind}(\mathcal{A})$ and $k \geq 0$. Then C_a^k is the k-bound approximation of msc(a). If there are no cycles in $G(a)$ then there exists k such that $C_a^k \equiv msc(a)$ otherwise there exists no msc(a).*

Considering DLs that allow for (even atomic) negation, the main problems are given by the existence of the Open World Assumption (OWA). Indeed, if an individual, say a has been asserted to belong to each concept of a given set of atomic concepts (say $CS \subset N_P$), there is no clue to say that it does not belong to some concept $P \in N_P \setminus CS$. In order to better explain this problem let be consider the following example. Let $C \equiv P \sqcap \exists r.(P \sqcap \exists r.\neg P)$ a concept and let $\mathcal{A} = \{P(a), P(b_1), \neg P(b_3), R(a, b_1), R(a, b_2), R(b_1, b_2), R(b_2, b_3)\}$ the reference ABox. Building the concept graph $\mathcal{G}_C$ for C and the individual graph $G(a)$ for a, it is easy to verify that does not exist any homomorphism between $\mathcal{G}_C$ and $G(a)$ though $\mathcal{A} \models C(a)$. This is due to the fact that there is no information about the membership (or non membership) of individuals, for instance like b_2, to primitive P. In such cases, the suggestion is to perform the so called *atomic completion* of the individual, that consists in adding, for every primitive concept P and for all nodes in the individual graph either P or $\neg P$ to their labels. If after every atomic completion, the concept graph of C is homomorphic to the completed individual graph then $\mathcal{A} \models C(a)$ (and the converse holds as well).

In [105] the following definition and sketch of the algorithm for computing an approximation of the msc (msc^* for short) is presented.

Definition 2.4.13 (approximated msc) *Given any ABox $\mathcal{A}$ and an individual $a \in \mathsf{Ind}(\mathcal{A})$, the approximation of $msc(a)$ is given by*

$$msc^*(a, \mathsf{SEEN}) = \mathsf{Real}(a) \sqcap \mathsf{RoleCncpt}(a, \mathsf{SEEN}) \sqcap \mathsf{CycleCncpt}(a, \mathsf{SEEN})$$

where

- $$\mathsf{\mathit{Real}}(a) = \bigsqcap_{C \in \mathsf{\mathit{CA}}(a)} C$$

- $$\mathsf{\mathit{RoleCncpt}}(a, \mathsf{\mathit{SEEN}}) = \bigsqcap_{\substack{R \in \mathsf{\mathit{IR}}(a) \\ b \in \mathsf{\mathit{RF}}(R,a) \setminus \mathsf{\mathit{SEEN}}}} \exists R.\mathsf{\mathit{mscInt}}(a, \mathsf{\mathit{SEEN}})$$

- $$\mathsf{\mathit{CycleCncpt}}(a, \mathsf{\mathit{SEEN}}) = \bigsqcap_{\substack{R \in \mathsf{\mathit{IR}}(a) \\ b \in \mathsf{\mathit{RF}}(R,a) \cap \mathsf{\mathit{SEEN}}}} \exists R.\top$$

and

- $$\mathsf{\mathit{mscInt}}(a, \mathsf{\mathit{SEEN}}) = \bigsqcap_{\substack{b \in \mathsf{\mathit{RF}}(R,a) \\ \mathcal{A} \models \exists R.\sqcap msc^*(b, \mathsf{\mathit{SEEN}})(a)}} msc^*(b, \mathsf{\mathit{SEEN}} \cup \{a, b\})$$

where the parameter named SEEN *is a set of individuals.* $msc^*(a, \emptyset)$ *is denoted simply by* $msc^*(a)$*.*

The msc^* operator takes as input an individual and a set of individuals and returns a concept consisting of a group of three main conjuncts. $\mathsf{Real}(a)$, which is the left-most conjunct in the definition represents the intersection of all the concepts in $\mathsf{CA}(a)$ that is the set of all the concepts to which the starting individual a is asserted to belong to. The middle conjunct, $\mathsf{RoleCncpt}(a, \mathsf{SEEN})$, is the intersection, for any role R involving a i.e. $R \in \mathsf{IR}(a)$, of all the qualified existential restrictions. Such role R can be obtained considering any R-filler b of R (i.e. $b \in \mathsf{RF}(R, a)$) provided that such b is not in SEEN, that means that b has not been already visited. The concepts in the scope of such existential restrictions are again msc, but computed, this time, on b. In the same time the individuals $\{a, b\}$ are added to the SEEN set. For each b the restrictions $\exists R.msc^*(b)$ are refined conjoining as many msc's as possible (as long as a belongs to this kind of restriction). The third conjunct, $\mathsf{CycleCncpt}(a, \mathsf{SEEN})$, is the intersection of as many $\exists R.\top$ as the number of R-fillers of a, for any role $R \in \mathsf{IR}(a)$, that appear also in the SEEN set.

The presented definition allows to compute an approximation of the msc both in case of acyclic ABoxes and in case of cyclic ones. This guarantee to have an approximation of the msc if for the case in which an exact msc needs not exists (namely for DLs allowing for existential and qualified number restrictions). Furthermore, the obtained concept from the msc^* computation can be easily translated into a concept graph that can be mapped by means of a homomorphism to the individual graph G(a), hence the individual will be an instance of the msc approximation.

The msc^* can be computed considering a fixed bound, as suggested by Cohen and Hirsh in [43]. They fix an integer k as the maximum path length in exploring relationships starting from the desired individual and the output is a concept containing at most k universal value restrictions on the roles the individual (and recursively its R-fillers) takes part to. It is straightforward to see that this approach is the same one to which def. 2.4.13 has been inspired. For clarity the following example is considered. Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a knowledge base, with $\mathcal{T} = \{A \sqsubseteq \top, B \sqsubseteq \top, C \sqsubseteq \top, \exists R.\top \sqsubseteq \top\}$ and $\mathcal{A} = \{A(a), B(b), C(c), R(a,b), R(b,c), R(c,a)\}$ and let $k := 2$ be the fixed value for the bound k. The computation of the concept D (following [43] and def. 2.4.13) as the k-bound approximation for the individual a will be: $D \equiv A \sqcap \forall R.(B \sqcap \forall R.C)$. This result can be easily understood noting that:
a belongs to the extension of A $(k = 0)$
a takes part to role R with b as filler and b belongs to the extension of B $(k = 1)$
b takes part to role R with c as filler and c belongs to the extension of C $(k = 2)$.

Looking at the last example two observations can be made:

- asking for $k \geq 3$ the next step in computing D will consist in considering $R(c, a)$ which implies to examine a again, consequently cycling on the same assertions. Therefore the bound k assures to avoid falling in infinite loops that might easily occur even in tiny ABoxes as the one in the example;
- asking to a reasoner whether a actually belongs to concept D (computed in the example), it would answer with a discouraging "`no`". This is due to the Open World Assumption (OWA).

Differently from the Closed World Assumption (CWA), usually adopted in database systems, DL reasoners are allowed to infer the truth of universal (or numeric) restrictions only if this is directly stated in the knowledge base, and not solely on the ground of the known assertions in the A-Box. Indeed, it may always happen that new assertions contradict an erroneous generalization. This is true also when dealing with negated restrictions (anything that is not asserted or is not provable cannot be assumed as false - negation as failure). To solve this problem three different solution can be considered:

- building up the k-bound abstraction using existential restrictions instead of universal ones (this ensures that the individual always belongs to its abstraction);
- building up the k-bound abstraction using universal restrictions and adding required assertions to the knowledge base;

- employing an epistemic operator [67] for relativizing an assertion to what is currently known to the knowledge base.

Note that choosing the second alternative, considered the concept D built in the previous example, the following assertions should be added to the ABox: $\{\forall R.(B \sqcap \forall R.C)(a), \forall R.C(b)\}$. This recalls a default reasoning setting (for more details about default reasoning see [8]), in which some unexpressed knowledge, if consistent with pre-existing one, may be added to the knowledge base.

2.4.3 Computing Unification and Matching of Concept Descriptions

Unification and matching are non-standard inferences that allow to replace certain concept names by concept descriptions before testing for equivalence or subsumption. This capability turns out to be useful when maintaining (large) knowledge bases, generally made by different knowledge engineer.

Unification

Concept unification [14] is an operation that can be regarded as weaking the equivalence between concept expressions. The introduction of this non-standard inference has been motivated by the following application problem: if several knowledge engineers are involved in defining new concepts, and the knowledge acquisition process takes rather long, it can happen that the same (intuitive) concept is introduced several times, often with slightly differing descriptions. In this context, testing for concept equivalence could be not enough to find out whether, for a given concept description, there already exists another concept description in the knowledge base describing the same notion. In order to overcome this problem the idea is that, given a concept description, some parts of it could be replaced by other concept names in order to make such concept definition equivalent to another. In order to clarify this notion an example is shown. Let be considered the following concept descriptions that are intuitively supposed to be equivalent:

$$\forall \mathsf{hasChild}.\forall \mathsf{hasChild}.\mathsf{Rich} \sqcap \forall \mathsf{hasChild}.\mathsf{Rmr}$$
$$\mathsf{Acr} \sqcap \forall \mathsf{hasChild}.\mathsf{Acr} \sqcap \forall \mathsf{hasChild}.\forall \mathsf{hasChild}.\mathsf{Rich}$$

Under the assumption that Rmr stands for "Rich and married rich" and Acr stands for "All children are rich", and replacing the concept name Rmr by the description $\mathsf{Rich} \sqcap \forall \mathsf{hasSpouse}.\mathsf{Rich}$ and Acr by $\forall \mathsf{hasChild}.\mathsf{Rich}$ yields the descriptions

$$\forall \mathsf{hasChild}.\forall \mathsf{hasChild}.\mathsf{Rich} \sqcap \forall \mathsf{hasChild}.(\mathsf{Rich} \sqcap \forall \mathsf{hasSpouse}.\mathsf{Rich})$$
$$\forall \mathsf{hasChild}.\mathsf{Rich} \sqcap \forall \mathsf{hasChild}.\forall \mathsf{hasChild}.\mathsf{Rich} \sqcap \forall \mathsf{hasChild}.\forall \mathsf{hasSpouse}.\mathsf{Rich}$$

which are clearly equivalent. Thus it is possible to conclude that both descriptions are meant to express the concept ”All grandchildren are rich and all children are rich and married rich”.

A *substitution* of concept descriptions for concept names that makes two concept descriptions C, D equivalent is called a *unifier* of C and D. Of course, before testing for unsatisfiability, it has to be decided which of the concept names the unifier is allowed to replace. These names are called *concept variables*, to distinguish them from the usual concept names that cannot be replaced. In the above example, the concept names Acr and Rmr were considered to be variables, whereas Rich was treated as a (non-replaceable) concept name. Concept descriptions containing variables are called *concept patterns*. Formally, a substitution (in a certain DL) is defined as a mapping from the concept variables into the set of concept descriptions. An example is the substitution $\{\mathsf{Rmr} \mapsto \mathsf{Rich} \sqcap \forall \mathsf{hasSpouse}.\mathsf{Rich},\ \mathsf{Acr} \mapsto \forall \mathsf{hasChild}.\mathsf{Rich}\}$ used in the example. The application of a substitution can be extended from variables to concept patterns. Currently, the results for unification have been formalized only for DLs with low expressivity (see [14, 15] for more details).

The intuition underlining concept unification is that, in order to find possible overlaps between concept definitions, certain concept names can be treated as variables, hence, via unification, can be discovered that two concepts (possibly independently defined by distinct knowledge designers) are equivalent. The knowledge base may consequently be simplified by introducing a single definition of the unifiable concepts. Of course, it can happen that concept descriptions unifiable in this way do not really mean to represent the same notion. However, an unsatisfiability test can be used to suggest to the knowledge engineer possible candidate descriptions.

Matching

Matching can be seen as a special case of unification, where one of the two expressions to be unified do not contain variables [14, 15]. Thus, a matching problem is of the form $C \equiv^? D$ where C is a concept description and D is a concept pattern. A substitution σ is a matcher of this problem iff $C \equiv \sigma(D)$. Moreover, it is possible to define matching (as well as also unification) by the use of the subsumption relation rather than the equivalence. The first approach is named *matching modulo equivalence*, while the second one *matching modulo subsumption* [28].

A matching problem modulo subsumption is of the form $C \sqsubseteq^? D$, where C is is a concept description and D is a concept pattern. Such a problem asks for a substitution σ such that $C \sqsubseteq \sigma(D)$. Since σ is a solution of $C \sqsubseteq^? D$ iff σ

solves $C \equiv^? C \sqcap D$, then matching modulo subsumption can be reduced to matching modulo equivalence and hence seen as a unification problem. Anyway, in the context of matching modulo subsumption, the interest is in finding the "minimal" solutions of $C \sqsubseteq^? D$, namely σ should satisfy the property that there does not exist another substitution δ such that $C \sqsubseteq \delta(D) \sqsubset \sigma(D)$.

Matching modulo equivalence was introduced to help filtering out unimportant aspects of complicated concepts appearing in large knowledge bases, and to specify patterns for explaining proofs carried out by DL systems [139]. For example, matching the concept pattern $D = \forall \mathsf{researchInterest}.X$ against the description $C = \forall \mathsf{pets.Cat} \sqcap \forall \mathsf{researchInterest.AI} \sqcap \forall \mathsf{hobbies.Gardeninn}$ yields the minimal matcher $\sigma = \{X \mapsto \mathsf{AI}\}$, thus finding the scientific interest described in the concept and consequently filtering out the other aspects described by C. Another motivation for matching as well as unification can be found in the area of integrating data or knowledge base schemata represented in some DL. In fact, an integrated schema can be viewed as the union of the local schemata along with some interschema assertions satisfying certain conditions. Finding inter-schema assertions can be supported by solving matching or unification problems, as proposed in [27].

2.4.4 Concept Rewriting and Approximation across DLs

Approximation in DLs was first mentioned by Baader, Küsters and Molitor in [13] as possible new inference. Approximating a concept, defined in a DL, means to translate this concept to another concept, defined in a second, typically less expressive, DL, such that both concepts are as closely related as possible with respect to some relation such as equivalence or subsumption (generally subsumption is used). Formally, concept approximation consist in: given a concept C defined in a certain DL $\mathcal{L}_s$ ("s" for source) find a concept D, the upper/lower approximation of C, in a DL $\mathcal{L}_d$ ("d" for destination) such that i) D subsumes/is subsumed by C, and ii) D is minimal/maximal concept in $\mathcal{L}_d$ (w.r.t. subsumption) with this property.

There is a number of different applications of this inference problem (see [36] for an overview). A first example is represented by the *translation of Knowledge Bases* which may become necessary to port KBs across different knowledge representation systems or to integrate different KBs. In these cases approximation could be used to (automatically) translate a KB written in an expressive DL into another (semantically closely related) KB described by a less expressive DL. Another important application of the approximation is enabling *non-standard inferences for expressive DLs*. In fact, as seen previously, non standard inferences in DLs are mostly restricted to quite inexpressive DLs (such as those that do not allow for concept disjunction). By first approximating KB written in expressive DLs to a less expressive DL, non-standard inferences, that cannot be applied to the original knowledge base, could be applied to the approximated knowledge base. In this way, even some information may be lost, it is possible to have some (even most general) results rather than to have no results.

Approximation can be also used for *computing commonalities between concepts.* Indeed, as seen in Sect. 2.4.1, typically the Least common subsumer is employed for this task, but for DL allowing disjunction it is simply given by the union of the considered concept descriptions, that gives very low information about concept commonalities. So, a possible solution is given by approximating concept descriptions to less expressive DLs not allowing for disjunction and hence compute the lcs on such approximated concepts. In this way concept commonalities can be made explicit. As it will be shown in Sect. 5.2.2, this thesis uses concept approximation for finding commonalities between $\mathcal{ALC}$ concept descriptions.

Computing approximation from $\mathcal{ALC}$ to $\mathcal{ALE}$ concept descriptions

In this section it is illustrated how $\mathcal{ALC}$ concept descriptions can be approximated by $\mathcal{ALE}$ concept descriptions [36]. The notion of concept approximation is formally defined.

Definition 2.4.14 (Concept approximation) *Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be two DLs, and let C be an $\mathcal{L}_1$- and D be an $\mathcal{L}_2$ concept descriptions. Then, D is called upper (lower) $\mathcal{L}_2$-approximation of C ($D = \mathsf{approx}_{\mathcal{L}_2}(C)$ for short) if*

1. *$C \sqsubseteq D$ ($C \sqsubseteq D$)*
2. *D is minimal (maximal) with this property, i.e., $C \sqsubseteq D'$ and $D' \sqsubseteq D$ ($D \sqsubseteq D'$) implies $D' \equiv D$ for all $\mathcal{L}_2$ concept descriptions D'.*

In this section the attention will be concentrated on *upper* $\mathcal{ALE}$-approximations of $\mathcal{ALC}$ concept descriptions.

First of all it is important to note that approximations need not exist in general. Indeed, let $\mathcal{L}_1 = \{\sqcap\}$ and $\mathcal{L}_2 = \{\sqcup\}$ be two DLs allowing respectively only conjunction and disjunction operators, and let A and B be two concept names. It is straightforward to see that there does not exist an upper level $\mathcal{L}_1$-approximation of the $\mathcal{L}_2$-concept description $A \sqcup B$. In the same way there does not exist a lower $\mathcal{L}_2$-approximation of the $\mathcal{L}_1$-concept description $A \sqcap B$.

Moreover, approximations need not be uniquely determined. Indeed, looking at the example just shown, it is easy to see that both A and B are lower $\mathcal{L}_1$-approximations of the concept definition $A \sqcup B$.

Anyway, considering $\mathcal{ALE}$ DL, since it allows for conjunction, it immediately follows that if upper $\mathcal{ALE}$-approximations exist, they are uniquely determined up to equivalence (as shown in [36]). In the following, the algorithm for obtaining an $\mathcal{ALE}$-approximation, starting from an $\mathcal{ALC}$ concept description is presented. It is based on the structural characterization of subsumption between an $\mathcal{ALC}$-concept description, say C, in $\mathcal{ALC}$-normal form and an $\mathcal{ALE}$-concept description, say D.

Theorem 2.4.15 (Subsumption between an $\mathcal{ALC}$ and an $\mathcal{ALE}$ concept) *Let $C = \bigsqcup_{i=1}^{n} C_i$ be an $\mathcal{ALC}$ concept description in $\mathcal{ALC}$ normal form and D an $\mathcal{ALE}$ concept description. Then*

- *$C \sqsubseteq D$ iff $C \equiv \bot$ or $D \equiv \top$*

or, for all $i = 1, \ldots, n$ it holds that

1. *$\mathsf{prim}(D) \subseteq \mathsf{prim}(C_i)$ and*
2. *for all $D' \in \mathsf{ex}(D)$ there exists $C' \in \mathsf{ex}(C_i)$ such that $C' \sqcap \mathsf{val}(C_i) \sqsubseteq D'$ and*
3. *$\mathsf{val}(C_i) \sqsubseteq \mathsf{val}(D)$.*

The idea[9] is that D is compared to every disjunct C_i in C. This comparison, in turn, is very similar to the structural characterization of subsumption between $\mathcal{ALE}$-concept descriptions (see Sect. 2.4.1).

Given the notion of subsumption between an $\mathcal{ALC}$ and $\mathcal{ALE}$ concept, it is now possible to show the algorithm "$\mathsf{c\text{-}approx}_{\mathcal{ALE}}$" for computing the approximation.

$\mathsf{c\text{-}approx}_{\mathcal{ALE}}$ (In: $\mathcal{ALC}$ concept C) : Out: upper $\mathcal{ALE}$-approximation of C

1. If $C \equiv \bot$, then $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C) := \bot$
2. If $C \equiv \top$, then $\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C) := \top$
3. Otherwise, transform C into $\mathcal{ALC}$-normal form $C_1 \sqcup \cdots \sqcup C_n$ and return

$$
\begin{aligned}
\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C) \;:=\; & \bigsqcap_{A \in \bigcap_{i=1}^{n} \mathsf{prim}(C_i)} A \;\sqcap \\
\sqcap \; & \bigsqcap_{(C'_1,\ldots,C'_n) \in \mathsf{ex}(C_1) \times \cdots \times \mathsf{ex}(C_n)} \exists r.\mathsf{lcs}\{\mathsf{c\text{-}approx}_{\mathcal{ALE}}(C'_i \sqcap \mathsf{val}(C_i)) \mid 1 \le i \le n\} \;\sqcap \\
\sqcap \; & \forall r.\mathsf{lcs}\{\mathsf{c\text{-}approx}_{\mathcal{ALE}}(\mathsf{val}(C_i)) \mid 1 \le i \le n\}
\end{aligned}
$$

[9]See [36] for the proof of the Theorem 2.4.15.

As an example, let be two $\mathcal{ALC}$ concepts in normal form:
$C_1 \equiv (\exists r.A \sqcap \forall r.B) \sqcup (\exists r.A \sqcap \exists r.B \sqcap \forall r.A)$
$C_2 \equiv \exists r.A \sqcap \exists r.B \sqcap \forall r.(\neg A \sqcup \neg B)$
It can be verified that:
$\textsf{c-approx}_{\mathcal{ALE}}(C_1) \equiv \exists r.(A \sqcap B)$
$\textsf{c-approx}_{\mathcal{ALE}}(C_2) \equiv \exists r.(A \sqcap \neg B) \sqcap \exists r.(B \sqcap \neg A)$
Hence, given an $\mathcal{ALC}$ concept description C, the $\textsf{c-approx}_{\mathcal{ALE}}$ algorithm finds an $\mathcal{ALE}$ concept description which is as specific as possible and satisfies the conditions of Theorem 2.4.15. Moreover, in [36] the following important result is proved.

Theorem 2.4.16 (Approximation existence) *For every $\mathcal{ALC}$ concept description C the $\mathcal{ALE}$-approximation exists. It is uniquely determined up to equivalence and can be computed by the $\textsf{c-approx}_{\mathcal{ALE}}$ algorithm, i.e., $\textsf{approx}_{\mathcal{ALE}}(C) \equiv \textsf{c-approx}_{\mathcal{ALE}}(C)$.*

Anyway in [36] it is also proved that the approximation of $\mathcal{ALC}$ concept descriptions may grow exponentially and the algorithm for computing such approximation has a double-exponential time complexity.

Chapter 3

Similarity and Dissimilarity Measures: Related Work

Defining similarity and dissimilarity measures is a topic that has been investigated for a long time and that plays a key role in different areas such as Natural Language Processing, Information Retrieval, Information Integration and Machine Learning. As Quine observed [162] "*Similarity is fundamental for learning, knowledge and thought, for only our sense of similarity allows us to order things into kinds so that these can function as stimulus meanings. Reasonable expectation depends on the similarity of circumstances and on our tendency to expect that similar causes will have similar effects.*" Similarity thus is fundamental in making predictions. Indeed this task is generally grounded on the assumption that similar things usually behave similarly. In Sect. 1.3 some learning methods, namely instance-based and clustering methods, have been presented and the importance that (dis-)similarity measures have for them has been highlighted.

In this chapter similarity and dissimilarity measures will be formally defined, and the set of properties that they satisfy will be explained. Hence the main models, proposed in the literature, for (dis-)similarity assessment will be presented.

Definition and evaluation of similarity and dissimilarity measures have been largely studied in the literature. They can be classified with respect two different dimensions. One is related to the kind of knowledge representation to which they are applied. The other is related to the way in which similarity and/or dissimilarity is computed.

Along the dimension of the representation, it is possible to distinguish: measures applied to feature vectors, measures applied to strings, measures applied to sets, measures applied to trees and measures applied to clauses (see [165] for a complete overview).

Along the dimension of the way for computing (dis-)similarity, it is possible to distinguish: measures based on geometric models, measures based on feature matching, measures based on semantic relations, measures based on Information Content, measures based on alignment and transformational models, and measures based on contextual information (following [169, 86]).

With respect to the representation dimension, the distinction that will be made is between propositional setting and relational setting. Hence, for each representational setting, the different models for computing (dis-)similarity will be presented. For each model, the most prominent measures in the literature will be analyzed. In the last part of this chapter, the attention will be focused on definition of measures applied to knowledge expressed in Description Logics i.e. in the ontological setting (see Sect. 1.1.1). It is important to note that main interest of this thesis is about assessing (dis-)similarity among concept definitions and among individuals, hence, aspects such as assessing similarity between ontologies will be not treated extensively. Even if such topic represents an interesting open issue for researchers w.r.t. tasks such as ontology mapping and alignment, it is not directly linked with the main motivations of this thesis.

3.1 Defining Similarity and Dissimilarity Measures

Similarity and dissimilarity measures are applied to objects of a considered domain in order to determine "somehow" how much they are similar or how much difference there is among them. From here it will be used the term *similarity measure* for determining the amount of similarity among objects and *dissimilarity measure* for determining the amount of difference between objects.

Intuitively, defining a similarity measure requires two steps. In the first step, a set of similarity values, e.g. the set {*far, near*} or the set {*equal, similar, dissimilar, totally different*}, has to be defined. A commonly used set of similarity values is the set of the real number $\mathbb{R}$ (where one has to specify whether larger values mean "more similar" or "more dissimilar"). The second step consist in defining a function from a pairs of objects to the set of similarity values. Following [26, 165] formal definitions of similarity and dissimilarity measures can be given.

Definition 3.1.1 (Similarity Measure) *Let D be a set of elements of a considered domain and let $(V, \leq)$ be a totally ordered set. A function $s : D \times D \to V$ is a similarity function iff there exists an element $0_V \in V$ and $1_V \in V$ such that:*

- *the function is positive: $\forall x, y \in D \ : \ s(x, y) \geq 0_V$*

- *the function is reflexive:* $\forall x \in D \ : \ s(x,x) = 1_V$ *and* $\forall y \in D \wedge x \neq y \ : \ s(x,x) \geq s(x,y)$
- *the function is symmetric:* $\forall x, y \in D \ : \ s(x,y) = s(y,x)$

Definition 3.1.2 (Dissimilarity Measure) *Let D be a set of elements of a considered domain and let $(V, \leq)$ be a totally ordered set. A function $d : D \times D \to V$ is a dissimilarity function iff there exists an element $0_V \in V$ such that:*

- *the function is positive:* $\forall x, y \in D \ : \ d(x,y) \geq 0_V$
- *the function is reflexive:* $\forall x \in D \ : \ d(x,x) = 0_V$ *and* $\forall y \in D \wedge x \neq y \ : \ d(x,x) \leq d(x,y)$
- *the function is symmetric:* $\forall x, y \in D \ : \ d(x,y) = d(y,x)$

In the sequel the terms *measure* and *function* will be used interchangeably as is usual in the literature. For the next presentation of the function properties, only the dissimilarity measure will be considered, since it is almost the same in the case of the similarity measure. In general, it will be denoted with s a similarity measure and with d a dissimilarity measure.

Definition 3.1.3 (Strictness property) *Let D be a set of elements of a considered domain and let $(V, \leq)$ be a totally ordered set. Let d be a dissimilarity function on D with minimum value $0_V \in V$. The function d is strict iff:*

$$\forall x, y \in D \ : \ d(x,y) = 0_v \Rightarrow x = y$$

The strictness property ensures that the minimum dissimilarity value is assigned only if the considered element are equal. If the strictness property is not satisfied then also elements that are different could assumes the lowest dissimilarity value.

Definition 3.1.4 (Triangle inequality property) *Let D be a set of elements of a considered domain and let $(V, +, \leq)$ be a totally ordered set equipped with an order-preserving addition operation such that $(V, +)$ is a commutative group. Let d be a dissimilarity function on D. The function d satisfies the triangle inequality iff:*

$$\forall x, y, z \in D \ : \ d(x,y) + d(y,z) \geq d(x,z)$$

The idea of the triangle inequality is that the distance from an object A to an object C, is not greater the sum of distances from A to another object B and from B to C. Measures that satisfy the triangle inequality often produce more intuitive results and also allow many optimizations in algorithms exploiting them[1].

Definition 3.1.5 (Pseudo-metric) *A dissimilarity function is a* pseudo-metric *(or equivalently a* semi-distance*) iff it satisfies the triangle inequality.*

Definition 3.1.6 (Metric) *A pseudo-metric is a* metric *(or equivalently a* distance*) iff it satisfies the strictness property.*

Definition 3.1.7 (Normalized Dissimilarity Function) *Let D be a set of elements of a considered domain. Let d be a dissimilarity function on D with value in $\mathbb{R}$. Then d is a normalized dissimilarity function if*

$$\forall x, y \in D \ : \ 0 \leq d(x, y) \geq 1$$

where $d(x, x) = 0$ *and* $\forall x \in D, \exists y \in D \ : \ d(x, y) = 1$

Note that, given a dissimilarity function d it is always possible to construct a dissimilarity function d' that is a normalized dissimilarity function.

About the opportunity that a measure satisfies the presented properties a long debate has been developed among researchers that is currently opened. Mathematicians assert the necessity that measures satisfy the presented properties because they guarantee the completeness of the measure from a theoretical point of view. On the contrary, researchers from other fields, such as Machine Learning, psychology and cognitive science assert that some properties are not only useless but also inappropriate. For instance, Tversky [197] argues that similarity is not always a symmetric relation. The same opinion is shared in [71] where the argumentation is that in the naive view of the world, distance as well as similarity, particularly if it is defined in terms of a conceptual distance, is frequently asymmetric. However, Rada et al. [163] argue that, when similarity is limited to a feature comparison process, it is symmetric. They believe that the asymmetric problem of similarity found by Tversky [197] is a result of the existence of another asymmetric relation. For example, a metaphor relating two concepts by a "like" relation involves a selective rather than an unconstrained comparison process.

Also the importance of the triangle inequality has been discussed. Indeed while the supporters of triangle inequality argue that it is fundamental both to produce

[1]In [64], for example, a speed-up of a hierarchical algorithm is presented in case in which the used measure satisfies the triangle inequality.

more intuitive results and also for allowing many optimizations in algorithms (as shown in [64]) another line of research, as in [108, 90], claims that triangle inequality is not necessary to guarantee good algorithm results.

Another characteristic of the similarity assessment often discussed in the literature is the relation between the notion of similarity and difference. In general, it is assumed an inverse relation between similarity and difference. Anyway Tversky [197] considers this assumption inaccurate. Specifically, he argues that, even if an increase in the measure of the common features increases the similarity and decreases the difference whereas an increase in the measure of distinction decreases similarity and increases difference, the relative values of these two semantic relations may differ. Indeed, while subjects may pay more attention to the similar features in the assessment of similarity among objects, they may pay less attention to their distinctive features in the assessment of difference [197, 122].

As will be shown in Ch. 4, this thesis focuses on the definition of (dis-)similarity measures following Def. 3.1.1 and Def. 3.1.2, so considering the symmetric property important for the fixed goal. On the contrary, the triangle inequality is not considered fundamental for defining a good measure, therefor no attention will be posed to satisfying this property. Moreover, even if not directly used, it is assumed that the notion of dissimilarity is just the contrary of the similarity and vice-versa.

3.2 Similarity and Dissimilarity Measures in the Propositional Setting

Distance based methods were mainly developed for the propositional setting. Consequently, a large literature exists on determining (dis-)similarity measures in this setting. Such measures are useful in Machine Learning area and particularly for clustering and instance based learning methods.

When a *propositional* (or equivalently said *attribute-value* or *features-vector*) language is used, examples are represented as rows in a single table. More formally, the space of possible examples is defined as the product $\xi = D_1 \times D_2 \times \dots D_n$ of the n domains $D_1, D_2, \dots D_n$ of the attributes, hence, each example E is an n-tuple $(x_1, x_2, \dots x_n) \in \xi$ of fixed length, where each position has a fixed type.

The best known distance measures on vector space are based on the geometric model and the feature-matching model. In the following such approaches will be analyzed separately.

3.2.1 Measures based on Geometric Model

Geometric models of similarity have been among the most influential approaches for assessing similarity [40, 193, 194]. In this approach objects are seen as point in an n-dimentional space. The similarity between a pair of objects is considered to be inversely related to the distance between two objects points in the space. The best known distance measures on vector space based on geometric model are the Minkowski measure, the Manhattan measure and the Euclidean measure.

Definition 3.2.1 (Minkowski Distance) *Let $D_1, D_2, \ldots, D_n$ be sets and let $(V, +, \leq)$ be a commutative group on which a total order is defined. Let $D = D_1 \times D_2 \times \ldots D_n$ be the set of all vectors $\mathbf{x} = (x_1, x_2, \ldots x_n)$ such that $x_i \in D_i$ for $i = 1, 2 \ldots n$. Then, given $q \in \mathbb{N}^*$, the distance function $d : D \times D \to V$*

$$d(\mathbf{x}, \mathbf{y}) := \| \mathbf{x} - \mathbf{y} \|_q := \left[\sum_{i=1}^{n} (x_i - y_i)^q \right]^{\frac{1}{q}}$$

is called Minkowski Distance*, where n is the number of dimensions, x_i is the value of the item w.r.t. dimension i and q is a parameter that allows different spatial metrics to be used.*

Definition 3.2.2 (Euclidean Distance) *A Minkowski distance where $q = 2$ is called* Euclidean Distance

Geometrically speaking, the Euclidean distance assigns, as distance between two points, the length of the straight line connecting the points.

Definition 3.2.3 (Manhattan Distance) *A Minkowski distance where $q = 1$ is called* Manhattan Distance

In the Manhattan distance, the notion of distance involves a city-block metric where the distance between two points is the sum of their distances on each dimension ("short-cut" diagonal paths are not allowed to directly connect points differing on more than one dimension).

Lemma 3.2.4 *Minkoswki distance, Manhattan distance and Euclidean distance are metrics*

Measures just presented are applied to vectors whose features are all continuous. Nevertheless, it is quite common, in ML, to have vectors with categorical features or vectors with mixed type features (categorical and continuous). In order to cope with this slightly more complex situation, many other measures have been developed[2]. Specifically, measures based on Feature Matching Model (see Sect. 3.2.3) represent an answer to this problem.

3.2.2 Kernel Functions

Kernel functions can be informally defined as similarity functions able to work with high dimensional feature spaces. They have been developed jointly with kernel methods[3]: efficient learning algorithms realized for solving classification, regression and clustering problems in high dimensional feature spaces. Two components of kernel methods have to be distinguished: the kernel machine and the kernel function. The kernel machine encapsulates the learning task and the way in which a solution is sought, the kernel function encapsulates the hypothesis language, i.e., how the set of possible solutions is made up. This means that the same kernel machine can be used with different kernel function by simply considering a new kernel function (high modularization). Different kernel functions embed different hypothesis spaces. Before analyzing kernel functions, the intuition from which they move is briefly illustrated in the following.

Kernel functions have been introduced in the field of pattern recognition whose goal, in its simplest formulation, is to estimate a function $f : \mathbb{R}^N \to \{-1, +1\}$ using input-output training data pairs generated independent identically distributed (i.i.d.) according to an unknown probability distribution $P(x, y)$, where training data are $(x_1, y_1), \ldots (x_N, y_N) \in \mathbb{R}^N \times Y$, $Y = \{-1, +1\}$, such that f will correctly classify unseen examples (x, y). An example is assigned to the class $+1$ if $f(x) \geq 0$ and to the class -1 otherwise. The test examples are assumed to be generated from the same probability distribution $P(x, y)$ as the training data; hence, y is determined such that (x, y) is in some sense similar to the training examples. To this end, a similarity measure $k : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}$ is necessary such that, given two examples x and x' returns a real number characterizing their similarity. For reasons that will be clarified later, the function k is called a *kernel* [31]. A type of similarity measure that is of particular mathematical appeal are inner products, namely given two vectors

[2]In [108, 26] a set of measures applicable to objects defined by non-numeric features are presented.

[3]The class of kernel methods is formed by learning algorithms like Gaussian processes [152], kernel principal component analysis [180] for feature extraction, kernel Fisher discriminant [147, 19], the most famous Support Vector Machines [31, 178] for supervised learning and Support Vector Clustering [20] for unsupervised learning. Different kernel machines tackle different learning tasks.

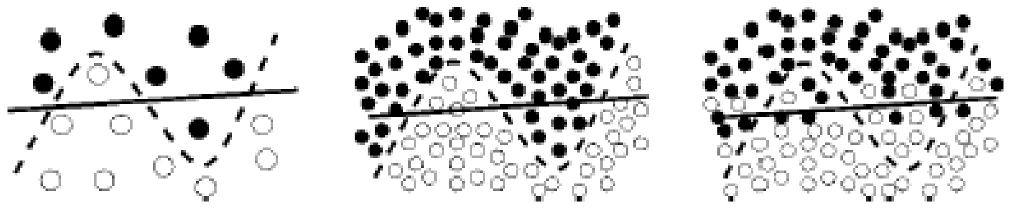

Figure 3.1: Illustration of the overfitting dilemma: Given only a small sample (left) either, the solid or the dashed hypothesis might be true, the dashed one being more complex, but also having a smaller training error. Only with a large sample it is possible to be able to see which decision reflects the true distribution more closely. If the dashed hypothesis is correct the solid would underfit (middle); if the solid were correct the dashed hypothesis would overfit (right).

$x, x' \in \mathbb{R}^N$ the canonical inner product is defined as: $(x \cdot x') = \sum_{i=1}^{N} x_i \cdot x'_i$ where x_i denotes the i-th entry of x.

One of the crucial points for solving the presented goal is, hence, the choice of the function for performing classification. It has to be chosen in a way such that it minimizes the *expected error (risk)*. Specifically, since the expected error cannot be computed, being P(x,y) unknown, the function has to be chosen such that it minimizes the *empirical risk* [152] (that is computable). This is since it is possible to give conditions on the learning machine which ensure that asymptotically (as the number of examples $n \to \infty$) the empirical risk will converge toward the expected risk. However, for small sample sizes, large deviations are possible and *overfitting* might occur, as shown in Fig. 3.1. Then a small generalization error cannot be obtained by simply minimizing the training error. One way to avoid the overfitting dilemma is to restrict the complexity[4] of the function class from which the function f is chosen [199]. The intuition which will be formalized in the following is that a "simple" (e.g., linear) function minimizing the error and that explains most of the data is preferable to a complex one (Occams razor).

Moving from such an intuition, algorithms in feature spaces target a linear function for performing the learning task. Considering that this choice cannot be always possible (see Fig. 3.1), algorithms use of the following trick. Via a nonlinear mapping $\phi : \mathbb{R}^N \to \mathcal{F}$ that $\forall x \in \mathbb{R}^N$ assigns $\phi(x)$, $(x_1, \ldots, x_N) \in \mathbb{R}^N$ are mapped

[4]A specific way for controlling the complexity of a function class is given by the Vapnik-Chervonenkis (VC) theory and the structural risk minimization (SRM) principle [199, 200, 201], that present a way for bounding the complexity of the function class from which f is chosen from. Complexity is captured by the notion of VC dimension, which measures how many training points can be shattered (i.e. separated), for all possible labelings, using a function of the class. Anyway sometimes this bound cannot be easily computed.

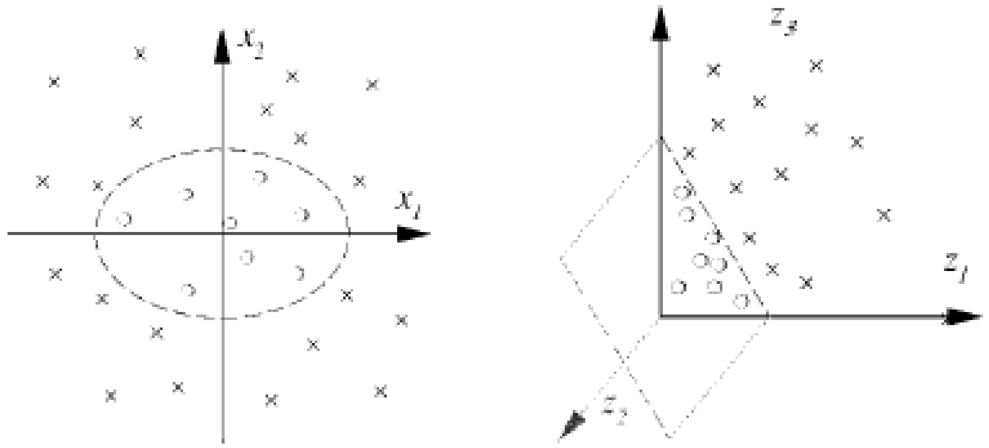

Figure 3.2: Two-dimensional classification example. (a) Function separation in the original space of features. (b) Linear function separation in the mapped feature space.

into a, potentially, much high dimensional space $\mathcal{F}$. Hence, for a given learning problem, the same algorithm can be considered in $\mathcal{F}$ rather than in $\mathbb{R}^N$, i.e., the algorithm works with the sample $(\phi(x_1), y_1), \ldots, (\phi(x_n), y_n) \in \mathcal{F} \times Y$. Given this mapped representation, a simple linear classification in $\mathcal{F}$ is to be learnt.

This idea can be clarified by means of the toy example in Fig. 3.2: in two dimensions a rather complicated nonlinear decision surface is necessary to separate the classes, whereas in a feature space of second-order monomials (see e.g., [181]) $\phi : \mathbb{R}^2 \to \mathbb{R}^3$ such that $\forall (x_1, x_2) \in \mathbb{R}^2 : (x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}x_1x_2, x_2^2)$ a linear decision surface is sufficient in order to separate the classes. In this example, it is easy to control both the statistical complexity (by using a simple linear hyperplane classifier) and the algorithmic complexity of the learning machine, as the feature space is only three dimensional. However, it becomes rather tricky to control the algorithmic complexity of the learning machine for large real-world problems. For instance, considering images of 16×16 pixels as patterns and fifth-order monomials as mapping ϕ, then one would map to a space that contains all fifth-order products of 256 pixels, i.e., to about a 10^{10}-dimensional space. So, even if the statistical complexity of this function class can be controlled, the result is an intractable problem executing an algorithm in this space. Fortunately, for certain feature spaces $\mathcal{F}$ and corresponding mappings ϕ there is a highly effective trick for computing scalar products using kernel functions [31, 199, 173]. For instance, considering the toy example, the computation of a scalar product between two feature space vectors, can be readily reformulated in terms of a kernel function k as follow:

$(\phi(\mathbf{x}) \cdot \phi(\mathbf{y})) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)((y_1^2, \sqrt{2}y_1y_2, y_2^2))^T = ((x_1, x_2)(y_1, y_2)^T)^2 = (\mathbf{x} \cdot \mathbf{y})^2$ $=: k(\mathbf{x}, \mathbf{y})$. Hence, a kernel function is a similarity measure that computes the inner product in a high dimensional feature space $\mathcal{F}$, which is in general different from the representation space of the instances.

The interesting aspect of the kernel functions is that the inner product can be *implicitly* calculated in $\mathcal{F}$, *without* explicitly computing, using or even knowing, the mapping ϕ. So, kernels compute inner products in spaces, where, otherwise, it could be hardly performed any computations, as seen for the extended version of the toy example above. A direct consequence of this is [179]: *every (linear) algorithm that only uses inner products can implicitly be executed by using kernels, i.e., one can very elegantly construct a nonlinear version of a linear algorithm*[5].

Until now $\mathbb{R}^N$ has been considered as representation space of the instances. Really whatever set $\mathcal{X}$ can be considered. **An important result** is claimed in [6]. Here, it is asserted that **any set, whether a linear space or not, that admits a *positive definite kernel* can be embedded into a linear space**. In the following (see also Sect. 3.3.6), the term "valid" is used to mean "positive definite". Here, the definition of a positive definite kernel[6] is reported.

Definition 3.2.5 (Positive Definite Kernel) *Let $\mathcal{X}$ be a set. A symmetric function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a positive definite kernel on $\mathcal{X}$ if, for all $n \in \mathbb{Z}^+$, $x_1, \ldots, x_n \in \mathcal{X}$, and $c_1, \ldots, c_n \in \mathbb{R}$, it follows that*

$$\sum_{i,j \in \{1,\ldots,n\}} c_i c_j k(x_i, x_j) \geq 0$$

namely, the matrix K defined by $K_{ij} = k(x_i, x_j)$ is positive definite.

While it is not always easy to prove the positive definiteness for a given kernel following the definition above, the validity of a kernel function could be showed exploiting some closure properties that have to be satisfied. In particular, kernel functions are closed under sum, direct sum, multiplication by a scalar, product, tensor product, zero extension, pointwise limits, and exponentiation [48, 99].

In the following, the traditionally used kernel on vector spaces are briefly reviewed.

[5]Even algorithms that operate on similarity measures k generating positive matrices $k(x_i, x_j)$ can be interpreted as linear algorithms in some feature space $\mathcal{F}$ [177].

[6]In Def. 3.2.5 a function with real value is considered. However many authors consider the general case of complex-valued kernels. The relationship between the definition used for that case and the ones used here is discussed in [21]. Virtually all the results extend naturally to the complex case.

Let $x, x' \in \mathbb{R}^N$ and let $\langle \cdot, \cdot \rangle$ denote the inner product in $\mathbb{R}^N$. Apart from the **Linear Kernel**

$$k(x, x') = \langle x, x' \rangle = \sum_{i=1}^{N} x_i \cdot x'_i$$

and the **Normalized Linear Kernel**

$$k(x, x') = \frac{\langle x, x' \rangle}{\sqrt{|| x || \, || x' ||}}$$

the two most frequently used kernels on vector spaces are the polynomial kernel and the Gaussian RBF kernel. Given two parameters $l \in \mathbb{R}, p \in \mathbb{N}^+$, the **polynomial kernel** is defined as:

$$k(x, x') = (\langle x, x' \rangle + l)^p$$

The intuition behind this kernel definition is that it is often useful to construct new features as products of original features. The parameter p is the maximal order of monomials making up the new feature space, while l can be used as a bias towards lower-order monomials; for instance, if $l = 0$ the feature space consists only of monomials of order p of the original features.

Given the parameter γ, the **Gaussian RBF kernel** is defined as:

$$k(x, x') = e^{-\gamma ||x - x'||^2}$$

Using this kernel function in a support vector machine can be seen as using a radial basis function network with Gaussian kernels centered at the support vectors. The images of the points from the vector space $\mathbb{R}^N$ under the map $\phi : \mathbb{R}^N \rightarrow \mathcal{H}$ with $k(x, x') = \langle \phi(x), \phi(x') \rangle$ lie all on the surface of a hyperball in the Hilbert space $\mathcal{H}$. Now two images are orthogonal and any set of images is linearly independent.

Others known kernels on vector spaces are: **sigmoidal kernel** where given $l, \theta \in \mathbb{R}$ it is defined as:

$$k(x, x') = \tanh(l \langle x, x' \rangle + \theta)$$

and **inverse multiquadratic kernel**, where given $c \in \mathbb{R}^+$, it is defined as:

$$\frac{1}{\sqrt{|| x - x' ||^2 + c^2}}$$

Kernel functions jointly with kernel methods have been largely used, particularly in pattern recognition, because they revealed high computational efficiency when other methods hardly perform any computation. Anyway, in this initial formulation, they are linked to feature vector representations and are not able to give any information about how the conclusions of the learning process have been obtained.

3.2.3 Measures Based on Feature Matching Model

Similarity plays a fundamental role in theories of knowledge and behavior. It serves as an organizing principle by which persons classify objects from concepts, and make generalizations. Particularly, psychological researches demonstrated that these tasks are often performed by recurring to common and distinguishing features that characterize the objects considered. The main work proposing an evaluation of similarity based on features matching has been proposed by Tversky [197]; where objects are represented as collections of features, and similarity is described as a feature matching process. Specifically, a set of qualitative assumptions is shown to imply the *contrast model*, which expresses the similarity between objects as a linear combination of the measures of their common and distinctive features. The model is used to discover, analyze, and explain a variety of empirical phenomena such as the role of common and distinctive features, the relations between judgments of similarity and difference and the presence of asymmetric similarities.

Differently from the classical theoretical analysis of similarity measures (see Sects. 3.1, 3.2.1) dominated by geometric models, Tversky shows that there are some situations in which such models are inappropriate. Indeed, as seen in Sect. 3.2.1, geometric models represent objects as points in some coordinate space such that the observed dissimilarities between objects correspond to the metric distances (see Def. 3.1.6) between the respective points. Conversely, Tversky argues that a geometrical dimensional representation cannot be used for whatever kind of data and settings. While it can be appropriately used for representing certain stimuli (such as colors, tones), it is not so appropriate for measuring other kinds of information such as faces, countries, or personalities. Indeed, in these cases, a representation in terms of many qualitative features is much more suitable. Considering this different kind of representation, the assessment of similarity in this settings may be better evaluated as a comparison of features rather than as the computation of metric distance between points. Moreover, in this comparison some properties of the geometrical models could be inappropriate. An example is represented by the symmetric property.

Tversky asserts that the similarity judgments can be regarded as extensions of similarity statements, that is, statements of the form "*a is like b*" which is directional, in the sense that it has a subject, a, and a referent, b, and it is not equivalent, in general, to the converse similarity statement "*b is like a*", since persons tend to select the prototype, as a referent, and the less salient stimulus (or the variant) as a subject. In fact, we say "*the portrait resembles the person*" rather than "*the person resembles the portrait*" or "*the son resembles the father*" rather than "*the father resembles the son*". This asymmetry in the choice of similarity statements is associated with the asymmetry in judgments of similarity. Sometimes both directions are used, but

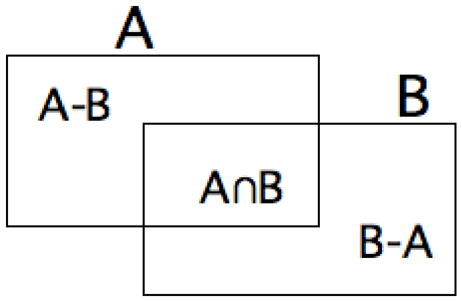

Figure 3.3: Graphical representation of the relation between two feature sets.

they carry different meanings. For example, considering metaphoric expressions, the statement "*Life is like a play*" says that people play roles, while the statement "*A play is like life*" says that a play can capture the essential elements of human life. Given such considerations, the theoretical approach to similarity based on feature matching is neither dimensional nor metric in nature.

In [197] it is assumed that every object $a, b, c, \dots$ can be represented by a set of features $A, B, C, \dots$ respectively, where features are not restricted to binary or nominal variables, but can be also ordinal or cardinal variables (i.e., dimensions) and set variables. Hence, a notion of similarity between two objects a and b is introduced. It is called "*contrast model*" and it is expressed as a linear combination of three basic components: the set of features that are common to a and b, let say $A \cap B$, the set of features that belong to a but not to b, let say $A - B$ and the set of features that belong to b but not to a, let say $B - A$. A schematic illustration of these components is presented in Fig. 3.3. A formal definition of contrast model is given in the following.

Definition 3.2.6 (Contrast Model) *Given a set of object E, for all $a, b \in E$, let A, B respectively, the set of their describing features. Then the similarity between a and b is given by:*

$$constrast_{tv}(a, b) = \theta \cdot f(A \cap B) - \alpha \cdot f(A - B) - \beta \cdot f(B - A)$$

where "$-$" is the set difference, θ, α, β are non-negative constants and $f(\cdot)$ is taken as the count of features in the set.

The similarity value between two objects is computed as a linear combination, or a "contrast", of the measures of the common and the distinctive features; hence, the representation is called the contrast model. Moreover, the function is expressed as an ordinal measure of similarity. That is, $constrast_{tv}(a, b) > constrast_{tv}(c, d)$ means that the object a is more similar to b than c is similar to d. Then, looking at the function definition it is straightforward to note that the similarity value increases with the addition of common features and/or deletion of distinctive features.

In the present theory, the assessment of similarity is described as a feature-matching process, namely it is formulated in terms of the set-theoretical notion of a matching function rather than in terms of the geometric concept of distance. An important assumption that is implicitly made here is the *solvability*, which requires that the feature space under study be sufficiently so rich that certain (similarity) equations can be solved. Finally, it important to note that the contrast model does not define a single similarity measure, rather a family of measures characterized by different values of the parameters θ, α and β. For example, if $\alpha = \beta = 0$ and $\theta = 1$, then $constrast_{tv}(a, b) = f(A \cap B)$; that is, the similarity between objects is the measure of their common features. If, on the other hand, $\theta = 0$ and $\alpha = \beta = 1$ $then constrast_{tv}(a, b) = f(A - B) + f(B - A)$; that is, the dissimilarity between objects is the measure of the symmetric difference between the respective feature sets, namely similarity is determined by distinctive features only. In this case the measure can be regarded as a measure of distance or diversity. When $\alpha = 1$ and $\beta = 0$, then the common features to a and b are compared with those unique to a. The reverse is true when $\alpha = 0$ and $\beta = 1$.

In general the contrast model expresses similarity between objects as a weighted difference of the measures of their common and distinctive features, thereby allowing for a variety of similarity relations over the same domain. Moreover, such wights allow to express the notion of asymmetry argued by Tversky. In case where asymmetry is not desired, all parameters must have the same value.

The contrast model is the easiest form of similarity measure. Another measure, called "***ratio model***" has been presented by Tversky [197]. The goal of this measure is to provide a normalized similarity value in the range $[0, 1]$. It is formally defined[7] in the following.

Definition 3.2.7 (Ratio Model) *Given a set of objects E, for all $a, b \in E$, let A, B respectively, the set of their describing features. Then the similarity between a and b is given by:*

$$sim_{tv}(a, b) = \frac{f(A \cap B)}{f(A \cap B) + \alpha \cdot f(A - B) + \beta \cdot f(B - A)}$$

As for the contrast model, the presence of weights in ratio model express the notion of asymmetry claimed by Tversky. Particularly, recalling the distinction between the subject and the referent in a directional assertion of similarity made

[7]Note that θ becomes 1 in the ratio model.

above, in the setting of the parameters, the features of the subject will be weighted more w.r.t. the features of the referent (namely $\alpha > \beta$). Consequently, similarity is reduced more by the distinctive features of the subject than by the distinctive features of the referent. If asymmetry is not desired then $\alpha = \beta = 0.5$.

Under the assumption that f is distributive over disjoint sets, namely $f(A \cup B) = f(A) + f(B)$ in the case of $A \cap B = \emptyset$, then the *ratio model* is more commonly written as[8]:

$$sim_{tv}(a, b) = \frac{2 \cdot f(A \cap B)}{f(A) + f(B)}$$

Note that, as for the contrast model, also the ratio model defines a family of measures depending of the values of the parameters.

A disadvantage of feature-based models is that two entities are seen to be similar if they have common features; however, it may be argued that the extent to which a concept possesses or is associated with a feature may be a matter of degree [122]. Consequently, a specific feature can be more important to the meaning of an entity than another. On the other hand, the consideration of common features between entity classes seems to be cognitively sensible for the way people assess similarity. Contrast and ratio models could be extended to more expressive knowledge representations by preliminarily defining what a feature is for them. However, feature-based models (as well as geometric models) are not able to capture expressive relationships among data that typically characterize most complex languages.

3.3 Similarity and Dissimilarity Measures in Relational Setting

As argued in [164], for a large number of applications it is very problematic to represent knowledge in a propositional language since having information in one table could cause loss of information and/or redundancy. From here, the necessity of working in more expressive settings. Particularly, in a relational language, multiple relations can be used for representing knowledge. Obviously, the choice of a new and more expressive language requires to define new algorithms and measures for working with it[9]. In the following, different models for determining (dis-)similarity in relational settings will be analyzed.

[8]Note that in this expression only the case of symmetric similarity is considered

[9]Chs. 4, 5 show that the choice of new and expressive languages such as DLs requires new measures for assessing similarity among objects and also new and/or modified learning methods.

3.3.1 Measures based on Semantic Relations

The measures based on semantic relations are also called *path distance measures*. They are typically applied to semantic networks [45] having a tree structure. Similarity measures in this setting usually involves measuring path lengths in the network. Distances between trees (see [165]) also belong to this category. They measure differences between trees by comparing the different paths that constitute the trees.

In the early formulations of measures based on semantic relations [163], background information was provided in the form of semantic network involving concepts and is-a edges. In this setting, semantic relatedness and semantic distance are equivalent and so it is possible to use the latter as a measure of the former. Particularly, in this context, the conceptual distance is the length of the shortest path between two nodes in the semantic network. Considering this assumption, Rada et al. [163] define the similarity between two nodes C and D (i.e. between two concepts) as the length of the path from C to E plus the length of the path from D to E, where E is the *most specific is-a ancestor (msa for brevity)*; namely, going up in the tree, E is the first ancestor that is common to C and D. Formally:

$$sim(C, D) = length(C, E) + length(D, E)$$

where $E = msa(C, D)$ and $length(A, B)$ is a function that, given two nodes A and B, returns the number of edges that link A and B. Note that, since the considered semantic network is a tree, the *msa* of two given nodes exists and is unique.

Further developments of the semantic-distance based models have been proposed by Lee et al. [126]. They argue that, in a realistic scenario, adjacent nodes are not necessarily equidistant and that a meaningful semantic-distance assessment needs to consider the underlying architecture of the hierarchical network. The result was a semantic distance model enriched by weighted indexing schema and variable edge weights. Weights can be defined w.r.t. different criteria such as the local density of the hierarchy, the depth of a node in the hierarchy, the type of the link (i.e., type of semantic relation) and the strength of an edge link (i.e., closeness between a child and its parent node). This is because, for example, the density effect suggests that the greater the density, the closer the distance between the nodes; w.r.t. the depth of a hierarchy, the distance shrinks as one descends the hierarchy, because the differences between nodes are based on finer details. Conversely, concepts in the middle to high sections of the hierarchical network, being spatially close to each other, are deemed to be conceptually similar to each other. In [109, 189] some weight are defined. They are assigned to the edges and are expressed as a function of the link strength, the depth of the node, the local density of a node, the overall density, and the type of link. Note that measures based on semantic-distance model satisfy all metric properties and are context independent (see Sect. 3.3.3).

Recently, path distance measures have been used in order to assess similarity between ontologies (in the easiest case in which an ontology is considered to be a taxonomy). This goal is accomplished considering two kinds of measures: local and global measures. The local measure compares the similarity of the positions of a concept in the two different ontologies. The global measure is computed by averaging the result of the local measure for concepts similarity. Local measures are computed by recurring to path distance approach. Specifically, in [98] the *Learning Accuracy* measure is proposed. It compares two concepts based on their distance in the tree (e.g. the length of the shortest path between the root and their most specific common abstraction). In [138], the *Balanced Distance Metric* measure is presented. It further develops the idea of the *Learning Accuracy* measure by taking into account further types of paths and a branching factor of the concepts.

Although the semantic distance model has been supported by a number of experiments, widely used in different fields such as information systems [24, 37, 44, 95], and has shown to be well suited for specific domains, it has the disadvantage of being highly sensitive to the predefined hierarchical network so to the semantic structure. Then it gives coarse values of similarity for concepts that have a same superordinate. Moreover, even if some works (for instance [109]) try to consider also more complex relations among concepts, basically the semantic distance model considers only is-a (and sometimes part-of) relations among concepts; this does not allow to exploit such model jointly with expressive language for knowledge representation.

3.3.2 Measures based on Information Content

Measures based on Information Content (*IC*) use an approach similar to those used by measures based on semantic relations (see Sect. 3.3.1), the main difference is represented by the use of the notion of *IC* rather than the notion of path length. The main motivation for purpose this approach is overcoming the problem of semantic distance-based model, namely its dependence from the structure of the network.

In the same way of measures based on semantic relations, measures based on *IC* consider knowledge represented by mean of a taxonomy of concepts where only is-a relationships are considered. The most important work based on such an approach has been proposed by Resnik [166, 167]. The main idea is that similarity between concepts can be measured with respect to the amount of information that they share. Particularly, the more information two concepts have in common, the more similar they are. The similarity between two concepts (classes) is approximated to the amount of information conveyed by the first superclass in the hierarchy that subsumes both classes. The amount of information of a concept is measured by recurring to the notion of Information Content. Following the standard argumentation of information theory [171], the IC of a concept c is measured as negative the log likelihood, namely $IC(c) = -log\ p(c)$. As the probability of occurrence of a concept in a corpus increases, the information conveyed by c decreases, such that the more abstract is a concept, the lower is its information content. Formally the similarity between concepts is defined as in the following.

Definition 3.3.1 *Let $\mathcal{C}$ be the set of concepts in an is-a taxonomy (also permitting multiple inheritance) and let $p : \mathcal{C} \to [0, 1]$ be the function that assigns, for any $c \in \mathcal{C}$, the probability $p(c)$ of finding an instance of the concept c. Given any $c_1, c_2 \in \mathcal{C}$:*

$$sim_{res}(c_1, c_2) = \max_{c \in S(c_1,c_2)} [-log\ p(c)]$$

where $S(c_1, c_2)$ is the set of concepts that subsume both c_1 and c_2.

Notice that although similarity is computed by considering all upper bounds for the two concepts, the information measure has the effect of identifying minimal upper bounds, since no concept is less informative that its superordinates. For this reason the similarity function can be also written[10] as $sim_{res} = IC(E) = -log\ p(E)$ where E is the *most specific ancestor* of c_1 and c_2: $E = msa(c_1, c_2)$. Moreover, if there is a unique top concept, its information content is 0. This is because, from the definition of IC, it is straightforward to see that the function p is monotonic as one moves up in the taxonomy, namely if c_1 is-a c_2 then $p(c_1) \leq p(c_2)$, consequently the probability value of a concept c is 1 if the taxonomy has a unique top node.

The principle underlining Resnik's measure is indirectly captured by the semantic-distance method (see Sect. 3.3.1). Indeed, if the minimal path of is-a links between two nodes is long, that means that it is necessary to go up highly in the taxonomy, to more abstract concepts, in order to find a least upper bound. For example, considering the taxonomy in Fig. 3.4, Nickel and Dime are both subsumed by Coin, whereas the most specific superconcept that Nickel and Credit Card share is Medium of Exchange[11]. This show that, measures based on IC are able to capture the same idea of semantic distance measure, but, taking into account probabilities of concepts and hence the IC of a concept, the problem of unreliability of edge distances (see Sect. 3.3.1) is avoided as no path lengths are counted.

[10]Notice that the situation is different in case of multiple inheritance, where could not exist an upper bound structurally distinguishable, so, in this case, the computation of the maximum value as reported in Def. 3.3.1 is necessary.

[11]In a feature-based setting (as proposed by Tversky in [197]) this would be reflected by explicit shared features: nickels and dimes are both small, round, metallic, and so on. These features are captured implicitly by the taxonomy in categorizing Nickel and Dimes as subordinates of Coin.

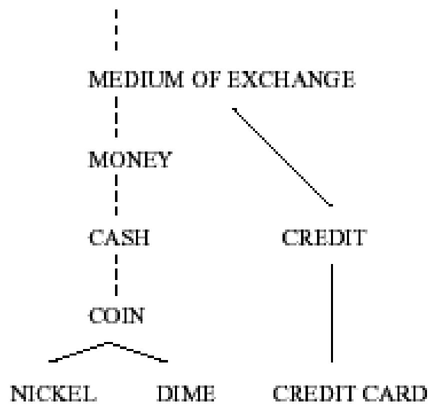

Figure 3.4: Fragment of the Wordnet taxonomy. Solid lines represent is-a links; dashed lines indicate that some intervening nodes were omitted to save space.

Considering the evolutions of Resnik's measure [167] in models based on semantic networks, taking into account factors such as concept depth and density [126, 109, 189], a simplified version of the measure proposed in [109] but dealing with IC is:

$$dist(c_1, c_2) = IC(c_1) + IC(c_2) - 2 \cdot IC(msa(c_1, c_2))$$

In [130] the following measure based on IC and able to take into account density and depth factors is also proposed:

$$sim_{lin}(c_1, c_2) = \frac{2 \cdot IC(msa(c_1, c_2))}{IC(c_1) + IC(c_2)}$$

This measure is a metric (see [130] for the demonstration).

As experimentally shown [167], measures based on IC generate better results w.r.t. measures based on semantic networks. Moreover the information-content model requires less information about the detailed structure of the network and the determination of information content can adapt a static knowledge structure to multiple contexts. Anyway, it is also true that for concepts in a hierarchy that contain an exaggerated information content, the information-content model can generate a corse result for the comparison of classes, because it does not differentiate the similarity values of any pair of classes in a sub-hierarchy as long as their "smallest common denominator" is the same. This could be, for example, the case of words in a WordNet [203] taxonomy, where many polysemous words and multi-worded classes surely have an exaggerated information content value.

3.3.3 Measures in Context-based Model

Cognitive and psychological studies have shown that the human process of assessing similarity between objects is often influenced by the context in which objects are considered. Anyway, the measures presented so far consider only the properties of the compared objects regardless of any context (the "environment" surrounding the objects) or concepts useful for characterizing object configurations, namely they are *context-free* measures. In ML, the necessity of having measures that are able to consider the context was first pointed out by conceptual clustering and successively by natural language processing.

One of the first works that introduces a *context-sensitive* similarity measure has been proposed by Gowda [92], where the similarity between two objects A and B depends not only on A and B, but also on the objects in the collection to be clustered. The similarity is expressed as the reciprocal of *mutual distance*. To determine the mutual distance from A to B, the objects in the collection are ranked according to the Euclidean distance to A and then according to the Euclidean distance to B. The mutual distance from object A to object B is the sum of the ranks of A w.r.t. B and the ranks of B w.r.t. A. Thus, the similarity between compared objects depends on their relation to the other objects[12]. Even if this measure[13] is *context-sensitve*, it is *concept-free*, that is, it depends only on the properties of individual objects and not on any external concepts which might be useful (particularly for conceptual clustering) to characterize object configurations. Thus, using this measure, for example jointly with clustering methods, does not allow to capture the "Gestalt properties" of object clusters, namely properties that characterize a cluster as a whole and are not derivable from properties of individual entities.

In order to solve this problem the intuition of Michalski and Stepp [144] was that the system must be equipped with the ability to recognize configurations of objects corresponding to certain concepts. An example of this intuition is shown in Fig. 1.7 where points are clustered with respect to the "concept" diamond. In [144] a *concept-sensitive* similarity measure between objects is presented. It is called *conceptual cohesiveness* and represents the corner stone of the conceptual clustering algorithm presented therein (see Sect. 1.3.2). The idea is that given two points A and B, the conceptual cohesiveness of A and B depends not only on those points and on surrounding points E, but also on a set of concepts C which are available for describing A and B together. Formally conceptual cohesiveness can be expressed as:

$$Conceptual\_Cohesiveness(A, B) = f(A, B, E, C)$$

[12]See [108] for an example of application of this measure.

[13]This measure is not a metric, it does not satisfy the triangle inequality. In spite of this, it has been successfully applied in several clustering applications.

To illustrate this measure, let us consider a set of points in a two-dimensional space and a set of concepts C consisting of geometrical figures (i.e. sequences of straight lines, circles, rectangles, triangles, etc.). A measure of conceptual cohesiveness could be defined, w.r.t. the considered example as: $f(A, B, E, C) = max_i\{(\#e(i) - 1)/area(i)\}$ where i indexes all geometrical figures that are specified in C and that cover points A and B, $\#e(i)$ is the total number of data points from E covers by figure i, and $area(i)$ is the area of figure i.

This measure is the most general similarity measure, it requires to define the relation between objects and concepts for every specific task. Anyway, considering its characteristics, it can be applied only for some particular tasks such as conceptual clustering. Indeed there are some situations in which no concepts are available; this consequently, does not make possible the usage of the presented measure.

In the natural language processing area, Miller and Charles [148], studying the relation between semantic similarity and contextual similarity, discuss a contextual approach to semantic similarity. A contextual representation of a word comprises syntactic, semantic, pragmatic, and stylistic conditions that affect the use of that word. Although they found that the similarity among contextual representations is one of the factors for similarity assessment among words, their work revealed a clear relationship between semantic and contextual similarity, when words belong to the same syntactic category (i.e., nouns, verbs, adjectives, or adverbs). For such words, the similarity assessment is defined in terms of the degree of substitutability of words in sentences. The more often a word can be substituted by another word in the same context, the more similar the words are. The problem with this similarity measure is that it is difficult to define a systematic way to calculate it.

3.3.4 Measures Based on Alignment and Transformational Models

Alignment-based models of (dis-)similarity have been realized in order to cope with structured descriptions. In these models, comparison is not just matching features, but determining the corresponding way of elements, or alignment way between elements. Matching features are aligned to the extent that they play similar roles within their entities. For example, a car with a green wheel and a truck with a green hood share *green* feature, but this matching feature may not increase their similarity because the car wheel does not correspond to the truck hood. Drawing inspiration from analogical reasoning [84, 101], in alignment-based models, matching features influence similarity more if they belong to parts that are placed in correspondence, and parts tend to be placed in correspondence if they have many features in common and are consistent with other emerging correspondences [85, 136].

Another empirically validated set of predictions stemming from an alignment-based approach to similarity concerns alignable and non-alignable differences [135]. Non-alignable differences between two entities are attributes of one entity that have no corresponding attribute in the other entity. Alignable differences are those that require that the elements of the entities are firstly placed in correspondence. For instance, comparing a police car to an ambulance, a non-alignable difference is that police cars have weapons in them, but ambulances do not. There is no clear equivalent of weapons in the ambulance. Alignable differences include the following: police cars carry criminals to jails rather than carrying sick people to hospitals, a police car is a car while ambulances are vans, and police car drivers are policemen rather than emergency medical technicians. Consistently with the role of structural alignment in similarity comparisons, alignable differences influence similarity more than non-alignable differences [137]. Knowing these correspondences affects not only how much a matching element increases similarity but also how much a mismatching element decreases similarity.

Transformational models are based on the assumption that the comparison process proceeds by transforming one representation into the other. A critical step for these models is to specify what transformational operations are possible. In Artificial Intelligence, Ullman [198] has argued that objects are recognized by being aligned with memorized pictorial descriptions. Once an unknown object has been aligned with all candidate models, the best match to the viewed object is selected. The alignment actions considered are rotate, scale, translate, and topographically warp object descriptions. For rigid transformations, full alignment can be obtained by aligning three points on the object with three points on the model description.

In transformational accounts that are explicitly designed to model data similarity, similarity is usually defined in terms of transformational distance. Particularly, similarity is often assumed to decrease monotonically as the number of transformations required to make one sequence identical to the other increases. An example of measure based on transformational model is the Levenshtein's edit distance [128] which computes the similarity between two strings as the number of deletions, insertions, or substitutions required to transform the one into the other. By the use of this measure Maedche and Staab propose [133] a syntactic similarity measure for strings defined as follows:

Definition 3.3.2 *Given two strings L_i and L_j in $\mathcal{L}$, the similarity between L_i and L_j is given by the function: $s : \mathcal{L} \times \mathcal{L} \to [0, 1]$ such that $\forall L_i, L_j \in \mathcal{L}$:*

$$s(L_i, L_j) = max(0, \frac{min(\mid L_i \mid, \mid L_i \mid) - ed(L_i, L_j)}{min(\mid L_i \mid, \mid L_i \mid)})$$

where $\mid \cdot \mid$ stands for the length of L_i and $ed(\cdot, \cdot)$ is the Levenshtein edit distance.

This measure returns a degree of similarity between 0 and 1, where 1 stands for perfect matching and 0 for bad match. It considers the number of changes that must be made to transform one string into the other and weights the number of these changes against the length of the shortest string of these two. Moreover, Maedche and Staab [133] propose a way for determining lexical similarity between two different ontologies, by computing the averaged string matching of the overall concept names contained in them.

It is important to note that both alignment-based and transformational approaches place elements into correspondence. Whereas the correspondences are explicitly stated in the structural alignment method, they are implicit in transformational alignment. The transformational account often produces globally consistent correspondences, for example correspondences that obey to a one-to-one mapping principle, but this consistency is a consequence of applying a pattern-wide transformation and is not enforced by interactions between emerging correspondences. For large data sets characterized by complex representations, both approaches could be computationally expensive.

3.3.5 Miscellaneous Approaches

Considering advantages and drawbacks of the presented models, many works propose a miscellaneous approach to take advantage from the usage of several mixed models. In the following, some examples of miscellaneous approaches will be presented.

Propositionalization and Geometrical Models

Geometrical models are largely used particularly for their efficiency (see Sect. 3.2.1). Anyway they can be applied only to propositional representations. In order to exploit the efficiency of geometrical models also when relational representations are considered, the propositionalization problem has been focused. Such a problem consists in finding a way, namely a mapping, for transforming a multi-relational representation into a propositional representation. Once that a propositional representation has been obtained, any method can be applied on this one rather than on the original representation. For instance, it is possible to compute (dis-)similarities between objects by the use of geometrical models applied to the obtained propositional representation, hence results are referred to the original problem.

Hypotheses-driven distances

One of the most interesting works based on propositionalization and usage of geometrical models, has been proposed by Sebag [182] where a method for building a distance on first-order logic representation by recurring to the propositionalization is presented; the measure is named *hypoteses-driven distance* (HDD). The distance is formalized starting from examples expressed as definite or constrained clauses. Let $\mathcal{L}_h$ denote the language of hypotheses, and let $\mathcal{H} = \{h_1, \dots, h_d\}$ be a set of d hypotheses. $\mathcal{H}$ induces a mapping π from $\mathcal{L}_h$ onto the boolean space of dimension d [183], by associating to each example E the vector of the boolean coding wether E is subsumed by h_i, namely $\pi : \mathcal{L}_h \to \{0,1\}^d$ and for all example $E \to \pi(E) = (\pi_1(E), \dots, \pi_d(E))$ where $\pi_i(E) = 1$ if h_i subsumes E, $\pi_i(E) = 0$ otherwise. The projection onto $\{0,1\}^d$ does not make any assumption on $\mathcal{L}_h$ besides $\mathcal{H}$, it only invokes the covering test. Considering that $\{0,1\}^d$ is a metric space, a distance on $\mathcal{L}_h$ naturally follows, by setting, for all examples E and F:

$$dist(E,F) = \sum_{i=1}^{d} \mid \pi_i(E) - \pi_i(F) \mid$$

By construction, *dist* is symmetric and satisfies the triangular inequality, but it does not satisfy the identity relation, namely $(dist(E,F) = 0) \not\Rightarrow (E = F)$. So *dist* is a semi-distance measure. Another projection onto a richer metric space is presented and the corresponding HDD is defined as the Euclidean distance [182].

It is important to note that HDDs are not so useful whenever they are based on a concise set of hypotheses $\mathcal{H}$: e.g. *dist* gets rather coarse if each example is covered by a single hypothesis. Hence, the granularity of a HDD increases with the redundancy of $\mathcal{H}$ (i.e. the average number of h_i covering each example) and more precisely with the number and diversity of hypothesis h_i. Moreover, a HDD does not involve in any way the conclusions associated to hypothesis h_i; this suggest that the relevance of a HDD is potentially independent from the relevance of $\mathcal{H}$. Furthermore, the structure of the boolean space does not reflect the structure of the problem domain. A hypothesis h_i usually covers less than half the problem space: having $\pi_i(E) = 1$ is thus less frequent than having $\pi_i(E) = 0$, whilst 1 and 0 play equivalent roles in the boolean space. So, mainly these measures, rather than syntactically comparing two examples, analyzes the way in which the examples semantically behave with respect to a set of hypotheses.

Mixing Path Distance and Feature Matching Approaches

As seen in Sect. 3.2.3, the feature-matching model is often able to capture similarity between object in a satisfactory way. Anyway, such model can be applied only to propositional representations. More complex representations can be considered only by previously defining what a feature is in the new context. On the other hand, as illustrated in Sect. 3.3.1, measures based on semantic relations are able to take into account relationships among objects (at least is-a links) but they strongly depends from the structure defining object links. Some works tried to improve the assessment of similarity of objects by mixing these two models.

In [169], an asymmetric *Matching Distance* (MD) measure was presented with the goal of providing a similarity measure between concepts within a single ontology. Concepts are represented by means of synonym sets, characterizing the set of features for a given concept and that are interrelated by hyponymy (is-a) and meronymy (part-whole) relations. Similarity is evaluated by combining a feature-matching process with a semantic distance measurement. Particularly, similarity is expressed as the number of common and different features between two entity-classes where the relevance of the different features is given in terms of the distance among entities in a hierarchical structure. This represents one of the first works able to treat also part-whole relationships, besides of is-a relationship. Anyway, considering these relationships is not enough in case of knowledge bases described by expressive languages. Moreover, since the *Matching Distance* measure is based on the comparison of distinguishing features, the lack of distinguishing features in an entity class definition produces a null similarity value w.r.t. any other entity class in the ontology. This is a common situation for entity classes that are general concepts located at the top level of the hierarchical structure. Although this can be seen as a drawback of the MD measure, the model strength is the capability to assess the similarity among concepts located at or below Rosch's basic level of a hierarchical structure [170], such as the concepts found in spatial catalogs. This characteristic of the MD model is in contrast to previous models based on semantic distance [163]. Indeed, while semantic distance can determine similarity among general concepts of a hierarchical structure, it usually assigns the same value to any pair of entity classes that have a common superclass.

Another interesting example of similarity measure defined by jointly using feature-matching and path distance models has been presented in [133]. Here, a measure for determining the taxonomic overlapping between two hierarchical ontologies is illustrated. First of all, the notion of *semantic cotopy* (SC) of a concept C (w.r.t. an ontology $\mathcal{O}$ where it is defined) is introduced. The SC is given by the set of all direct super and sub-concepts of C in $\mathcal{O}$. Given the semantic cotopy of the concept C in the ontology $\mathcal{O}_1$ and the semantic cotopy of C in the ontology $\mathcal{O}_2$, the taxonomic overlapping of $\mathcal{O}_1$ and $\mathcal{O}_2$ with respect to C is computed as the ratio between the intersection of the SC of C w.r.t. $\mathcal{O}_1$ and the SC of C w.r.t. $\mathcal{O}_2$ and the union of such computed SC. So, mainly the SC is determined by recurring to the path distance model while the taxonomic overlap is computed by recurring to the feature matching model. Then, the overall taxonomic overlapping between $\mathcal{O}_1$ and $\mathcal{O}_2$ is given by the averaged taxonomic overlapping computed w.r.t. to all concepts the ontologies. A modified version of such measure has been proposed in [62], where the *common semantic cotopy* rather than the semantic cotopy is considered.

Mixing Feature Matching, Context-based and Information Content-based Approaches

Experiences in determining measures that use jointly feature matching and path-distance approaches demonstrated experimentally interesting results (see [169, 133]). Anyway, they sometimes suffer from the drawback of the path distance approach that is the strong dependence on the hierarchical structure representing the KB (see Sect. 3.3.1). In order to overcome this problem, the Information Content based approach could be used (see Sect. 3.3.2). Moreover, it is shown that, in many fields such as natural language processing and ML, the availability of measures able to take into account contextual information, could improve the effectiveness of several tasks such as clustering and words disambiguation. Moving from these observations, Rodriguez [169, 168] proposed a measure for determining similarity of concepts within an ontology that mixes feature matching and information content based approaches and is able to take into account contextual information.

Concepts are represented by means of a set of features given by their synonym sets. Concepts are then interrelated by means of hyponymy (is-a) and meronymy (part-whole) relations. User queries are considered, representing context specifications that define the application domain. The application domain helps to select among senses of a term with multiple meanings (i.e. polysemous terms), thus reducing the problem of word-sense ambiguity, since only these entity classes are considered in the similarity assessment. Similarity is expressed as the number of common and different features between two concepts (also called entity-classes). Features are weighted w.r.t. their relevance in the domain. Feature relevance is determined by recurring to the notion of information content used for measuring the informativeness of a feature. Namely, if a feature is shared by all entity classes of the domain, its relevance decreases. On the contrary, if a feature characterizes only some entity classes of the domain, its relevance increases. So, the feature weights are determined by analyzing the variability of distinguishing features within the application domain. Consequently distinguishing features that present great variability are more important in the similarity assessment than features that do not contribute significantly to distinguishing entity classes. The experimental evaluation demonstrated that this measure really improves the previous measure proposed by Rodriguez [169] based on feature matching and path distance approaches. Anyway, when the application domain has the maximum variability, that is no feature is shared by entity classes or only one entity class is part of the application domain, the relevance is equally assigned to the features. Similar result occurs without variability. Moreover, this measure, and particularly the method for weighting features, shows sensitivity to the set of entity classes defined in the ontology. This sensitivity becomes more important for a narrow application domain, where the omission on one entity class may affect the determination of common and different features of the domain.

3.3.6 Relational Kernel Functions

Kernel methods have been revealed very efficient in solving many ML tasks such as classification, clustering and regression. The key element of the kernel methods is given by the kernel functions that are defined on any set and are able to perform an implicit mapping of the feature data into a high dimensional feature space where a learning task can be performed exploiting a linear function (see Sect. 3.2.2). Specifically, in the new feature space, the computation of the kernel function corresponds to compute the inner product in this space. In the first formulation, kernel functions have been defined for propositional representation (see Sect. 3.2.2). Nevertheless, as argued in Sect. 3.3, in a variety of AI problems there is a need to learn, represent and reason w.r.t. definitions over structured and relational data. The development of kernels functions for structured data have been motivated by the necessity of solving real-world problems in an efficient way.

The best-known kernel for representation spaces that are not mere attribute-value tuples is the **convolution kernel** proposed by Haussler [99]. The basic idea of convolution kernels is that the semantics of a composite object can often be captured by a relation R between the object and its parts. The kernel on such objects is composed of kernels defined on different parts. Specifically, convolution kernels are obtained by composing other existing kernels by a certain sum over products, exploiting the closure properties of the class of positive definite functions.

Particularly, the class of kernels on a set $\mathcal{X} \times \mathcal{X}$ is closed under addition, multiplication by a positive constant, pointwise limits and product (i.e. if $k_1(x, y)$ and $k_2(x, y)$ are kernels then $k(x, y) = k_1(x, y)k_2(x, y)$ is a kernel). Moreover, since kernels are closed under product, they are also closed under tensor product[14]. Similarly, since kernels are closed under sum, they are also closed under direct sum [15]. Furthermore, if $k((x, u), (y, v))$ is a kernel on $(\mathcal{X} \times \mathcal{X}) \times (\mathcal{X} \times \mathcal{X})$ then the *diagonal projection* $k^{\triangle}(x, y) = k((x, x), (y, y))$ is a kernel on $\mathcal{X} \times \mathcal{X}$. Lastly, if $S \subseteq \mathcal{X}$ and k is a kernel on $S \times S$, then k may be extended to a kernel on $\mathcal{X} \times \mathcal{X}$ by defining $k(x, y) = 0$ if either x or y is not in S. This follows directly from the definition of positive definite function. This is called *zero extension of* k (for more details about the closure properties of kernels see [99, 21]). In the following, convolution kernel is formally defined.

Let X be a domain (set), let $x \in X$ be a composite structure and $x_1, \ldots, x_D$ are its "parts", where x_d is in the set X_d for each $1 \leq d \leq D$, and D is a positive integer. $X_1, \ldots, X_D$ are non empty, separable metric spaces. The relation "$x_1, \ldots, x_d$ are parts of x" can be represented by a relation R on the set $X_1 \times \cdots \times X_D \times X$,

[14] i.e. if $k_1(x, y)$ is a kernel on $\mathcal{X} \times \mathcal{X}$ and $k_2(u, v)$ is a kernel on $U \times U$ then $k_1 \otimes k_2((x, u), (y, v)) = k_1(x, y)k_2(u, v)$ is a kernel on $(\mathcal{X} \times U) \times (\mathcal{X} \times U)$.

[15] $k_1 \oplus k_2((x, u), (y, v)) = k_1(x, y) + k_2(u, v)$ is a kernel on $(\mathcal{X} \times U) \times (\mathcal{X} \times U)$.

where $R(x_1, \ldots, x_D, x)$ is true iff $x_1, \ldots, x_D$ are the parts of x. For brevity, let $\overrightarrow{x} = x_1, \ldots, x_D$, and denote $R(x_1, \ldots, x_D, x)$ by $R(\overrightarrow{x}, x)$. R is said finite if $R^{-1}(x)$ is finite for all $x \in X$.

Now suppose that $x, y \in X$ and for some decomposition of x and y, $\overrightarrow{x} = x_1, \ldots, x_D$ are the parts of x, and $\overrightarrow{y} = y_1, \ldots, y_D$ are the parts of y. Suppose also that, for each $1 \leq d \leq D$, a kernel k_d in X_d is available in order to measure the similarity $k_d(x_d, y_d)$ between the part x_d and the part y_d. Then, the similarity $k(x, y)$ between x and y is defined as the following generalized convolution:

$$k(x,y) = \sum_{\overrightarrow{x} \in R^{-1}(x), \overrightarrow{y} \in R^{-1}(y)} \prod_{d=1}^{D} k_d(x_d, y_d) \tag{3.1}$$

This defines a symmetric function on $S \times S$, where $S = \{x : R^{-1}(x) \textit{ is not empty}\}$. Hence the *R-convolution* of $k_1, \ldots, k_D$ is defined. It is denoted by $k_1 \star \cdots \star k_D(x, y)$ and represents the zero order extension of k to $X \times X$. k is called finite convolution if R is finite.

Theorem 3.3.3 *If $k_1, \ldots, k_D$ are kernels on $X_1 \times X_1, \ldots, X_D \times X_D$, respectively, and R is a finite relation on $X_1 \times \cdots \times X_D \times X$, then $k_1 \star \cdots \star k_D$ is a kernel on $X \times X$.*

Lemma 3.3.4 *Let k be a kernel on a set $U \times U$ and for all finite, nonempty $A, B \subseteq U$, define $k'(A, B) = \sum_{x \in A, y \in B} k(x, y)$. Then k' is a kernel on the product of the set of all finite, nonempty subsets of U with itself.*

The presented results[16] ensures that the function defined in (3.1) is a kernel on X. Hence, the term "convolution kernel" refers to a class of kernels that can be formulated as shown in (3.1). This formalization defines a powerful method for defining kernel for structured data, indeed the advantage of convolution kernels is that they are very general and can be applied in several situations. Even if, because of their generality, they require a significant amount of work to adapt them to a specific problem, which makes choosing R in real-world applications a non-trivial task.

Another important result is the relation between kernel functions and distances satisfying the metric conditions [99].

[16]For the proof of the results see [99].

Definition 3.3.5 (Distances induced by kernels) *Let $k : X \times X \to \mathbb{R}$ be a kernel on X. The distance measure induced by k is defined as:*

$$d_k(x, y) = \sqrt{k(x, x) - 2k(x, y) + k(y, y)}$$

If k is a valid kernel (see Sect. 3.2.2) then d_k is well-behaved in that it satisfies the conditions of a pseudo-metric: (1) $d(x, z) \leq d(x, y) + d(y, z)$ (triangle inequality); (2) $d(x, y) = d(y, x)$ (symmetry); (3) $x = y \Rightarrow d(x, y) = 0$. (For a metric this latter implication must be an equivalence).

This relation is practically important as it extends the applicability of kernel functions to distance-based methods such as k-nearest neighbour.

Moving from the notion of convolution kernel, many kernel functions have been defined for different type of structured data. The simplest structured representation considered is the string structure. The traditional kernel function used for text classification is simply the scalar product of two texts in their bag-of-words representation [112], this kernel function does not take the structure of the text, or the words of the text into account but only the number of times each word occurs.

More sophisticated approaches (based on convolution kernels) try to define a kernel function on the sequence of characters. The idea behind this kernel is to base the similarity of two strings on the number of common subsequences. The subsequences can be also not continuous in the strings, because the more gaps in the occurrence of the sequence, the less weight is assigned in the kernel function [132, 205]. An alternative to this kernel has been presented in [158, 127], where only continuous substrings of a given string are considered. Applications and empirical results regarding these kernel functions can be found in [127, 132] (for a detailed survey about string kernel functions see [79]).

Kernel functions on tree structures exploiting convolution kernels have also been formulated. In [46], a kernel function on tree structures (that can be applied in many natural language processing tasks) is presented. Here, the instances of the learning tasks are considered to be labeled ordered directed trees. The key idea to capture structural information about the trees in the kernel function is to consider all subtrees occurring in a parse tree. A subtree is defined as a connected subgraph of a tree such that either all children or no child of a vertex is in the subgraph. The children of a vertex are the vertices that can be reached from the vertex by traversing one direct edge. Thee kernel function is the inner product in the space which describes the number of occurrences of all possible subtrees.

A generalization of this kernel to take into account also other substructures of the trees is described in [115]. Another generalization considered in this paper allows labels to partially match. Promising results have been achieved with this kernel function in HTML document classification tasks[17].

Labeled graph are widely used in computer science to model data. Graph kernels, exploiting convolution kernel, have been presented [78, 81, 114, 116]. Conceptually, they are based on a measure of the walks in two graphs that have some or all labels in common. In [78] walks with equal initial and terminal label are counted, in [114, 116] the probability of random walks with equal label sequences is computed, and in [81] walks with equal label sequences, possibly containing gaps, are counted. One interesting application of such graph kernels have been shown in [80], where an experiment in a relational reinforcement learning setting is presented[18].

A framework that allows for the application of kernel methods to different kinds of structured data has been presented [82]. The approach is based on the idea of availability of a representation that allows for modeling the semantics of an object by means of the syntax of the representation. The underlying principle is representing individuals as (closed) terms in a typed higher-order logic [131]. The individuals-as-terms representation is a natural generalization of the attribute-value representation and collects all information about an individual in a single term. The key idea is to have a fixed type structure. This type structure expresses the semantics and composition of individuals from their parts. The type structure is made up by function types, product types, and type constructors. Function types are used to represent types corresponding to sets, multisets, and so on. Product types are used to represent types corresponding to fixed size tuples. Type constructors are used to represent types corresponding to arbitrary size structured objects such as lists, trees, and so on. The set of type constructors also contains types corresponding to symbols and numbers. It is important to note that the elements of these sets, lists, etc. are not restricted to be mere symbols but can again be structured objects. Each type defines a set of terms that represent instances of that type, these are called the basic terms. The terms of the logic are the terms of the typed λ-calculus, which are formed in the usual way by abstraction, tupling, and application. The basic term kernel is then defined inductively on the structure of terms. Particularly, for each possible part of a term, an existing kernel function is considered. Hence, the basic term kernel is defined, analogously to convolution kernel, as product and sum of the kernel functions considered for every part of a term.

This framework constitutes an interesting result for defining kernel functions that are applicable to very expressive representation language and that are able to exploit also semantics. Anyway, the expressiveness of the used language is far from the expressiveness of very powerful representation language such as DLs.

[17]For a detailed survey about kernel functions applied on tree structures see [79].

[18]For a detailed survey on graph kernels see [79].

3.3.7 Measures for Simple Description Logics

As seen in the previous sections, the idea of measuring concept similarity has received considerable attention in several domains, such as psychology, cognitive science, computational linguistics and information retrieval. Recently, also in the field of Information Integration it is increasing the necessity of having measure able to determine the similarity between concepts. Such field often relies on ontologies and hence concepts are described in DLs (see Sect. 1.1.1). Most past work has concentrated on the similarity of "atomic" concepts (see previous section), rather than composite, defined concepts, which are the stock-in-trade of DLs.

One of the first works that focuses on the problem of assessing similarity between complex definite concepts expressed in DL has been proposed by Borgida et al. [29]. Its goal is to generalize previous efforts in defining similarity for primitive concepts in order to obtain a way for assessing similarity between complex concept descriptions. For reaching this goal a very simple DL, allowing only concept conjunction (such logic is called $\mathcal{A}$), is considered. Hence, the main measures presented in Sects. 3.2.3, 3.3.1, 3.3.2 are modified in a way such that they can be applied to $\mathcal{A}$ concept definitions. Particularly, w.r.t. the feature matching model (see Sect.3.2.3), features are represented by atomic concepts in this setting. An ordinary concept is just the conjunction of its features. Considering that set intersection and difference of the atom sets corresponds, at least in this simple case, to the *least common subsumer* and *concept difference* in $\mathcal{A}$ (see Sect. 2.4.1), the Tversky's measures can be translated into DL notation as follows:

Definition 3.3.6 (Contrast Model) *Given a set of concept definitions $\mathcal{L}$, for all $C, D \in \mathcal{L}$, the similarity between C and D is given by:*

$$constrast_{tv}(C, D) = \theta \cdot f(lcs(C, D) - \alpha \cdot f(dif(C, D)) - \beta \cdot f(dif(D, C))$$

where θ, α, β are non-negative constants, $f(\cdot)$ counts the features in the set.

Definition 3.3.7 (Ratio Model) *Given a set of concept definitions $\mathcal{L}$, for all $C, D \in \mathcal{L}$ the similarity between C and D is given by:*

$$sim_{tv}(C, D) = \frac{2 \cdot f(lcs(C, D))}{2 \cdot f(lcs(C, D)) + f(dif(C, D)) + f(dif(D, C))}$$

where $f(\cdot)$ is taken as the count of features in the set.

In case of a semantic network model (as presented in Sect.3.3.1), if the network is a tree, then it is possible to consider $\mid C \mid$ that represents the length of the path from C to the root ($\top$). Hence, given the concepts C and D in the network, it is possible to consider the paths from C and D to the root. These paths first intersect at the most specific ancestor $msa(C, D)$ which, is the same as the $lcs(C, D)$. Hence, the Rada's distance applied to concepts definition in $\mathcal{A}$ logic can be written as:

$$dist_{rada}(C, D) = \mid C \mid + \mid D \mid - 2 \cdot \mid lcs(C, D) \mid$$

For the IC models (see Sect. 3.3.2), it is again possible to note the parallel between $msa(C, D)$ in a semantic network and $lcs(C, D)$ in DL. So translating IC measures to $\mathcal{A}$ logic yields:

$$dist(C, D) = IC(C) + IC(D) - 2 \cdot IC(lcs(C, D))$$

$$sim_{lin}(C, D) = \frac{2 \cdot IC(lcs(C, D))}{IC(C) + IC(D)}$$

Considering that, the DLs standardly used in the applications are much more expressive than $\mathcal{A}$ logic, Borgida et al. [29] try to generalize the presented reformulation to a complex DL. The first results of these efforts converge to a set of open questions, for each analyzed approach, that have to be solved in order to define measures for complex DL.

In feature based models, as highlighted in Sect.3.2.3, the key issues are what counts as a feature, and what valid decompositions into features are. In a propositional DL (like $\mathcal{A}$), one might take the minimal disjunctive normal form, and count literals; but it is much less clear what to do with terms constructed using roles. For example, if $(\leq 3R)$, $(\leq 4R)$ and $(\leq 9R)$ each count as a single feature, there is no way to express that the first and second would be judged to be more similar than the second and the third. Much more complicated is to assess similarity in presence of nested role restrictions such as $\forall R.(\forall R.A)$ vs. $\forall R.A$. In this case more information is required concerning the salience of roles, and/or how to combine such measures in the cases of enumeration and nesting to produce a legitimate measure of feature set size'. Similarly, in network-based measures a key problem is that of assigning a useful size for the various concepts in the description. Arbitrariness in choosing a size measure for complex concepts could be a substantial obstacle to this approach. Once again, some mechanism for measuring the size besides what is available in the pure structural form is necessary. The use of IC based approach could overcome the problem of the network based approach but, on the other side, this approach requires a way for determining the probability $p(C)$ of an object being an instance of an arbitrary concept C in order to define the information content of a concept.

In the next chapter, some measures applicable to expressive DL (mainly $\mathcal{ALC}$ logic) are proposed. They are able to solve some of the issues illustrated above. Based on this approach, a recent work [110] defines a new measure for $\mathcal{ALCNR}$ logic that is applicable in Geo-Spatial context.

Chapter 4

Similarity and Dissimilarity Measures for Description Logics

The application of learning methods to a knowledge base most of the times requires the availability of (dis-)similarity measures to assess the (dis-)similarity of the considered elements. To this purpose, measures are strictly related to the adopted representation language, since they need to capture all the expressiveness of the language in order to evaluate similarity.

As seen in Ch. 3, a lot of work has been done in determining (dis-)similarity measures by the use of various representation languages. Anyway most of the work has been dedicated to propositional representations. However, most of the things that can be done efficiently in a propositional setting, cannot be done efficiently in relational setting. Hence, new measures for relational settings have been often formalized. Main models for computing (dis-)similarity measures in relational settings have been shown in Sect. 3.3, anyway, most of the measures "implementing" such models are not able to exploit the high expressiveness of DLs, because often measure definitions are strictly connected with the representation language. Some initial works for defining measures able to cope with DL representations have been done (see Sect. 3.3.7), anyway they refer to very low expressive DLs. Moreover, such works do not deal with the problem of assessing (dis-)similarity between individuals, only concepts are considered.

In this chapter a set of developed similarity and dissimilarity measures for DLs is presented. They are able to compute (dis-)similarity between complex concept definitions, between individuals and concept and individual, asserted in the same ontology. Mainly $\mathcal{ALC}$ description logic[1] has been taken into account, since it is considered a good compromise between expressiveness and consequent computational

[1]See Sect. 2.2 for more details about $\mathcal{ALC}$ logic.

complexity. Moreover, since numeric representation are almost always treated in real world application, a measure for $\mathcal{ALN}$ logic[2] is also presented. At the end of the section, a measure that is able to deal with whatever DLs is defined.

All the measures that will be presented are symmetric[3]. Indeed, Tversky [197] argues that asymmetric property is necessary when, searching the similarity between elements, one element is considered more important than the one to which the element is compared. On the contrary, in the considered scenario, learning tasks are applied to ontological knowledge where no information is available about the importance of the considered elements. So, in this scenario, it is fundamental that the considered measures satisfy the symmetric property, since the absence of this property could generate different results with respect to the order in which the elements are considered. Moreover, most of the presented measures do not satisfy triangle inequality. This is because, as argued by Jain et al. [108], triangle inequality is not fundamental in order to guarantee the fine work of a measure. In the following all the measures will be presented in detail, jointly with the properties they satisfy.

4.1 A Semantic Similarity Measure for $\mathcal{ALC}$

As argued in Ch. 2, 3, a merely syntactic approach for determining the (dis-)similarity value for elements described by means of expressive DLs is too weak, because it is not able to exploit the expressiveness of the language. So a different approach (based on semantics) has to be taken into account.

Considering semantics in DLs means to deal with concept and role extensions (besides of the available inference operators). On the other hand, the main idea of Tversky's *contrast model* (see Sect.3.2.3) is shared in this thesis. Namely:

- common features increase the perceived similarity of two concepts
- feature differences diminish the perceived similarity
- feature commonalities increase the perceived similarity more than feature differences can diminish it

This intuition could be applied to a concept in an ontology, where its meaning is given by its extension. Thus, two concept definitions in an ontology are more similar as higher is the amount of the extension they share. Their differences are given by

[2]See Sect. 2.2 for more details about $\mathcal{ALN}$ logic.

[3]See Sect. 3.1 for the properties of a (dis-)similarity measure and the opportunity that a measure satisfies all or some of these properties.

the parts of the extensions that are not in common. Hence, in this vision, the "features" of a concept are represented by the individuals in its extension. Because, *commonalities increase the perceived similarity more than feature differences can diminish it*, in order to accomplish to this principle, a weight that increases the amount of common extensions could be set.

Moving form this intuition, a measure for assessing concepts similarity is defined. This measure employs the basic set theory and is mainly grounded on concept commonalities. Particularly, the base criterion for this measure is: *the similarity between concepts is not only the result of the common features, but also the result of the different characteristics.*

Let C and D be two concept descriptions in a TBox $\mathcal{T}$, expressed in the $\mathcal{ALC}$ logic. As seen in Sect. 2.3.2, given a concept C in $\mathcal{T}$, it is possible to consider its extension $C^{\mathcal{I}}$. Here the canonical interpretation of the ABox is considered and the *unique names assumption* (UNA) is made, namely: constants in the ABox are interpreted as themselves and different names for individuals stand for different domain objects (*canonical interpretation*[4]). The semantic similarity measure [53] is defined as in the following:

Definition 4.1.1 (Semantic Similarity Measure) *Let $\mathcal{L}$ be the set of all concepts in $\mathcal{ALC}$ and let $\mathcal{A}$ be an ABox with canonical interpretation $\mathcal{I}$. The* Semantic Similarity Measure *s is a function $s : \mathcal{L} \times \mathcal{L} \mapsto [0, 1]$ defined as:*

$$s(C, D) = \frac{|I^{\mathcal{I}}|}{|C^{\mathcal{I}}| + |D^{\mathcal{I}}| - |I^{\mathcal{I}}|} \cdot \max(\frac{|I^{\mathcal{I}}|}{|C^{\mathcal{I}}|}, \frac{|I^{\mathcal{I}}|}{|D^{\mathcal{I}}|}) \tag{4.1}$$

where $I = C \sqcap D$ and $(\cdot)^{\mathcal{I}}$ stands for the concept extension wrt the interpretation $\mathcal{I}$.

The presented measure assigns the maximum value 1 if the two input concepts C and D are equivalent (namely if $C \sqsubseteq D$ and $D \sqsubseteq C$), while it assigns the minimum similarity value 0 if the considered concepts are totally disjoint (namely if $C \sqcap D \equiv \bot$), because in this case they are totally different, indeed they are semantically unrelated, being their intersection equivalent to the bottom concept and their extensions do not overlap. In case of overlapping concepts, a value in the range $]0, 1[$ is computed. This value expresses the similarity between the two concepts (given by the factor $|I^{\mathcal{I}}|/(|C^{\mathcal{I}}|+|D^{\mathcal{I}}|-|I^{\mathcal{I}}|)$) reduced by a quantity ($\max(|I^{\mathcal{I}}|/|C^{\mathcal{I}}|, |I^{\mathcal{I}}|/|D^{\mathcal{I}}|)$) which represents the major incidence of the intersection with respect to either concept. Indeed, the higher such factor is the more one of the concepts is likely to be subsumed by the other.

[4]See [8] for more details about the canonical interpretation.

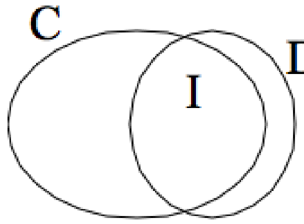

Figure 4.1: Geometrical interpretation of similarity between concepts C and D. Similarity will be high because D is almost near to be subsumed by C.

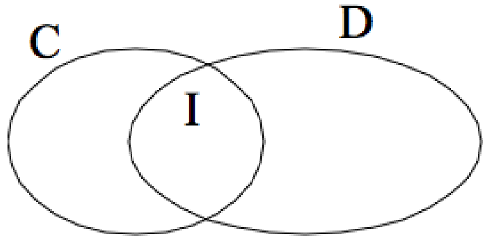

Figure 4.2: Geometrical interpretation of similarity between concepts C and D. Similarity will be not so high because they are differentiated by many "features" (namely individuals not shared as concept instances).

Hence the presented measure assesses similarity between concepts considering similarity not as an absolute value (namely considering only the amount of the overlapping extension) but as weighted with respect to a degree of non-similarity. A geometrical interpretation of the measure is shown in Fig. 4.1 and Fig. 4.2. Circles represent the extension of the concepts C and D, I represents the extension shared by the concepts, it is the same both for Fig. 4.1 and Fig. 4.2. Anyway, in case of Fig. 4.1, the similarity of C and D will be higher than the similarity computed in Fig. 4.2. This is because in the first case differences between C and D (namely the parts of the extensions outside of I) are less than differences in Fig. 4.2.

In case of an incomplete knowledge base, the measure could suffer from the lack of information. For example a concept could have a few individuals in its extension that are almost all shared by another concept, hence they result to be very similar, instead, really (w.r.t. the considered domain) they are not so similar. In order to minimize such an effect, a "normalization" factor could be considered, for instance by dividing by the cardinality of set of individuals in the ABox. Anyway, it has to be considered that the illustrated drawback depends from an inadequate state of the knowledge base rather than from the measure itself.

It is important to note that this measure is totally semantic. Indeed, it uses only semantic inferences like instance checking (and retrieval) to determine the concept extensions, and it does not make use of the syntactic structure of the concept description. Hence it is independent from the granularity level of descriptions. This fact reflects the intrinsic complexity of expressive DL languages like $\mathcal{ALC}$ for which a merely structural approach to reasoning is ineffective (subsumption is computed using a tableaux algorithm rather than a structural algorithm). Therefore, the measure definition employs set theory and semantic services, so it make use of numeric approach despite its application on a symbolic DL representation. Moreover it can be applied to knowledge base written in $\mathcal{ALC}$ logic as well as to any DL endowed with the basic reasoning services required by its definitions.

Below it is proved that the function presented in Def. 4.1.1 is really a similarity measure. Then, an example of its application is given. Hence, the extension of the measure for computing the similarity between individuals will be illustrated. At the end of the section, complexity issues related to the computation of the measure will be discussed.

Recalling Def. 3.1.1 in Sect. 3.1, a function s is a similarity measure if the following three properties are satisfied:

1. s is definite positive
2. s is symmetric
3. s satisfies the minimality property, namely: $\forall C, D \ : \ s(C, D) \leq s(C, C)$

The first property is trivially satisfied by the function definition 4.1, as it has values in the real interval $[0, 1]$. Then, it is easy to prove the minimality property, simply substituting the concept C to D in the definition and computing $s(C, C)$. The symmetry property is also trivially verified. Indeed set intersection, sum, product and maximum are commutative. It is straightforward to note that given two concepts C and D, it holds that:

$$\begin{aligned} s(C,D) = \ & \tfrac{|I^{\mathcal{I}}|}{|C^{\mathcal{I}}|+|D^{\mathcal{I}}|-|I^{\mathcal{I}}|} \cdot \max\left(\tfrac{|I^{\mathcal{I}}|}{|C^{\mathcal{I}}|}, \tfrac{|I^{\mathcal{I}}|}{|D^{\mathcal{I}}|}\right) = \\ & \tfrac{|I^{\mathcal{I}}|}{|D^{\mathcal{I}}|+|C^{\mathcal{I}}|-|I^{\mathcal{I}}|} \cdot \max\left(\tfrac{|I^{\mathcal{I}}|}{|D^{\mathcal{I}}|}, \tfrac{|I^{\mathcal{I}}|}{|C^{\mathcal{I}}|}\right) = \ s(D,C) \end{aligned}$$

note that I remains the same because of the commutativity of the intersection.
In order to clarify Def. 4.1.1, the following example is presented.

Example 4.1.2 *Let be consider the knowledge base with the* TBox *and* ABox *reported below.*

Primitive Concepts: $N_C = \{\mathsf{Female}, \ \mathsf{Male}, \ \mathsf{Human}\}$.
Primitive Roles: $N_R = \{\mathsf{HasChild}, \ \mathsf{HasParent}, \ \mathsf{HasGrandParent}, \ \mathsf{HasUncle}\}$.

TBox: $\mathcal{T} = \{$ $\mathsf{Woman} \equiv \mathsf{Human} \sqcap \mathsf{Female};$ $\quad \mathsf{Man} \equiv \mathsf{Human} \sqcap \mathsf{Male};$
$\mathsf{Parent} \equiv \mathsf{Human} \sqcap \exists \mathsf{HasChild.Human};$ $\mathsf{Father} \equiv \mathsf{Man} \sqcap \mathsf{Parent};$
$\mathsf{Mother} \equiv \mathsf{Woman} \sqcap \mathsf{Parent} \ \exists \mathsf{HasChild.Human};$
$\mathsf{Child} \equiv \mathsf{Human} \sqcap \exists \mathsf{HasParent.Parent};$

$\mathsf{Grandparent} \equiv \mathsf{Parent} \sqcap \exists \mathsf{HasChild.}(\exists\ \mathsf{HasChild.Human});$
$\mathsf{Sibling} \equiv \mathsf{Child} \sqcap \exists \mathsf{HasParent.}(\exists\ \mathsf{HasChild} \geq 2);$
$\mathsf{Niece} \equiv \mathsf{Human} \sqcap \exists \mathsf{HasGrandParent.Parent} \sqcup \exists \mathsf{HasUncle.Uncle};$
$\mathsf{Cousin} \equiv \mathsf{Niece} \sqcap \exists \mathsf{HasUncle.}(\exists\ \mathsf{HasChild.Human});$
$\}.$

ABox: $\mathcal{A} = \{$Woman(Claudia), Woman(Tiziana), Father(Leonardo), Father(Antonio), Father(AntonioB), Mother(Maria), Mother(Giovanna), Child(Valentina), Sibling(Martina), Sibling(Vito), HasParent(Claudia,Giovanna), HasParent(Leonardo,AntonioB), HasParent(Martina,Maria), HasParent(Giovanna,Antonio), HasParent(Vito,AntonioB), HasParent(Tiziana,Giovanna), HasParent(Tiziana,Leonardo), HasParent(Valentina,Maria), HasParent(Maria,Antonio), HasSibling(Leonardo,Vito), HasSibling(Martina,Valentina), HasSibling(Giovanna,Maria), HasSibling(Vito,Leonardo), HasSibling(Tiziana,Claudia), HasSibling(Valentina,Martina), HasChild(Leonardo,Tiziana), HasChild(Antonio,Giovanna), HasChild(Antonio,Maria), HasChild(Giovanna,Tiziana), HasChild(Giovanna,Claudia), HasChild(AntonioB,Vito), HasChild(AntonioB,Leonardo), HasChild(Maria,Valentina), HasUncle(Martina,Giovanna), HasUncle(Valentina,Giovanna)
$\}$

Considered this knowledge base, it is possible to compute the similarity value between concepts as shown:

$$\begin{aligned} s(\mathsf{Grandparent}, \mathsf{Father}) &= \frac{|(\mathsf{Grandparent} \sqcap \mathsf{Father})^{\mathcal{I}}|}{|\mathsf{Granparent}^{\mathcal{I}}| + |\mathsf{Father}^{\mathcal{I}}| - |(\mathsf{Grandparent} \sqcap \mathsf{Father})^{\mathcal{I}}|} \cdot \\ &\cdot \quad max(\frac{|(\mathsf{Grandparent} \sqcap \mathsf{Father})^{\mathcal{I}}|}{|\mathsf{Grandparent}^{\mathcal{I}}|}, \frac{|(\mathsf{Grandparent} \sqcap \mathsf{Father})^{\mathcal{I}}|}{|\mathsf{Father}^{\mathcal{I}}|}) = \\ &= \frac{2}{2+3-2} \cdot max(\frac{2}{2}, \frac{2}{3}) = 0.67 \end{aligned}$$

In the same way it is possible to compute the similarity value among all concepts defined above. $\square$

4.1.1 Derived Similarity Measure Involving Individuals

The measure in Def. 4.1.1 is able to the assess similarity between concepts. Until now, there is no way for computing similarity value between individuals. Determining the similarity between individual could be important for many learning tasks such as classification and clustering and hence a similarity measure able to do this could be very helpful for performing learning applied to ontological knowledge.

As seen in Sect. 2.4.2, for every individual in the ABox, it is possible to compute its Most Specific Concept (msc) or at least its upper approximation[5] (msc^*). Namely, given a and b, two individuals in a given ABox, it is possible to compute $A^* = \mathsf{msc}^*(a)$ and $B^* = \mathsf{msc}^*(b)$. In this way concept definitions are available and hence, it is possible to apply Def. 4.1.1, thus yielding the similarity value for the two individuals:

$$\forall a, b: \ s(a,b) = s(A^*, B^*) = s(\mathsf{msc}^*(a), \mathsf{msc}^*(b))$$

Hence, the similarity value between two individuals is set to be equal to the similarity value computed considering their mscs. Analogously, the similarity value between a concept description C and an individual a can be computed as the similarity value between the concept C and the $\mathsf{msc}^*(a)$:

$$\forall a: \ s(a,C) = s(\mathsf{msc}^*(a), C)$$

In order to clarify the extension of Def. 4.1.1 to the case involving individuals, the following example is given.

Example 4.1.3 *Let $\mathcal{T}$ the TBox and $\mathcal{A}$ the ABox defined in Example 4.1.2 and let Claudia and Tiziana the considered individuals. In order to determine the similarity value of these two individuals their msc^*s have to be computed. They are given by:*

$$\begin{aligned}\textsf{msc*(Claudia)} &= \textsf{Woman} \sqcap \textsf{Sibling} \sqcap \exists\ \textsf{HasParent(Mother} \sqcap \textsf{Sibling} \sqcap \\ &\sqcap\ \exists\textsf{HasSibling(C1)} \sqcap\ \exists\textsf{HasParent(C2)} \sqcap \exists\textsf{HasChild(C3)})\end{aligned}$$

where

- $\textsf{C1} \equiv \textsf{Mother} \sqcap \textsf{Sibling} \sqcap \exists\textsf{HasParent(Father} \sqcap \textsf{Parent)} \sqcap \exists\textsf{HasChild(Cousin} \sqcap \exists\textsf{HasSibling(Cousin} \sqcap \textsf{Sibling} \sqcap \exists\textsf{HasSibling}.\top))$
- $\textsf{C2} \equiv \textsf{Father} \sqcap \exists\textsf{HasChild(Mother} \sqcap \textsf{Sibling)}$
- $\textsf{C3} \equiv \textsf{Woman} \sqcap \textsf{Sibling} \sqcap \exists\textsf{HasSibling}.\top \sqcap \exists\textsf{HasParent(C4)}$

[5] In some cases msc and msc^* are equivalent. In general $\mathsf{msc} \sqsubseteq \mathsf{msc}^*$.

- $\mathsf{C4} \equiv \mathsf{Father} \sqcap \mathsf{Sibling} \sqcap \exists \mathsf{HasSibling(Uncle} \sqcap \mathsf{Sibling} \sqcap \exists \mathsf{HasParent(Father} \sqcap \mathsf{Granparent))} \sqcap \mathsf{HasParent(Father} \sqcap \mathsf{Grandparent} \sqcap \exists \mathsf{HasChild(Uncle} \sqcap \mathsf{Sibling}))$

And for the individual Tiziana:

$$\begin{aligned}
\mathsf{msc*(Tiziana)} &= \mathsf{Woman} \sqcap \mathsf{Sibling} \sqcap \exists \mathsf{HasSibling(Woman} \sqcap \mathsf{Sibling} \sqcap \\
&\sqcap \exists \mathsf{HasParent(C5))} \sqcap \exists \mathsf{HasParent(Father} \sqcap \mathsf{Sibling} \sqcap \mathsf{HasSibling(C6)} \sqcap \\
&\sqcap \exists \mathsf{HasParent(C7))} \sqcap \mathsf{HasParent(Uncle} \sqcap \mathsf{Mother} \sqcap \mathsf{Sibling} \sqcap \\
&\sqcap \exists \mathsf{HasChild(Woman} \sqcap \mathsf{Sibling}))
\end{aligned}$$

where

- $\mathsf{C5} \equiv \mathsf{Mother} \sqcap \mathsf{Sibling} \sqcap \exists \mathsf{HasSibling(C8)} \sqcap \exists \mathsf{HasParent(Father} \sqcap \exists \mathsf{HasChild(Mother} \sqcap \mathsf{Sibling}))$

- $\mathsf{C8} \equiv \mathsf{Mother} \sqcap \mathsf{Sibling}\, \exists \mathsf{HasParent(Father} \sqcap \mathsf{Grandparent)} \sqcap \exists \mathsf{HasChild(Cousin} \sqcap \exists \mathsf{HasSibling(Cousin} \sqcap \mathsf{Sigling} \sqcap \exists \mathsf{HasSibling}.\top))$

- $\mathsf{C6} \equiv \mathsf{Uncle} \sqcap \mathsf{Sibling} \sqcap \exists \mathsf{HasParent(Father} \sqcap \mathsf{Grandparent})$

- $\mathsf{C7} \equiv \mathsf{Father} \sqcap \mathsf{Granparent} \sqcap \exists \mathsf{HasChild(Uncle} \sqcap \mathsf{Sibling)}$

Note that it holds that $\mathsf{msc}^*\mathsf{(Tiziana)} \not\sqsubseteq \mathsf{msc}^*\mathsf{(Claudia)}$ *and* $\mathsf{msc}^*\mathsf{(Claudia)} \not\sqsubseteq \mathsf{msc}^*\mathsf{(Tiziana)}$. *Now, since* $(\mathsf{msc}^*(\mathsf{Tiziana}))^{\mathcal{I}} = \{\mathsf{Tiziana}\}$ *and* $(\mathsf{msc}^*(\mathsf{Claudia}))^{\mathcal{I}} = \{\mathsf{Claudia}, \mathsf{Tiziana}\}$, *the similarity of these individuals is:*

$$s(\mathsf{Claudia}, \mathsf{Tiziana}) = \frac{1}{1+2-1} \cdot max\left(\frac{1}{2}, \frac{1}{1}\right) = 0.5$$

In the same way it could be calculate the similarity between a concept and an individual. □

4.1.2 Computational Complexity

In order to assess the complexity of the presented measure, three different cases of application of the measure are discussed separately. Considering that the measure mainly use a numerical approach, the only source of complexity is given by the retrieval operator, used for computing concept extensions. Since it is grounded on the instance checking operator (IChk), then the only source of complexity for the measure is given by the complexity of instance checking for the adopted DL language, hereafter indicated with $C(\mathsf{IChk})$. The complexity of the presented measure could be summarized as in the following:

Similarity between concepts: $C(s) = 3 \cdot C(\mathsf{IChk})$
because the instance check is repeated three times: for the concept descriptions C, D and I.

Similarity individual - concept: $C(s) = C(\mathsf{msc}^*) + 3 \cdot C(\mathsf{IChk})$
this is because, in this case, besides of the instance checking operations required by the previous case, the msc approximation of the considered individual is to be computed; the complexity of the msc approximation is denoted by $C(\mathsf{msc}^*)$.

Similarity between individuals: $C(s) = 2 \cdot C(\mathsf{msc}^*) + 3 \cdot C(\mathsf{IChk})$
this case is analogous to the previous one, the only difference is that now two msc approximations have to be computed for the arguments.

As clearly shown by these formulæ, the measure complexity is sensible to the choice of the reference DL. For instance, for the $\mathcal{ALC}$ logic, $C(\mathsf{IChk})$ is PSPACE (see [8], Ch. 3). For the cases involving individuals, it suffices to recall that also the computation of the (approximation of) the most specific concept depends on instance checking, besides of the specific algorithm (see Sec. 2.4.2 and [134] for more details about the complexity of the msc approximation).

Experimental evaluation has shown that the measure can be effectively used to compute the similarity between concepts. Anyway it presents some problems when similarity between individuals is computed. This is due to the use of the (approximated) msc which often turns out to be so specific that its extension likely includes only the considered individual. This phenomenon consequently provokes a dissimilarity even if the individuals are assertions of concepts semantically very similar.

As a consequence of some investigation for solving this issue, the more reliable solution seemed to measure the similarity (or dissimilarity) of the concepts as a combination of the similarity value of the sub-concepts that constitute the msc concepts. The underling idea to this solution is that concepts defined in terms of the same sub-concepts could be intuitively similar in their turn.

Anyway, the realization of such solution could be not so straightforward since, as known, semantically equivalent concepts may be described in many syntactically different ways. The solution to these problems is given by the measure presented in the next section.

4.2 A Semantic and Structure driven Dissimilarity measure for $\mathcal{ALC}$

The assessment of similarity value between individuals cannot be effectively done by "simply comparing" the extensions of their (approximated) msc. This is because msc is so specific that includes in its extension only the individual for which it is computed and it does not cover other individuals semantically related.

In order to overcome this problem the idea is to compute the (dis-)similarity value between individuals as a "combination" of the (dis-)similarity of the sub-concepts that constitute their (approximated) msc. This idea moves from the intuition that concepts defined by almost the same sub-concepts will be probably similar.

This intuition is not so different from those introduced in other relevant works in relational settings. Indeed, for example, in [72] the computation of the similarity between objects (described by features and by relations with other objects) depends on the similarity of their attribute values and the similarity of the objects related to them. The similarity of the related objects, in turn, depends on the attribute values of these objects and their relation to other objects and so on. In [74] a distance measure between structural descriptions is presented. It is defined according to a top-down evaluation scheme: distance between disjunction of conjunctions, distance between conjunctions, and literals. At the lowest level, the similarity is computed based on a probabilistic interpretation of a matching predicate. An approach similar to those of the just described previous works has been presented in [165] for computing the similarity of concept descriptions in First-Order-Logic.

Differently from these works, in which similarity is computed mainly in a structural way, the measure that will be shown here [50, 52] computes the similarity value using a semantic approach, namely using the notion of the extension of the considered concepts.

Anyway, considering sub-concepts of a description, a structural based approach is also adopted, that is known to be not reliable when the representation language is DL because it allow for semantically equivalent concepts written in many syntactically different ways. However, as seen in Sect. 2.4, given a DL, which is $\mathcal{ALC}$ in our case, every concept descriptions can be turned into an equivalent concept description in normal form by the use of rewriting rules. The normalization of concepts assures that semantically equivalent concepts will be written all in the same way and hence a structural approach can be used mitigating this drawback. In the following the dissimilarity measure for $\mathcal{ALC}$ will be presented.

4.2.1 Overlap Function

In order to define a dissimilarity measure for $\mathcal{ALC}$ descriptions, the notion of *overlap function* is introduced. This is a definite positive function on the set of $\mathcal{ALC}$ normal form descriptions (see Def. 2.4.2), defined making use of the structure and semantics of the concepts (and roles) involved in the descriptions. Consequently, from such overlap function, the dissimilarity measure will be derived. In the following, the overlap function is formally defined[6].

Definition 4.2.1 (Overlap Function) *Let $\mathcal{L} = \mathcal{ALC}/_{\equiv}$ be the set of all concepts in $\mathcal{ALC}$ normal form and let $\mathcal{A}$ be an A-Box with canonical interpretation $\mathcal{I}$. f is a function*[7] *$f : \mathcal{L} \times \mathcal{L} \mapsto R^+$ such that for all $C, D \in \mathcal{L}$, with $C = \bigsqcup_{i=1}^{n} C_i$ and $D = \bigsqcup_{j=1}^{m} D_j$ it assigns:*

$$\underline{\textit{Disjunctive level}}: \quad f(C,D) := f_{\sqcup}(C,D) = \begin{cases} \infty & \text{if } C \equiv D \\ 0 & \text{if } C \sqcap D = \bot \\ \max_{\substack{i \in [1,n] \\ j \in [1,m]}} f_{\sqcap}(C_i, D_j) & \text{o.w.} \end{cases}$$

$$\underline{\textit{Conjunctive level:}} \quad f_{\sqcap}(C_i, D_j) := f_P(\mathsf{prim}(C_i), \mathsf{prim}(D_j)) + f_{\forall}(C_i, D_j) + f_{\exists}(C_i, D_j)$$

<u>Primitive concepts:</u>

$$f_P(\mathsf{prim}(C_i), \mathsf{prim}(D_j)) := \frac{|PE(C_i) \cup PE(D_j)|}{|(PE(C_i) \cup PE(D_j)) \setminus (PE(C_i) \cap PE(D_j))|}$$

where,

- $PE(C_i) := (\bigsqcap_{P \in \mathsf{prim}(C_i)} P)^{\mathcal{I}}$ *and* $PE(D_j) := (\bigsqcap_{P \in \mathsf{prim}(D_j)} P)^{\mathcal{I}}$ *(extension of the primitive concepts conjunctions)*
- $f_P(\mathsf{prim}(C_i), \mathsf{prim}(D_j)) = \infty$ *when* $(\mathsf{prim}(C_i))^{\mathcal{I}} = (\mathsf{prim}(D_j))^{\mathcal{I}}$
- $f_P(\mathsf{prim}(C_i), \mathsf{prim}(D_j)) = 0$ *when* $PE(C_i) \cap PE(D_j) = \emptyset$

$$\underline{\textit{Value restrictions}}: \quad f_{\forall}(C_i, D_j) := \sum_{R \in N_R} f_{\sqcup}(\mathsf{val}_R(C_i), \mathsf{val}_R(D_j))$$

$$\underline{\textit{Existential restrictions}}: \quad f_{\exists}(C_i, D_j) := \sum_{R \in N_R} \sum_{k=1}^{N} \max_{p=1,\ldots,M} f_{\sqcup}(C_i^k, D_j^p)$$

where $C_i^k \in \mathsf{ex}_R(C_i)$ and $D_j^p \in \mathsf{ex}_R(D_j)$ and it is supposed that $N = |\mathsf{ex}_R(C_i)| \geq |\mathsf{ex}_R(D_j)| = M$, otherwise the indices N and M are to be exchanged in the formula.

[6]Note that the notations used are those introduced in Sec. 2.4.

[7]The name $\mathcal{A}$ of the A-Box is omitted for keeping the notation as simple as possible.

The function f represents a measure of the overlap between two concept descriptions (namely C and D) expressed in $\mathcal{ALC}$ normal form. It is defined recursively beginning from the top level of the concept descriptions (a disjunctive level) up to the bottom level represented by (conjunctions of) primitive concepts. In this way, at every level, sub-concepts are compared and their similarity is computed. Hence the "combination" of the similarity values computed at every level will give the final similarity value of the considered concepts.

In case of disjunction, the overlap amount is obviously null if the two concepts are totally disjoint (namely $C \sqcap D \equiv \bot$), while the overlap is set to infinity if the two concepts are equivalent, indeed in this case their extensions are exactly the same. In case of partial overlap between the two concepts, the amount is computed as the maximum of the overlaps computed between all couples of disjuncts (C_i, D_j) that make up the top level of the considered concepts. The rationale of this third case is given by the semantics of the disjunction operator. Since the disjunction requires that at least one of the disjuncts is verified, in the same way it is assumed that the overlap is given by the maximum of the computed values between all couples of disjuncts.

Since each disjunct is a conjunction of descriptions, it is necessary to define how to compute the overlap between conjunctive concepts. This overlap is computed as the sum of the overlap measure computed on the parts that make up the conjunctive concept descriptions. The rationale of this definition is given by the semantics of the conjunction operator. In fact as conjunction requires that all of its terms are verified, in the same way the overlap is measured as the sum of the overlap between all terms constituting the conjunctions.

Specifically, a conjunctive form can have three different types of terms: primitive concepts, universal restrictions and existential restrictions. Since conjunction is a symmetric operator by definition, it is possible to put together every type of restriction (occurring at the same level) so it is possible to consider the conjunctions of primitive concepts $(\mathsf{prim}(C_i), \mathsf{prim}(D_j))$, the conjunctions of existential restrictions and the conjunction of universal restrictions as specified in the definition of $\mathcal{ALC}$ normal form. Next, the computation of overlap for the three different type of terms is defined.

The overlap between two conjunctions of (possibly negated) primitive concepts is minimal if the they do not share any individual in their extensions. Conversely, if the two concepts share some individuals, the overlap between them is computed as a measure of the union of their extensions with respect to the complementary set of the intersection with respect to their union. Hence, as the number of shared individuals increases consequently, the denominator decreases and thus the amount of the overlap computed for two conjunctions of (negated) primitive concepts increases.

This is exactly the desired behavior of a function that measures the intuitive notion of "overlap" between concepts. Moreover it is important to note that the function returns 0 or a value that is greater than 1.

The computation of the overlap between, respectively, concept descriptions expressed by universal and existential restrictions is a bit more complex. Considering the conjunction of universal restrictions, it is worthwhile to recall the rewriting rules (see Sect. 2.4) for which, in a conjunction, every universal restriction is relative to a different role (since it is possible to write $\forall R.C \sqcap \forall R.D = \forall R.(C \sqcap D)$). Moreover, recall that the scope of each universal restriction is expressed in normal form. Thus, the overlap between two concepts (within the disjuncts C_i and in D_j, resp.) that are in the scope of a universal restriction w.r.t. a certain role R can be computed as the overlap between two concepts expressed in normal form recursively computed by $f_\sqcup$, already discussed. Of course, if no disjunction occurs at the top level, it is possible to regard the concept description as a disjunction of single term to which $f_\sqcup$ applies in a simple way. Anyway, since a conjunction of different universal restrictions may occur (one per different role), the amount of the overlap of this conjunction is given by the sum of the overlap computed for every universal restriction, namely for every scope of a universal restriction. It is worthwhile noting that, when a universal restriction on a certain role (say R) occurs in a disjunct (e.g. in C_i), but no such restriction on the same role occurs in the other description (say D_j), then $\mathsf{val}_R(D_j)$ is considered equal to the top concept, namely $\mathsf{val}_R(D_j) = \top$.

Now, the overlapping computation between two descriptions made up of conjunctions of existential restrictions is analyzed. In this case the notion of *existential mapping* [124] is considered. It is supposed that $N = |\mathsf{ex}_R(C_i)| \geq M = |\mathsf{ex}_R(D_j)|$. Such a mapping can be defined as a function: $\alpha : \{1, \ldots, N\} \mapsto \{1, \ldots, M\}$. If each element of $\mathsf{ex}_R(C_i)$ and $\mathsf{ex}_R(D_j)$ is indexed with an integer in the ranges $[1, N]$ and $[1, M]$, respectively, then α maps each concept description $C_i^k \in \mathsf{ex}_R(C_i)$ with a description $D_j^p \in \mathsf{ex}_R(D_j)$. Since each C_i^k (resp. D_j^p) is in normal form, again, it is possible to compute the amount of their overlap applying $f_\sqcup$ to such corresponding descriptions. Namely, fixed a role R and considered a certain C_i^k (with $k \in [1, N]$), the overlap between C_i^k and D_j^p (with $p \in [1, M]$) is computed. Remembering that the hypothesis in the function definition at existential restrictions level is $N \geq M$, thus each existential restriction on R is coupled with the one on the same role in other description scoring the maximum amount of overlap. These maxima are summed up per role, then the sum is made also varying the role considered. In case of one role restriction on a certain role S is absent from either description then the concept scope of such "hypothetical" role (S) is considered as the concept $\top$.

4.2.2 Defining the Dissimilarity Measure

After clarifying the definition of the overlap function f, a dissimilarity measure is derived from it as shown in the following.

Definition 4.2.2 (Semantic Dissimilarity Measure) *Let $\mathcal{L}$ be the set of all concepts $\mathcal{ALC}$ normal form and let $\mathcal{A}$ be an A-Box. The* dissimilarity measure *d is a function $d : \mathcal{L} \times \mathcal{L} \mapsto [0,1]$ defined as follows: $\forall\ C = \bigsqcup_{i=1}^{n} C_i$ and $D = \bigsqcup_{j=1}^{m} D_j$ concept descriptions in $\mathcal{ALC}$ normal form, let*

$$d(C,D) := \begin{cases} 1 & \textit{if } f(C,D) = 0 \\ 0 & \textit{if } f(C,D) = \infty \\ 1/f(C,D) & \textit{otherwise} \end{cases} \tag{4.2}$$

where f is the function defined above.

The function d measures the dissimilarity between two concepts, say C and D, in $\mathcal{ALC}$ normal form, using the overlap function f. Particularly, if the overlap function assumes its minimum value ($f(C,D) = 0$), this means that there is no overlap between the considered concepts, therefore d must indicate that the two concepts are totally different, indeed $d(C,D) = 0$, that is minimum value of d. If $f(C,D) = \infty$ this means that the two concepts are totally overlapped and consequently $d(C,D) = 0$ that means that the two concepts are undistinguishable, indeed d assumes the minimum value of its range. If the considered concepts have a partial overlap then their dissimilarity, defined as $1/f(C,D)$ is lower as much as the two concept are more overlapped[8]. Indeed, as noticed in the previous section, the value of f will be always greater than one in this case and it grows with the increasing of the overlapping amount, consequently, for Def. 4.2.2, d will be $0 < d(C,D) < 1$.

The function d is really a dissimilarity measure. Remembering Def. 3.1.1 a function d is a dissimilarity measure if the following three properties are satisfied: 1) d is definite positive; 2) d is symmetric; 3) d satisfies the minimality property namely: $\forall C, D\ :\ d(C,D) \geq d(C,C)$. The first property is trivially satisfied by d, as it has value in the real interval $[0,1]$. Moreover, by construction, d computes dissimilarity by means of sums of positive quantities and maxima computed on sets of such values. The symmetry property is trivially verified by the commutativity of the sum and maximum operators. Then, it is easy to prove the minimality property. Indeed, by the definition, it holds that $\mathrm{d}(C,C) = 0$ and $\mathrm{d}(C,C') = 0$ if C is semantically equivalent to C'. In all other cases, $\forall D \in \mathcal{L}$ and D not semantically equivalent to D ($C \not\equiv D$), results: $d(C,D) > 0$. To better clarify the usage of the presented dissimilarity measure, an example is illustrated in the following.

[8]Remember that the overlap is measured considering the entire concept definitions and their sub-concepts too.

Example 4.2.3 *Let C and D be two $\mathcal{ALC}$ concepts in normal form defined as follows:*

$C \equiv A_2 \sqcap \exists R.B_1 \sqcap \forall T.(\forall Q.(A_4 \sqcap B_5)) \sqcup A_1$

$D \equiv A_1 \sqcap B_2 \sqcap \exists R.A_3 \sqcap \exists R.B_2 \sqcap \forall S.B_3 \sqcap \forall T.(B_6 \sqcap B_4) \sqcup B_2$

where A_i and B_j are all primitive concepts.

To compute the dissimilarity value between C and D, first of all, the overlap function has to be computed. Let C and D be neither equivalent nor disjoint. Hence the third case of $f_\sqcup$ has to be applied. For sake of simplicity it is denoted $C_1 := A_2 \sqcap \exists R.B_1 \sqcap \forall T.(\forall Q.(A_4 \sqcap B_5))$ and $D_1 := A_1 \sqcap B_2 \sqcap \exists R.A_3 \sqcap \exists R.B_2 \sqcap \forall S.B_3 \sqcap \forall T.(B_6 \sqcap B_4)$. The overlap will be given by the maximum value of the overlap computed among all couples of disjunctive terms:

$$f(C,D) := f_\sqcup(C,D) = \max\{\ f_\sqcap(C_1,D_1), f_\sqcap(C_1,B_2), f_\sqcap(A_1,D_1), f_\sqcap(A_1,B_2)\ \}$$

For brevity, only the computation of $f_\sqcap(C_1,D_1)$ will be considered, which is also the most complex case. $f_\sqcap$ is computed as the sum of f_P, $f_\forall$, $f_\exists$ i.e., respectively. Let it be $(A_2)^\mathcal{I} \neq (A_1 \sqcap B_2)^\mathcal{I}$ and also $(A_2)^\mathcal{I} \cap (A_1 \sqcap B_2)^\mathcal{I} \neq \emptyset$. Hence, f_P is given by:

$$\begin{aligned} f_P(C_1,D_1) &= f_P(\mathsf{prim}(C_1), \mathsf{prim}(D_1)) = f_P(A_2, A_1 \sqcap B_2) = \\ &= \frac{|(A_2)^\mathcal{I} \cup (A_1 \sqcap B_2)^\mathcal{I}|}{|((A_2)^\mathcal{I} \cup (A_1 \sqcap B_2)^\mathcal{I}) \setminus ((A_2)^\mathcal{I} \cap (A_1 \sqcap B_2)^\mathcal{I})|} \end{aligned}$$

In order to compute $f_\forall$ it is necessary to note that there are two roles at the same level: T and S; so the summation over the different roles is made by two terms. The role S is only in D_1 and not in C_1, consequently $val_R(C_1) = \top$. Thus, in this case,

$$\begin{aligned} f_\forall(C_1,D_1) &= \sum_{R \in N_R} f_\sqcup(\mathsf{val_R}(C_1), \mathsf{val_R}(D_1)) = \\ &= f_\sqcup(\mathsf{val_T}(C_1), \mathsf{val_T}(D_1)) + f_\sqcup(\mathsf{val_S}(C_1), \mathsf{val_S}(D_1)) = \\ &= f_\sqcup(\forall Q.(A_4 \sqcap B_5), B_6 \sqcap B_4) + f_\sqcup(\top, B_3) \end{aligned}$$

The computations of $f_\sqcup(\forall Q.(A_4 \sqcap B_5), B_6 \sqcap B_4)$ and $f_\sqcup(\top, B_3)$ are the same seen above.

Now, the computation of $f_\exists$ has to be performed. It is necessary to note that here there is only one role R, so the first summation collapses to a single element. Then N and M, that are the numbers of conjunctive existential concept descriptions w.r.t. the same role (S in this case), are respectively $N = 2$ and $M = 1$, so it would have to find the max *in a singleton, that can be simplified. Hence the computation of $f_\exists$ for the current example will be:*

$$f_\exists(C_1,D_1) = \sum_{k=1}^{2} f_\sqcup(\mathsf{ex_R}(C_1), \mathsf{ex_R}(D_1^k)) = f_\sqcup(B_1, A_3) + f_\sqcup(B_1, B_2)$$

Also in this case, the computation of $f_\sqcup$ is the same seen above.

In order to determine the dissimilarity value of C and D the same operations have to be performed for computing $f_\sqcap(C_1, B_2), f_\sqcap(A_1, D_1), f_\sqcap(A_1, B_2)$. Once that such the overlap amounts have been computed, the final overlap value between C and D will be given by the maximum of the computed overlap values for all couples of disjunctive terms. Hence, the dissimilarity value is immediately obtained as the inverse of the chosen maximum value. □

As seen in Sect. 4.1.1, the presented dissimilarity measure can be easily applied to compute dissimilarity between individuals or between an individual and a concept, by recurring to the notion of the (approximated) msc of an individual w.r.t. the considered ABox. The obtained msc can be normalized as whatever concept description and consequently the dissimilarity value can be computed as seen in Ex. 4.2.3. Computing dissimilarity between individuals or between concept and individual may be turn useful both in inductive reasoning (construction, repairing of knowledge bases) and in information retrieval. Differently from the measure defined in Sect. 4.1, the presented measure is really able to determine the dissimilarity value between individuals. By recurring to a structure driven approach, the dissimilarity value is computed considering dissimilarities between sub-concepts of the concept definitions. Even if a structural approach has been used, the computed dissimilarity value does not depend from the way in which a concept is described, as concepts in normal form are considered. Indeed the normal form of a concept ensures that semantically equivalent concepts are also written in the same syntactic way. This aspect allows to use also a semantic approach that is mainly given by the use of the concept extensions when the dissimilarity between conjunctive (negated) primitive concepts is computed. Moreover, as seen for the measure presented in Sect. 4.1, the usage of the concept extensions allows to define a semantic dissimilarity measure by means of a numeric approach.

Experimental evaluations (see Sect. 5.1.1) of the presented measure showed that it is effectively able to determine dissimilarity values between concepts, between individuals and between concepts and individuals. However, in case of complex concept descriptions (such as msc), deeply nested sub-concepts could increase the dissimilarity value. This is because, with the increasing of the nested sub-concepts, increases also the dissimilarity values (between subconcepts) to sum up. Anyway, deeply nested sub-concepts are less semantically related to the considered concept. A solution to this problem is presented in Sect. 4.2.4. In the following, the computational complexity of the measure presented in this section is discussed.

4.2.3 Computational Complexity

The computational complexity of the dissimilarity measure d is strictly related to the computational complexity of the overlap function f defined in Sect 4.2.1. Besides, the overlap function relies on some reasoning services, namely subsumption and retrieval. Assuming the most trivial computation of the retrieval by the use of the instance checking inference operator (IChk), it is easy to note that the complexity of f (and hence the complexity of d) depends on the complexity of these inferences. Specifically, in order to define the complexity of d, three cases, descending from being d grounded on f (namely $\mathit{Compl}(d) = \mathit{Compl}(f_\sqcup)$), will be distinguished.

Let $C = \bigsqcup_{i=1}^{n} C_i;\ D = \bigsqcup_{j=1}^{m} D_j$ be two $\mathcal{ALC}$ normal form concept descriptions:

Case 1: *C and D are semantically equivalent.* In this case only subsumption is involved in order to verify the semantic equivalence of the concepts. Thus

$$\mathit{Compl}(d) = 2 \cdot \mathit{Compl}(\sqsupseteq)$$

where $\mathit{Compl}(\cdot)$ represents the complexity.

Case 2: *C and D are disjoint.* In this case subsumption and conjunction are involved. Anyway, being the conjunction complexity constant in time, the complexity of d is the same as in the previous case.

Case 3: *C and D are neither semantically equivalent nor disjoint.* In this case the complexity depends on the structure of the concepts involved. Particularly, $f_\sqcap$ has to be computed for $n \cdot m$ times (for determining the maximum overlap value among all couples of disjunctive terms); hence, the complexity is:

$$\mathit{Compl}(d) = n \cdot m \cdot \mathit{Compl}(f_\sqcap) \ = \ n \cdot m \cdot [\mathit{Compl}(f_P) + \mathit{Compl}(f_\forall) + \mathit{Compl}(f_\exists)]$$

Thus, it is necessary to analyze the complexity of f_P, $f_\forall$, $f_\exists$.

The dominant operation in computing f_P is given by the *Instance Checking* (IChk), used for determining the concept extensions. Considering that computing f_P, two concepts are involved, then the complexity of f_P is given by:

$$\mathit{Compl}(f_P) = 2 \cdot \mathit{Compl}(IChk)$$

The computation of $f_\forall$ and $f_\exists$ applies recursively the definition of $f_\sqcup$ on less complex descriptions. Specifically, $|N_R|$ calls of $f_\sqcup$ are necessary for computing $f_\forall$, while the number of invocations of $f_\sqcup$ necessary for computing $f_\exists$ is equal to $|N_R| \cdot N \cdot M$, where $N = |\mathsf{ex}_R(C_i)|$ and $M = |\mathsf{ex}_R(D_j)|$ as in Def. 4.2.1. Hence, summing up, the complexity of d is given by:

$$\mathit{Compl}(d) = nm[\ (2\mathit{Compl}(IChk)) + (|N_R|\mathit{Compl}(f_\sqcup)) + (|N_R|MN\,\mathit{Compl}(f_\sqcup))\ ]$$

The complexity of the dissimilarity measure d strongly depends on the complexity of the instance checking for $\mathcal{ALC}$, which is P-space [8]. Nevertheless, in practical applications, these computations may be efficiently carried out exploiting statistics that are maintained by the DBMSs query optimizers. Besides, the counts that are necessary for computing the concept extensions could be estimated by means of the probability distribution over the domain. Another way for optimizing the computation of d involves pre-computing the extensions of the primitive concepts, and hence determining the extensions of complex concept description by means of the set operations. In this way the complexity of d can be decreased. In the next section, a modified version of the presented measure will be illustrated. This has been developed in order to avoid that deeply nested concept definitions affect the real dissimilarity value. Nevertheless, the weighted version of the dissimilarity measure does not increase its complexity.

4.2.4 A Weighted Dissimilarity Measure for $\mathcal{ALC}$

Experimental evaluations[9] of the dissimilarity measure presented in Sect. 4.2.2 showed that it performs well, assessing the dissimilarity between concepts, between individuals and between concepts and individuals. However, for complex concept descriptions, deeply nested sub-concepts could increase the dissimilarity value between two concepts more than their "real" value. This is because, the dissimilarity measure is defined on the ground of the overlap function (see Def. 4.2.1 and Def. 4.2.2) which is recursively defined, beginning from the top level of the concept descriptions (a disjunctive level) up to the bottom level represented by (conjunctions of) primitive concepts. Hence concept descriptions constituted by deeply sub-concepts could increase the overlap w.r.t. another concept adding further information. Anyway very deeply sub-concepts (such as those contained in different nested levels of existential and universal concept restrictions) are less semantically related to the initial concepts (say C and D), while they are more related to other aspects represented by sub-concepts of C and D. In order to clarify this intuition, the following description is considered:

$$\mathsf{C} \equiv \mathsf{Woman} \sqcap \mathsf{Sibling} \sqcap \exists\ \mathsf{HasParent}(\mathsf{Mother} \sqcap \mathsf{Sibling} \sqcap \exists \mathsf{HasSibling}(\mathsf{C1}))$$

where

$$\mathsf{C1} \equiv \mathsf{Mother}\ \sqcap\ \mathsf{Sibling}\ \sqcap\ \exists \mathsf{HasParent}(\mathsf{Father}\ \sqcap\ \mathsf{Parent})$$

The sub-concepts Father and Parent in the scope of $\exists \mathsf{HasParent}(\mathsf{Father} \sqcap \mathsf{Parent})$ refer to a specification of the concept $\mathsf{C1}$, hence they are less semantically important

[9]See Ch. 5 for applications and experimental evaluations of the measures presented in this chapter.

for C than concepts Mother and $\mathsf{Sibling}$ that specify directly the concept C (indeed they are at the top level of the concept definition). Anyway computing the overlap between C and another concept, the sub-concepts Mother, $\mathsf{Sibling}$, Father and Parent will have the same importance in determining such value. With the increase of the nesting level of the sub-concepts, this phenomenon becomes more evident.

In order to solve this problem the measure can be weighted by the use of a weighting factor that decreases the "importance" of the overlap between sub-concepts with the increase of their nesting levels in the initial descriptions [51]. Specifically, more sub-concepts are nested, more the overlapping will be decreased by the associated weight; consequently such amount will be less determinant in computing the final dissimilarity. Moreover, in order to make Def. 4.2.1 and Def. 4.2.2 really computable, the value ∞ has to be quantified in a computable way. Using the knowledge base as source of information, the value ∞ can be substituted by the cardinality of the ABox ($|\Delta|$) which represents all the knowledge about the domain. Hence the overlap function is modified as in the following:

Disjunctive level:

$$f'(C,D) := f'_{\sqcup}(C,D) = \begin{cases} |\Delta| & \text{if } C \equiv D \\ 0 & \text{if } C \sqcap D = \bot \\ \max_{\substack{i \in [1,n] \\ j \in [1,m]}} 1 + \lambda \cdot f'_{\sqcap}(C_i, D_j) & \text{o.w.} \end{cases}$$

where λ is a weighting factor.

Primitive concepts:

$$f'_P(\mathsf{prim}(C_i), \mathsf{prim}(D_j)) := \frac{|PE(C_i) \cup PE(D_j)|}{|(PE(C_i) \cup PE(D_j)) \setminus (PE(C_i) \cap PE(D_j))|}$$

where,

- $PE(C_i) := (\bigsqcap_{P \in \mathsf{prim}(C_i)} P)^{\mathcal{I}}$ and $PE(D_j) := (\bigsqcap_{P \in \mathsf{prim}(D_j)} P)^{\mathcal{I}}$ (extension of the primitive concepts conjunctions)
- $f_P(\mathsf{prim}(C_i), \mathsf{prim}(D_j)) = |\Delta|$ when $(\mathsf{prim}(C_i))^{\mathcal{I}} = (\mathsf{prim}(D_j))^{\mathcal{I}}$
- $f_P(\mathsf{prim}(C_i), \mathsf{prim}(D_j)) = 0$ when $PE(C_i) \cap PE(D_j) = \emptyset$

The definitions of the overlap function for the conjunctive level, value restriction and existential restrictions remain the same as in Def. 4.2.1.

Analogously, the definition of the dissimilarity measure is modified as follow:

$$d(C, D) := \begin{cases} 1 & \text{if } f'(C, D) = 0 \\ 0 & \text{if } f'(C, D) = \Delta \\ 1/f'(C, D) & \text{otherwise} \end{cases}$$

where f' is the overlap function defined just above.

The new definition of the overlap function at disjunctive level determines the overlap between the two concepts as the maximum of the overlaps among all couples of disjuncts C_i, D_j that make up the top level of the considered concepts; such amounts are multiplied by the weighting factor λ used to decrease the importance of the overlap of the sub-concepts, particularly the importance of the overlap decreases with the increasing of the nesting level. The weighting factor can be defined as a function of the level where the sub-concepts occur within the overall concept descriptions (e.g. $\lambda = 1/level$). Moreover it is important to note that the amount of the overlap (as discussed in Sect. 4.2.1) continues to be zero or greater that one. This ensure that the dissimilarity function d, for its definition, has value in the range $[0, 1]$.

Summarizing, a dissimilarity measure able to compute dissimilarity values between concepts, individuals and concept and individual described in a $\mathcal{ALC}$ knowledge base has been defined. It is grounded on a overlap function that measure the overlap between concepts by computing the amount of overlap between their sub-concepts. The importance of such amounts depends on the nesting level of the sub-concepts in the descriptions; it decreases with the increase of the nesting level. Hence, primitive concepts and restrictions play a different role in determining the dissimilarity value. Moreover, as seen in the previous section, the overlap function (and consequently also the dissimilarity measure) is based both on the semantics and on the structure of the concepts involved. It is semantic because it is grounded on the concept extensions, as retrieved from the current ABox. It is structural because the dissimilarity value is determined by computing the overlap of the sub-concepts constituting the descriptions. The use of normal form concept descriptions ensures that the computed dissimilarity value does not depends from the structural concept representations.

An important aspect to note is that the presented measure may represent a solution to one of the open questions illustrated by Borgida et al. in [29] namely "*how differences in concept structure might impact concept (dis-)similarity? For example considering the series $dist(B, B \sqcap A), dist(B, B \sqcap \forall R.A), dist(B, B \sqcap \forall R.\forall R.A)$ this should become smaller since more deeply nested restrictions ought to represent smaller differences.*" These problem has been solved by exploiting the structural approach and weighting the nesting level of sub-concepts.

4.3 An Information Content based Dissimilarity Measure for $\mathcal{ALC}$

One of the most prominent work analyzing (dis-)similarity measures for DLs concept descriptions has been proposed by Borgida et al. in [29]. The most important point argued in this paper is that the empirical success of Information Content (IC) based measures[10] for simple concept descriptions indicates that such model could be the best solution for determining the (dis-)similarity between complex concept descriptions. Furthermore, the necessity of considering concepts in normal form for computing their (dis-)similarity is also explained.

The latter argumentation confirms the choice, presented in Sect. 4.2, of considering concepts in normal form. Moving from the former argumentation, a measure grounded on the notion of IC that is able to determine the dissimilarity value between complex concept descriptions in $\mathcal{ALC}$ normal form, has been formalized. The measure [55] is defined making use of the structure and semantics of the concepts. Following the approach presented in Sect. 4.2, the measure elicits the underlying semantics, by querying the knowledge base for assessing the IC of concept descriptions w.r.t. the knowledge base, as proposed also in [17]. A function of the IC gap between concepts is first defined, hence a dissimilarity measure is derived from it. Moreover, an extension of this measure to cope with individuals and concept and individual is proposed.

4.3.1 Measuring the IC Gap between Concepts

The IC depends on the probability of an individual to belong to a certain concept (see Sect. 3.3.2). Differently from other works [17, 29] which assume that concepts are mutually independent and that a probability distribution for the concepts in an ontology is known, here, a way to derive such probability from the knowledge base, namely from the distribution that can be estimated therein, is proposed.

Specifically, in order to approximate the probability $p(C)$ for a certain concept C, the notion of concept extension w.r.t. the considered ABox in a fixed interpretation is used. The chosen interpretation is the *canonical interpretation* $\mathcal{I}$, which is the one adopting the set of individuals mentioned in the ABox as its domain and the identity as its interpretation function [8, 134]. Particularly, given a concept C, its probability is estimated by: $pr(C) = |C^{\mathcal{I}}|/|\Delta^{\mathcal{I}}|$. Once that $p(C)$ has been estimated, the IC of a concept C can be easily computed by recurring to its definition, namely : $IC(C) = -\log pr(C)$.

[10]See Sect. 3.3.2 for more details about measures based on IC.

Given these premises, a function for computing the variation of the IC between concepts in $\mathcal{ALC}$ normal form[11] is defined as follows.

Definition 4.3.1 (*IC* gap function) *Let $\mathcal{L} = \mathcal{ALC}/_{\equiv}$ be the set of all concepts in $\mathcal{ALC}$ normal form and let $\mathcal{A}$ be an ABox with canonical interpretation $\mathcal{I}$. g is a function*[12] *$g : \mathcal{L} \times \mathcal{L} \mapsto R^+$ defined recursively as follows: $\forall C, D \in \mathcal{L}$, with $C = \bigsqcup_{i=1}^{n} C_i$ and $D = \bigsqcup_{j=1}^{m} D_j$ it has:*

$$\underline{\textit{Disjunctive level}}: \quad g(C,D) := g_{\sqcup}(C,D) = \begin{cases} 0 & \textit{if } C \equiv D \\ \infty & \textit{if } C \sqcap D = \bot \\ \max_{\substack{i \in [1,n] \\ j \in [1,m]}} g_{\sqcap}(C_i, D_j) & \textit{o.w.} \end{cases}$$

$$\underline{\textit{Conjunctive level:}} \quad g_{\sqcap}(C_i, D_j) := g_P(\mathsf{prim}(C_i), \mathsf{prim}(D_j)) + g_{\forall}(C_i, D_j) + g_{\exists}(C_i, D_j)$$

$$\underline{\textit{Primitive concepts}}: \quad g_P(\mathsf{prim}(C_i), \mathsf{prim}(D_j)) := \begin{cases} \infty & \textit{if } \mathsf{pc}(C_i) \sqcap \mathsf{pc}(D_j) \equiv \bot \\ \frac{IC(\mathsf{pc}(C_i) \sqcap \mathsf{pc}(D_j)) + 1}{IC(LCS(\mathsf{pc}(C_i), \mathsf{pc}(D_j))) + 1} & \textit{o.w.} \end{cases}$$

where, $\mathsf{pc}(C_i) := (\bigsqcap_{P \in \mathsf{prim}(C_i)} P)$ and $\mathsf{pc}(D_j) := (\bigsqcap_{P \in \mathsf{prim}(D_j)} P)$ represent the primitive concepts conjunctions.

$$\underline{\textit{Value restrictions}}: \quad g_{\forall}(C_i, D_j) := \sum_{R \in N_R} g_{\sqcup}(\mathsf{val}_R(C_i), \mathsf{val}_R(D_j))$$

$$\underline{\textit{Existential restrictions}}: \quad g_{\exists}(C_i, D_j) := \sum_{R \in N_R} \sum_{k=1}^{N} \max_{p=1,\ldots,M} g_{\sqcup}(C_i^k, D_j^p)$$

where $C_i^k \in \mathsf{ex}_R(C_i)$ and $D_j^p \in \mathsf{ex}_R(D_j)$ and we suppose w.l.o.g. that $N = |\mathsf{ex}_R(C_i)| \geq |\mathsf{ex}_R(D_j)| = M$, otherwise the indices N and M are to be exchanged in the formula above.

The function g represents a measure of the variation of IC between two descriptions expressed in $\mathcal{ALC}$ normal form. Following the same criterion illustrated in Sect. 4.2, it is defined recursively beginning from the top level of the descriptions (a disjunctive level) up to the bottom level represented by (conjunctions of) primitive concepts.

[11]Every $\mathcal{ALC}$ concept description can be equivalently written in $\mathcal{ALC}$ normal form by means of semantic preserving rewriting rules (see Sect. 2.4 for more details).

[12]The name $\mathcal{A}$ of the ABox is omitted for keeping the notation as simple as possible.

In case of disjunctive descriptions three different possibilities have to be taken into account. If the considered concepts C and D are semantically equivalent then g is set to 0. This is because the knowledge of both of them rather than only one of them does not add any information. If C and D are disjoint they convey a great amount of *IC*, they could be seen as one the complementary of the other, hence the value of g, in this case, is set to ∞. If they have something in common, the amount of additional information that they supply is computed as the maximal[13] variation of the *IC* among all couples of disjuncts (C_i, D_j) that make up the top level of the considered concepts.

Since every disjunct is a conjunction of descriptions, it is necessary to calculate the *IC* gap between conjunctive concepts. This is computed as the sum of the *IC* gap among the parts that make up the conjunctive description that are primitive concepts, universal restrictions and existential restrictions.

The amount of the gap between two conjunctions of (negated) primitive concepts is the ratio of the informative content of the conjunction of the two concepts over their Least Common Subsumer (see Def. 2.4.4), which simply amounts to their disjunction in the case of $\mathcal{ALC}$ (see Sect. 2.4.1). The intuition is that the information content conveyed by two concepts is inversely proportional to their semantic similarity (measured as the overlap of the respective extensions). Specifically, the gap between two conjunctions of (negated) primitive concepts is measured as the variation of the *IC* of the considered concepts at this level w.r.t. the *IC* conveyed by their lcs. Indeed, if C and D share most of their extension, consequently $|(\mathsf{pc}(C_i) \sqcap \mathsf{pc}(D_j))^{\mathcal{I}}| \simeq |(lcs(\mathsf{pc}(C_i), \mathsf{pc}(D_j)))^{\mathcal{I}}| \Rightarrow g_P(\mathsf{prim}(C_i), \mathsf{prim}(D_j)) \simeq 1$. This means that very low *IC* gap exists between the considered conjunctions of (negated) primitive concepts. As the amount of extensions they share decreases, $IC(pc(C_i) \sqcap pc(D_j))$ increases and consequently $g_P(\mathsf{prim}(C_i), \mathsf{prim}(D_j))$ increases as well (as the denominator $IC(lcs(\mathsf{pc}(C_i), \mathsf{pc}(D_j)))$ is constant). It important to note that g_P is always greater than one.

The computation of the gap between descriptions expressed by universal and existential restrictions is defined as explained in Sect. 4.2.1. Namely, in case of conjunction of universal restrictions, the *IC* gap between concepts that are scope of universal concept restriction with respect to the same role[14] (say R) is computed by applying $g_{\sqcup}$. This operation is legal because the scope of each restriction is expressed in normal form. Of course, if a single disjunct occurs at the top level, it is possible to regard the concept description as a disjunction of a single term to which $g_{\sqcup}$ applies in a simple way. Since a conjunction of concepts with universal restrictions (one per different role) can be found, the gap of the conjuncts is given by the sum of

[13]It would be appropriate to consider the minimal dissimilarity (or an average one) as well.

[14]Remember that, in an $\mathcal{ALC}$ normal form concept, for each role there is a unique universal concept restriction, because of the rewriting rule $\forall R.A \sqcap \forall R.B \Rightarrow \forall R.(A \sqcap B)$

the IC gap yielded by each restriction. When a universal restriction on a role (S for example) occurs only in one of the descriptions (i.e. in C_i and not in D_j), then the computation assumes $\top$ as the corresponding concept in the other description (namely $\mathsf{val}_S(D_j) = \top$).

In case of conjunction of existential restrictions the *existential mapping* function $\alpha : \{1, \dots, N\} \mapsto \{1, \dots, M\}$, that maps each concept description $C_i^k \in \mathsf{ex}_R(C_i)$ to $D_j^p \in \mathsf{ex}_R(D_j)$, is used. Where $N = |\mathsf{ex}_R(C_i)|$; $M = |\mathsf{ex}_R(D_j)|$ and $N \geq M$ is supposed. Again, since each C_i^k (resp. D_j^p) is in normal form, fixed a role R and considered a certain C_i^k (with $k \in [1, N]$), the IC gap between C_i^k and D_j^p (with $p \in [1, M]$) is computed applying $g_\sqcup$. Each existential restriction on role R is coupled with the one on the same role in the other description, scoring the maximal IC gap (see footnote 13). These maxima are summed up per single role. In case of absence of role restrictions on a certain role from either description then it is considered as the top concept ($\top$).

4.3.2 Information Content based Dissimilarity Measure

After clarifying the definition of the function g and determined the IC gap between complex descriptions in $\mathcal{ALC}$ normal form, a dissimilarity measure is derived from g as shown in the following.

Definition 4.3.2 (IC based Dissimilarity Measure) *Let $\mathcal{L}$ be the set of all concepts in normal form in $\mathcal{ALC}$ and let $\mathcal{A}$ be an ABox. The* dissimilarity measure d_g *is a function $d_g : \mathcal{L} \times \mathcal{L} \mapsto [0, 1]$, such that given the concept descriptions in $\mathcal{ALC}$ normal form $C = \bigsqcup_{i=1}^n C_i$ and $D = \bigsqcup_{j=1}^m D_j$, let*

$$d_g(C, D) := \begin{cases} 0 & \text{if } g(C, D) = 0 \\ 1 & \text{if } g(C, D) = \infty \\ 1 - 1/g(C, D) & \text{otherwise} \end{cases} \tag{4.3}$$

where g is the function defined above.

The function d_g measures the level of dissimilarity between two concepts (say C and D) in $\mathcal{ALC}$ normal form, using the function g that expresses the IC gap between the two concepts, say C and D. Particularly, if $g(C, D) = 0$ this means that one concept (say D) does not add further information to that given by C, therefore $d_g(C, D) = 0$, which is the minimum value in its range. If $g(C, D) = \infty$ this means that one concept is the contrary of the other one, thus $d_g(C, D) = 1$ i.e. it amounts to the maximum value of its range, which indicates that the two concepts are totally different. In the third case the dissimilarity value is given by one minus the inverse of the amount of IC gap between C and D. This can be easily understood supposing to have concepts made by only conjunctive (negated) primitive concepts. Remembering that, as discussed in Sect. 4.3.1, if they share most of their extensions $g_P \simeq 1$ consequently, in this case, also $g_\sqcup = g \simeq 1$. Generalizing, the more the considered concepts share their extensions the less dissimilar they are, hence the dissimilarity value d_g have to be near to 0.

Proposition 4.3.3 *d_g is a dissimilarity measure for $\mathcal{ALC}/_\equiv$.*
PROOF.
Three properties of Def. 3.1.1 have to be satisfied: 1) d_g is definite positive; 2) d_g is symmetric; 3) d_g satisfies the minimality property ($\forall C, D : d_g(C, D) \geq d_g(C, C)$).
1. trivial: by construction d_g computes a dissimilarity by using sums of positive quantities and maxima computed on sets of such values;
2. by the commutativity of the operations involved;
3. by the definition of d_g, it holds that $d_g(C, C) = 0$ and $d_g(C, C') = 0$ if C is semantically equivalent to C'. In all other cases, $\forall D \in \mathcal{L}$ and D not semantically equivalent to C ($C \not\equiv D$), it results: $d_g(C, D) > 0$.

To clarify the usage of the presented measure, an example is illustrated.

Example 4.3.4 *Let C and D be the concepts used in the Example 4.2.3:*
$C \equiv A_2 \sqcap \exists R.B_1 \sqcap \forall T.(\forall Q.(A_4 \sqcap B_5)) \sqcup A_1$
$D \equiv A_1 \sqcap B_2 \sqcap \exists R.A_3 \sqcap \exists R.B_2 \sqcap \forall S.B_3 \sqcap \forall T.(B_6 \sqcap B_4) \sqcup B_2$
where A_i and B_j are all primitive concepts.

In order to compute the dissimilarity value between C and D, the IC *gap function has to be firstly computed. As in Ex. 4.2.3, C and D are neither equivalent nor disjoint. Hence the third the case of the $g_\sqcup$ has to be computed. For sake of simplicity it is denoted $C_1 := A_2 \sqcap \exists R.B_1 \sqcap \forall T.(\forall Q.(A_4 \sqcap B_5))$ and $D_1 := A_1 \sqcap B_2 \sqcap \exists R.A_3 \sqcap \exists R.B_2 \sqcap \forall S.B_3 \sqcap \forall T.(B_6 \sqcap B_4)$. The* IC *gap will be given by the maximum value of the gap computed among all couples of disjunctive terms:*

$$g(C, D) := g_\sqcup(C, D) = \max\{\ g_\sqcap(C_1, D_1), g_\sqcap(C_1, B_2), g_\sqcap(A_1, D_1), g_\sqcap(A_1, B_2)\ \}$$

For brevity, only the computation of $g_\sqcap(C_1, D_1)$ will be considered. This is computed as the sum of g_P, $g_\forall$, $g_\exists$. The computations of $g_\forall(C_1, D_1)$ and $g_\exists(C_1, D_1)$ are the same for the Ex. 4.2.3. The computation of $g_P(C_1, D_1)$ is shown here. Suppose that $A_2 \sqcap (A_1 \sqcap B_2) \not\equiv \bot$. Then:

$$g_P(\mathsf{prim}(C_1), \mathsf{prim}(D_1)) = g_P(A_2, A_1 \sqcap B_2) = \frac{IC(A_2 \sqcap (A_1 \sqcap B_2)) + 1}{IC(lcs(A_2, A_1 \sqcap B_2)) + 1}$$

Hence the value of g_P will be computed by determining the extension of $A_2 \sqcap (A_1 \sqcap B_2)$ and the extension of $A_2 \sqcup (A_1 \sqcap B_2) = lcs(A_2, A_1 \sqcap B_2)$ and then applying the definition *of* IC*,* $IC(A_2 \sqcap (A_1 \sqcap B_2)) = -log\ p(A_2 \sqcap (A_1 \sqcap B_2)) = -log(\frac{|(A_2 \sqcap A_1 \sqcap B_2)^{\mathcal{I}}|}{|\Delta^{\mathcal{I}}|})$ *and analogously for* $IC(A_2 \sqcup (A_1 \sqcap B_2))$*.*

To determine the dissimilarity of C *and* D *the same operations have to be performed for computing* $g_{\sqcap}(C_1, B_2)$*,* $g_{\sqcap}(A_1, D_1)$*,* $g_{\sqcap}(A_1, B_2)$*. Once that such amounts have been computed, the* IC *gap between* C *and* D *will be given by the maximum of the gap for all couples of disjunctive elements. Hence, the dissimilarity value is immediately computed as one minus the inverse of the chosen maximum value.* □

As seen in Sect. 4.1.1, the presented dissimilarity measure can be easily applied to compute dissimilarity value between individuals and between an individual and a concept by recurring to the notion of the (approximated) msc of an individual w.r.t. the considered ABox. The presented dissimilarity measure, as the measure presented in Sect. 4.2, is able to overcome the issues that the measure presented in Sect. 4.1 show, when the similarity between individuals has to be computed. This is obtained by giving a more structural definition of dissimilarity. Moreover a semantic approach is used. It is mainly given by the use of the notion of concept extensions when dissimilarity value between conjunctive (negated) primitive concepts is computed, which represent the base of the recursion of the definition of the measure, rather of the definition of the *IC* gap function (to which the measure is derived from). The retrieval inference operator is used for determining the concept extensions. It is important to note that, as seen in Sect. 4.2.4, also for the measure presented in this section, a modified weighted form can be thought.

4.3.3 Computational Complexity

The computational complexity of d_g is the same seen in Sect. 4.2.3 for d, as both depend from the overlap function f and the *IC* gap function g respectively, that are defined in the same way except for the base of the recursion (given by f_P and g_P respectively). The main source of complexity for f_P is represented by the double invocation of the retrieval operator whose complexity has been approximated to the complexity of the instance checking. The computation of g_P requires a double computation of the *IC* of a concept and the computation of the lcs of the two definitions. The computation of the *IC* is based on the determination of concept extensions whose complexity has been approximated with the complexity of the instance checking. The computation of the lcs is constant in time. Thus, $Compl(f_P) = Compl(g_P) = 2 \cdot Compl(IChk)$. Hence the complexity of f and g (that is the same for d and d_g), mainly depends from the complexity of the instance checking operator which is P-Space in $\mathcal{ALC}$ [68]. Anyway, as said in Sect. 4.2.3, in practical applications, these computations may be efficiently carried out exploiting the statistics maintained by the DBMSs query optimizers and the set operations.

4.4 A Semantic and Structure driven Similarity Measure for $\mathcal{ALN}$

The experimental evaluations (see Sect. 5.1.1) of the measures presented in the previous sections have shown their effectiveness in determining (dis-)similarity between concepts, individuals and concept and individual. Such measures refer to $\mathcal{ALC}$ concept descriptions. This represents an interesting result from a theoretical and practical point of view, as they are, for the best of the knowledge, among the first works, at the time of writing this thesis, assessing (dis-)similarity in quite expressive description logic. Anyway, in order to apply (dis-)similarity measures to real-world problems, numeric restrictions have to be treated. The measure that will be presented in this section has been developed moving by this consideration. Particularly, this work aims at investigating and extending previous ideas to languages endowed with numeric restrictions, starting from the simplest description logic allowing for these numeric restrictions, namely $\mathcal{ALN}$ logic.

$\mathcal{ALN}$ is a DLs language which allows for the expression of universal features and numeric constraints (see Sect. 2.2 for more details). It has been adopted because of the tractability of the main reasoning services (see Sect. 2.4 and [66] for more details). Furthermore, it has already been adopted in other frameworks for learning in hybrid representations such as CARIN-$\mathcal{ALN}$ [172] or IDLP [119].

4.4.1 Measure Definition

Using the structural notion of $\mathcal{ALN}$ normal form (see Def. 2.4.3 where the notations used in the definition are introduced) and the world-state as represented by the KB, a similarity measure for the space of (equivalent) descriptions $\mathcal{L} = (\mathcal{ALN}\ |_{\equiv})$ can be defined as follows [76]:

Definition 4.4.1 ($\mathcal{ALN}$ similarity measure) *The function $s : \mathcal{L} \times \mathcal{L} \mapsto [0, 1]$ is defined as follows. Given $C, D \in \mathcal{L}$:*

$$\begin{aligned} s(C, D) \ := \lambda \ \cdot \Bigg[& s_P(\mathsf{prim}(C), \mathsf{prim}(D)) + \frac{1}{|N_R|} \sum_{R \in N_R} s(\mathsf{val}_R(C), \mathsf{val}_R(D)) \ + \\ & + \frac{1}{|N_R|} \sum_{R \in N_R} s_N((\mathsf{min}_R(C), \mathsf{max}_R(C)), (\mathsf{min}_R(D), \mathsf{max}_R(D))) \Bigg] \end{aligned}$$

where $\lambda \in]0, 1]$ ($\lambda \leq 1/3$),

$$s_P(\mathsf{prim}(C), \mathsf{prim}(D)) := \frac{|\bigcap_{P_C \in \mathsf{prim}(C)} P_C^{\mathcal{I}} \cap \bigcap_{Q_D \in \mathsf{prim}(D)} Q_D^{\mathcal{I}}|}{|\bigcap_{P_C \in \mathsf{prim}(C)} P_C^{\mathcal{I}} \cup \bigcap_{Q_D \in \mathsf{prim}(D)} Q_D^{\mathcal{I}}|}$$

and if $\min(M_C, M_D) > \max(m_C, m_D)$ *then*

$$s_N((m_C, M_C), (m_D, M_D)) := \frac{\min(M_C, M_D) - \max(m_C, m_D) + 1}{\max(M_C, M_D) - \min(m_C, m_D) + 1}$$

else

$$s_N((m_C, M_C), (m_D, M_D)) := 0$$

The rationale for the measure is the following. Due to the relative simplicity of the language, the definition of operators working on $\mathcal{ALN}$ may be given structurally, as seen in the Sect. 2.4 for $\mathcal{ALE}$. Thus, the measure s is defined by recursively decomposing the normal form of the concept descriptions under comparison. Hence, separately, per each level, the similarity of the sub-concepts are measured, namely: primitive concepts, value restrictions, and number restrictions. The contribution of the similarity at a given level is combined jointly with a fixed[15] rate λ. Actually, in order to have s ranging over $[0, 1]$, λ should be less or equal to $1/3$.

The similarity of the primitive concept sets is computed as the ratio of the number of common individuals (belonging to both primitive conjuncts) w.r.t. the number of the individuals belonging to either conjunct. For those sub-concepts that are related through a role (say R) the similarity of the concepts made up by the fillers is computed recursively by applying the measure to $\mathsf{val}_R(\cdot)$. Finally, the similarity of the numeric restrictions is computed as a measure of the overlap between the two intervals. Namely it is the ratio of the amounts of individuals in the overlapping interval and those the larger one, whose extremes are minimum and maximum. Note that some intervals may be unlimited above: $\max = \infty$. In this case such upper level could be approximated with an upper limit N greater than $|\Delta| + 1$.

The baseline of this measure is the extension of primitive concepts. Since such extensions cannot be known beforehand due to the OWA, as for the measures illustrated in the previous sections, an epistemic adjustment is made, by assuming that it is approximated by retrieving the concept instances based on the current world-state (i.e. according to the ABox $\mathcal{A}$); formally $P^I \leftarrow \{a \in \mathsf{Ind}(\mathcal{A}) \mid I \models_{\mathcal{A}} P(a)\}$ where P is a considered concept. The interpretation is not decisive because of the *unique names assumption* (UNA) holding for the individual names. Hence, the *canonical interpretation*[16] is considered for counting the retrieved individuals.

Furthermore, it can be foreseen that, per each level, before summing the three measures assessed on the three parts, these figures be normalized. Moreover, a lowering factor $\lambda_R \in]0, 1[$ may be multiplied so to decrease the impact of the sets of individuals related to the top-level ones through some role R.

[15]Actually different rates could be assigned to the similarity of primitive concepts, the similarity of numerical restrictions and the similarity of concepts for the role fillers.

[16]The interpretation where individual names occurring in the ABox stand for themselves [8].

In order to clarify the function definition, an example is illustrated below. The example shows that the measure exploits the ABox which can be supposed complete according to the TBox descriptions (e.g. $\mathsf{Female} \sqsubseteq \neg\mathsf{Male}$).

Example 4.4.2 *Let $\mathcal{A}$ be the considered Abox defined as in the following:*

$$\mathcal{A} = \left\{ \begin{array}{l} \mathit{Person}(\mathit{Meg}), \neg\mathit{Male}(\mathit{Meg}), \mathit{hasChild}(\mathit{Meg,Bob}), \mathit{hasChild}(\mathit{Meg,Pat}), \\ \mathit{Person}(\mathit{Bob}), \mathit{Male}(\mathit{Bob}), \mathit{hasChild}(\mathit{Bob,Ann}), \\ \mathit{Person}(\mathit{Pat}), \mathit{Male}(\mathit{Pat}), \mathit{hasChild}(\mathit{Pat,Gwen}), \\ \mathit{Person}(\mathit{Gwen}), \neg\mathit{Male}(\mathit{Gwen}), \\ \mathit{Person}(\mathit{Ann}), \neg\mathit{Male}(\mathit{Ann}), \mathit{hasChild}(\mathit{Ann,Sue}), \mathit{marriedTo}(\mathit{Ann,Tom}), \\ \mathit{Person}(\mathit{Sue}), \neg\mathit{Male}(\mathit{Sue}), \\ \mathit{Person}(\mathit{Tom}), \mathit{Male}(\mathit{Tom}) \end{array} \right\}$$

and let two descriptions be:

$$\begin{array}{rcl} C & \equiv & \mathit{Person} \sqcap \forall \mathit{marriedTo}.\mathit{Person} \sqcap \leq 1.\mathit{hasChild} \\ D & \equiv & \mathit{Male} \sqcap \forall \mathit{marriedTo}.(\mathit{Person} \sqcap \neg\mathit{Male}) \sqcap \leq 2.\mathit{hasChild} \end{array}$$

Noted that $|N_R| = |\{marriedTo, hasChild\}| = 2$, and let $\lambda = 1/3$, the similarity between C and D in the knowledge base is computed as follows:

$$\begin{array}{rl} s(C,D) = \frac{1}{3} & \cdot \quad [s_P(\mathsf{prim}(C), \mathsf{prim}(D)) + \frac{1}{2}\sum_{R \in N_R} s(\mathsf{val}_R(C), \mathsf{val}_R(D)) + \\ & +\frac{1}{2}\sum_{R \in N_R} s_N((\mathsf{min}_R(C), \mathsf{max}_R(C)), (\mathsf{min}_R(D), \mathsf{max}_R(D)))] \end{array}$$

Now, the three parts are computed separately:

$$\begin{array}{rcl} s_P(\mathsf{prim}(C), \mathsf{prim}(D)) & = & s_P(\{\mathit{Person}\}, \{\mathit{Male}\}) = \\ & = & \dfrac{|\{\mathit{Meg}, \mathit{Bob}, \mathit{Pat}, \mathit{Gwen}, \mathit{Ann}, \mathit{Sue}, \mathit{Tom}\} \cap \{\mathit{Bob}, \mathit{Pat}, \mathit{Tom}\}|}{|\{\mathit{Meg}, \mathit{Bob}, \mathit{Pat}, \mathit{Gwen}, \mathit{Ann}, \mathit{Sue}, \mathit{Tom}\} \cup \{\mathit{Bob}, \mathit{Pat}, \mathit{Tom}\}|} = 3/7 \end{array}$$

For the number restrictions on role $\mathit{hasChild}$*:*

$$s_N((m_C, M_C), (m_D, M_D)) \quad = \quad s_N((0,1),(0,2)) = \frac{\min(1,2) - \max(0,0) + 1}{\max(1,2) - \min(0,0) + 1} = 2/3$$

For the number restrictions on role $\mathit{marriedTo}$*:*

$$s_N((m'_C, M'_C), (m'_D, M'_D)) = s_N((0, |\Delta| + 1), (0, |\Delta| + 1)) = s_N((0,8),(0,8)) = 1$$

As regards the value restrictions on $\mathit{marriedTo}$*,* $\mathsf{val}_{\mathsf{marriedTo}}(\mathsf{C}) = \mathit{Person}$ *and* $\mathsf{val}_{\mathsf{marriedTo}}(\mathsf{D}) = \mathit{Person} \sqcap \neg\mathit{Male}$*, hence:*

$$s(\mathit{Person}, \mathit{Person} \sqcap \neg\mathit{Male}) = 1/3 \cdot (s_P(\{\mathit{Person}\}, \{\mathit{Person}, \neg\mathit{Male}\}) + 1 + 1)$$

and

$$s_P(\{\mathsf{Person}\},\{\mathsf{Person},\neg\mathsf{Male}\}) = \frac{|\{\mathsf{Meg},\mathsf{Bob},\mathsf{Pat},\mathsf{Gwen},\mathsf{Ann},\mathsf{Sue},\mathsf{Tom}\}\cap\{\mathsf{Meg},\mathsf{Gwen},\mathsf{Ann},\mathsf{Sue}\}|}{|\{\mathsf{Meg},\mathsf{Bob},\mathsf{Pat},\mathsf{Gwen},\mathsf{Ann},\mathsf{Sue},\mathsf{Tom}\}\cup\{\mathsf{Meg},\mathsf{Gwen},\mathsf{Ann},\mathsf{Sue}\}|} = 4/7$$

As there are no value restrictions on **hasChild***, the similarity is maximal (*$\mathsf{val}_{\mathsf{hasChild}}(\mathsf{C}) = \mathsf{val}_{\mathsf{hasChild}}(\mathsf{D}) = \top$*).*

Summing up:

$$s(C,D) \;=\; \frac{1}{3}\left[\frac{3}{7}+\frac{1}{2}\left(\frac{1}{3}\left(\frac{4}{7}+1+1\right)+\frac{1}{3}(1+1+1)\right)+\frac{1}{2}\left(1+\frac{2}{3}\right)\right] = \frac{92}{126} \simeq .7301 \;\square$$

4.4.2 Discussion

It can be proven that s is really a similarity measure by demonstrating the three properties of Def. 3.1.1: 1) s is definite positive; 2) s is symmetric; 3) s satisfies the minimality property ($\forall\, C, D\ :\ s(C,D) \leq s(C,C)$)

Proposition 4.4.3 *The function s is a similarity measure for the space $\mathcal{L}$.*

PROOF: the three properties of the definition are proved in the following

1. *It is straightforward to see that s is positive definite since it is defined recursively as a sum of non-negative values.*

2. *s is also symmetric because of the commutativity of the operations involved, namely sum, minimum, and maximum (note that the value of s_N in Def. 4.4.1 does not change by exchanging C with D).*

3. *It has to be proven that $\forall C, D \in \mathcal{L}:\ \ s(C,D) \leq s(C,C)$. This property can be proved by structural induction on D. The base cases are those related to primitive concepts and number restrictions, the inductive ones are those related to value restrictions and conjunctions:*

   - *if D is primitive then* $s(C,D) = \lambda[s_P(\mathsf{prim}(C),\mathsf{prim}(D)) + s_1 + s_2] \leq$ $\lambda[\frac{|\bigcap_{P_C\in\mathsf{prim}(C)} P_C^{\mathcal{I}} \cap \bigcap_{Q_D\in\mathsf{prim}(D)} Q_D^{\mathcal{I}}|}{|\bigcap_{P_C\in\mathsf{prim}(C)} P_C^{\mathcal{I}} \cup \bigcap_{Q_D\in\mathsf{prim}(D)} Q_D^{\mathcal{I}}|} + 1 + 1] \leq$ $\lambda[\frac{|\bigcap_{P_C\in\mathsf{prim}(C)} P_C^{\mathcal{I}} \cap \bigcap_{P_C\in\mathsf{prim}(C)} P_C^{\mathcal{I}}|}{|\bigcap_{P_C\in\mathsf{prim}(C)} P_C^{\mathcal{I}} \cup \bigcap_{P_C\in\mathsf{prim}(C)} P_C^{\mathcal{I}}|} + 1 + 1] = \lambda[1+1+1] = s(C,C).$
   - *if D is a number restriction the proof is analogous to the previous one, observing that* $0 \leq \frac{\min(M_C,M_D)-\max(m_C,m_D)}{\max(M_C,M_D)-\min(m_C,m_D)} \leq \frac{\min(M_C,M_C)-\max(m_C,m_C)}{\max(M_C,M_C)-\min(m_C,m_C)} \leq 1$

- *if D is a value restriction, then supposing by induction hypothesis that the property holds for descriptions whose depth is less than D's depth. This is the case of the sub-concept $\mathsf{val}_R(D)$.*
  Thus $s(\mathsf{val}_R(C), \mathsf{val}_R(D)) \leq s(\mathsf{val}_R(C), \mathsf{val}_R(C))$ from which it is possible to conclude that the property holds.
- *if D is a conjunction of two simpler concepts, say $\exists D_1, D_2 \in \mathcal{L} : D = D_1 \sqcap D_2$, then assuming by induction hypothesis that the property holds for descriptions whose depth is less than D's depth such as $D_{1,2}$. This means that $\forall i \in \{1,2\} : s(C, D_i) \leq s(C, C)$. It can be proven that $\forall i \in \{1,2\} : s(C, D) \leq s(C, D_i)$. Hence the property holds.* $\square$

Note that, following [26], a dissimilarity measure can be easily derived from s.

Definition 4.4.4 ($\mathcal{ALN}$ dissimilarity measure) *The* dissimilarity function $d : \mathcal{L} \times \mathcal{L} \mapsto [0,1]$ *is defined as follows. Given $C, D \in \mathcal{L}$:*

$$d(C, D) = 1 - s(C, D)$$

From a computational point of view, as the approach presented in Sect. 4.2.3 has been followed, consequently, in the same way it is possible to assert that the computational complexity of the measure presented in this section strongly depends from the computational complexity of the instance checking operator for $\mathcal{ALN}$. Anyway, in order to control the computational cost of these functions, it could be assumed that the retrieval of the primitive concepts is computed beforehand, on the ground of the current knowledge base, and then the similarity measure can be computed bottom-up through a procedure based on dynamic programming.

Moreover, as seen in the previous sections, the presented measure can be extended for assessing similarity between individuals, and between a concept and an individual, by recurring to the (approximated) msc of an individual w.r.t. an ABox.

The presented measure can be refined introducing a weighting factor, useful for decreasing the impact of the similarity between nested sub-concepts in the descriptions on the determination of the overall value.

Another natural extension may concern the definition of (dis-)similarity measures for more expressive languages. For example, a normal form for $\mathcal{ALCN}$ can be obtained based on those for $\mathcal{ALN}$ and $\mathcal{ALC}$. Then, by exploiting the presented measures for $\mathcal{ALC}$ and the current measure, a new measure for $\mathcal{ALCN}$ could be obtained. Anyway, doing this, some aspects have to be considered such as, how to manage the expression $\exists R.A \sqcap \geq 3R$. Indeed in this cases an approach that considers separately the different kinds of restriction is not semantically correct, being one restriction related to the other one.

4.5 A Relational Kernel Function for $\mathcal{ALC}$

As discussed in Sects. 3.2.2, 3.3.6, kernel functions are a means to express a notion of similarity is some feature space. Generally, they are used jointly with a kernel method. Kernel methods are a family of efficient and effective algorithms (including the *support vector machines* – SVMs) that have been applied to a variety of problems and recently also to those requiring structured representations[17]. One of the advantages of kernel methods is that the learning algorithm (inductive bias) and the choice of the kernel (language bias) are almost completely independent. Thus, an efficient algorithm for attribute-value instance spaces can be converted into one suitable for structured spaces (e.g. trees, graphs) by merely replacing the kernel function. This motivates the increasing interest addressed to the SVMs and other kernel methods that reproduce learning in high-dimensional spaces while working as in a vectorial (propositional) representation.

Hence, the definition of kernel functions for structured data, parametrized on a description language, allows for the employment of algorithms such as SVM that can simulate feature generation. These functions transform the initial representation of the instances into the related active features, thus making possible to learn the classifier (in the case of SVM) directly from the structured data.

Moving from such a intuition and considering the efficiency and effectiveness that characterize kernel methods, a kernel function for $\mathcal{ALC}$ concept descriptions has been defined. It is based both on the syntactic structure (exploiting the *convolution* kernel; see Sect. 3.3.6) and on the semantics which can be derived from the knowledge base (in particular, from the ABox). It is proved to be valid and efficiently computable, that it makes is eligible for kernel machines.

Furthermore, it is important to note that the original algorithm for feature extraction produces only active features acting as positive examples for the adopted propositional learners, thus making a sort of CWA, which contrasts with the mainstream in DLs reasoning: an inactive feature should be explicitly inferred from the knowledge base. By allowing negation in the language, it becomes natural to represent also negative examples. Moreover, since kernels reflect a notion of similarity in an unknown embedded space of features, distance measures can be derived from them[18].

[17]See Sect. 3.3.6 for more details about relational kernel functions.

[18]See Sect. 3.3.6 for the motivation about the derivation of a distance measure from a kernel.

4.5.1 Kernel Function Definition

In this section the definition of the kernel function applied to $\mathcal{ALC}$ concept definitions will be presented. Based on convolution kernel (that deals with compound data by decomposing structured data into parts, for which valid kernels have already been defined), it is structure driven. It is also semantic based, indeed the notion of concept extensions is used in its definition.

In order to avoid dependance of the kernel function from the syntactic concept representations, concepts in $\mathcal{ALC}$ normal form are considered. This guarantees that semantically equivalent concepts are also written in the same way[19]. Considered an $\mathcal{ALC}$ normal form description, a related $\mathcal{ALC}$ *description tree* can be derived (as seen in Sect. 2.4.1 for less expressive description logics). It is an AND-OR tree, rooted in a disjunctive node (according to the normal form) and made by alternate disjunctive and conjunctive nodes. Conjunctive nodes are labeled by a $\mathsf{prim}(\cdot)$ set and, per each $R \in N_R$, branching one edge labeled $\forall R$ to the subtree of $\mathsf{val}_R(\cdot)$ and for each $E \in \mathsf{ex}_R(\cdot)$, an edge labeled $\exists R$ to the subtree related to E. An empty label in a node is equivalent to the top concept.

Hence, one way for defining a kernel for $\mathcal{ALC}$ descriptions, could be to formalize a kernel based on the AND-OR tree structure of the descriptions, like for the standard tree kernels [83] where (as seen in Sect. 3.3.6) similarity between trees depends on the number of similar subtrees (or paths unraveled from such trees). Yet this would end in a merely structural measure which does not fully capture the semantic nature of expressive DLs languages such as $\mathcal{ALC}$. The adopted approach, by recurring to the idea of the convolution kernel (see Sect. 3.3.6), uses the normal form to decompose complex descriptions level-wise into sub-descriptions. Valid kernels already known, or kernels obtained by convolution are then applied to the sub-descriptions. Particularly, for each level three different situations can be found: the level is dominated by the disjunction of concepts, the level is a conjunction of concepts, the level is simply made by primitive concepts. In this way. a family of valid kernels for the space X of $\mathcal{ALC}$ descriptions can be obtained. In the following, the kernel function is formally defined [75].

Definition 4.5.1 ($\mathcal{ALC}$ kernel) *Given two concept descriptions in normal form $D_1 = \bigsqcup_{i=1}^{n} C_i^1$ and $D_2 = \bigsqcup_{j=1}^{m} C_j^2$, and an interpretation $\mathcal{I}$, the $\mathcal{ALC}$* kernel *based on $\mathcal{I}$ is the function $k_{\mathcal{I}} : X \times X \mapsto \mathbb{R}$ inductively defined as follows.*

disjunctive descriptions: $k_{\mathcal{I}}(D_1, D_2) = \lambda \sum_{i=1}^{n} \sum_{j=1}^{m} k_{\mathcal{I}}(C_i^1, C_j^2)$ *with* $\lambda \in]0, 1]$

[19]This is guaranteed since concepts in a single ontology are considered, so the same vocabulary is used for defining several concepts.

conjunctive descriptions:

$$k_{\mathcal{I}}(C^1, C^2) = \prod_{\substack{P_1 \in \mathsf{prim}(C^1) \\ P_2 \in \mathsf{prim}(C^2)}} k_{\mathcal{I}}(P_1, P_2) \cdot \prod_{R \in N_R} k_{\mathcal{I}}(\mathsf{val}_R(C^1), \mathsf{val}_R(C^2)) \cdot$$

$$\cdot \prod_{R \in N_R} \sum_{\substack{C^1_i \in \mathsf{ex}_R(C^1) \\ C^2_j \in \mathsf{ex}_R(C^2)}} k_{\mathcal{I}}(C^1_i, C^2_j)$$

primitive concepts: $k_{\mathcal{I}}(P_1, P_2) = k_{\text{set}}(P_1^{\mathcal{I}}, P_2^{\mathcal{I}})/|\Delta^{\mathcal{I}}| = |P_1^{\mathcal{I}} \cap P_2^{\mathcal{I}}|/|\Delta^{\mathcal{I}}|$

where k_{set} is the kernel for set structures defined in [83]. This case includes also the negation of primitive concepts using set difference: $(\neg P)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus P^{\mathcal{I}}$

The rationale for this kernel is that the similarity between disjunctive descriptions is treated by taking the sum of the cross-similarities between all couple of disjuncts from either description. The term λ is employed to downweight the similarity of the sub-descriptions on the grounds of the level where they occur.

As regards the conjunctive level, the kernel computes the similarity between two input descriptions, distinguishing among primitive concepts, those referred in the value restrictions and those referred in the existential restrictions. These similarity values are multiplied reflecting the fact that all the restrictions have to be satisfied at a conjunctive level.

The similarity between primitive concepts is measured in terms of the intersection of their extensions, weighted by the number of individuals occurring in the ABox. Another possible weight for the rate of similarity could be $1/|P_1^{\mathcal{I}} \cup P_2^{\mathcal{I}}|$ which means weight the similarity value with the size of the concepts, measured in terms of the individuals belonging to their extensions. Remind that the *unique names assumption* is made on the individuals occurring in the ABox $\mathcal{A}$. Hence, it is possible to take into account the canonical interpretation $\mathcal{I}$, using $\mathsf{Ind}(\mathcal{A})$ as its domain ($\Delta^{\mathcal{I}} = \mathsf{Ind}(\mathcal{A})$).

In order to clarify the definition of the presented kernel function, an example is presented below.

Example 4.5.2 *Consider the following descriptions (whose trees are depicted in Fig. 4.3):*

$C \equiv (P_1 \sqcap P_2) \sqcup (\exists R.P_3 \sqcap \forall R.(P_1 \sqcap \neg P_2))$

$D \equiv P_3 \sqcup (\exists R.\forall R.P_2 \sqcap \exists R.\neg P_1)$

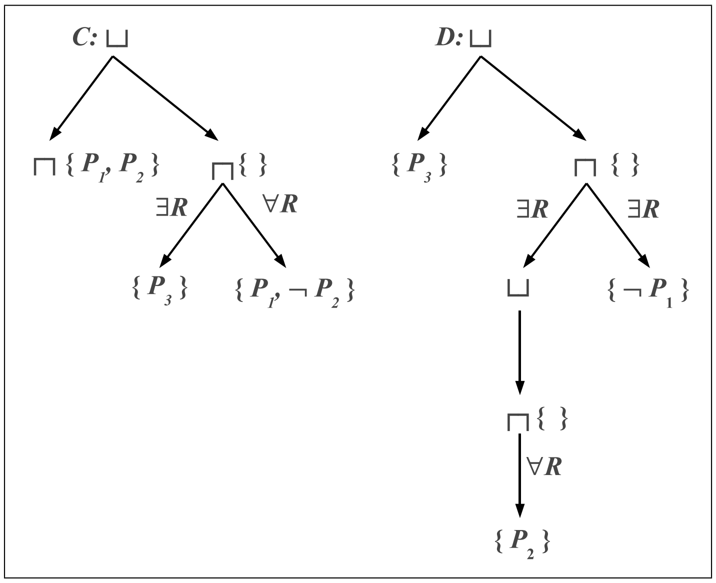

Figure 4.3: The (compacted) tree representation for the descriptions used in Example 4.5.2.

Note that the parenthesis have been employed to emphasize the level-wise structure of the descriptions.

Now, suppose

$P_1^{\mathcal{I}} = \{a, b, c\}, \quad P_2^{\mathcal{I}} = \{b, c\}, \quad P_3^{\mathcal{I}} = \{a, b, d\}, \quad \Delta^{\mathcal{I}} = \{a, b, c, d, e\}$

The application of the kernel function to such concepts will be as in the following. Since C and D are already in normal form, the first step will be consist in computing $k_{\mathcal{I}}$ for every couple of disjunct, namely:

$$k_{\mathcal{I}}(C, D) = \lambda \sum_{i=1}^{2} \sum_{j=1}^{2} k_{\mathcal{I}}(C_i, D_j) = \lambda \cdot (k_{\mathcal{I}}(C_1, D_1) + k_{\mathcal{I}}(C_1, D_2) + k_{\mathcal{I}}(C_2, D_1) + k_{\mathcal{I}}(C_2, D_2))$$

where

$C_1 \equiv P_1 \sqcap P_2, \quad C_2 \equiv \exists R.P_3 \sqcap \forall R.(P_1 \sqcap \neg P_2),$
$D_1 \equiv P_3, \quad D_2 \equiv \exists R.\forall R.P_2 \sqcap \exists R.\neg P_1.$

Noted that C_1, C_2, D_1, D_2 are conjunctive, the kernel for the conjunctive level has to be compute for every couple C_i, D_j as follows:

$$\begin{aligned}
k_{\mathcal{I}}(C_1, D_1) &= \prod_{P_1^C \in \mathsf{prim}(C_1)} \prod_{P_1^D \in \mathsf{prim}(D_1)} k_{\mathcal{I}}(P_1^C, P_1^D) \cdot k_{\mathcal{I}}(\top, \top) \cdot k_{\mathcal{I}}(\top, \top) = \\
&= k_{\mathcal{I}}(P_1, P_3) \cdot k_{\mathcal{I}}(P_2, P_3) \cdot 1 \cdot 1 = \\
&= \frac{|\{a, b, c\} \cap \{a, b, d\}|}{a, b, c, d, e} \cdot \frac{|\{b, c\} \cap \{a, b, d\}|}{a, b, c, d, e} = \frac{2}{5} \cdot \frac{1}{5} = \frac{2}{25}
\end{aligned}$$

No contribution comes from value and existential restrictions: the factors amount to 1 since $\mathsf{val}_R(C_1) = \mathsf{val}_R(D_1)) = \top$ *and* $\mathsf{ex}_R(C_1) = \mathsf{ex}_R(D_1) = \emptyset$ *which make those equivalent to* $\top$ *too.*

Hence, the conjunctive kernel on next couple of disjuncts, namely C_1 *and* D_2 *has to be computed. In this case note that there are no universal restrictions, moreover* $N_R = \{R\} \Rightarrow |N_R| = 1$ *this means that all products on varying* $R \in N_R$ *can be simplified because they are products of a single element.*

$$\begin{aligned}
k_{\mathcal{I}}(C_1, D_2) &= [k_{\mathcal{I}}(P_1, \top) \cdot k_{\mathcal{I}}(P_2, \top)] \cdot k_{\mathcal{I}}(\top, \top) \cdot \sum_{\substack{E_C \in \mathsf{ex}_R(C_1) \\ E_D \in \mathsf{ex}_R(D_2)}} k_{\mathcal{I}}(E_C, E_D) = \\
&= (3 \cdot 2) \cdot 1 \cdot [k_{\mathcal{I}}(\top, \forall R.P_2) + k_{\mathcal{I}}(\top, \neg P_1)] = \\
&= 6 \cdot [\lambda \sum_{\substack{C' \in \{\top\} \\ D' \in \{\forall R.P_2\}}} k_{\mathcal{I}}(C', D') + 2] = \\
&= 6 \cdot [\lambda \cdot (1 \cdot k_{\mathcal{I}}(\top, P_2) \cdot 1) + 2] = \\
&= 6 \cdot [\lambda \cdot (\lambda \cdot 1 \cdot 2/5 \cdot 1) + 2] = 6 \cdot (2\lambda^2/5 + 2) = 12(\lambda^2/5 + 1)
\end{aligned}$$

Note that an empty prim *is equivalent to* $\top$.

Now, it is computed:

$$\begin{aligned}
k_{\mathcal{I}}(C_2, D_1) &= k_{\mathcal{I}}(\top, P_3) \cdot k_{\mathcal{I}}(\mathsf{val}_R(C_2), \top) \cdot \sum_{\substack{E_C \in \mathsf{ex}_R(C_2) \\ E_D \in \mathsf{ex}_R(D_1)}} k_{\mathcal{I}}(E_C, E_D) = \\
&= 3/5 \cdot k_{\mathcal{I}}(P_1 \sqcap \neg P_2, \top) \cdot k_{\mathcal{I}}(P_3, \top) = \\
&= 3/5 \cdot [\lambda(k_{\mathcal{I}}(P_1, \top) \cdot k_{\mathcal{I}}(\neg P_2, \top))] \cdot 3/5 = \\
&= 3/5 \cdot [\lambda(3/5 \cdot 3/5)] \cdot 3/5 = 81\lambda/625
\end{aligned}$$

Note that, again, the absence of the prim *set is equivalent to* $\top$ *and, since one of the sub-concepts has no existential restriction the product gives no contribution.*

Finally, the kernel function on the last couple of disjuncts is computed

$$\begin{aligned}
k_{\mathcal{I}}(C_2, D_2) &= k_{\mathcal{I}}(\top, \top) \cdot k_{\mathcal{I}}(P_1 \sqcap \neg P_2, \top) \cdot \sum_{\substack{C'' \in \{P_3\} \\ D'' \in \{\forall R.P_2, \neg P_1\}}} k_{\mathcal{I}}(C'', D'') = \\
&= 1 \cdot 9\lambda/25 \cdot [(k_{\mathcal{I}}(P_3, \forall R.P_2) + k_{\mathcal{I}}(P_3, \neg P_1)] = \\
&= 9\lambda/25 \cdot [\lambda \cdot k_{\mathcal{I}}(P_3, \top) \cdot k_{\mathcal{I}}(\top, P_2) \cdot k_{\mathcal{I}}(\top, \top) + 1/5] = \\
&= 9\lambda/25 \cdot [\lambda \cdot 3/5 \cdot 2\lambda/5 \cdot 1 + 1/5] = 9\lambda/25 \cdot [6\lambda^2/25 + 1/5]
\end{aligned}$$

Again the absence of restrictions was evaluated as the top concept.

By collecting the four intermediate results, the value for the computed kernel function on definitions of C and D can be computed:

$$k_{\mathcal{I}}(C, D) = 2/25 + 12(\lambda^2/5 + 1) + 81\lambda/625 + 9\lambda/25 \cdot [6\lambda^2/25 + 1/5$$

□

Observe that the function could be also specialized to take into account the similarity between different relationships. This would amount to consider each couple of existential and value restrictions with one element from each description (or equivalently from each related AND-OR tree) and computing the convolution of the sub-descriptions in the restriction. As previously suggested for λ, this should be weighted by a measure of similarity between the roles, measured on the grounds of the available semantics. Particularly, given two roles $R, S \in N_R$, a possible weight can be: $\lambda_{RS} = |R^{\mathcal{I}} \cap S^{\mathcal{I}}|/|\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}|$. Anyway, in the same way seen before, the intersection could be also measured on the grounds of the relative role extensions with respect to the whole domain of individuals, as follows: $\lambda_{RS} = |R^{\mathcal{I}} \cap S^{\mathcal{I}}|/|R^{\mathcal{I}} \cup S^{\mathcal{I}}|$. It is also worthwhile to recall that some DLs knowledge bases support also the so called *R-box* (see Sect. 2.2) with assertions concerning the roles, thus it could be possible to know beforehand that say $R \sqsubseteq S$, and compute their similarity consequently.

As seen for the measures presented in the previous sections, the kernel can be simply extended to the case of individuals $a, b \in \mathsf{Ind}(\mathcal{A})$ by taking into account their (approximated) most specific concepts, namely $k_{\mathcal{I}}(a, b) = k_{\mathcal{I}}(msc(a), msc(b))$. In this way, it is possible to move from a graph representation, like the ABox portion containing an individual, to an intensional tree-structured representation.

In the next section the validity of the kernel will be proven. Then a distance measure, derived from the $\mathcal{ALC}$ kernel function is presented.

4.5.2 Discussion

The attractiveness of kernel methods comes from the fact that efficient learning can be applied to highly dimensional feature spaces by transforming relational data by means of a certain function ϕ. Particularly, kernel methods are able to perform learning leaving unknown this function. They really require only of a method for computing the inner product in the original feature space. Valid kernels are those which can be embedded in a linear space.

In this section, the validity of the presented $\mathcal{ALC}$ kernel function is proved. A kernel function is valid (see Def. 3.2.5) if it is symmetric and *positive definite*, namely for any $n \in \mathbb{Z}^+$, $x_1, \ldots, x_n \in X$ and $(c_1, \ldots, c_n) \in \mathbb{R}^n$: $\sum_{i,j=1}^{n} c_i c_j k(x_i, x_j) \geq 0$. Positive definiteness is often hard to be proved. Anyway, as argued in Sect. 3.2.2, it can be also demonstrated by exploiting some closure properties of the class of positive definite kernel functions. Namely, multiplying a valid kernel by a constant, adding or multiplying two valid kernels yields another valid kernel. Here, the validity of the presented kernel for the considered hypothesis space X of $\mathcal{ALC}$ descriptions in normal form will be proved.

Observe that the core function is that relative to primitive concept extensions. It is essentially a set kernel [83] which has been proven to be valid. The versions for top-level conjunctive and disjunctive descriptions are also positive definite being essentially based on the kernel on primitive concepts levels. Descending through the levels there is an interleaving of the employment of these functions up the basic case of the kernel for primitive descriptions.

Proposition 4.5.3 *The function $k_{\mathcal{I}}$ is a valid kernel for the space X of $\mathcal{ALC}$ normal form descriptions.*

PROOF: The kernel validity is verified by demonstrating symmetry and positive definiteness for $k_{\mathcal{I}}$.

(symmetry) The kernel function $k_{\mathcal{I}}$ is symmetric because of the commutativity of the operations involved in its computation.

(positive definiteness) This can be proven by structural induction on the depth d of the descriptions (maximum number of levels).

- *($d = 0$) This case is related to the function for primitive descriptions (represented as a flat tree). It is a convolution based on the set kernel between single primitive concept extensions. The multiplication by a constant ($1/|\Delta^{\mathcal{I}}|$) does not affect this property.*

- *($d > 0$) Assuming by hypothesis that $k_{\mathcal{I}}$ is positive definite for descriptions of depth up to $d - 1$. Now, if the descriptions are in normal form, $k_{\mathcal{I}}$ can be computed based on the disjunctive or the conjunctive version (depending on the top-level). In the former case, a convolution of a function for simpler descriptions (maximum depth $< d$) is verified that is positive definite by hypothesis of induction. Hence it must be positive definite for this case also, by the closure of the class of positive definite functions.*
  In the latter case, a product of positive definite functions for primitive (first factor) or sums of positive definite functions for simpler descriptions (maximum depth $< d$) is required, which is positive definite by hypothesis of induction. Again, by the closure of the class of positive definite functions, the function at depth d is positive definite.

By induction, then, the function is positive definite for descriptions of any depth $\Box$

As regards the efficiency, by pre-calculating the extension of all primitive concepts, and so applying set operations for computing the kernel function at primitive level, the kernel function $k_{\mathcal{I}}$ can be computed in time $O(|N_1||N_2|)$ where $|N_i|$, $i = 1, 2$, is the number of nodes of the concept AND-OR trees, by means of dynamic programming.

Summarizing, the presented kernel function for $\mathcal{ALC}$ descriptions has been proved to be valid. Differently from other kernel functions for structured data presented in the literature, that are mainly structure driven, it is both structure and semantic-driven. Moreover, for the best knowledge of the writer, it represent one of the first kernel function able to cope with expressive DL. Furthermore, being $k_{\mathcal{I}}$ valid, it constitutes the basis for the definition of semantic distances suitable for the same representation. Such distances could be employed in unsupervised as well as supervised methods for learning from $\mathcal{ALC}$ knowledge base.

The main weakness of the approach is on its scalability towards more complex DL languages. While computing msc approximations might be feasible, it may be more difficult to define a normal form for comparing descriptions. Indeed, as long as the expressive power increases, it also becomes more redundant, hence the semantics becomes more and more detached from the syntactic structure of the descriptions. Anyway the final goal could be to extend the presented kernel function towards more expressive languages like $\mathcal{SHIQ}$ or $\mathcal{SHOIN}(\mathcal{D})$ for a full support to the OWL ontology language.

4.5.3 A Distance Induced from the Kernel Function

Given a kernel function, it is possible to define an induced distance measure[20]. It is derived as follows.

Definition 4.5.4 (Induced Distance) *Given two descriptions C and D, and the canonical interpretation $\mathcal{I}$, the* induced distance*, based on the $\mathcal{ALC}$ kernel $k_{\mathcal{I}}$, is the function $d_{\mathcal{I}} : X \times X \mapsto \mathbb{R}$ defined as follows:*

$$d_{\mathcal{I}}(C, D) = \sqrt{k_{\mathcal{I}}(C, C) - 2k_{\mathcal{I}}(C, D) + k_{\mathcal{I}}(D, D)}$$

Moreover, as said in Sect. 3.3.6, being $k_{\mathcal{I}}$ a valid kernel (as proved in the previous section), it follows that the induced distance $d_{\mathcal{I}}$ is a metric, namely the following three properties are satisfied: (1) $d_{\mathcal{I}}(C, D) \leq d_{\mathcal{I}}(C, E) + d_{\mathcal{I}}(E, D)$, (2) $d_{\mathcal{I}}(C, D) = d_{\mathcal{I}}(D, C)$, (3) $C = D \Leftrightarrow d_{\mathcal{I}}(C, D) = 0$.

As proposed in [99], the normalization of kernels to the range $[0, 1]$ can be very important in practical cases. By normalizing the $\mathcal{ALC}$ kernel function as follows:

$$\widetilde{k}_{\mathcal{I}}(C, D) = \frac{k_{\mathcal{I}}(C, D)}{\sqrt{k_{\mathcal{I}}(C, C)}\sqrt{k_{\mathcal{I}}(D, D)}}$$

it is obtained a valid kernel such that: $0 \leq \widetilde{k}_{\mathcal{I}}(C, D) \leq 1$. This allows to derive another distance measure, called *radial distance*, which is induced by the normalized $\mathcal{ALC}$ kernel function in the following way:

$$d^2_{\mathcal{I}}(C, D) = \frac{1}{2}(\log \widetilde{k}_{\mathcal{I}}(C, C) + \log \widetilde{k}_{\mathcal{I}}(D, D)) - \log \widetilde{k}_{\mathcal{I}}(C, D)$$

These distances may have lots of practical applications spanning from instance-based classification and clustering to raking of query answers retrieved in a knowledge bases. This constitutes a new approach in the Semantic Web area. Indeed in this way it is possible to combine the efficiency of the numerical approaches and the effectiveness of a symbolic representation. Instead currently, almost all tasks applied to the Semantic Web area use a logic-based approach, mainly based on deductive reasoning. Moreover the presented function can be used in inductive tasks i.e. classify individuals of an A-Box. This can help to make complete large ontology that are incomplete hence it is possible to apply deductive reasoning to such complete knowledge bases.

The main source of complexity for the presented function is the computation of the concepts extensions. However this complexity could be decreased by precalculating the extension of all primitive concepts, and then using these as sets and apply set operations.

[20]See Sect. 3.3.6 and [83] for more details.

4.6 A Semantic Semi-Distance for Individuals in Any DLs Knowledge Base

In the previous sections of the current chapter, many similarity and dissimilarity measures for specific DL concept descriptions have been proposed. They can be used in order to measure (dis-)similarity between concepts, individuals and between a concept and an individual. Although they turned out to be quite effective, as will be shown in Sect. 5, most of them are characterized by a drawback that is they are partly based on structural criteria (given by the use of the normal form). This determines their main weakness: they are hardly scalable to deal with standard languages, such as OWL-DL, commonly used for knowledge bases.

In this section, a semi-distance measure for assessing dissimilarity between individuals asserted in an ontology is presented, and it is shown that it is able to overcome the limitations illustrated above. It is grounded on the principles on which the Hypothesis-driven distances (See Sect. 3.3.5) have been defined and can be applied to a wide range of ontology languages (RDF through OWL) since it is merely based on the discernibility of the input individuals w.r.t. a fixed set of features, that are represented by concept definitions (hypotheses). As such, the new measure totally depends on semantic aspects of the individuals in the knowledge base. It is not absolute, on the contrary it depends on the knowledge base it is applied to. In the following the measure is formally defined.

4.6.1 Measure Definition

The measures presented in the previous sections make use of the (approximated) msc of the individuals in order to compute their (dis-)similarity value. This is because individuals do not have a syntactic structure that can be compared and consequently, to lift them to the concept level before the comparison is fundamental.

On the contrary, the main intuition on which the semi-distance measure is defined is that *on a semantic level, similar individuals should behave similarly with respect to the same concepts*. Hence, the (dis-)similarity of individuals in a knowledge base can be computed by comparing their semantics along a number of dimensions represented by a committee of concept descriptions. This way of assessing (dis-)similarity between individuals can be easily seen as the way of determining distances between examples by the use of the Hypotheses-driven distances (HDDs) [182]. In the current context, hypothesis can be represented by (some of) the concepts of the knowledge base. Thus, following the ideas borrowed from HDDs, a totally semantic distance measures can be defined, that is able to assess dissimilarity value between individuals in the context of a knowledge base.

More formally, the rationale of the new measure is to compare individuals on the grounds of their behavior w.r.t. a given set of hypotheses, that is a collection of (primitive or defined) concept descriptions, say $\mathsf{F} = \{F_1, F_2, \ldots, F_m\}$, which stands as a group of discriminating *features* expressed in the language taken into account.

Consequently, by the use of the HDDs definitions and the Minkowski's distance, a family of distance functions for individuals asserted in a knowledge base can be defined as follows:

Definition 4.6.1 (Family of Semi-Distance Measures) *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a knowledge base and let $\mathsf{Ind}(\mathcal{A})$ be the set of the individuals occurring in $\mathcal{A}$. Given sets of concept descriptions $\mathsf{F} = \{F_1, F_2, \ldots, F_m\}$ in $\mathcal{T}$, a family of semi-distance functions $d_p^{\mathsf{F}} : \mathsf{Ind}(\mathcal{A}) \times \mathsf{Ind}(\mathcal{A}) \mapsto \mathbb{R}$ is defined as follows:*

$$\forall a, b \in \mathsf{Ind}(\mathcal{A}) \quad d_p^{\mathsf{F}}(a, b) := \frac{1}{m} \left[\sum_{i=1}^{m} \mid \pi_i(a) - \pi_i(b) \mid^p \right]^{1/p}$$

where $p > 0$ and $\forall i \in \{1, \ldots, m\}$ the projection function *π_i is defined by:*

$$\forall a \in \mathsf{Ind}(\mathcal{A}) \quad \pi_i(a) = \begin{cases} 1 & F_i(x) \in \mathcal{A} \\ 0 & \neg F_i(x) \in \mathcal{A} \\ \frac{1}{2} & \textit{otherwise} \end{cases}$$

The superscript F will be omitted when the set of hypotheses is fixed.

Note that a family of semi-distance measures is obtained (rather than a single measure), because the measure d_P^{F} depends on the chosen set of hypotheses F. Moreover, since the definition of d_p^{F} is based on the Minkowski's distance, its interpretation can be easily understood in geometrical terms as seen in Sect. 3.2.1. Intuitively speaking, more similar the considered individuals are, more similar the project function values are, consequently the difference of the projection values will be close to 0 and the value of the semi-distance measure will be close to 0 as well. On the contrary, more different the considered individuals are, more different the projection values are, hence their differences will increase and consequently the computed value of d_p^{F} will increase as well.

Moreover, remembering that Euclidean distance and Manhattan distance are specialization of Minkowski distance, obtained by setting $p = 2$ and $p = 1$ respectively, the following versions of d_p^{F} can be considered:

$$\forall a, b \in \mathsf{Ind}(\mathcal{A}) \quad d_1(a, b) := \frac{1}{m} \sum_{i=1}^{m} \mid \pi_i(a) - \pi_i(b) \mid$$

or:

$$\forall a, b \in \mathsf{Ind}(\mathcal{A}) \quad d_2(a,b) := \frac{1}{m}\sqrt{\sum_{i=1}^{m} (\pi_i(a) - \pi_i(b))^2}$$

Note that the definition of the measures can be made more accurate by considering entailment rather than the simple ABox look-up, when determining the values of the projection functions:

$$\forall a \in \mathsf{Ind}(\mathcal{A}) \quad \pi_i(a) = \begin{cases} 1 & \mathcal{K} \models F_i(x) \\ 0 & \mathcal{K} \models \neg F_i(x) \\ \frac{1}{2} & \textit{otherwise} \end{cases}$$

Obviously, this requires more computational effort than the simple ABox look-up.

An important assumption made here is that the feature-set F represents a sufficient number of (possibly redundant) features that are able to discriminate really different individuals. This is because, as seen in Sect. 3.3.5, HDDs do not present any interest whenever they are based on a concise set of hypotheses: e.g. the function *dist* (see Sect. 3.3.5) gets rather coarse if any example is covered by a single hypothesis. Then the granularity of a HDD increases with the redundancy of the set of hypotheses (i.e. the average number of h_i covering any example) and more precisely with the number and diversity of hypothesis h_i.

The choice of the concepts to be included in F – *feature selection* – is beyond the scope of this work. Experimentally, good results was obtained by using the set of both primitive and defined concepts found in the ontology (see Sect.5.1.1).

4.6.2 Discussion

In this section it is proved that the function defined in Def. 4.6.1 is really a semi-distance measure. In order to prove this, the following three properties have to be proved (see Def. 3.1.5):

Proposition 4.6.2 (semi-distance) *For a fixed hypothesis set and $p > 0$, given any three instances $a, b, c \in \mathsf{Ind}(\mathcal{A})$. it holds that:*

1. $d_p(a,b) \geq 0$
2. $d_p(a,b) = d_p(b,a)$
3. $d_p(a,c) \leq d_p(a,b) + d_p(b,c)$

PROOF:

1. *Trivial. By definition of Minkoski's distance.*
2. *Trivial, as the absolute value of the difference projection values is considered.*
3. *Noted that*

$$\begin{aligned}(d_p(a,c))^p &= \frac{1}{m}\sum_{i=1}^{m} \mid \pi_i(a) - \pi_i(c) \mid^p = \frac{1}{m}\sum_{i=1}^{m} \mid \pi_i(a) - \pi_i(b) + \pi_i(b) - \pi_i(c) \mid^p \\ &\leq \frac{1}{m}\sum_{i=1}^{m} \mid \pi_i(a) - \pi_i(b) \mid^p + \frac{1}{m}\sum_{i=1}^{m} \mid \pi_i(b) - \pi_i(c) \mid^p \\ &\leq (d_p(a,b))^p + (d_p(b,c))^p \leq (d_p(a,b) + d_p(b,c))^p\end{aligned}$$

then the property follows for the monotonicity of the power function.

It cannot be proved that $d_p(a, b) = 0$ iff $a = b$. This is the case of *indiscernible* individuals with respect to the given set of hypotheses F.

The presented measure is very powerful, indeed it can be applied to whatever DLs as it is neither structure driven nor dependent on the constructors of a specific language. Rather, it requires only retrieval (through instance-checking) service used for deciding whether an individual asserted in the knowledge base is belonging to a concept extension (or, alternatively, if this could be derived as a logical consequence). Hence, the complexity of d_p^{F} is given by $Compl(d_p^{\mathsf{F}}) = |\mathsf{F}| \cdot 2 \cdot Compl(\mathsf{IChk})$ for the chosen DL and where IChk stands for the instance checking inference operator which is invoked $|\mathsf{F}|$ times for every individual.

Various developments for the measure can be foreseen as concerns its definition. Namely, since it is very dependent on the concepts included in the committee of features F, two immediate lines of research arise: 1) reducing the number of concepts saving those concepts which are endowed of a real discriminating power; 2) learning optimal sets of discriminating features. Both these objectives can be accomplished by means of machine learning techniques especially when ontologies with a large set of individuals are available.

Chapter 5

Applying the Measures: Classification and Clustering in the Semantic Web Domain

Most of the research in the Semantic Web and Semantic Web Services fields focus on deductive-based reasoning methods. However, important tasks that are likely to be provided by new generation knowledge-based systems, such as classification, construction, revision, population, evolution are supported also by inductive methods.

To support these tasks and overcome the inherent complexity of the classic logic-based inference other forms of reasoning are being investigated, both deductive, such as *non-monotonic*, *paraconsistent* [97], *approximate* [100], *case-based reasoning* [58] and inductive-analogical forms such as inductive *generalization* [43] and *specialization* [73]. Anyway, in general, inductive reasoning and knowledge discovery have received less attention, although it may assist most of the tasks mentioned above, such as clustering (that can be used for improving the efficiency of the service discovery process), classification (that can be used to enforce concept retrieval[1] besides of KB population and evolution), mapping and alignment of knowledge bases.

In the perspective of knowledge/functionality sharing and reuse of the *social* vision of the SW, new inference services are required, aiming at noise-tolerant and efficient forms of reasoning. Two kinds of noise may be identified. The first one may be introduced by inconsistency in the KB. A second kind of noise is due to incorrect knowledge that does not strictly cause inconsistency, nevertheless it may yield incomplete/inconsistent conclusions with respect to the intended meaning of the concepts in the considered domain. From this perspective, inductive clustering

[1]Retrieval may be regarded from both a deductive perspective and from an opposite one, through approximate reasoning [100].

methods and instance-based inductive methods applied to multi-relational domains appear particularly well suited. Indeed, with special reference to instance based-methods, they are known to be both very efficient and noise-tolerant. These are very interesting characteristics, considered that noise is always harmful in contexts where knowledge is to be acquired from distributed sources.

Moving from these considerations, a similarity-based relational instance-based framework for the SW context has been devised to derive (by analogy) both consistent consequences from the KB and, possibly, also new assertions which were not previously logically derivable. The main idea is that similar individuals, by analogy, should likely belong to similar concepts. Specifically, a classification procedure has been derived by the use of different approaches: a relational form of the *k-Nearest Neighbor* algorithm (*K-NN*, henceforth) (see Sect. 1.3.1 and App. A for details about classical K-NN setting) and a Support Vector Machine [31, 178] applicable to $\mathcal{ALC}$ knowledge bases, by the use of a defined kernel function. Particularly, classification can be performed even in absence of a definition for the target concept in the KB, by analogy with a set of training assertions on such a concept (provided by an expert).

The cited approaches have been formalized in the literature for propositional representations. Upgrading these algorithms to work on multi-relational representations, like the languages used in the SW, is not straightforward at all. Novel (dis-)similarity measures suitable for such representations are necessary, since they play a key role for the correct prediction of the classification results. Moreover, in the standard setting of the instance-based classification methods, instances are assumed to be disjoint. This typically cannot hold in a SW context, where an individual may be instance of more that one concept. Furthermore, a theoretical problem has been posed by the *Open World Assumption* (OWA) that is generally made in the target context, differently from the typical ML settings (particularly Logic Programming and Data Bases setting) where the *Closed World Assumption* (CWA) is the standard. The usage of the realized instance-based framework can bring various advantages:

- it can give a better insight in the specific domain of the ontology;
- it may help ontology population which is time consuming and error-prone;
- it may trigger concept induction/revision by means of supervised and unsupervised machine learning methods [73].
- it may be used to improve the concept retrieval inference service.

In the context of the SWSs, clustering methods and semantic (dis-)similarity measures can be used to improve the effectiveness of the service discovery process besides of the ranking of the retrieved services. Traditionally, service discovery is performed by a syntactic matching between the service request and all provided services. Hence, the services satisfying the match are returned. This approach is characterized by some heavy drawbacks.

- some available services could not be discovered due to trivial syntactic differences even if they semantically perform the searched functionality
- the linear complexity (in the number of all available services) of the matching process is unusable with the increasing of the number of the provided services
- services selected by the matching process need to be ranked with respect to a certain criterion rather than returned as a (long)flat list.

In order to solve these problems, services can be described by means of DLs and service discovery could be performed exploiting DLs inference services. (Conceptual) clustering methods could be applied to such service descriptions, giving as output subsets of services homogeneously grouped and intensionally described. Hence, the discovery process can be performed by matching the request to the cluster descriptions rather than to all available services. Once that the cluster of interest is found, the matching process could be performed on the descriptions populating it, ignoring the service descriptions contained in the other clusters. The services retrieved by the matching process can then be ranked by the use of a similarity criterion w.r.t. the service request. In this way the efficiency of the discovery process is strongly increased. Moreover, the availability of a ranked list of the selected services facilitates the choice the right one, decreasing the complexity of the negotiation process. Indeed, by providing the most similar service to the request, the probability of iterating the search of the right service among the matched ones decreases.

The application of clustering methods to complex DL representations requires the availability of suitable (dis-)similarity measures, in order to ensure a meaningful partitioning of the services. Moreover, a crucial point is represented by finding a way for determining meaningful and readable intensional cluster descriptions. In this chapter various aspects are treated:

- it is shown how the defined similarity and dissimilarity measures for DLs can be helpful and effective, in important domains such as the SW and the SWSs
- it is experimentally shown the validity of the presented measures by embedding them in inductive learning algorithms
- it is shown that inductive learning methods can be effectively used to improve many open issues in the Semantic Web and Semantic Web Services context

In the next section, classification methods for improving the concept retrieval inference task in the SW context will be illustrated. In Sect. 5.2 clustering methods for improving the SWS discovery will be analyzed.

5.1 Analogy Reasoning to Improve Concept Retrieval and Induce New Knowledge

Many inference tasks, such as concept retrieval as well as semi-automatize the ABox population task, can be effectively performed in the context of the SW by the use of inductive inference methods and particularly by means of instance-based learning methods. Indeed many studies [118, 59, 206] have pointed out a number of advantages of instance-based learning methods: (1) they have often excellent performance, (2) they are characterized by the ability of coping with symbolic as well as continuous attributes and class values, and (3) they are robust with respect to noise in the data or missing attribute values.

In this section, an instance-based framework applicable to ontological knowledge is proposed. Exploiting the defined (dis-)similarity measures (see Chapt. 4), the proposed framework can derive inductively (by analogy) both consistent consequences from the knowledge base and also new assertions which may not be logically derived. Such a framework is based on a classification process whose goal is to classify individuals asserted in the knowledge base with respect to the concepts defined therein. Hence, the framework can be effectively used to semi-automatize the task of populating ontologies that are partially defined (in terms of assertions). Moreover, classification can be performed even in absence of a definition for the target concept in the knowledge base, by analogy with a set of training assertions on such a concept provided by an expert. Consequently, it can be effectively used in order to improve concept retrieval. Indeed, as will be shown experimentally, besides of having comparable results with the classical deductive approach, new knowledge is also induced, even in presence of noise. In turn, this enables other related (bottom-up) services such as learning and/or revision of faulty knowledge bases, ontology construction and evolution. The framework has been realized in a modular way:

- fixed a classification algorithm, different measures can be used
- fixed the instance-based approach, different classification algorithms can be used.

Particularly, two different classification algorithms have been realized: a relation K-NEAREST NEIGHBOR algorithm and a Support Vector Machine. In the following they will be analyzed in detail, jointly with their experimental evaluations.

5.1.1 Relational K-Nearest Neighbor

K-NN algorithms have been set to cope with propositional representations (see Sect. 1.3.1). The extension of such algorithm to more complex representation [4] and particularly to a relational setting is not trivial. Considering the advantages that characterize this algorithm, namely efficiency and noise tolerance, many efforts have been made for determining a version that is able to work in a relational setting. Particularly, the First Order Logic has been focused as representation language (see [72, 165]). The main problem highlighted in such works has been the lack of suitable measures that are able to cope with the increase of the expressiveness of the representation language.

For the best knowledge, no efforts have been made to extend the K-NN to cope with ontological representations. This task requires to solve other two issues, besides of those to have suitable measures, that are: to deal with the OWA, generally made in the semantic web context, and to deal with the non-disjointness of classes (concepts), as an individual can belong to more than one concept in an ontology. Below, the classification problem is formally defined, hence the solutions to the mentioned problems are described, jointly with the formalization of the relational K-NN for DLs [49, 54, 56].

The Classification Problem: Let KB = $(\mathcal{T},\ \mathcal{A})$ a knowledge base, let $C = \{C_1, \ldots, C_s\}$ be the set of concepts in $\mathcal{T}$ and let $\mathsf{Ind}(\mathcal{A})$ the set of all individuals asserted in $\mathcal{A}$.
Considered $a \in \mathsf{Ind}(\mathcal{A})$ determining the set of concepts (classes) $C' \subseteq C$ to which a belongs to, namely the set of concepts of which a is instance.

Considered the classification problem and the specification of the K-NN algorithm in its classical setting (see App. A, the classification process of a query instance (namely an individual) x_q could be performed as follows. Given a dissimilarity measure for DL, the set of k nearest pre-classified examples is selected. The objective is to learn a discrete-valued target function $h : IS \mapsto C$ from a space of instances IS to a set of concepts $C = \{C_1, \ldots, C_s\}$. The value of h for x_q is determined on the ground of the value that h assumes in the neighborhood of x_q, i.e. the k closest instances to x_q, in terms of the chosen dissimilarity measure. More precisely, considering the *weighted K-NN*, the value of h for classifying x_q is assigned according to the (weighted) value which is *voted* by the majority of instances in the neighborhood. Formally, this is expressed by:

$$\hat{h}(x_q) \leftarrow \operatorname*{argmax}_{C_j \in C} \sum_{i=1}^{k} w_i \delta(C_j, h(x_i))$$

where, δ is the Kronecker delta that returns 1 in case of matching arguments and

0 otherwise, w_i is usually given by $w_i = 1/d(x_i, x_q)$ or $w_i = 1/d(x_i, x_q)^2$,where d is the chosen dissimilarity measure.

This is a limited definition of the classification problem, as the strong assumption of this setting is that it can be employed to assign one value (e.g. one class) to a query instance among a set of values which can be regarded as a set of pairwise disjoint concepts/classes. On the contrary, in a SW setting, an individual could be instance of more than one concept. Hence, the set $C = \{C_1, \ldots, C_s\}$ has to be considered as made by not necessarily pairwise disjoint classes C_j $(1 \leq j \leq s)$ that may be assigned to a query instance.

In this more general case, in which nothing is known about the disjointness of the classes (unless explicitly stated in the TBox), a new answering procedure is proposed. It is based on the decomposition of the multi-class problem into smaller binary classification problems (one per class). Therefore, a simple binary value set ($V = \{-1, +1\}$) can be employed. Then, for each concept, a hypothesis $\hat{h}_j : IS \mapsto V$ is computed, for each class $C_j \in C$:

$$\hat{h}_j(x_q) \leftarrow \underset{v \in V}{\operatorname{argmax}} \sum_{i=1}^{k} \frac{\delta(v, h_j(x_i))}{d(x_q, x_i)^2} \qquad \forall j \in \{1, \ldots, s\} \tag{5.1}$$

where each function h_j $(1 \leq j \leq s)$, simply indicates the occurrence (+1) or absence (−1) of the corresponding assertion in the ABox: $C_j(x_i) \in \mathcal{A}$. Note that also $h_j : IS \mapsto V$. As a possible alternative[2], h_j may return +1 when $C_j(x_i)$ can be inferred from the knowledge base $\mathcal{K}$, and −1 otherwise.

Anyway, even if this formulation of the K-NN algorithm is able to solve the problem of non-explicitly disjoint concepts, it makes an implicit assumption of *Closed World*. To deal with the OWA, the absence of information on whether a certain instance x belongs to the extension of concept C_j should not be interpreted negatively, as see before, rather, it should count as neutral information. Thus, one can still adopt the decision procedure in Eq. (5.1), however another value set has to be considered for the h_j's, namely $V = \{-1, 0, +1\}$, where the three values denote, respectively, non-occurrence, absence and occurrence of the opposite assertion. Formally:

$$h_j(x) = \begin{cases} +1 & C_j(x) \in \mathcal{A} \\ -1 & \neg C_j(x) \in \mathcal{A} \\ 0 & \textit{otherwise} \end{cases}$$

Occurrence can be easily computed with a lookup in the ABox, therefore the overall complexity of the procedure depends on the number $k \ll |\mathsf{Ind}(\mathcal{A})|$, that is the number of times the distance measure is needed.

[2]For the sake of simplicity and efficiency, this case will not be considered in the following.

Note that, as the procedure is based on a majority vote of the individuals in the neighborhood, it is less error-prone in case of noise in the data (i.e. incorrect assertions in the ABox), therefore it may be able to give a correct classification even in case of (partially) inconsistent knowledge bases.

Again, a more complex procedure may be devised by simply substituting the notion of occurrence (absence) of assertions in (from) the ABox with the one of derivability (denoted with $\vdash$) from the whole KB, i.e. $\mathcal{K} \vdash C_j(x)$ ($\mathcal{K} \not\vdash C_j(x)$), $\mathcal{K} \not\vdash C_j(x)$ *and* $\mathcal{K} \not\vdash \neg C_j(x)$, respectively. Although this may improve the precision of inductive reasoning, it is also more computationally expensive, since the simple lookup in the ABox must be replaced with instance checking. Besides, this method could be extended with different (yet tractable) answering procedures based on statistical inference, to control the degree of confidence on the answer correctness.

The classification results could be useful for various purposes. The first one is that, classifying an individual w.r.t. the set of all possible concepts make possible to induce new knowledge. Indeed, the classification process reveals that the considered individuals belongs to other concepts besides of those determined by deductive reasoning. Consequently, by exploiting the inducted knowledge and classifying all the individuals in the ABox w.r.t. to a fixed concept, the retrieval inference service is improved. Furthermore, the most important point is that, not only classification w.r.t. concepts defined in the KB can be considered, but also the classification of individuals w.r.t. a totally new query concept, for example built on the fly from the considered KB. Anyway, in this case, the notion of derivability from the KB has to be employed, during the classification process, rather than the simply look up of the ABox, as seen above.

Summarizing, the defined relational K-NN algorithm for DL is implemented as a generalization of the propositional, distance-weighted k-NN algorithm[3] to a DL relational representation. It stores all training cases in its KB. It is "distance-weighted" since the votes of neighbors further away from the query are weighted less than the votes of nearby neighbors, as the weight is inversely proportional to the distance values of the neighbors from the query instance x_q. During classification, for each known class, the k nearest neighbors of each query vote on the class of the query instance. When asked to classify x_q, the classifier computes its similarity to each x_i in the training set. The k most similar neighbors are then retrieved and they vote on the class of x_q. Moreover, the classifier is able to cope with the problem of non-disjointeness classes and the OWA. As regards the used inference services, like all other instance-based methods, the presented method may require performing *instance-checking*, in order to determine whether an individual, say a, belongs to a concept extension, i.e. whether $C(a)$ holds for a certain concept C.

[3]See App. A for more details about the distance-weighted k-nearest neighbor algorithm.

Experimental Evaluation

In order to assess the validity of the presented method and the validity of the measures presented in Chap. 4, the method has been applied to the instance classification problem. The measures chosen for the experimentation have been the dissimilarity measure based on the overlap function presented in Sect. 4.2, and the dissimilarity measure based on Information Content presented in Sect. 4.3. Both measures refer to $\mathcal{ALC}$ logic. The classification has been performed on four different ontologies represented in OWL: FSM, SURFACE-WATER-MODEL from the Protégé library[4], the FINANCIAL ontology [5] employed as a testbed for the PELLET reasoner and the FAMILY ontology written by hand. Although they are represented in languages that are different from $\mathcal{ALC}$, these details are simply discarded,in order to be able to apply the cited measures.

FAMILY is an $\mathcal{ALCF}$ ontology describing *kinship* relationships. It is made up of 14 concepts (both primitive and defined), some of them are declared to be disjoint, 5 object properties, 39 distinct individual names. Most of the individuals are asserted to be instances of more than one concept, and are involved in more than one role assertions. This ontology has been written to have a small yet more complex case with respect to the following ones. Indeed, while the other ontologies are more regular, i.e. only some concepts are employed in the assertions (the others are defined only intensionally), in the FAMILY ontology every concept has at least one instance asserted. The same happens for the assertions on roles; particularly, there are some cases where role assertions constitute a chain from an individual to another one, by means of other intermediate assertions.

The FSM ontology describes the domain of *finite state machines* using the $\mathcal{SOF}(D)$ language. It is made up of 20 (both primitive and defined) concepts (some of them are explicitly declared to be disjoint), 10 object properties, 7 datatype properties, 37 distinct individual names. About half of the individuals are asserted as instances of a single concept and are not involved in any role (object property).

SURFACE-WATER-MODEL is an $\mathcal{ALCOF}(D)$ ontology describing the domain of the surface water and the water quality models. It is based on the *Surface-water Models Information Clearinghouse* (SMIC) of the USGS. Namely, it is an ontology of numerical models for surface water flow and water quality simulation. The application domain of these models comprises lakes, oceans, estuaries etc.. It is made up of 19 concepts (both primitive and defined) without any specification about disjointness, 9 object properties, 115 distinct individual names; each of them is an instance of a single class and only some of them are involved in object properties.

[4]See the webpage: `http://protege.stanford.edu/plugins/owl/owl-library`
[5]See the webpage: `http://www.cs.put.poznan.pl/alawrynowicz/financial.owl`

FINANCIAL is an $\mathcal{ALCIF}$ ontology that describes the domain of eBanking. It is made up of 60 (both primitive and defined) concepts (some of them are declared to be disjoint), 17 object properties, and no datatype property. It contains 17941 distinct individual names. From the original ABox, is has been randomly extracted assertions for 652 individuals.

The classification method was applied to all the individuals in each ontology; namely, the individuals were checked to assess if they were instances of the concepts in the ontology through the analogical method. The performance was evaluated comparing its responses to those returned by a standard reasoner[6] used as baseline. Specifically, for each individual in the ontology the msc is computed and enlisted in the set of training (or test) examples. Each example is classified applying the adapted k-NN method presented above. The chosen value of k has been $\sqrt{|\mathsf{Ind}(\mathcal{A})|}$, as advised in the instance-based learning literature [89]. The experiment has been repeated twice, adopting both the cited dissimilarity measures and a leave-one-out cross validation procedure. For each concept in the ontology, the following parameters have been measured for the evaluation:

- *match rate*: number of cases of individuals that got exactly the same classification by both classifiers with respect to the overall number of individuals;
- *omission error rate*: amount of unlabeled individuals (namely the method could not determine whether it was an instance or not) while it was to be classified as an instance of that concept;
- *commission error rate*: amount of individuals (analogically) labeled as instances of a concept, while they (logically) belong to that concept or vice-versa
- *induction rate*: amount of individuals that were found to belong to a concept or its negation, while this information is not logically derivable from the knowledge base

The average rates obtained (using both dissimilarity measures) over all the concepts in each ontology are reported, jointly with their standard deviation.

By looking at Tab. 5.1, reporting the experimental outcomes with the dissimilarity measure based on the overlap (see Def. 4.2.2), preliminarily it is important to note that, for every ontology, the commission error was quite low. This means that the classifier did not make critical mistakes i.e. cases when an individual is deemed as an instance of a concept while it really is an instance of another disjoint concept.

In particular, by looking at the outcomes related to the FAMILY ontology, it can be observed that the match rate is the lowest while the highest rate of omission

[6]PELLET: `http://pellet.owldl.com`

Table 5.1: Results (average±std-dev.) of the experiments with the method employing the measure based on overlap.

	Match Rate	**Commission Rate**	**Omission Rate**	**Induction Rate**
FAMILY	.654±.174	.000±.000	.231±.173	.115±.107
FSM	.974±.044	.026±.044	.000±.000	.000±.000
S.-W.-M.	.820±.241	.000±.000	.064±.111	.116±.246
FINANCIAL	.807±.091	.024±.076	.000±.001	.169±.076

errors was reported. This may be due to two facts: 1) very few individuals were available w.r.t. the number of concepts[7]; 2) sparse data situation: instances are irregularly *spread* over the concepts, that is some concepts have a lot of instances while other concepts have very few instances[8]. Hence the msc approximations that were computed also resulted very different one from another, which reduces the possibility of significantly matching similar mscs. This is a known drawback of the Nearest-Neighbor methods. This is more clear by looking at Tab. 5.2 where the higher match rate values correspond to concepts having the higher number of instances. However, as mentioned above, it is important to note that the algorithm did not make any commission error and it is able to infer new knowledge (11%).

Classification results of individuals asserted in the FSM ontology reveal the maximum match rate w.r.t. the classification given by the logic reasoner. Moreover, differently from the other ontologies, both the omission error rate and induction rate were null. A very limited percentage of incorrect classification cases was observed. These outcomes were probably due to the fact that individuals in this ontology are quite regularly divided by the assertions on concepts and roles, namely most of the individuals are instances of a single concept or a single role, so computing their mscs, these are all very similar to each other and consequently the amount of information they convey is very low. A choice of a lower number k of neighbors could probably help committing those residual errors.

For the same reasons, also for the SURFACE-WATER-MODEL ontology quite a high rate of matching classifications was reported (yet less than with the previous ontology); moreover, some cases of omission error (6%) were observed. The induction rate was about 12% which means that, for this ontology, the classifier always assigned individuals to the correct concepts and, in some cases, it could also induce new assertions. Since this rate represents assertions that were not logically deducible

[7]Instance-based methods make an intensive use of the information about the individuals and improve their performance with the increase of the number of instances considered.

[8]Specifically, there is a concentration of instances of concepts like Human, Child and GrandChild.

Table 5.2: Outcomes of the trials with the FAMILY ontology employing the measure based on overlap.

	Match Rate	**Commission Rate**	**Omission Rate**	**Induction Rate**
Father	0.590	0.000	0.359	0.051
Man	0.436	0.000	0.487	0.077
Parent	0.692	0.000	0.231	0.077
Female	0.436	0.000	0.410	0.154
Male	0.436	0.000	0.487	0.077
Human	0.974	0.000	0.000	0.026
Child	0.590	0.000	0.000	0.410
UncleAunt	0.846	0.000	0.154	0.000
Woman	0.436	0.000	0.410	0.154
Sibling	0.718	0.000	0.128	0.154
Grandchild	0.718	0.000	0.103	0.179
Grandparent	0.923	0.000	0.077	0.000
Mother	0.641	0.000	0.333	0.026
Cousin	0.718	0.000	0.051	0.231
average	0.654	0.000	0.231	0.115

from the ontology and yet they were inferred inductively by the analogical classifier, these figures may be a positive outcome (provided this knowledge were deemed as correct by an expert). In this case the increase of the induction rate has been due to the presence of assertions of mutual disjointness for some of the concepts.

Results are no different also for the experiments with the FINANCIAL ontology that largely exceeds the others in terms of number of concepts and individuals. The observed match rate is again above the 80% and the rest of the cases are comprised in the induction rate (17%), leaving a limited margin to residual errors. This corroborates a fact about the NN learners, that is they reaching better performance in the limit, as long as new training instances become available. Actually, performing a 10-fold cross validation, the obtained results are almost the same.

The average results obtained by adopting the classification procedure and the measure based on information content (see Def. 4.3.2) are reported in Table 5.3. By analyzing this table it is possible to note that no sensible variation was observed in the classifications performed using the dissimilarity measure based on overlap. Particularly, with both measures, the method correctly classified all the individuals, almost without commission errors. The reason is that, in most of the cases, the individuals of these ontologies are instances of one concept only and they are involved

Table 5.3: Results (average ± std-dev.) of the experiments with the method employing the measure based on information content.

	Match Rate	**Commission Rate**	**Omission Rate**	**Induction Rate**
FAMILY	.608±.230	.000±.000	.330±.216	.062±.217
FSM	.899±.178	.096±.179	.000±.000	.005±.024
S.-W.-M.	.820±.241	.000±.000	.064±.111	.116±.246
FINANCIAL	.807±.091	.024±.076	.000±.001	.169±.046

in a few roles (object properties). Some of the figures are slightly lower than those observed in the other experiment: this is confirmed by a higher variability.

Surprisingly, the results on larger ontologies (S.-W.-M. and FINANCIAL) perfectly match those obtained with the other measure. This is probably due to the fact that the leave-one-out cross-validation mode has been used, which yielded a high value for the number k of training instances to consider as neighborhood. It is well known that the NN procedure becomes more precise as more instances can be considered. The price to be paid was a longer computation time.

Considered the experimental results, it is possible to assert that the classifier is really able induce new knowledge that is not logically derivable. Consequently, this relational classifier can be naturally exploited for predicting/suggesting missing information about individuals thus enabling a sort of inductive retrieval. Particularly, an increase in accuracy was observed when the instances increase in number and are homogeneously spread. Furthermore, the k-NN method in its classical form may be particularly suitable for the automated induction of missing values for (scalar or numeric) datatype properties of an individual as an estimate derived from the values of the datatypes for the surrounding individuals.

Moreover, the realized classifier can be employed to perform the retrieval of a new query concept, defined by the use of concepts and roles asserted in the considered ontology. In the following, the experimental evaluation for such a task are reported. The measure used in this context is the semi-distance measure presented in Sect. 4.6. The proposed method has been applied to a number of retrieval problems. To these purpose, besides of the ontologies used in the experimentation above[9], other two ontologies have also been used: NEWTESTAMENTNAMES and SCIENCE ontology, taken from the Protégé library[10]. NEWTESTAMENTNAMES ontology describes facts related to the New Testament. It is a $\mathcal{SHIF}(D)$ ontology consisting of 47 concepts, 27 object properties, 676 individual names. SCIENCE ontology describes

[9]The FAMILY ontology has not been considered because too small.

[10]`http://protege.stanford.edu/plugins/owl/owl-library`

Table 5.4: Ontologies employed in the experiments.

ontology	*DL*	*#concepts*	*#obj. prop*	*#data prop*	*#individuals*
FSM	$\mathcal{SOF}(D)$	20	10	7	37
S.-W.-M.	$\mathcal{ALCOF}(D)$	19	9	1	115
SCIENCE	$\mathcal{ALCIF}(D)$	74	70	40	331
FINANCIAL	$\mathcal{ALCIF}$	60	17	0	652
NTN	$\mathcal{SHIF}(D)$	47	27	8	676

scientific facts in $\mathcal{ALCIF}(D)$. It is made up of 74 concepts, 70 object properties, 331 individual names. Table 5.4 summarizes all details concerning the ontologies employed in the experimentation.

In is important to note that, differently from the previous experimentation, in which measure for $\mathcal{ALC}$ have been used, the measure considered in the current experimentation is language independent (see Sect. 4.6), hence the entire expressiveness of each ontology can be considered.

The experiment was quite intensive, involving the classification of all the individuals in each ontology; namely, the individuals were checked through the inductive procedure to assess whether they were to be retrieved as instances of a query concept. Therefore, 15 queries were randomly generated by conjunctions/disjunctions of primitive or defined concepts of each ontology. The performance, as in the previous case, was evaluated comparing the procedure responses to those returned by a standard reasoner[11] employed as a baseline.

The experiment has been repeated twice adopting different procedures according to the size of the corresponding ABox (measured by $|\mathsf{Ind}(\mathcal{A})|$): the leave-one-out cross validation for the smaller ontologies (FSM and S.-W.-M.) and the ten-fold cross validation for the larger ones. As for the previous experimentation the amount of neighbors (k) to select has been set as $\sqrt{|\mathsf{Ind}(\mathcal{A})|}$. Yet it has been experimentally found that much smaller values for k could be chosen, resulting in the same classification. The employed version of the measure formalized in Def. 4.6.1 is the simplest one, namely those obtained by setting $p = 1$ (Manhattan distance d_1), and all the concepts in the ontology for determining the set F have been used. The classification evaluation has been performed, as in the previous experimentation, by measuring for each concept in the ontology: *match rate*, *omission error rate*, *commission error rate* and *induction rate*. In the following the average rates obtained over all the

[11] PELLET: `http://pellet.owldl.com`

Table 5.5: Results (average±std-dev.) of the experiments with the method employing the semi-distance semantic measure.

	match rate	*commission rate*	*omission rate*	*induction rate*
FSM	97.7 ± 3.00	2.30 ± 3.00	0.00 ± 0.00	0.00 ± 0.00
S.-W.-M.	99.9 ± 0.20	0.00 ± 0.00	0.10 ± 0.20	0.00 ± 0.00
Science	99.8 ± 0.50	0.00 ± 0.00	0.20 ± 0.10	0.00 ± 0.00
Financial	90.4 ± 24.6	9.40 ± 24.5	0.10 ± 0.10	0.10 ± 0.20
NTN	99.9 ± 0.10	0.00 ± 7.60	0.10 ± 0.00	0.00 ± 0.10

concepts in each ontology jointly with their standard deviation are discussed.

By looking at Tab. 5.5 reporting the experimental outcomes, preliminarily it is important to note that, as for the previous experimentation, for every ontology, the commission error was low. This means that the procedure is quite accurate: it did not make critical mistakes i.e. cases when an individual is deemed as an instance of a concept while it really is an instance of another disjoint concept.

By comparing these outcomes with those reported for the previous experiments (see Tab. 5.1 and Tab. 5.3), where the average match rate on the same was slightly higher than 80%, it is straightforward to note a significant increase of the performance due to the accuracy of the measure used in the current experiment. Also, the elapsed time (not reported here) was lowered because, once the values for the projection functions π's (see Def. 4.6.1) are pre-computed, the efficiency of the classification, which depends a lot on the computation of the dissimilarity, gains a lot of speed-up. Anyways, it is also possible to note a decrease of the induction rate, with respect to the previous experiment, which means that the semi-distance measure is highly comparable with a reasoner but less able to induce new knowledge.

The usage of all concepts for the set F made the measure accurate, which is the reason why the procedure resulted conservative as regards inducing new assertions. It matched rather faithfully the reasoner decisions. A noteworthy difference was observed for the case of the Financial ontology for which the lowest match rate and the highest variability in the results over the various concepts are found. On a careful examination of the experimentation with this ontology, it has been found that the average results were lowered by a concept whose assertions, having been poorly sampled from the initial ontology, could not constitute enough evidence to the inductive method for determining the correct classification.

The same problem, to a lesser extent, was found also with the FSM ontology which was the one with the least number of assertions. This shows that the weaker side of any instance-based procedure results when data are too sparse or non evenly distributed. Moreover, as mentioned above, it has been also found that a lower value for k could have been chosen, as in many cases the decision on the correct classification was easy to make even on account of a few (the closest) neighbor instances.

In the last experiment presented, all concepts involved in an ontology were used for inclusion in the hypothesis set F, for computing distance values. This is because the inherent redundancy helps a lot the measure accuracy (see discussion about the Hypotheses-driven distance in Sect. 3.3.5). Yet larger sets yield more effort to be made for computing the measures. Nevertheless, it is well known that the NN approach suffers when lots of irrelevant attributes for describing the instances are considered. Thus, it has also been tested how the variation of hypotheses (concept descriptions) belonging to the set F could affect the performance of the measure. The expected result was that with an increasing number of considered hypotheses for F, the accuracy of the measure would increase accordingly. To test this claim experimentally, one of the ontologies used for the previous experiment has been considered. A leave-one-out cross validation has been performed repeatedly (three times) with an increasing percentage of concepts randomly selected for F w.r.t. the overall number of primitive and/or defined concept names in the ontology. The average results returned by the system are depicted in Fig. 5.1. Numerical details of such outcomes are given in Table 5.6.

As expected, it is possible to note that the accuracy of the decisions (*match rate*) is positively correlated with the number of concepts included in F. The same outcomes were obtained by repeating similar experiments with other ontologies. It should be observed that in some cases the concepts randomly selected for inclusion in F actually turned out to be a little redundant (by subsumption or because of a simple overlap between their extension). This suggests a line of further investigation that will concern finding minimal subsets of concepts to be used for the measure.

Table 5.6: Average results varying the number of hypotheses in the set F.

	match	*commission*	*omission*	*Induction*
% of concepts	*rate*	*rate*	*rate*	*rate*
20%	79.1	20.7	0.00	0.20
40%	96.1	03.9	0.00	0.00
50%	97.2	02.8	0.00	0.00
70%	97.4	02.6	0.00	0.00
100%	98.0	02.0	0.00	0.00

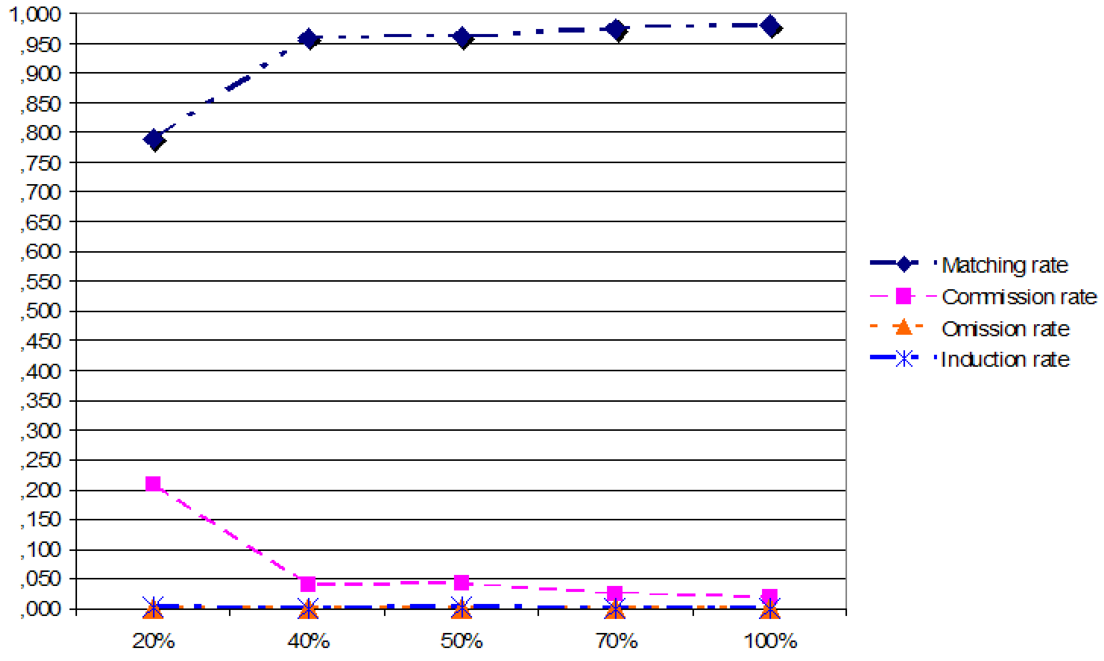

Figure 5.1: Average results varying the number of hypotheses in the set F.

5.1.2 Concept Retrieval by means of Kernel Methods

In order to make concept retrieval more effective by the use of an inductive classifier, a kernel method, specifically a Support Vector Machine (SVM) has also been used. The reasons of this choice are many. First of all kernel methods, and particularly SVMs, are well known efficient inductive learning methods. They can be developed in a modular way. Indeed two components of kernel methods have to be distinguished: the kernel machine and the kernel function. The kernel machine encapsulates the learning task and the way in which a solution is looked for, the kernel function encapsulates the hypothesis language, i.e., how the set of possible solutions is made up. Different kernel functions implement different hypothesis spaces or even different knowledge representations.

Moving from this consideration, a SVM can be used jointly with the kernel function for $\mathcal{ALC}$ (presented in Sect. 4.5) in order to perform classification of individuals asserted in a considered ontology.

From a computational point of view, the attractiveness of kernel methods comes from the fact that they map, by means of the kernel function, the original feature space of the considered data set into a high dimensional feature space where the execution of the learning task is easier. Anyway this is done without suffering

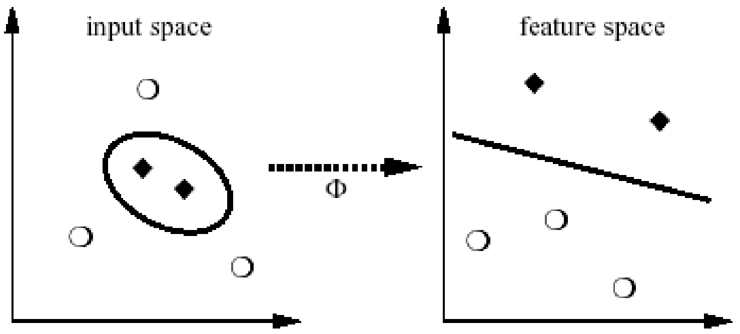

Figure 5.2: The idea of SVMs.

the high cost of explicitly computing the mapped data. The *kernel trick* is to define a positive definite kernel on any feature set. For such functions it is know that there exists an embedding of the feature set in a linear space such that the kernel on the elements of the set corresponds to the inner product in this space.

Particularly, SVMs[12] are classifiers that, by means of a mapping function ϕ, map the training data into a higher dimensional feature space where they can be classified using a linear classifier (see Fig. 5.2). This is done by constructing a separating hyperplane with the maximum margin in such new feature space, which yields a nonlinear decision boundary in input space. By the use of a kernel function[13], it is possible to compute the separating hyperplane without explicitly carrying out the map into the feature space. In this section two main goals are to be achieved:

- concept retrieval is effectively performed by the use of inductive kernel-based learning algorithms, namely SVMs
- the $\mathcal{ALC}$ kernel function (presented in Sect. 4.5) is experimentally validated.

In order to reach these goals, a SVM from the LIBSVM library[14] has been considered jointly with the the presented kernel $\mathcal{ALC}$. The problem to be solved is defined as follow.

Given a knowledge base $KB = (\mathcal{T}, \mathcal{A})$, let $\mathsf{Ind}(\mathcal{A})$ be the set of all individuals in the ABox $\mathcal{A}$ and $C = \{C_1, \ldots, C_s\}$ the set of all concepts (both primitive and defined) in the TBox $\mathcal{T}$. The problem to solve is: considered an individual $a \in \mathsf{Ind}(\mathcal{A})$ determine the set of concepts $\{C_1, \ldots, C_t\} \subseteq C$ to which a belongs to.

[12]To give a detailed examination of the kernel methods and particularly of the SVMs is out of the scope of this thesis. Here only the main idea of the SVMs is given in order to understand the setting of the used classification method.

[13]See Sect. 3.2.2, 3.3.6 for more details about kernel functions.

[14]Software downloadable at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`

In order to cope with the non-disjointness of the concepts (classes) and the OWA (see discussion in Sect. 5.1.1 for more details about such aspects), the classification problem has been decomposed in a set of s ternary classification problems, each one returning a value from the set $\{-1, 0, +1\}$. Specifically, considered the query instance x_q, for every every concept $C_j \in C$ the classifier will return $+1$ if x_q is an instance of C_j, -1 if x_q is an instance of $\neg C_j$, and 0 otherwise, namely if no information is available from the KB. The classification is performed on the ground of a set of training examples from which such information can be derived.

In the following, how the SVM classifies a single individual w.r.t. a fixed class will be briefly explained. This is given only for sake of completeness. Indeed, due to the modularization that characterizes SVMs and more in general kernel methods, classification can be performed without knowing how the SVM really works.

Given the set of training examples and the kernel function k to be considered, the SVM builds the Gram Matrix, namely the matrix containing all computed value $k(x_i, x_j)$ where x_i and x_j are training examples. Hence, when x_q has to be classified, the kernel function values $k(x_q, x_i)$ is computed, where $i = \{1, \ldots, |Tr|\}$ and Tr is the set of training examples. At this point the linear classifier is built and the class of x_q is consequently determined.

In the following the experimental evaluation performed by applying the cited SVM to ontological knowledge, through the developed $\mathcal{ALC}$ kernel will be discussed.

Experimental Evaluation

In order to assess the validity of the presented method and particularly the validity of the presented kernel function for $\mathcal{ALC}$ logic (see Sect. 4.5), the method has been applied to the instance classification problem. The classification has been performed on nine different ontologies represented in OWL: FAMILY handmade ontology, FSM, SURFACE-WATER-MODEL, NEWTESTAMENTNAMES, SCIENCE from the Protégé library[15], that are the same ontologies seen for the previous experiments; moreover the UNIVERSITY (handmade) ontology, PEOPLE, NEWSPAPER and WINES ontologies (also from from the Protégé library) have been considered. Table 5.7 summarizes all the details concerning the ontologies employed in the experimentation. Although they are represented in languages that are different from $\mathcal{ALC}$, further constructors are simply discarded, in order to be able to apply the kernel function for $\mathcal{ALC}$.

The classification method was applied to all the individuals in each ontology; namely, the individuals were checked to assess if they were instances of the concepts in the ontology through the SVM. As in the previous experiments, the performance

[15]See the webpage: `http://protege.stanford.edu/plugins/owl/owl-library`

Table 5.7: Ontologies employed in the experiments.

ontology	*DL*	*#concepts*	*#obj. prop*	*#data prop*	*#individuals*
PEOPLE	$\mathcal{ALCHIN}(D)$	60	14	1	21
UNIVERSITY	$\mathcal{ALC}$	13	4	0	19
FAMILY	$\mathcal{ALCF}$	14	5	0	39
FSM	$\mathcal{SOF}(D)$	20	10	7	37
S.-W.-M.	$\mathcal{ALCOF}(D)$	19	9	1	115
SCIENCE	$\mathcal{ALCIF}(D)$	74	70	40	331
NTN	$\mathcal{SHIF}(D)$	47	27	8	676
NEWSPAPER	$\mathcal{ALCF}(D)$	29	28	25	72
WINES	$\mathcal{ALCIO}(D)$	112	9	10	188

was evaluated comparing its responses to those returned by a standard reasoner[16] used as a baseline.

Specifically, for each individual in the ontology the msc is computed and enlisted in the set of training (or test) examples. Each example is classified applying the SVM and the $\mathcal{ALC}$ kernel function with $\lambda = 1$ (see Def. 4.5.1). The experiment has been repeated twice, adopting the leave-one-out cross validation procedure for less populated ontologies (with less then 50 individuals), while the ten-fold cross validation procedure has been used for the other ontologies. For each concept in the ontology, the classification results have been measured by the use of the same rates in the previous evaluations.

By looking at Tab. 5.8, reporting the experimental outcomes with the $\mathcal{ALC}$ kernel function with $\lambda = 1$, preliminarily it is important to note that, for every ontology, the commission error was quite low. This means that the classifier did not make critical mistakes, i.e. cases when an individual is deemed as an instance of a concept while it really is an instance of another disjoint concept. Particularly, the commission error rate is not null mainly in two cases, that are UNIVERSITY and FSM ontologies. By looking Tab. 5.7, it is straightforward to note that these ontologies are those having almost the lowest number of individuals concepts. Specifically, the number of concepts is almost similar to the number of individuals, this means that there is not enough information for separating the feature space producing a correct classification. However, also in this condition the commission error is quite low, the matching rate is considerably high and the classifier is able to induce new knowledge (induction rate not null).

[16]PELLET: `http://pellet.owldl.com`

Table 5.8: Results (average and range) of the experiments with the SVM employing the $\mathcal{ALC}$ kernel function with $\lambda = 1$.

Ontology	*measure*	*match rate*	*induction rate*	*omis. err. rate*	*comm. err. rate*
People	*avg.*	**0.866**	**0.054**	**0.08**	**0.00**
	range	0.66 - 0.99	0.00 - 0.32	0.00 - 0.22	0.00 - 0.03
University	*avg.*	**0.789**	**0.114**	**0.018**	**0.079**
	range	0.63 - 1.00	0.00 - 0.21	0.00 - 0.21	0.00 - 0.26
FSM	*avg.*	**0.917**	**0.007**	**0.00**	**0.076**
	range	0.70 - 1.00	0.00 - 0.10	0.00 - 0.00	0.00 - 0.30
Family	*avg.*	**0.619**	**0.032**	**0.349**	**0.00**
	range	0.39 - 0.89	0.00 - 0.41	0.00 - 0.62	0.00 - 0.00
NewsPaper	*avg.*	**0.903**	**0.00**	**0.097**	**0.00**
	range	0.74 - 0.99	0.00 - 0.00	0.02 - 0.26	0.00 - 0.00
Wines	*avg.*	**0.956**	**0.004**	**0.04**	**0.00**
	range	0.65 - 1.00	0.00 - 0.27	0.01 - 0.34	0.00 - 0.00
Science	*avg.*	**0.942**	**0.007**	**0.051**	**0.00**
	range	0.80 - 1.00	0.00 - 0.04	0.00 - 0.20	0.00 - 0.00
S.-W.-M.	*avg.*	**0.871**	**0.067**	**0.062**	**0.00**
	range	0.57 - 0.98	0.00 - 0.42	0.00 - 0.40	0.00 - 0.00
N.T.N.	*avg.*	**0.925**	**0.026**	**0.048**	**0.001**
	range	0.66 - 0.99	0.00 - 0.32	0.00 - 0.22	0.00 - 0.03

Moreover, by looking at the outcomes related to the University and FSM ontologies, it can be observed that the match rate is the lowest. As for the commission error, this is probably due to the fact that very few individuals were available w.r.t. the number of concepts. In general, by jointly analyzing Tab. 5.7 and Tab. 5.8 it is possible to note that the match rate increases with the increase of the number of individuals in the considered ontology with a consequent strong decrease of the commission error rate, almost null in such cases. Almost always the classifier is able to induce new knowledge. Anyway it presents also a conservative behavior, indeed the omission error rate is very often not null. To decrease the tendency to a conservative behavior of the classifier, a threshold could be introduced for the consideration of the "unknown" (namely labeled with 0) training examples.

To evaluate the impact of the parameter λ in the definition of the kernel function, the experiment has been repeated by setting $\lambda = 0.5$ and applying the leave-one-out procedure to the ontologies with less than 50 individuals (see Tab. 5.7), and the ten-fold cross validation procedure to the other ontologies. The average results are reported in Tab. 5.9. From this table and particularly, by looking Fig. 5.3

Table 5.9: Results (average) of the experiments employing the SVM jointly with the $\mathcal{ALC}$ kernel function with $\lambda = 0.5$.

	match rate	*induction rate*	*omission rate*	*commission rate*
People	86.6	5.4	8.0	0.0
University	71.1	11.4	4.4	13.2
FSM	90.3	0.7	0.0	9.0
Family	61.5	3.6	34.9	0.0
Newspaper	90.4	0.0	9.6	0.0
Wines	95.2	0.6	4.2	0.0
Science	87.7	6.5	5.8	0.0
S.-W.-M.	86.2	8.0	5.8	0.0
NTN	90.5	4.3	4.9	0.3

where the mean rates w.r.t. the various ontology are reported, it is possible to note that the variation of the λ value does not generally influence the classification results and that sometimes the match rate also decrease (i.e. University, FSM and NewTestamentNames) w.r.t. the classification performed using $\lambda = 1$.

Another experiment has been done, to test the method as a means for performing inductive concept retrieval with respect to new query concepts built from a considered ontology. Particularly, the method has been executed to perform a number of retrieval problems applied to the ontologies illustrated in Tab. 5.7 using $\lambda = 1$ for the kernel function.

The experiment was quite intensive involving the classification of all the individuals in each ontology; namely, the individuals were checked through the inductive procedure to assess whether they were retrieved as instances of a query concept. Therefore, 15 queries were randomly generated by means of conjunctions/disjunctions of primitive and/or defined concepts of each ontology. The classification performance was evaluated comparing responses to those returned by a standard reasoner (Pellet) employed as a baseline. For each classified query concept, *match rate*, *omission error rate*, *commission error rate* and *induction rate* have been measured.

The experiment has been repeated twice, adopting, as for the previous experiment, the leave-one-out procedure in case of ontologies with less than 50 individuals and ten-fold cross validation for the others.

The outcomes of the experiment are reported in Tab. 5.10, from which it is possible to observe that the behavior of the classifier mainly remains the same as in the experiment whose outcomes are reported in Tab. 5.8.

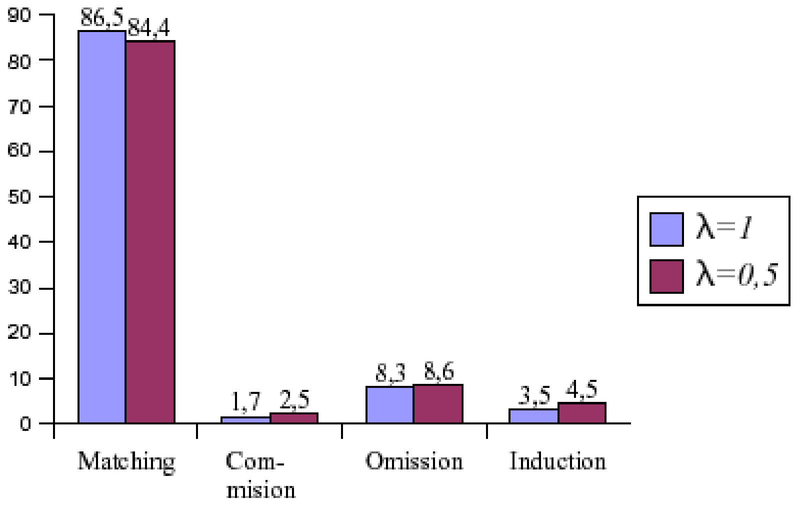

Figure 5.3: Means of the outcomes of the classification rates with respect to all considered ontologies. Classification has been performed by the a SVM jointly with the $\mathcal{ALC}$ kernel function with $\lambda = 1$ and $\lambda = 0.5$

Concluding, it can be asserted that the proposed $\mathcal{ALC}$ kernel function can be effectively used, jointly with a SVM, to perform inductive concept retrieval, guaranteeing almost null commission error and interestingly the ability to induce new knowledge. The performance of the classifier increases with the increase of the number of individuals populating the considered ontology that have to be preferable homogeneously spread w.r.t. the concept in the ontology.

Table 5.10: Results (average) of the experiments.

Ontology	*match rate*	*ind. rate*	*omis. err. rate*	*comm. err. rate*
People	88.6	4.0	7.4	0.0
University	72.0	16.0	0.9	11.1
FSM	87.8	0.9	0.0	11.4
Family	66.3	4.5	29.2	0.0
Newspaper	77.9	0.0	22.1	0.0
Wines	94.3	0.0	5.7	0.0
Science	97.8	0.5	1.6	0.0
S.-W.-M.	80.4	13.4	6.2	0.0
NTN	90.6	2.2	7.2	0.0

5.2 Improving the Service Discovery and Ranking Processes

In the last few years, the Web had two revolutionary changes, Web Service technology and the Semantic Web technology, that transformed it from a static document collection into an intelligent and dynamically integrated collection of resources. The former has allowed uniform access via Web standards to software components residing on various platforms and written in various programming languages. The latter has enriched existing Web data with their meaning, logically expressed with formal descriptions that are machine processable, thus facilitating access and integration. The major limitation of Web Services is that their retrieval and composition still require manual effort. To solve this problem, researchers have augmented Web Services with a semantic description of their functionality (see Sect. 1.2.1). By reusing a common vocabulary, service modelers can produce semantic service descriptions that can be shared and understood on the Web. Such vocabulary is defined by upper-level ontologies such as OWL-S[17] and WSMO[18] for *Semantic Web Services* (henceforth SWS).

The definition of SWS descriptions is a complex and time consuming task, that requires a significant amount of expertise. In this thesis a DLs based framework for describing services is proposed. It aims at supporting modelers in the service description task and bridging the gap between the formal semantics of service descriptions and human intuition.

The reasons of the choice of DLs as representation language for describing SWS are multiple: (1) DLs are endowed by a formal semantics, this allow to guarantee expressive service descriptions, besides of precise definition of the semantics of the service descriptions; (2) DLs are the theoretical foundation of OWL, hence they ensure compatibility with existing ontology standards; (3) DLs service descriptions could be easily mapped to other largely used representation formalisms for describing services, such as OWL-S; (4) the service discovery task can be performed by algorithms defined in terms of standard and non-standard DL inferences. In the following, moving from these considerations, three different aspects will be analyzed:

- service description distinguishing *Hard Constraints* and *Soft Constraints*;
- efficient service discovery exploiting inductive (conceptual) clustering methods;
- effective ranking of the discovered services by the use of constraint hardness and semantic (dis-)similarity measures for DLs.

[17]http://www.daml.org/services/owl-s/1.0/
[18]http://wsmo.org

5.2.1 Modeling DLs-based Service Descriptions by the use of Constraint Hardness

In this section a DLs-based framework for describing services is introduced. The main reason of the attention to service descriptions is the need of automating processes such as service discovery and composition[19]. The use of DLs in service descriptions and discovery task is not new [129, 88, 195, 159, 196, 93, 39]. However in [129] it has been showed that primitives, modeled by DLs, sometimes produce counterintuitive results. This issue has been analyzed in [93], where preliminary guidelines for modeling DLs-based service descriptions are presented. The framework proposed here [57] enriches the guidelines of [93] by extending them to the formalization of *Hard* and *Soft Constraints* in service descriptions.

Indeed, in a real scenario it is important to express forms of variability in service descriptions (and particularly in the service request side), represented by the optional and the mandatory aspects of a service description. For this reason, the notion of *Hard* and *Soft Constraints* are introduced. *Hard Constraints* (HC) are those features of a service description that have to be necessarily satisfied by the target services, while *Soft Constraints* (SC) are those features whose satisfaction is only preferable. To be able to distinguish *HC* and *SC* is important both for *business-to-consumer* interaction and for service discovery task. In fact with respect to business-to-consumer interaction, *HC* and *SC* allow to express the real necessities of the user; with respect to the discovery process, the distinction between *HC* and *SC* allows to relax some needs, increasing the possibility of satisfying a request. In the following, a way to express these kind of constraints is showed.

A *service description* is expressed as a set of constraints that have to be satisfied by the service providers. It can be thought as an abstract class acting as a template for *service instances*; to be specific, a service description defines a space of possible service instances (as in [161]), thus introducing *variance* [93], namely a service description usually represents numerous variants of a concrete service.

Thus, variance is the phenomenon of having more than one instance and/or more than one interpretation for a service description. Following [93], it is possible to distinguish between *variance due to intended diversity* and *variance due to incomplete knowledge*. To explain these concepts, the notion of *possible worlds* (borrowed from the first-order logic semantics) is used. Under open-world semantics, a modeler must explicitly state which service instances are not covered by the service description. For each aspect of the service description that is not fully specified there are *several possible worlds*, reflecting a way of resolving incompleteness (*variance due to incomplete knowledge*). Besides, given a possible world, the lack of constraints

[19]See Sect. 1.2.2 for more details about this tasks.

possibly allows for many instances satisfying a service description (*variance due to intended diversity*). In order to clarify the two different kinds of variance the following example is considered in which an informal service description is reported. Here, a flight service request is specified and some examples of its service instances[20] are reported.

Flight(flight) *and* operatedBy(flight,company)*and* departureTime(flight,time) *and* arrivalTime(flight,time) *and* from(flight,Germany) *and* to(flight,Italy)

and the *Service instances:*

- Flight(0542) *and* operatedBy(0542,ryanair) *and* departureTime(0542,8:00) *and* arrivalTime(0542,9:40) *and* from(0542,Hahn) *and* to(0542,Bari)
- Flight(0721) *and* operatedBy(0721,hlx) *and* departureTime(0721,12:00) *and* arrivalTime(0721,13:10) *and* from(0721,Cologne) *and* to(0721,Milan)
- Flight(9312) *and* operatedBy(9312,airBerlin) and departureTime(9312,17:00) *and* arrivalTime(9312,19:30) *and* from(0721,Berlin) *and* to(0721,Rome)

The service description reported above represents the request of flights from Germany to Italy, independently of departure and arrival time, company and cities involved. This lack of constraints, for example about arrival and departure location, company, arrival and departure time, allows many possible instances (as above), inducing *variance due to intended diversity.* Now, the following service instance is considered:

Flight(512) *and* operatedBy(512,airBerlin)*and* departureTime(512,18:00) *and* arrivalTime(512,19:30) *and* from(512,Berlin) *and* to(512,London)

This is also a correct instance of the service request reported above, because the fact that London is not an Italian city is left unspecified in the KB. So there can be a possible world in which London is an Italian city. Here, the absence of constraints induces *variance due to incomplete knowledge.*

So, variance due to incomplete knowledge is resolved by allowing many different possible worlds, each one resolving unspecified issues in a different way. Variance due to intended diversity is resolved by allowing alternative service instances within one possible world.

In order to cope with the effects of the *variance* on the semantics of a service description, it is necessary to adopt a language for service representation characterized by well-defined semantics. This is one of the peculiarities of the DL family,

[20]Note that a service instance contains the exact information about the model expressed by the service description.

from here the choice of Grimm et al. in [93] of representing services by means of DLs. Such framework is reported in the following, hence an extension for dealing with the notions of HC and SC of a service will be presented.

- A service description is expressed by a set of DL-axioms $D = \{S, \phi_1, \phi_2, ..., \phi_n\}$, where the axioms ϕ_i impose restrictions on an atomic concept S, which represents the service to be performed;

- Domain-specific background knowledge is represented by a *knowledge base* (KB) that contains all relevant domain-level facts;

- A *possible world*, resolving incomplete knowledge issues, is represented by a single DL model (interpretation) I of $KB \sqcup D$;

- The service instances that are acceptable w.r.t. a service description D, are the individuals in the extension S^I of the concept S representing the service;

- Variance due to intended diversity is given by S^I containing different individuals;

- Variance due to incomplete knowledge is reflected by $KB \sqcup D$ having several models $I_1, I_2, ...$.

The axioms in a service description D constrain the set of acceptable service instances in S^I. These constraints are generally referred to the properties used in a description. Here, various ways for constraining a property using DL are reported.

Variety: a property can be either restricted to a fixed value or it can range over instances of a certain class. This is expressed by $\forall r.i$ (or $\exists r.i$) and $\forall r.C$ (or $\exists r.C$), respectively. For any acceptable service instance, the value of such a property must either be an individual or a member of a class.

Multiplicity: a property can be either multi-valued, allowing service instances with several property values, or single-valued, requiring service instances to have at most one value for the property. By the number restriction $\leq 1\ r$, a property is marked as single-valued. Using the restrictions $\leq m\ r$ (with $m \geq 2$) $\geq n\ r$, $\exists r.\top$, $\exists r.C$, and $\forall r.C$ a property is marked as multi-valued.

Coverage: a property can be explicitly known to cover a range. If a property is *range-covering*, the service description enforces that in every possible world, for any value in the range, there is an acceptable service instance with this property value. This introduces variance due to intended diversity. This kind of constraint is expressed by an axiom of the form $C \sqsubseteq \exists r^-.S$ in D, where the concept C is the range of the property r to be covered[21]. A non-range-covering property induces variance due to incomplete knowledge, as in distinct possible worlds different subsets of the range will be covered.

Example 5.2.1 *In order to make more clear the illustrated framework, it is used for describing the following service request jointly with the reference knowledge base.*

$$\begin{array}{ll} D_r = \{ & S_r \equiv \mathsf{Company} \sqcap \exists \mathsf{payment.EPayment} \sqcap \exists \mathsf{to.}\{\mathsf{bari}\} \sqcap \\ & \quad \sqcap \exists \mathsf{from.}\{\mathsf{cologne,hahn}\} \sqcap \leq 1\ \mathsf{hasAlliance} \sqcap \\ & \quad \sqcap \forall \mathsf{hasFidelityCard.}\{\mathsf{milesAndMore}\}; \\ & \{\mathsf{cologne,hahn}\} \sqsubseteq \exists\ \mathsf{from}^{-}.S_r \qquad \} \\ KB = \{ & \mathsf{cologne{:}Germany,\ hahn{:}Germany,\ bari{:}Italy,\ milesAndMore{:}Card}\} \end{array}$$

D_r is the requested service, described as a set of axioms that impose restrictions on S_r, that is the service that has to be performed. Here, the requester asks for companies that fly from Cologne and Hahn to Bari and accept electronic payment when selling tickets. Moreover, it is required that a company has at most one alliance with another company and, if it has a fidelity program, it has to be "Miles and More". In this service description, several kinds of constraints have been used. Variety *constraints are used with the properties* to, from *and* hasFidelityCard, *indeed these properties are restricted to a fixed value. The* at-most *number restriction (≤ 1) for the property* hasAlliance *is a* Multiplicity *constraint with which the property* hasAlliance *is declared to be single-value. A* Coverage *constraint is expressed by the last axiom in D_r which makes explicit the range covered by the property* from. *Namely, this axiom asserts that* {cologne,hahn} *is the range coverage of the property* from.

If one rather requires that, for all companies, the payment method is specified and that the unique method allowed is electronic payment, the service description has to be:

$$\begin{array}{ll} D_r = \{ & S_r \equiv \mathsf{Company} \sqcap \exists \mathsf{payment.EPayment} \sqcap \forall \mathsf{payment.EPayment} \sqcap \\ & \quad \sqcap \exists \mathsf{from.}\{\mathsf{cologne,hahn}\} \sqcap \exists \mathsf{to.}\{\mathsf{bari}\} \sqcap \leq 1\ \mathsf{hasAlliance} \sqcap \\ & \quad \sqcap \forall \mathsf{hasFidelityCard.}\{\mathsf{milesAndMore}\}; \\ & \{\mathsf{cologne,hahn}\} \sqsubseteq \exists\ \mathsf{from}^{-}.S_r \qquad \} \end{array}$$

In this way the existence of a payment method has been forced. Moreover it as been constrained by allowing a unique form of payment, that is electronic payments.□

The services presented in the example represent relatively simple descriptions. Indeed, in real scenarios a service request is typically characterized by some needs

[21]Obtained by transforming the axiom $\forall x | C(x) \rightarrow \exists y : [r(y,x) \wedge S(y)]$ into DL by standard manipulation of first-order formulæ.

that *must* be necessarily satisfied and others that *may* be satisfied (expressing a preference). The former will be considered as *Hard Constraints* and the latter as *Soft Constraints*. Specifically, *HC* represent necessary and sufficient conditions for selecting requested service instances, while *SC* represent only necessary conditions. Taking this difference into account makes the service description and management more complex. A possible solution is to describe services (and particularly service requests) by the use of two different sets: the *HC* set and the *SC* set, whose elements are expressed in DLs as seen above.

More formally, let $D_r^{HC} = \{S_r^{HC}, \sigma_1^{HC}, ..., \sigma_n^{HC}\}$ be the set of *HC* for a requested service description D_r and let $D_r^{SC} = \{S_r^{SC}, \sigma_1^{SC}, ..., \sigma_m^{SC}\}$ be the set of *SC* for D_r. Every element in D_r^{HC} and in D_r^{SC} is expressed as previously seen. The complete description of D_r is given by $D_r = \{S_r \equiv S_r^{HC} \sqcup S_r^{SC}, \sigma_1^{HC}, ..., \sigma_n^{HC}, \sigma_1^{SC}, ..., \sigma_m^{SC}\}$. In this description, new information on constraint hardness has been added.

Example 5.2.2 *In order to understand service descriptions by distinguishing between* HC *and* SC *the following example is considered. It is a slightly modified version of the example 5.2.1:*

$$\begin{aligned} D_r = \{ \quad & S_r \equiv \textsf{Flight} \sqcap \ \exists \textsf{from}.\{\textsf{cologne, hahn, frankfurt}\} \sqcap \exists \textsf{to}.\{\textsf{bari}\} \sqcap \\ & \quad \sqcap \forall \textsf{hasFidelityCard}.\{\textsf{milesAndMore}\}; \\ & \{\textsf{cologne, hahn, frankfurt}\} \sqsubseteq \exists \ \textsf{from}^{-}.S_r; \quad \{\textsf{bari}\} \sqsubseteq \exists \ \textsf{to}^{-}.S_r \quad \} \end{aligned}$$

where

$$\begin{aligned} HC_r = \{ \quad & \textsf{Flight} \sqcap \exists \textsf{to}.\{\textsf{bari}\} \sqcap \exists \textsf{from}.\{\textsf{cologne, hahn, frankfurt}\}; \\ & \{\textsf{cologne, hahn, frankfurt}\} \sqsubseteq \exists \ \textsf{from}^{-}.S_r; \quad \{\textsf{bari}\} \sqsubseteq \exists \ \textsf{to}^{-}.S_r \quad \} \\ SC_r = \{ \quad & \textsf{Flight} \sqcap \forall \textsf{hasFidelityCard}.\{\textsf{milesAndMore}\}\}; \\ KB = \{ \quad & \textsf{cologne,hahn,cologne:Germany, bari:Italy, milesAndMore:Card}\} \end{aligned}$$

With this service description a requester asks for flights starting from Frankfurt or Cologne or Hahn and arriving at Bari. The use of "Miles And More" card would be preferred. Departure and arrival places are expressed as HC*. This means that provided services must fulfill these constraints. This is understandable thinking, for instance, of a requester who wants to go from Koblenz to Bari. He/she is interested in Cologne, Hahn and Frankfurt airports because they have the same distance from Koblenz, while he/she is not interested in other airports because much more faraway. Instead, the use of "Miles And More" card is expressed as* SC*, namely flights that allow the use of this card are preferred, but the requester accepts also flights that do not allow the use of this card. This is because the use of* Miles and More *card is advantageous for the requester but it is not the primary need; his/her primary need is to have a flight for reaching Bari.* □

This new representation constitutes a flexible framework for service modeling as it allows to better model the requester's needs, expressing them as real-life preferences; a feature that is not considered in the original framework [93]. Moreover expressing SC allows to have service instances satisfying a request even if part of the request is ignored; this increases the possibility of having plausible responses.

5.2.2 Efficient Service Discovery by means of Clustering Methods

Service Discovery is the task of locating service providers that can satisfy the requester's needs. It consists of matching the description of a service request to the descriptions of published service in a registry. Traditionally such match has been performed on the ground of syntactic characteristics. Namely, it has featured syntax-based searches taking into account keywords and taxonomies. Specifically, given one or more keywords, descriptions containing the input keywords are returned as search results. Anyway, this kind of search can often results poor in performance, because it can fail to retrieve services described by synonyms (as well as singular/plural) of the searched string. Then, the requester must select the service which satisfies his requirements by manually filtering the returned services.

In this scenario, semantic service descriptions can be used to automate the discovery task. Discovery is performed by matching a requested service description to the service descriptions of potential providers, in order to detect relevant ones. Two service descriptions match if there is an acceptable instance for both descriptions [196, 161, 93, 129]. Hence, the service instance provides a basis for a business interaction between the provider and the requester. Semantic matching techniques are analyzed in more details in the following. Considering the framework presented in Sect. 5.2.1, let D_r and D_p respectively a requested service description and a provided service description, expressed as a set of axioms imposing restrictions on the services that have to be performed, say S_r and S_p, respectively. The matching process (w.r.t. a KB) can be defined as a boolean function $match(KB, D_r, D_p)$ which specifies how to apply DLs inferences to perform matching. Various matching procedures, based on DLs inferences, have been proposed [129, 196, 161].

Here, the attention is focussed on the semantic matching process proposed in [93]. Differently from the others, this procedure is able to treat *variance* (particularly variance due to incomplete knowledge) without being too weak or too strong. The assumption on which the matching procedure is based is that precise control of *variance* in service description is crucial to ensure quality of the discovery process. The other matching procedures [195, 196, 88] consider a match valid if there exists a common instance service at least in *one possible world.* It can be formalized as:

$$KB \cup D_r \cup D_p \cup \{\exists x : S_r(x) \wedge S_p(x)\} \ is\ consistent \Leftrightarrow$$

$$\Leftrightarrow KB \cup D_r \cup D_p \cup \{i : S_r \sqcap S_p\} \ is\ satisfiable$$

This match is called *Satisfiability of Concept Conjunction.* It is the weakest check w.r.t. both kinds of variance. Indeed, along the dimension of intended diversity, it is sufficient to find one common service instance. Along the dimension of incomplete knowledge, it is sufficient to find one possible world in which such a service instance exists, regardless of all other possible worlds.

Another type of matching procedure [129, 159, 157] executes the match by checking for subsumption, either of the requester's description by the provider's or vice versa. It can be formalized as:

$$KB \cup D_r \cup D_p \models \forall x : S_r(x) \rightarrow S_p(x) \Leftrightarrow KB \cup D_r \cup D_p \cup \{i : (S_r \sqcap \neg S_p)\} \text{ is unsatisfiable}$$

It is called *Entailment of Concept Subsumption.* This check is very strong, since it requires one of the service descriptions to be more specific than the other, for all service instances in all possible worlds.

Conversely, a valid match for the procedure in [93] occurs when there exists a common instance service between a provider's service description D_p and a requester's service description D_r w.r.t. KB, in *every possible world.* It can be formalized as:

$$KB \cup D_r \cup D_p \models \exists x S_r(x) \wedge S_p(x) \Leftrightarrow KB \cup D_r \cup D_p \cup \{S_r \sqcap S_p \sqsubseteq \bot\} \text{unsatisfiable}$$

This check is called *Entailment of Concept Non-Disjointness.* It is stronger than *Satisfiability of Concept Conjunction* because checks for an intersection in every possible world, but it is not as strong as *Entailment of Concept Subsumption*, because it does not require one of the sets of acceptable service instances to be fully contained in the other set. This match increases (w.r.t *Entailment of Concept Subsumption*) the possibility to find interesting provided services, decreasing the error due to variety (more present in *Satisfiability of Concept Conjunction*). It is important to note that for this procedure, two service descriptions match if their conjunction is not subsumed by the *bottom* concept, hence it is basically grounded on the subsuption inference service. Consequently, the complexity of the matching procedure depends from the complexity of the subsumption operator for the chosen DL.

Normally, service discovery is performed by applying the matching procedure to every provided services. This means that the discovery process has a linear complexity in the number of the provided services (besides of the complexity of the matching procedure itself). With the increasing number of available services, this can constitute an efficiency problem. To cope with this problem, here, it is proposed to apply a (conceptual) clustering method to the available services, in order to obtain subsets of services, homogeneously grouped and intensionally described. Hence, the service discovery can be performed by matching the request to the intensional cluster descriptions rather than to all services, thus decreasing the number of comparison.

As seen in Sect. 1.3.2, many different methods exist in order to solve clustering problems. Among the others, hierarchical methods have been analyzed, as they are versatile and suitable for clustering non-large amount of data with few available information [108]. They produce a nested series of partitions based on similarity criteria for merging or splitting clusters; the clustering structure obtained is called *dendrogram*[22] that is basically a tree structure.

In the considered DLs-based framework for service description (see Sect. 5.2.1), the set of all provided services can be seen as a set of DLs concept descriptions[23] on which a hierarchical clustering method can be applied. A clustering method generally requires the availability of (dis-)similarity measures able to cope with and capture the semantics of the objects to cluster (see Sect. 1.3.2 for more details). In Chap. 4 a set of (dis-)similarity measures able to capture the semantics of DL concept descriptions has been presented. Hence, a clustering method could be "easily" applied to service descriptions in order to obtain meaningful and homogeneous subsets of services. Specifically, a hierarchical method can be used, obtaining as output a dendrogram of the provided services, where the actual service descriptions are in the leaves, while intermediate nodes represent an intensionally described superset of all service descriptions in the leaves below. Particularly, an inner node groups all service descriptions below. The root node contains all services. Intensional description of the nodes can be constructed by computing the least common subsumer[24] (lcs) of the descriptions belonging to this node[25]. Once that such dendrogram is obtained, the matching process can be formalized as in the following [192].

A user request R is formulated as a service description fulfilling the needs of the requester. Following the Entailment of Concept Non-Disjointness matching process, a service description D may fulfill the given request if their extensions overlap. In the dendrogram, each inner node N represents a superset of all service descriptions below. If $R \sqcap N \sqsubseteq \bot$, the branch below can be ignored, because there is no match between R and the underlying service descriptions; otherwise the children node of N

[22]A dendrogram is a nested grouping of patterns and similarity levels at which grouping change. The dendrogram could be broken at different levels to yield different clustering of the data.

[23]Such concept descriptions refer to the same vocabulary, hence they can be seen as concept definitions in the same ontology.

[24]See Sect. 2.4.1 for more details about such non-standard inference service.

[25]Note that the computation of lcs is constant in time for DLs allowing concept disjunction, which is realistic to be considered for describing services.

are recursively explored, until the leaves of the tree are reached. Hence, following the path of overlapping nodes leads to the service descriptions that satisfy the request.

In this way, the search space can be drastically reduced, indeed, during the query processing, matchmaking can be restricted to the branches of the tree where tree nodes indicate an overlapping between user requests and service descriptions. Particularly, a good clustering of n available service descriptions may significantly reduce the number of necessary comparisons for satisfying a request, thus improving retrieval time from $O(n)$ to $O(log\ n)$ for concise queries.

Modeling services requires expressive DLs in order to describe their functionalities in the best possible way. Hence DLs allowing the use of concept disjunctions have to be realistically considered. However, in this case, the computation of the lcs of a set of service descriptions (computed for building the intensional cluster descriptions) is simply given by their disjunction (see Sect. 2.4.1). This could provoke the overfitting phenomenon in generalizing cluster descriptions. In order to avoid this possible risk, a different generation of the intensional cluster descriptions could be performed.

Here, service description in $\mathcal{ALC}$ logic have been considered. As seen in Sect. 2.4.4, it is always possible, given an $\mathcal{ALC}$ concept description to obtain an $\mathcal{ALE}$ approximation of it. $\mathcal{ALE}$ logic is less expressive than $\mathcal{ALC}$, it does not allow concept disjunction (see Sect. 2.2) and hence, the lcs is computed in a structural way (see Sect. 2.4.1). Considered such theoretical results, a different and more general intensional cluster description can be generated. Particularly, given an inner node in the tree, every service description (which is expresses in $\mathcal{ALC}$ logic) belonging to it, can be approximated to an $\mathcal{ALE}$ concept description. Hence, such approximations can be considered for computing the lcs (in $\mathcal{ALE}$). In this way a more general and compact cluster representation is obtained. Fig. 5.4 shows a tree (dendrogram) for four clustered service descriptions.

In the following, the realized clustering methods, applied to DLs knowledge bases, will be examined in details.

Applying Clustering Methods

Here, the clustering process is performed without the use of any information besides of the (dis-)similarity values among the service descriptions that have to be clustered. Neither the number of clusters to generate is known, nor the classes to learn. In order to cope with this situation, the hierarchical agglomerative clustering approach has been chosen. As *simple-link* and *complete-link* algorithms represent the corner stones of such an approach (see Sect. 1.3.2), a modified version of such algorithms has been realized. Main variations regard their applicability to DLs knowledge

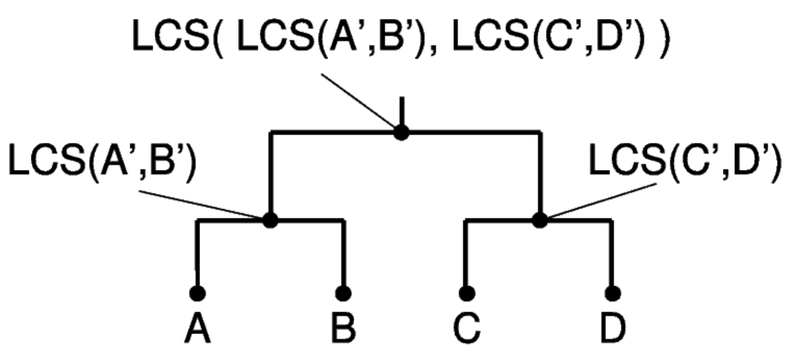

Figure 5.4: Tree (dendrogram) returned by a hierarchical clustering method applied to the services A, B, C, and D. For computation of the lcs, which is performed in $\mathcal{ALE}$, they are mapped from $\mathcal{ALC}$ to $\mathcal{ALE}$ leading to A', B', C', and D'.

representations and the introduction, at each dendrogram level, of intensional cluster descriptions, thus obtaining a form of conceptual clustering [146].

Indeed, conceptual clustering is mainly defined as a process of constructing a concept network characterizing a collection of objects, with nodes marked by concepts describing object classes and links marked by the relationship between the classes. In the case developed, the links between classes represent generalization-specialization relationships. Moreover, in conceptual clustering, not only the inherent structure of the data drives cluster formation, but also the description language which is available to the learner. This aspect has been successfully treated by the presented measure for DLs (see Chapt. 4) that effectively exploit not only structural representation of the data, but also their meaning. Moreover such measures are also *context-sensitive*[26], as they do not consider only object features but also the context of reference, namely the KB. Indeed, because the measures use DLs inference operator, they also take into account external aspects not directly linked with the considered element, such as, (assertion of) concept disjunction or subsuption in the KB. Furthermore, it is important to note that a conceptual clustering in which the concepts to learn are known (as those proposed by Michalski in [146]), cannot be applied in this case because there is no available information. In the following the realized clustering algorithms are detailed.

Single-Link Algorithm

Let $S = \{S_1, \dots, S_n\}$ a set of available ($\mathcal{ALC}$) service descriptions.

1. Let $\mathcal{C} = \{C_1, \dots, C_n\}$ the set of the initial clusters obtained by considering every service description as a single cluster

[26]For more details about characteristic of context sensitive measures see Sect. 1.3.2.

```
2. Let $\mathcal{C}' = \{approx_{\mathcal{ALE}}(C_1), \ldots, approx_{\mathcal{ALE}}(C_n)\}$ be the $\mathcal{ALE}$ approximations of
   the descriptions in $\mathcal{C}$
3. For $i := 1$ to NumberOfClusters - 1 consider cluster $C'_i$
   (a) For $j := i + 1$ to NumberOfClusters consider cluster $C'_j$
       i. compute[27] $d_{kl}(s_{ik}, s_{jl})$ on varying $k = \{1, \ldots, NumObjOfC'_i\}$ and
          $l = \{1, \ldots, NumObjOfC'_j\}$
       ii. compute $\min_{ij} = \min_{k=1,\ldots,NumObjOfC'_i,\ l=1,\ldots,NumObjOfC'_j}\{d_{kl}\}$
4. compute $\min_{hk} = \min_{i,j=1,\ldots,NumOfClusters} \min_{ij}$
   where $h$ and $k$ are the clusters with minimum distance
5. create the intensional cluster description $C'_m = lcs_{\mathcal{ALE}}(C'_h, C'_k)$
6. populate with $C'_h$ and $C'_k$
7. link $C'_m$ to $C'_h$ and $C'_k$
8. add $C'_m$ in $C'$ and delete $C'_h$ and $C'_k$ from $C'$
9. if $|\mathcal{C}'| \neq 1$ go to 3
```

Complete-link Algorithm

As seen in Sect. 1.3.2, the complete-link algorithm differs from the single link algorithm only for the way of computing cluster distances. This is also valid for the new modified version of the single and complete link algorithms. The complete-link algorithm is obtained from the algorithm above, by simply substituting line 3(a)ii with: "compute $\max_{ij} = \max_{k=1,\ldots,NumObjOfC'_i,\ l=1,\ldots,NumObjOfC'_j}\{d_{kl}\}$" and line 4 with "compute $\min_{hk} = \min_{i,j=1,\ldots,NumOfClusters} \max_{ij}$ where h and k are the clusters with maximum distance". Namely the complete link algorithm is obtained by considering the maximum distance among clusters rather than the minimum one.

The principal source of complexity of these algorithms is represented by the computation of the matrix containing the dissimilarity values of all elements. Hence, by first computing such matrix, the algorithms have to mainly compute only the maximal or the minimum distance among clusters which is a tractable computational operation. Moreover, the computation of the matrix can be optimized[28]. Furthermore, it is important to note that in the presented algorithms, the intensional cluster descriptions are built by approximating service descriptions to a less

[27]Here, one of the measures presented in Chapt. 4 can be used. Let be assumed that the measure based on Information Content (see Sect. 4.3) is applied.

[28]As the dissimilarity value of an element w.r.t. itself is null and the measures are symmetric, only the inferior diagonal of the matrix can be computed.

expressive DL, which is $\mathcal{ALE}$, and then computing the structural $\mathcal{ALE}$ lcs. If no concept approximations are performed, the algorithms work as well by computing $\mathcal{ALC}$ lcs.

Even if the presented algorithms realize a form of conceptual clustering, they are not able to exploit the intensional cluster representation, indeed the criterion for merging clusters is based on distances among elements belonging to the existing clusters. The intensional description are only used by the further application (which is service matchmaking in the presented case). Moreover, the criterion used, by the single and complete link algorithms, for merging clusters, could sometimes provokes drawbacks in presence of noisy data (see Sect. 1.3.2). In order to overcome this limitation and exploit the intensional cluster descriptions, a new clustering algorithm has been realized, on the ground of single-link and complete link algorithms. This new algorithm, called *lcs-based clustering algorithm* is able to exploit the intensional cluster descriptions during the clustering process and, for this reason, hopefully it can also overcome the drawbacks. The lcs-based algorithm is detailed in the following.

lcs-based Clustering Algorithm

Let $S = \{S_1, \ldots, S_n\}$ a set of available ($\mathcal{ALC}$) service description.

```
1. Let $\mathcal{C} = \{C_1, \ldots, C_n\}$ the set of initial clusters obtained by considered every
   service description as a single cluster
2. Let $\mathcal{C}' = \{approx_{\mathcal{ALE}}(C_1), \ldots, approx_{\mathcal{ALE}}(C_n)\}$ the approximated descriptions
   from $\mathcal{ALC}$ DL to $\mathcal{ALE}$
3. For $i := 1$ to NumberOfClusters - 1 consider cluster $C'_i$
   (a) For $j := i + 1$ to NumerOfClusters consider cluster $C'_j$
       i. compute the dissimilarity values $d_{ij}(C'_i, C'_j)$
4. compute $\min_{hk} = \min_{i,j=1,\ldots,NumOfClusters} d_{ij}$ where $h$ and $k$ are the clusters
   with minimum distance
5. create the intensional cluster description $C'_m = lcs_{\mathcal{ALE}}(C'_h, C'_k)$
6. link $C'_m$ to $C'_h$ and $C'_k$
7. add $C'_m$ in $C'$ and delete $C'_h$ and $C'_k$ from $C'$
8. if $|\mathcal{C}'| \neq 1$ go to 3
```

Differently from the algorithms illustrated above, the lcs-based algorithm uses the intensional cluster descriptions during the clustering phase. Indeed the two clusters with minimum distance are found and merged in single cluster. They are used for the construction of the intensional cluster description, after that they are discarded. The generated description is considered as a cluster made by a single element in the next clustering step. These operations are repeated until a single cluster is obtained. Such algorithm is more expensive than the previous ones[29], but it is also more powerful. Indeed, it considers intensional cluster description during the clustering process rather than representative elements, realizing a really concept-driven approach.

It is important to note that all the presented algorithms merge, at every step, two existing clusters into a single one, thus returning a binary tree as result of the clustering process. Finding a way for merging more than two clusters, at every step, could be useful to speed up the clustering process and consequently decrease its complexity. An important result, in this direction, has been reported in [64], where it is proved that, if the measure used for performing the clustering satisfies the *cluster aggregate inequality property*, then a multiple merging of clusters can be performed at each level. It is easy to verify that this result equally holds for the lcs-based clustering algorithm.

All the clustering algorithms proposed here can be used also for clustering individuals asserted in an ontology. This can be done by firstly computing the (approximated) msc of the individuals and then by applying the algorithms.

In the following the experimental evaluation of algorithms will be shown. Unfortunately, due to lack of data sets regarding SWS, an experimentation regarding also the matching process has not been performed. Anyway, proved the validity of the obtained clusters, it is straightforward to understand that the matching process works as well.

Experimental Evaluation

In order to assess the validity of the presented clustering algorithms and of the used dissimilarity measure, they have been applied to a conceptual clustering problem. Namely, concepts asserted in an ontology have been grouped into clusters by the use of the single-link, the complete-link and the lcs-based algorithm. They have been used jointly with the dissimilarity measure for $\mathcal{ALC}$ concept based on information content (see Sect. 4.3). The clustering process has been performed on 9 ontologies

[29]Even if all dissimilarity values are pre-computed, at each step the dissimilarity values between the new element (intensional cluster description) and all remaining not-clustered descriptions have to be computed.

Table 5.11: Ontologies employed in the experiments.

ontology	*DL*	*#concepts*	*#obj. prop*	*#data prop*	*#individuals*
PEOPLE	$\mathcal{ALCHIN}(D)$	60	14	1	21
UNIVERSITY	$\mathcal{ALC}$	13	4	0	19
FAMILY	$\mathcal{ALCF}$	14	5	0	39
FSM	$\mathcal{SOF}(D)$	20	10	7	37
S.-W.-M.	$\mathcal{ALCOF}(D)$	19	9	1	115
SCIENCE	$\mathcal{ALCIF}(D)$	74	70	40	331
NTN	$\mathcal{SHIF}(D)$	47	27	8	676
NEWSPAPER	$\mathcal{ALCF}(D)$	29	28	25	72
WINES	$\mathcal{ALCIO}(D)$	112	9	10	188

represented in OWL, analyzed in the previous experiments (see Sect. 5.1.2). Table 5.11 summarizes all details concerning the employed ontologies. Although they are represented in languages that are different from $\mathcal{ALC}$, these details are simply discarded, in order to be able to apply the $\mathcal{ALC}$ dissimilarity measure.

Given the set of concepts (primitive and defined) in an ontology, they were clustered using the measure formalized in Def. 4.3.2. No other information was used, such as: concept labels or number of concepts to learn. Hence, since no external information (i.e. concept labels) was available, there was no possibility to evaluate the *external clustering quality*, namely how well the clustering is working, by comparing the obtained groups w.r.t. the known classes. Only the *internal quality* of the obtained clusters was evaluated. The internal quality of the clusters is generally measured in terms of the cohesiveness of clusters. It expresses the *overall clusters similarity*. One of the most common methods for computing the cluster cohesiveness is to use the weighted (dis-)similarity of the internal cluster (dis-)similarity [188]. It can be computed as $1/|S^2| \sum_{c_i, c_j \in S} d(c_i, c_j)$ where S is the considered cluster and d is the used (dis-)similarity measure. Considered a dissimilarity measure, as much the overall similarity is close to zero, best the quality of the cluster is. As much the similarity value is close to 1, worst the quality of the cluster is.

The experimentation results are shown in Tab. 5.12, where the average overall similarity values have been computed for each ontology w.r.t. the used algorithm. The results have been summarized in Fig. 5.5. By looking at the table it is possible to see that the average of the overall similarity values for all ontologies, and w.r.t. all three clustering algorithms, are very close to 0; the maximum value is 0.186, reached for the FAMILY ontology. This means that the clustering methods can be considered

Table 5.12: Average overall clusters similarity for each considered ontology and with respect to the employed clustering algorithm.

	single-link	*complete-link*	*lcs-link*
People	0.064	0.109	0.061
University	0.094	0.092	0.159
FSM	0.073	0.076	0.079
Family	0.157	0.171	0.186
Newspaper	0.144	0.134	0.158
Wines	0.055	0.060	0.077
Science	0.050	0.047	0.053
S.-W.-M.	0.105	0.092	0.157
NTN	0.137	0.105	0.142

valid. Particularly, the highest similarity values are reached for the University and Family ontologies, that contain the lowest number of concepts. Hence, it could be possible to deduce that the overall similarity increases with the increase of the number of concepts to cluster. Particularly, looking at Fig. 5.5, it seems that lcs-based clustering algorithm suffer from the lack of concepts in a particular way, while it is comparable with complete-link and single-link algorithms in the other cases.

In general, the obtained results confirm the validity of the proposed clustering methods and hence, they can be effectively used in order to improve the efficiency of the matching services process.

5.2.3 Effective Service Ranking based on DLs measures and Constraint Hardness

The provided services, selected by the matching process, are returned to the requester, in order to start the negotiation process[30] between the requester and providers.

Generally, such services are returned in a flat list where no ranking is provided. However, a ranked list of the selected services is fundamental to decrease the complexity of the negotiation process. Some proposals for ranking procedures have been formulated. They are grounded on the subsumption relationship between the request and the provided service description [93]. Namely services that are more general w.r.t. the request will be suggested as more adequate for the further negotiation process. This criterion is based on the intuition that more general a service

[30]The negotiation process is finalized to further refine service parameters. See Sect. 1.2.2 for more details about such process.

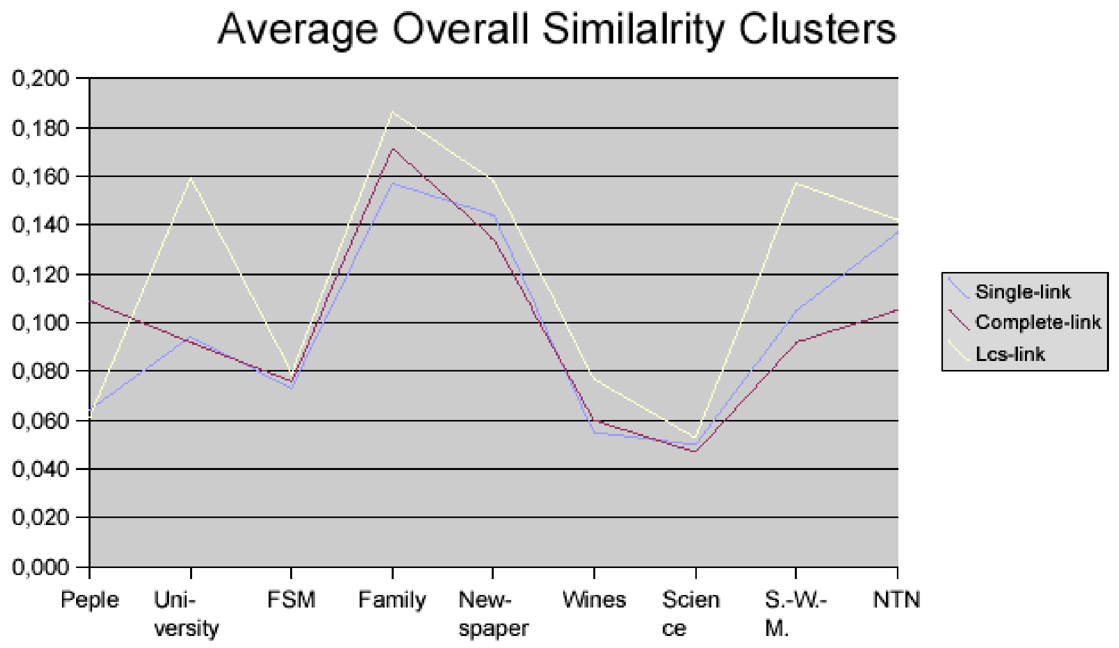

Figure 5.5: Means of the overall clusters similarity computed for every ontology with respect to the used clustering algorithm.

is, with high probability it will be able to satisfy the request. Even if this criterion could be adequate, it has to be noted that a general service could be less able to perform a request w.r.t. a more specific one, that could better match the requested needs. Moreover this criterion is not able to cope with the notions of *HC* and *SC*.

Here, a ranking procedure is proposed [57]. Based on the use of a semantic similarity measure for DL descriptions, it is able to manage *HC* and *SC*. Particularly, it assigns higher ranks to services that are more similar to the request and that satisfy both *HC* and *SC*, while services that are less similar and/or satisfy only *HC* receive a low rank.

One of the measures for DLs presented in Chap. 4 could be used. Here the simplest one is considered, that is the semantic similarity measure[31] presented in Sect. 4.1. This measure is able to assess a similarity value between concepts, individuals and concept and individual asserted in the same ontology. Considering Sect. 5.2.1, service descriptions can be viewed as DL descriptions asserted in a *TBox* and their instances can be regarded as concept assertions in an *ABox*, hence the *Canonical Interpretation* [8], which maps every service description to its instances,

[31]This measure has been defined for $\mathcal{ALC}$ descriptions. However, as discussed in Sect. 4.1, it can be applied to any DLs allowing for the instance checking operator.

can be considered. Thus, given a requested service description S_r and a provided service description S_p, their similarity can be computed as in the following:

$$s(S_r, S_p) = \frac{|(S_r \sqcap S_p)^{\mathcal{I}}|}{|(S_r \sqcup S_p)^{\mathcal{I}}|} \cdot \max(\frac{|(S_r \sqcap S_p)^{\mathcal{I}}|}{|S_r^{\mathcal{I}}|}, \frac{|(S_r \sqcap S_p)^{\mathcal{I}}|}{|S_p^{\mathcal{I}}|)})$$

where $(\cdot)^{\mathcal{I}}$ returns the concept extension w.r.t. $\mathcal{I}$

Remember that this measure assigns the maximum value in case of semantic equivalence of the descriptions. Otherwise it assigns a value in the range $[0, 1[$. This value grows with the increase of the cardinality of the set of service instances shared. The similarity value is computed between a requested service description S_r and a provided service description S_p, so the instance checking operator has to compute the set of instances for them. For every provided service, the set of instances is known. It is necessary to determine the extension of S_r. Note that the measure is applied after the matching process and that the chosen matching procedure (see Sect.5.2.2) selects all provided services S_p that have at least one instance satisfying S_r. More specifically, the matching process selects all provided services that have at least one instance satisfying S_r^{HC}, while SC cannot be also satisfied. So, the extension of S_r is given by the union of the provided service instances that satisfy S_r. Namely:

$$S_r^I = \bigcup_{j=1}^{n} \{x | S_r^{HC}(x) \wedge S_p^j(x)\} \cup \bigcup_{j=1}^{n} \{x | (S_r^{HC} \sqcap S_r^{SC})(x) \wedge S_p^j(x)\}$$

where n is the number of provided services selected by the matching process.

The rationale of the procedure consists in measuring the similarity between the requested and the provided services selected during the matching phase: a higher similarity will result in higher rankings.

The presented measure assigns highest values to services that share most of the instances with S_r so, as in [93], the criterion used is based on *variance*, namely, a service is better than another if the variance it provides is greater than the other. However, differently from [93], this procedure is able to supply a total order of the provided services (rather than a partial order). Anyway this is not enough for ensuring that provided services satisfying both HC and SC of S_r will be in the higher positions, while services satisfying only HC will be in the lower positions. To make more clear the reason, the following scenario is considered: let S_r be the requested service and let S_p^l and S_p^k two provided services, selected by the matching procedure. As seen in Sect. 5.2.1, a service is described by the set of HC and SC. Particularly, a service can also be described only by HC[32]. Let S_r described by both HC and SC, let S_p^l be a provided service whose instances all satisfy only the HC of S_r, and let S_p^k be a provided service whose instances all satisfy both HC and SC of S_r. So,

[32]A service can not be described only by SC because it means ask for a service that contains only optional constraints and this does not make sense.

considered the canonical interpretation, it is easy to see that
$\forall x : S_p^k(x) \rightarrow S_p^l(x) \Leftrightarrow (S_p^k)^I \subseteq (S_p^l)^I \Rightarrow |(S_p^k)^I| \leq |(S_p^l)^I| \Rightarrow s(S_r, S_p^k) \leq s(S_r, S_p^l)$.
This is the opposite result w.r.t. the fixed criterion, that is provided services satisfying both HC and SC of S_r and that are more similar to S_r have to be on top of the ranking. For achieving this goal, the set up ranking procedure is:

```
given $S_r = \{S_r^{HC}, S_r^{SC}\}$ service request; $S_p^i (i = 1, .., n)$ provided services selected
by $match(KB, D_r, D_p^i)$;
for $i = 1, \ldots, n$ do compute $\bar{s}_i := s(S_r^{HC}, S_p^i)$
let be $S_r^{new} \equiv S_r^{HC} \sqcap S_r^{SC}$
for $i = 1, \ldots, n$ do
    compute $\bar{\bar{s_i}} := s(S_r^{new}, S_p^i)$
    $s_i := (\bar{s}_i + \bar{\bar{s_i}})/2$
```

S_r^{HC} represents the set of HC of the request, S_r^{SC} the set of SC for the request. For all S_p^i, the similarity values $\bar{s}_i := s(S_r^{HC}, S_p^i)$ are computed. Hence, a new service description $S_r^{new} \equiv S_r^{HC} \sqcap S_r^{SC}$ is considered. It is defined as the conjunction of HC and SC of S_r. The instances of S_r^{new} satisfy both HC and SC of S_r. So, for all S_p^i the similarity values $\bar{\bar{s}} := s(S_r^{new}, S_p^i)$ are computed. Note that this value expresses how many and which provided services satisfy also SC of S_r besides of its HC. Consequently, it is straightforward to understand that a S_p^i satisfying only HC of S_r will has $\bar{\bar{s}} = 0$. For all S_p^i, the final similarity value s_i is given by the average between $\bar{s}$ and $\bar{\bar{s}}$. This last value s_i is used for setting the rank of the services. In order to clarify the presented procedure, the following example is considered.

Example 5.2.3 *Let D_r be a request and let D_p^l and D_p^k be two provided services, selected by the matching process. D_p^l and D_p^k have to be ranked. Here D_p^l and D_p^k are described specifying their* HC *and* SC. *However, the procedure can rank services even if they are described without any specification about their constraint hardness.*

$$
\begin{aligned}
D_r = \{ \quad & S_r \equiv \mathsf{Flight} \sqcap \forall \mathsf{operatedBy.LowCostCompany} \sqcap \exists \mathsf{to.}\{\mathsf{bari}\} \sqcap \\
& \quad \sqcap \exists\ \mathsf{from.}\{\mathsf{cologne,hahn}\} \sqcap \forall \mathsf{applicableToFlight.Card}; \\
& \{\mathsf{cologne,hahn}\} \sqsubseteq \exists\ \mathsf{from}^{-}.S_r \qquad \}
\end{aligned}
$$

where

$$
\begin{aligned}
HC_r = \{ \quad & \mathsf{Flight} \sqcap \exists \mathsf{to.}\{\mathsf{bari}\} \sqcap \exists\ \mathsf{from.}\{\mathsf{cologne,hahn}\} \\
& \{\mathsf{cologne,hahn}\} \sqsubseteq \exists\ \mathsf{from}^{-}.S_r \qquad \}
\end{aligned}
$$

$$SC_r = \{ \quad \mathsf{Flight} \sqcap \forall \mathsf{operatedBy.LowCostCompany} \sqcap \forall \mathsf{applicableToFlight.Card}\ \};$$

$D_p^l = \{ \quad S_p^l \equiv \textsf{Flight} \sqcap \exists \textsf{to.Italy} \sqcap \exists \textsf{from.Germany};$
$\quad \textsf{Germany} \sqsubseteq \exists \textsf{ from}^- .S_p^l; \quad \textsf{Italy} \sqsubseteq \exists \textsf{ to}^- .S_p^l \quad \}$

where

$HC_p^l = \{ \quad \textsf{Flight} \sqcap \exists \textsf{to.Italy} \sqcap \exists \textsf{from.Germany};$
$\quad \textsf{Germany} \sqsubseteq \exists \textsf{ from}^- .S_p^l; \quad \textsf{Italy} \sqsubseteq \exists \textsf{ to}^- .S_p^l \quad \}$ $\qquad SC_p^l = \{\}$

$D_p^k = \{ \quad S_p^k \equiv \textsf{Flight} \sqcap \forall \textsf{operatedBy.LowCostCompany} \sqcap \exists \textsf{to.Italy} \sqcap$
$\quad \sqcap \exists \textsf{from.Germany};$
$\quad \textsf{Germany} \sqsubseteq \exists \textsf{ from}^- .S_p^k; \quad \textsf{Italy} \sqsubseteq \exists \textsf{ to}^- .S_p^k \}$

where

$HC_p^k = \{ \quad \textsf{Flight} \sqcap \exists \textsf{to.Italy} \sqcap \exists \textsf{from.Germany};$
$\quad \textsf{Germany} \sqsubseteq \exists \textsf{ from}^- .S_p^k; \quad \textsf{Italy} \sqsubseteq \exists \textsf{ to}^- .S_p^k \quad \}$

$SC_p^k = \{ \quad \textsf{Flight} \sqcap \forall \textsf{operatedBy.LowCostCompany}\};$

$KB = \{ \quad \textsf{cologne,hahn:Germany, bari:Italy, LowCostCompany} \sqsubseteq \textsf{Company} \}$

Now, S_r, S_p^l and S_p^k are considered. Note that S_p^l satisfies only HC *of S_r while S_p^k satisfies both* HC *and* SC *of S_r. It is supposed that the extensions of S_p^l and S_p^k are: $|(S_p^l)^{\mathcal{I}}| = 8$; $|(S_p^k)^{\mathcal{I}}| = 5$ and that all instances satisfy S_r. Note that $S_p^k \sqsubseteq S_p^l \Rightarrow (S_p^k)^{\mathcal{I}} \subseteq (S_p^l)^{\mathcal{I}}$. Consequently, $|(S_r)^{\mathcal{I}}| = 8$. Furthermore, $S_r \not\equiv S_p^l$ and $S_r \not\equiv S_p^k$. It is considered S_r^{HC} and S_r^{SC}. Known that all the instances of S_p^l and S_p^k satisfy S_r and particularly, S_p^l satisfies only* HC *of S_r; S_p^k satisfies both* HC *and* SC *of S_r, it is straightforward to see that $|(S_r^{HC} \sqcap S_p^l)^I| = 8$ and that $|(S_r^{new} \sqcap S_p^l)^I| = 0$. In the other case, it is known that $|(S_p^k)^{\mathcal{I}}| = 5$ and that S_p^k satisfies both* HC *and* SC *of S_r. So, some instances of S_p^k can satisfy only* HC *of S_r and others satisfy both* HC *and* SC *(in the better case all instances of S_p^k satisfy both* HC *and* SC*). It is supposed that instances of S_p^k that satisfy both* HC *and* SC *of S_r, namely that satisfy $S_r^{new} \equiv S_r^{HC} \sqcap S_r^{SC}$ are 3. Hence, applying the procedure it has:*

$$\bar{s_l} := s(S_r^{HC}, S_p^l) = \frac{|(S_r^{HC} \sqcap S_p^l)^{\mathcal{I}}|}{|(S_r^{HC} \sqcup S_p^l)^{\mathcal{I}}|} \cdot \max(\frac{|(S_r^{HC} \sqcap S_p^l)^{\mathcal{I}}|}{|(S_r^{HC})^{\mathcal{I}}|}, \frac{|(S_r^{HC} \sqcap S_p^l)^{\mathcal{I}}|}{|(S_p^l)^{\mathcal{I}}|}) = \frac{8}{8} \cdot \max(\frac{8}{8}, \frac{8}{8}) = 1$$
$$\bar{s_k} := s(S_r^{HC}, S_p^k) = \frac{|(S_r \sqcap S_p^k)^{\mathcal{I}}|}{|(S_r \sqcup S_p^k)^{\mathcal{I}}|} \cdot \max(\frac{|(S_r \sqcap S_p^k)^{\mathcal{I}}|}{|S_r^{\mathcal{I}}|}, \frac{|(S_r \sqcap S_p^k)^{\mathcal{I}}|}{|(S_p^k)^{\mathcal{I}}|}) = \frac{5}{8} \cdot \max(\frac{5}{8}, \frac{5}{5}) = 0.625$$

The next step is computing $\overline{\overline{s_l}}$ and $\overline{\overline{s_k}}$. Considered the observation above it has:

$$\overline{\overline{s_l}} = 0; \qquad \overline{\overline{s_k}} := \frac{|(S_r^{new} \sqcap S_p^k)^{\mathcal{I}}|}{|(S_r^{new} \sqcup S_p^k)^{\mathcal{I}}|} \cdot \max(\frac{|(S_r^{SC} \sqcap S_p^k)^{\mathcal{I}}|}{|(S_r^{new})^{\mathcal{I}}|}, \frac{|(S_r^{new} \sqcap S_p^k)^{\mathcal{I}}|}{|(S_p^k)^{\mathcal{I}}|}) = \frac{3}{5} \cdot \max(\frac{3}{3}, \frac{3}{5}) = 0.6$$

Hence, the final similarity values are: $s_l = 0.5$ and $s_k = 0.8125$, and the ranking of

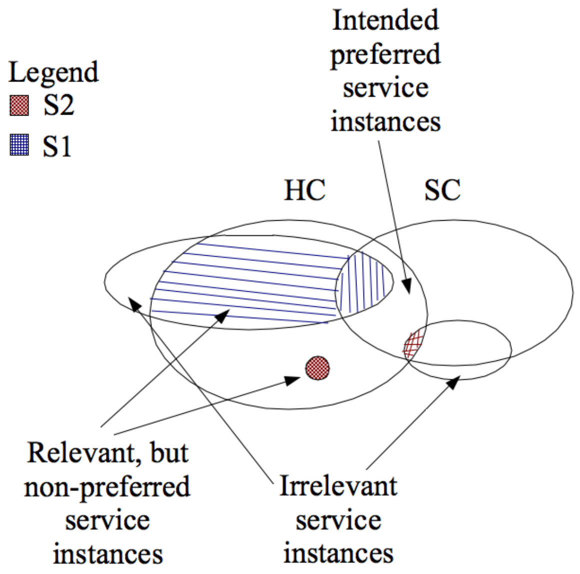
Legend
S2
S1
Intended preferred service instances
HC
SC
Relevant, but non-preferred service instances
Irrelevant service instances

Figure 5.6: Common instances between requested service and provided services for their ranking

the provided services is:

1.	S_p^k	Similarity Value	0.6125
2.	S_p^l	Similarity Value	0.5

This result is consistent with the goal. Namely, using this procedure, provided services are ranked w.r.t. both variance and satisfaction of S_r's SC. □

The rational of the ranking procedure is showed in Fig. 5.6. Since the considered services have been selected by the matching process, all of them have at least one instance satisfying S_r. In the figure, *HC* and *SC* represent the Hard and Soft Constraints of the request and S_1 and S_2 represent services to rank. All the instances of S_1 or S_2 that are in *HC* are relevant instance services for S_r, because they satisfy its *HC*. However they are not the preferred instance services for S_r because they do not satisfy also *SC*. For example, if the *HC* of S_r ask for flights from Cologne to Bari and the *SC* of S_r ask for flights that allow the use of Miles and More card then all the instances of S_1 and S_2 that are in *HC* are all flights from Cologne to Bari operated by two different company. This instances are relevant because they satisfy the main need, however flights from Cologne to Bari that allow the use of Miles and More card will be preferred w.r.t. flights that do not allow the use of this card. Thus the preferred instance services for S_r are all the instances of S_1 and S_2 that are in the intersection between *HC* and *SC*. These instances are all the flights from Cologne to Bari of S_1 and S_2 that allow for the use of Miles and More Card. The parts of S_1 and S_2 outside *HC* represent all the instances that do not satisfy *HC* and thus irrelevant service instances for S_r; for example flights having a departure and/or arrival city different from those requested. In the same way the part of S_2 outside *HC* but in *SC* represents irrelevant service instances for S_r because these instances satisfy *SC* without satisfying *HC*; for example represents flights that allow the use of Miles and More card but that do not arrive at Bari.

Initially, the procedure ranks provided services that satisfy *HC* w.r.t. the variance criterion, indeed provided services that share most of instances with S_r have higher similarity value. Hence *SC* are considered. The procedure assigns an additional similarity value to provided services satisfying also *SC*. This similarity value is assigned, again using the variance criterion. Note that in computing the additional similarity value are not considered all the service instances satisfying *SC* of S_r but only the service instances satisfying both *HC* and *SC* of S_r. This avoid to have in higher ranking position services that are very similar to *SC* but dissimilar from *HC*, whose instances are obviously not preferred w.r.t. services mostly similar to *HC*. Indeed the latter can have a lot of instances satisfying *SC* but that are not relevant at all for the main request. The realized ranking procedure constitutes a useful result for improving the discovery and negotiation processes. In the following, its computational complexity is analyzed.

The dominant operation of the ranking procedure is the computation of the similarity values, for which the semantic similarity measure (s) is invoked twice for every matched provided service. Hence, $Compl(\mathsf{Rank}) = n \cdot 2 \cdot Compl(s)$ where n is the number of services to rank. The complexity of s (see Sect. 4.1.2) mainly depends from the complexity of the *instance checking* operator (for the considered DL), used for computing the extensions of the service descriptions and the extension of their conjunction and disjunction. Substituting the complexity of the measure it has: $Compl(\mathsf{Rank}) = n \cdot 2 \cdot [4 \cdot Compl(IChK)]$. Note that the complexity of the ranking procedure could be decreased by reducing the number of calls to the instance checking operator, since the extensions of all available services can be computed beforehand. Consequently, at request-time, only the extension of the requested service description has to be computed. The extensions of the conjunction and disjunction of descriptions can be computed by means of set theory applied to the extensions already determined.

The realized ranking procedure turns out to be useful for supplying to the requester the most appropriate provided service. To this purpose, the procedure takes into account the presence of *HC* and *SC* and the *variance*, by exploiting a measure that assesses the similarity between service descriptions and allows to yield a total order among the selected services.

Chapter 6

Conclusions

This thesis reached two main objectives: the definition of new similarity and dissimilarity measures applicable to DLs Knowledge Bases and able to assesses the similarity value between concepts, individuals and between concept and individual asserted in an ontology; the application of inductive instance based learning methods in the Semantic Web and Semantic Web Services domains.

Both aspects constitute a novelty in the literature. Indeed, very few measures exist for DLs and particularly for the most expressive ones. As regard reasoning, the deductive approach is typically used in the context of the Semantic Web and Semantic Web Services. However, deduction-based inference services sometimes require high computational complexity, hence new form of reasoning are necessary. Inductive reasoning can be helpful in this way, but unfortunately it has not been deeply investigated. This thesis analyzed the applicability of inductive learning methods to ontological representation. Specifically, instance-based learning methods have been set up to improve various tasks such as: concept retrieval (to semi-automatize the ontology population) and service discovery process.

6.1 Summary of the Thesis

This thesis defined a novel approach for semantic similarity assessment between concepts, individuals and concept and individual asserted in an ontology. This approach is mainly based on semantics, sometimes used jointly with structural concept information. A set of similarity and dissimilarity measures have been defined. They are characterized by the use of a numeric approach applied to symbolic representations. Specifically, by recurring to the instance check inference operator, the measures exploit concept extensions (that constitute the semantics of a concept) to compute commonalities in terms of set theory (numerical approach).

A totally semantic similarity measure w.r.t. $\mathcal{ALC}$ KBs was defined (see Sect. 4.1). Anyway, it has been proved that it is mainly language independent, so it can be applied to KBs described by most expressive DLs. This is because the only requirement of the measure is the availability of the instance check operator for the chosen DL (and the computation of the Most Specific Concept in case of similarity assessment between individuals). Experimental evaluations proved that the measure is suitable to compute similarity between concepts while it is less able to compute similarity between individuals, due to the high specificity of the mscs.

This weakness has been solved by other two dissimilarity measures applicable to $\mathcal{ALC}$ KBs (see Sect. 4.2 and Sect. 4.3). One based on the overlap of the concepts and the other one based on the variation of the Information Content between the descriptions. The weakness has been overcome by exploiting also the concept structures, besides of their semantics. Specifically, the dissimilarity value is computed considering also the dissimilarity of subconcepts that built a concept description. This is particularly useful when mcss are considered, as they are characterized by many nested subconcepts. In order to avoid that deeply nested subconcepts modify the "real" dissimilarity value, a weighted (w.r.t. concept levels) version of both measures has been formalized. These measures resulted suitable for assessing dissimilarity value between concepts and individuals. Nevertheless, as they are also structure-driven, they cannot be used for more expressive DLs than $\mathcal{ALC}$.

Following the same approach presented above, a similarity measure for $\mathcal{ALN}$ logic has been defined (see Sect. 4.4.1). It allows to treat numerical restrictions that are often present in real-world application problems. As for the previous measures, such a measure cannot be used for more expressive DLs than $\mathcal{ALN}$.

A kernel function for $\mathcal{ALC}$ concept descriptions has been defined (see Sect. 4.5), with the goal of exploiting the efficiency of kernel methods. Based on the notion of convolution kernel, it allows to compute the dissimilarity value between concepts on the ground of a semantic and structure-driven approach. It constitutes a very interesting result as it is (one of) the first kernel functions defined for an expressive DL. Experimental evaluation of the function demonstrated that it can be effectively used in order to assess similarity between concepts, individuals, and concept and individual. Anyway it cannot be applied to more expressive DLs than $\mathcal{ALC}$.

In order to overcome the limitation of the previous measures, that is their dependance from a specific representation language, a totally semantic semi-distance measure has been defined (see Sect. 4.6). Based on the principles of the Hypothesis-driven distance, the measure is able to assess similarity between individuals asserted in an ontology without recurring to the computation of their mscs. Experimental evaluations proved that such a measure outperforms the previous ones, anyway, at the moment it cannot be used to determine the similarity value between concepts asserted in an ontology.

This thesis focused also an the application of inductive learning methods to the Semantic Web and Semantic Web Services contexts. This is for a double reason: to evaluate the defined measures in an objective way (the alternative could be to recur to a subjective human judgement); to show the effective applicability of inductive learning methods in these contexts, where deductive approach is generally employed. In order to reach these goals, different instance-based learning methods has been developed.

The first efforts involved the definition of an instance-based classifier that is applicable in the Semantic Web context. Specifically, a modified version of the k-Nearest Neighbor algorithm for DLs KBs has been developed (see Sect. 5.1.1). It classifies individuals of an ontology w.r.t. the concepts defined therein; moreover it is able to perform concept retrieval of a new query concept defined, on the fly, by means of concepts and roles in the reference ontology. The application of such algorithm to the SW context needed to solve two non trivial issues: 1) classes (concepts) w.r.t. the classification is performed are not disjoint; 2) the Open World Assumption, characterizing the SW context, has to be treated. The realized classifier has been applied, jointly with most of the defined measures, to several (online available) ontologies, both for classifying individuals and for solving query answering problems. The experimental evaluations proved that the classification results are comparable to those returned by a standard deductive-based reasoner; moreover the classifier is also able to induce new knowledge, not logically derivable. Its performance are less reliable when ontologies are not homogeneously populated. However this is a well known drawback of the K-NN algorithm. The realized classifier represents a form of uncertain reasoning. It can be effectively used to semi-automatize the ontology population task (nowadays manually made) and to improve the retrieval inference service.

With the goal of exploiting the well known efficiency of kernel methods and evaluate the defined $\mathcal{ALC}$ kernel function, a classifier has been realized by the use of a SVM. As for the previous case, the problems of non-disjointeness of the classes and the Open World Assumption have been solved. The system classifies individuals of an ontology w.r.t. the concepts therein, as well as it is able to return the extension of new concepts built on the ground of the reference ontology. Experimental evaluation showed that it is comparable with a deductive reasoner and moreover it is able to induce new assertions not logically derivable. It is less reliable in case in which many concepts have very few assertions.

As regards Semantic Web Services, different aspects have been treated. Firstly, modeling service descriptions has been analyzed. On the ground of a set of guidelines suggested by Grimm et al. [93], a DL-based framework for describing services has been formalized. The main peculiarity of the framework is the possibility to describe a service by distinguishing between Hard Constraints and Soft Constraints. This is particularly important when a service request has to be described. Indeed, by recurring to the distinction between HC and SC, it is possible to distinguish between the constraints of the request that have to be necessarily satisfied (HC), and constraints of the request that have to be preferable satisfied, but not necessarily (SC).

Considered the service descriptions, the service discovery process can be performed by means of a matching procedure grounded on standard and non-standard DLs inferences. The use of a semantic matching, rather than a syntactic one, ensures that the discovered services better satisfy the request. Anyway, in both cases, the matching is generally performed by checking if every available service is able to satisfy the request. However, with the increasing amount of available services such an approach could not be able to guarantee replies in a reasonable time. In order to improve the efficiency of the service discovery process, various hierarchical agglomerative clustering algorithms have been developed. Indeed, the set of all available services can be firstly grouped into homogeneous clusters, represented by means of a dendrogram (as the output of a hierarchical agglomerative clustering algorithm) whose leaves are the available service descriptions. Every cluster can be then intensionally described. Hence, the service discovery, for a given service request, can be performed by matching the request to intensional cluster descriptions rather than to each available service, thus heavily reducing the search space. Particularly, in the best case the complexity of the discovery process can decrease from linear to logarithmic in the number of the available service descriptions. The experimental evaluations of the clustering algorithms showed their validity; the obtained clusters have been measured by the use of the *overall clusters similarity*.

Moreover, a procedure for returning matched service in a ranked list (with respect to a fixed criterion) has also been developed. By exploiting the notion of constraint hardness of a service request and the semantic similarity measures, the procedure ranks in the highest positions provided services that are able to satisfy both HC and SC and that are more similar to the request, while provided services satisfying only HC and/or that are less similar to the service request are ranked in the lowest positions. The availability of such ranking process allows to improve both discovery and further negotiation process as increases the probability of finding the really required service descriptions.

Summarizing, this thesis defined a new approach for assessing similarity among elements of an ontology, proved the validity of such measures and showed that the application of inductive learning methods to the Semantic Web and Semantic Web Services domains can effectively solve many different open problems in such contexts.

6.2 Further Work

Considering the novelty of the arguments treated in this thesis, the work done constitutes only the starting point of a more wide research line. Indeed many improvements and open points need to be solved.

Specifically, measures for more expressive DLs (such as $\mathcal{ALCN}$), exploiting a semantic and structure driven approach, need to be defined. Also kernel functions for most expressive DLs could be useful, as they allow to treat inductive learning problems in a efficient way. The availability of measures applicable to more expressive DLs allows to cope with a wide range real life problems. Particularly, an open issue still remain the treatment of the role. Indeed, they currently are only implicitly considered, as part of a concept descriptions. Treating them in an explicit way could improve the quality of the assessed (dis-)similarity values.

Furthermore, the experimental evaluation of the defined semi-distance measure showed that it gives the most appropriate dissimilarity values w.r.t. the other developed measures. Anyway, it can be applied only to assess dissimilarity between individuals. A formalization of this measure that is able to cope also with concept descriptions could be very interesting, as it is totally semantic and language independent, and so it could be applied to DL KBs. Moreover, further experimentations of such measure showed that not all concepts of the knowledge base are necessary for building the set of hypotheses fundamental to compute a dissimilarity value. This suggest a line of further investigation that will concern finding minimal subsets of concepts to be used for the measure. This means reducing the number of considered concepts, saving those that are endowed of a real discriminating power. This could be also done by learning optimal sets of discriminating features, allowing also their composition on the ground of the specific constructors of the chosen representation language. Both these objectives can be accomplished by means of machine learning techniques.

Another important aspect to solve concerns the applicability of the measures to concepts and individuals asserted in different ontologies. Indeed, the availability of such measures could constitute a precious tool for tasks such as ontology matching and alignment, that are key issues for making systems and KBs really interoperable.

The developed inductive learning methods could be also improved w.r.t. different aspects. Specifically, the realized classifier, based on a modified version of the k-NN algorithm, could be extended with different (yet still computationally tractable) answering procedures grounded on statistical inference (non-parametric tests based on ranked distances) in order to accept answers as correct with a high degree of confidence. Moreover, it could be extended in a way such that the probability that an individual belongs to one or more concepts are given. Furthermore, the k-NN method, in its classical form, is particularly suitable for the automated induction of missing values for (scalar or numeric) datatype properties of an individual, as an estimate derived from the values of the datatypes for the surrounding individuals.

As regards the topic of the service discovery process, many aspects need to be investigated. First of all, an experimental evaluation of the proposed matching procedure exploiting clustering methods needs to be performed. This is also in order to check the opportunity of the usage of intensional cluster descriptions based on $\mathcal{ALE}$ logic rather than $\mathcal{ALC}$.

Moreover, the presented matching process is characterized by a set of open problems not treated in this thesis. Firstly, an implicit assumption has been made: a single branch of the tree contains the requested services. Anyway, really, it can happen that more that one branch satisfies the matching condition. In this case two different possibilities can be taken into account. One is to consider a search in the tree that allows backtracking. In this way, all the services satisfying the request will be found. The other solution consist in a search strategy driven by a heuristic, used for the choice of the branch to explore. A possible heuristic can be given by the similarity between the intensional node description and the request. Similarity can be measured by means of one of the defined measures for DLs.

Furthermore, an incremental clustering process could be considered. Specifically, after all available services have been clustered, the matching process is applied, and a set of services satisfying the request is found. All the clusters used for finding the required services could be updated by adding the description of the request and updating the intensional cluster descriptions; this is in order to better satisfy the next requests. Obviously, after a certain number of update, the clusters have to be re-computed.

Moreover, a new matching process could be useful for further increase the quality of the discovery process and reduce the noise in the selection of services.

The definition of semantic (dis-)similarity measures for DLs and the application of inductive learning methods to the Semantic Web domain constitute a very young line of research. I hope that this thesis will be only the starting point of a more wide research area in the future.

Bibliography

[1] David W. Aha. Lazy learning. *Artificial Intelligence Review*, 11:7–10, 1997.

[2] H. Akkermans, Z. Baida, and J. Gordijn. A shared service terminology for online service provisioning. In *Proceedings of the Sixth Intern. Conf. on Electronic Commerce (ICEC04)*, pages 1–10, New York, NY, USA, 2004. ACM Press.

[3] T. Andrews and et al. Business process execution language for web services (version 1.1), May 2003. .

[4] A. Appice, C. d'Amato, F. Esposito, and D. Malerba. Classification of symbolic objects: A lazy learning approach. *Journal of Intelligent Data Analysis*, 10:301–324, 2006.

[5] A. Arkin and et al. Web service choreography interface (wsci) version 1.0, August 2002. http://www.w3.org/TR/wsci/.

[6] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society.*, 68, 1950.

[7] K.D. Ashley. *Modeling legal argument: Reasoning with cases and hypotheticals.* MIT Press, 1990.

[8] F Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.

[9] F. Baader and R. Küsters. Computing the least common subsumer and the most specific concept in the presence of cyclic $\mathcal{ALN}$ concept descriptions. In O. Herzog and A. Güenter, editors, *Proceedings of the 22th Annual German Conference on Artificial Intelligence*, volume 1504 of *LNAI*, pages 129–140. Springer, 1998.

[10] F. Baader and R. Küsters. Matching in description logics with existential restrictions. In *Proceedings of Description Logic Workshop (DL'99)*, 1999.

[11] F. Baader and R. Küsters. Non-standard inferences in description logics: The story so far. In D. M. Gabbay, S. S. Goncharov, and M. Zakharyaschev, editors, *Mathematical Problems from Applied Logic I. Logics for the XXIst Century*, volume 4 of *International Mathematical Series*, pages 1–75. Springer-Verlag, 2006.

[12] F. Baader, R. Küsters, and R. Molitor. Computing least common subsumers in description logics with existential restrictions. In T. Dean, editor, *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 96–101. Morgan Kaufmann, 1999.

[13] F. Baader, R. Küsters, and R. Molitor. Rewriting concepts using terminologies. In *KR*, pages 297–308, 2000.

[14] F. Baader and R. Narendran. Unification of concept terms in description logics. In H. Prade, editor, *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI98)*, pages 331–335. John Wiley & Sons, 1998.

[15] F. Baader and R. Narendran. Unification of concepts terms in description logics. *Journal of Symbolic Computation*, 31(3):277–305, 2001.

[16] F. Baader, R. Sertkaya, and Y. Turhan. Computing least common subsumers w.r.t. a background terminology. In V. Haarslev and R. Möller, editors, *Proceedings of Proceedings of the 2004 International Workshop on Description Logics (DL2004)*. CEUR-WS.org, 2004.

[17] F. Bacchus. Lp, a logic for representing and reasoning with statistical knowledge. *Computational Intelligence*, 6:209–231, 1990.

[18] R.A. Baeza-Yates. Introduction to data structures and algorithms related to information retrieval. *Information Retrieval: Data Structures and Algorithms*, pages 13–27, 1992.

[19] G. Baudat and F. Anouar. Generalized discriminant analysis using a kernel approach. *Neural Computing*, 12(10):2385–2404, 2000.

[20] A. Ben-Hur, D. Horn, H. T. Siegelmann, and V. Vapnik. Support vector clustering. *Journal of Machine Learning Research*, 2:125–137, 2001.

[21] C. Berg, J. Christensen, and P. Ressel. *Harmonic Analysis on Semigroups: Theory of Positive Definite and Related Functions.* Springer, 1984.

[22] T. Berners-Lee. Semantic Web talk at xml 2000, 2000. http://www.dajobe.org/talks/sw-vienna/slide-10.html.

[23] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 2001.

[24] Y. Bishr. Semantic aspects of interoperable GIS. Technical report, Wageningen Agricultural University and ITC - Netherland, 1997.

[25] G. Biswas, J.B. Weinberg, and D.H. Fisher. Iterate: A conceptual clustering algorithm for data mining. *IEEE Transaction of System, Man and Cybernetics (Part C: Applications and Reviews).*, 28:100–111, 1998.

[26] H.H. Bock and E. Diday. *Analysis of symbolic data : exploratory methods for extracting statistical information from complex data.* Springer-Verlag, 2000.

[27] A. Borgida and R. Küsters. Whats not in a name: Some properties of a purely structural approach to integrating large dl knowledge bases. In *Proceedings of the 2000 Description Logic Workshop (DL 2000)*, volume 33, pages 65–78. CEUR Electronic Workshop Proceedings, http://ceur-ws.org/, 2000.

[28] A. Borgida and D.L. McGuinness. Asking queries about frames. In *Proceedings of the 5th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR96)*, pages 340–349, 1996.

[29] A. Borgida, T. Walsh, and H. Hirsh. Towards measuring similarity in description logics. In I. Horrocks, U. Sattler, and F. Wolter, editors, *Proceedings of the 2005 International Workshop on Description Logics (DL2005)*, volume 147 of *CEUR Workshop Proceedings.* CEUR-WS.org, 2005.

[30] Alexander Borgida and David W. Etherington. Hierarchical knowledge bases and e?cient disjunctive reasoning. In Ron J. Brachman, Hector J.Levesque, and Ray Reiter, editors, *Proceedings of the 1st Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR89)*, pages 33–43, 1989.

[31] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *In Proceeding of the 5th Annual ACM Workshop on Computing Learning Theory*, pages 144–152, 1992.

[32] R.J. Brachman. A structural paradigm for representing knowledge, 1977. PhD thesis.

[33] R.J. Brachman. Whats in a concept: Structural foundations for semantic networks. *Int. Journal of Man-Machine Studies*, 9(2):127–152, 1977.

[34] R.J. Brachman, editor. *On the epistemological status of semantic networks.* Academic Press, New York, 1979.

[35] R.J. Brachman and J. G Schmolze. An overview of the kl-one knowledge representation systems. *Int. Journal of Cognitive Science*, 9(2):171–216, 1985.

[36] S. Brandt, R. Küsters, and A.-Y. Turhan. Approximation and difference in description logics. In D. Fensel, F. Giunchiglia, D. McGuinness, and M.-A. Williams, editors, *Proceedings of the 8th International Conference on Principles of Knowledge Representation and Reasoning, KR02*, pages 203–214. Morgan Kaufmann, 2002.

[37] M. W. Bright, A. R. Hurson, and Simin H. Pakzad. Automated resolution of semantic heterogeneity in multidatabases. *ACM Transaction on Database Systems*, 19(2):212–253, 1994.

[38] L Cabral, J Domingue, E Motta, T.R. Payne, and F Hakimpour. Approaches to semantic web services: an overview and comparisons. In *Proceedings of First European Semantic Web Symposium (ESWS)*, volume 3053 of *LNCS*, pages 225–239. Springer, 2004.

[39] Andrea Calì, Diego Calvanese, Simona Colucci, Tommaso Di Noia, and Francesco M. Donini. A description logic based approach for matching user profiles. In *Description Logics*, 2004.

[40] J. D. Carroll and M. Wish. Models and methods for three-way multidimensional scaling. *Contemporary developments in mathematical psychology*, 2:57–105, 1974.

[41] OWL-S Coalition. Owl-s 1.1 release, 2004. http://www.daml.org/services/owl-s/1.1/.

[42] William W. Cohen, Alexander Borgida, and Haym Hirsh. Computing least common subsumers in description logics. In *Proceeding of AAAI*, pages 754–760, 1992.

[43] W.W. Cohen and H. Hirsh. Learning the CLASSIC description logic: Theoretical and experimental results. In P. Torasso, J. Doyle, and E. Sandewall, editors, *Proceedings of the 4th International Conference on the Principles of Knowledge Representation and Reasoning*, pages 121–133. Morgan Kaufmann, 1994.

[44] Christine Collet, Michael N. Huhns, and Wei-Min Shen. Resource integration using a large knowledge base in carnot. *IEEE Computer*, 24(12):55–62, 1991.

[45] A. Collins and M. Quillian. Retrieval time from semantic memory. *Journal of Verbal Learning and Verbal Behavior.*, 8:240–247, 1969.

[46] M. Collins and N. Duffy. Convolution kernels for natural language. In T. G. Dietterich, S. Becker, and Z. Ghahramani (Eds.), editors, *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2002.

[47] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13:21–27, 1967.

[48] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines (and other kernel-based learning methods).* Cambridge University Press, 2000.

[49] C. d'Amato and N. Fanizzi. Lazy learning from terminological knowledge bases. In In F. Esposito, Z. W. Raś, D. Malerba, and G. Semeraro (Eds.), editors, *Proceedings of the 16th International Symposium on Methodologies for Intelligent Systems, ISMIS2006*, volume 4203 of *Lecture Notes in Computer Science*, pages 570–579, Bari, Italy, 2006. Springer.

[50] C. d'Amato, N. Fanizzi, and F. Esposito. A dissimilarity measure for concept descriptions in expressive ontology languages. In H. Alani, C. Brewster, N. Noy, and D. Sleeman, editors, *Proceedings of the KCAP2005 Ontology Management Workshop*, Banff, Canada, 2005.

[51] C. d'Amato, N. Fanizzi, and F. Esposito. A dissimilarity of $\mathcal{ALC}$ concept descriptions. In *SWAP 2005, the 2nd Italian Semantic Web Workshop*, Trento, Italy, 2005. CEUR. online[1].

[52] C. d'Amato, N. Fanizzi, and F. Esposito. A semantic dissimilarity measure for concept descriptions in ontological knowledge bases. In M. Ackermann, B. Berendt, M. Grobelnik, and V. Svtek (Eds.), editors, *Proceedings of the second International Workshop on Knowledge Discovery and Ontologies (at ECML/PKDD 2005)*, Porto, Portugal, 2005.

[53] C. d'Amato, N. Fanizzi, and F. Esposito. A semantic similarity measure for expressive description logics. In A. Pettorossi, editor, *Proceedings of Convegno Italiano di Logica Computazionale, CILC05*, Rome, Italy, 2005. electronic edition available[2].

[54] C. d'Amato, N. Fanizzi, and F. Esposito. Analogical reasoning in description logics. In *Proceedings of the Second ISWC Workshop on Uncertainty Reasoning for the Semantic Web.*, Athens, Georgia (USA), 2006. CEUR. online[3].

[1] `http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS//Vol-166/19.pdf`

[2] `http://www.disp.uniroma2.it/CILC2005/downloads/papers/15.dAmato_CILC05.pdf`

[3] `http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS//Vol-218/paper4.pdf`

[55] C. d'Amato, N. Fanizzi, and F. Esposito. A dissimilarity measure for $\mathcal{ALC}$ concept descriptions. In *Proceedings of the 21st Annual ACM Symposium of Applied Computing, SAC2006*, Dijon, France, 2006. ACM Press.

[56] C. d'Amato, N. Fanizzi, and F. Esposito. Reasoning by analogy in description logics through instance-based learning. In *Proceedings of SWAP 2006, the 3rd Italian Semantic Web Workshop*, Pisa, Italy, 2006. CEUR. online[4].

[57] C. d'Amato and S. Staab. Modelling, matching and ranking services based on constraint hardness. In In J. Eder and S. Dustdar (Eds.), editors, *Proceeding of Advances in Semantics for Web services Workshop (at BPM06).*, volume 4103 of *Lecture Notes in Computer Science*, pages 471–482, Vienna, Austria, 2006. Springer.

[58] M. d'Aquin, J. Lieber, and A. Napoli. Decentralized case-based reasoning for the Semantic Web. In Y. Gil, V. Motta, E. Benjamins, and M. A. Musen, editors, *Proceedings of the 4th International Semantic Web Conference, ISWC2005*, number 3279 in LNCS, pages 142–155. Springer, 2005.

[59] B. Dasarathy. *Nearest Neighbor(NN) Norms: NN Pattern Classi*
cation Techniques. IEEE Computer Society Press, 1991.

[60] S. de Battle and et al. Semantic web service ontology (swso), September 2005. http://www.w3.org/Submission/SWSF-SWSO/.

[61] j. de Bruijn and et al. Web service modeling ontology (wsmo), June 2005. http://www.w3.org/Submission/WSMO/.

[62] K. Dellschaft and S. Staab. On how to perform a gold standard based evaluation of ontology learning. In I. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, and L. Aroyo, editors, *Proceedings of the 5th International Semantic Web Conference (ISWC-2006)*, volume 4273 of *Lecture Notes in Computer Science*. Springer, 2006.

[63] E. Diday and J.C. Simon. Clustering analysis. *Digital Pattern Recognition*, pages 47–94, 1976.

[64] Chris H. Q. Ding and Xiaofeng He. Cluster aggregate inequality and multilevel hierarchical clustering. In Alípio Jorge, Luís Torgo, Pavel Brazdil, Rui Camacho, and João Gama, editors, *Proceeding of the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases, PKDD*, volume 3721 of *Lecture Notes in Computer Science*, pages 71–83. Springer, 2005.

[4]`http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS//Vol-201/25.pdf`

[65] F. M. Donini, B. Hollunder, M. Lenzerini, A. Marchetti Spaccamela, D. Nardi, and W. Nutt. The complexity of existential quantification in concept languages. *Int. Journal of Artificial Intelligence*, 2(3):309–327, 1992.

[66] F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. *Information and Computation*, 134(1):1–58, 1997.

[67] F.M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. An epistemic operator for description logics. *Artificial Intelligence*, 100(1-2):225–274, 1998.

[68] F.M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Deduction in concept languages: From subsumption to instance checking. *Int. Journal of Logic and Computation*, 4(4):423–452, 1994.

[69] C. Dorai and A.K. Jain. Shape spectra based view grouping for free-form objects. In *In Proceeding of the International Conference on Image Processing (ICIP-95)*, pages 240–243, 1995.

[70] R. Duda and P. Hart. *Pattern classification and scene analysis.* New York: John Wiley & Sons., 1973.

[71] M. Egenhofer and D. Mark. Spatial information theory - a theoretical basis for geographic information systems. In A. Frank and W. Kuhn, editors, *Proceedings of the International Conference COSIT'95*, pages 1–14. Springer-Verlag, 1995.

[72] W. Emde and D. Wettschereck. Relational instance-based learning. In *I proceeding of the 13th International Conference on Machine Learning (ICML)*, pages 122–130, 1996.

[73] F. Esposito, N. Fanizzi, L. Iannone, I. Palmisano, and G. Semeraro. Knowledge-intensive induction of terminologies from metadata. In In F. van Harmelen, S. McIlraith, and D. Plexousakis Eds., editors, *In Proceeding of the the 3rd International Semantic Web Conference*, volume 3298 of *LNCS*, pages 441–455. Springer, 2004.

[74] F. Esposito, D. Malerba, and G. Semeraro. Classification in noisy environment using a distance measure between structural symbolic descriptions. *IEEE Transaction on Pattern Analysis and Machine Intelligence.*, 14(3):390–402, 1992.

[75] N. Fanizzi and C. d'Amato. A declarative kernel for $\mathcal{ALC}$ concept descriptions. In F. Esposito, Z. W. Raś, D. Malerba, and G. Semeraro (Eds.), editors,

In Proceedings of the 16th International Symposium on Methodologies for Intelligent Systems., volume 4203 of *Lecture Notes in Computer Science*, pages 322–331. Springer, 2006.

[76] N. Fanizzi and C. d'Amato. A similarity measure for the aln description logic. In *Proceedings of Convegno Italiano di Logica Computazionale, CILC05*, Bari, Italy, 2006. online[5].

[77] D.H. Fisher, H. Douglas, and M.J. Pazzani. *Concept Formation: Knowledge and Experience in Unsupervised Learning*. Morgan Kaufmann, 1991.

[78] T. Gärtner. Exponential and geometric kernels for graphs. In *In NIPS Workshop on Unreal Data: Principles of Modeling Nonvectorial Data*, 2002.

[79] T. Gärtner. A survey of kernels for structured data. *SIGKDD Explorations*, 5(1):49–58, 2003.

[80] T. Gartner, K. Driessens, and J. Ramon. Graph kernels and gaussian processes for relational reinforcement learning. In *In Proceedings of the 13th International Conference on Inductive Logic Programming*, 2003.

[81] T. Gärtner, P.A. Flach, and S. Wrobel. On graph kernels: Hardness results and efficient alternatives. In *Proceedings of the 16th Annual Conference on Computational Learning Theory and the 7th Kernel Workshop*, 2003.

[82] T. Gärtner, J.W. Lloyd, and P.A. Flach. Kernels for structured data. In S. Matwin and C. Sammut, editors, *Proceedings of 12th International Conference on Inductive Logic Programming, ILP2002*, volume 2583 of *LNCS*. Springer, 2002.

[83] T. Gärtner, J.W. Lloyd, and P.A. Flach. Kernels and distances for structured data. *Machine Learning*, 57(3):205–232, 2004.

[84] D. Gentner. Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, 7:155–170, 1983.

[85] R. L. Goldstone. Similarity, interactive activation, and mapping. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 20:3–28, 1994.

[86] R. L. Goldstone. Similarity, 2005. In K. Holyoak and R. Morrison (Eds.). Cambridge Handbook of Thinking and Reasoning. Cambridge: Cambridge University Press: p. 13-36.

[5]`http://cilc2006.di.uniba.it/download/camera/15_Fanizzi_CILC06.pdf`

[87] A. Gomez Perez and V.R. Benjamins. Overview of knowledge sharing and reuse components: Ontologies and problem-solving methods. In *Proceedings of Ontology and Problem-Solving Methods: Lesson learned and Future Trends, Workshop at IJCAI*, volume 18, pages 1.1–1.15. CEUR Pubblications, Amsterdam, 1999.

[88] J Gonzales-Castillo, D Trastour, and C. Bartolini. Description logics for matchmaking of services. In *Proceedings of the KI-2001 Workshop on Applications of Description Logics*, volume 44, 2001.

[89] G. Gòra and A. Wojna. Riona: A classifier combining rule induction and k-nn method with automated selection of optimal neighbourhood. In *Proceedings of the Thirteenth European Conference on Machine Learning, ECML 2002*, Lecture Notes in Artificial Intelligence, pages 111–123. Springer-Verlag, 2002.

[90] K. C. Gowda and E. Diday. Symbolic clustering using a new dissimilarity measure. *IEEE Transaction of System, Man and Cybernetics.*, 22:368–378, 1992.

[91] K.C. Gowda and G. Krishna. Agglomerative clustering using the concept of mutual nearest neighborhood. *Pattern Recognition.*, (10):105–112, 1977.

[92] K.C. Gowda and G. Krishna. Disaggregative clustering using the concept of mutual nearest neighborhood. *IEEE Transaction of System, Man and Cybernetics.*, SMC-8(12):888–894, 1978.

[93] S. Grimm, B. Motik, and C. Preist. Variance in e-business service discovery. In *Proceedings of the ISWC Workshop on Semantic Web Services*, 2004. http://www-106.ibm.com/developerworks/library/ws-bpel.

[94] T.R. Gruber. A translation approach to portable ontology specifications, 1993.

[95] N. Guarino, C. Masolo, and G. Verete. Ontoseek: Content-based access to the web. *IEEE Intelligent Systems*, 3(14):70–80, 1999.

[96] Volker Haarslev and Ralf Möller. RACER system description. In *Proceeding of Automated Reasoning, International Conference, IJCAR 2001*, volume 2083, pages 701–705, 2001.

[97] P. Haase, F. van Harmelen, Z. Huang, H. Stuckenschmidt, and Y. Sure. A framework for handling inconsistency in changing ontologies. In Y. Gil, V. Motta, E. Benjamins, and Mark A. Musen, editors, *Proceedings of the 4th International Semantic Web Conference, ISWC2005*, number 3279 in LNCS, pages 353–367, Galway, Ireland, November 2005. Springer.

[98] U. Hahn and K. Schnattinger. Toward text knowledge engineering. In *Proceeding of the 15th National Conference on Articial Intelligence (AAAI-98)*, 1998.

[99] D. Haussler. Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10, Department of Computer Science, University of California – Santa Cruz, 1999.

[100] P. Hitzler and D. Vrandecic. Resolution-based approximate reasoning for owl dl. In Y. Gil, E. Motta, V. R Benjamins, and M. A. Musen, editors, *In Proceeding of the 4th International Semantic Web Conference*, volume 3729 of *Lecture Notes in Computer Science*, pages 383–397. Springer, 2005.

[101] K. Holyoak and P. Thagard. Analogical mapping by constraint satisfaction. *Cognitive Science*, 13:295–355, 1989.

[102] I Horrocks, P. Patel-Schneider, and F. van Harmelen. From $\mathcal{SHIQ}$ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.

[103] I. Horrocks, P.F. Patel-Schneider, S. Bechhofer, and D. Tsarkov. Owl rules: A proposal and prototype implementation. *Web Semantics*, (3):23–40, 2005.

[104] Ian Horrocks. The fact system. In Harrie de Swart, editor, *Proceeding of Automated Reasoning with Analytic Tableaux and Related Methods, International Conference, TABLEAUX'98*, volume 1397 of *Lecture Notes in Computer Science*, pages 307–312. Springer, 1998.

[105] L. Iannone. Machine learning for ontology engineering, April 2006. PhD. Thesis.

[106] M. Ichino and H. Yaguchi. Generalized minkowski metrics for mixed feature-type data analysis. *System, Man and Cybernetics*, 24(4):698–708, 1994.

[107] A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data.* Prentice-Hall, 1988.

[108] A.K. Jain, M.N. Murty, and P.J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[109] J. Jang and D. Conrath. Semantic symilarity based on corpus statistic and lexical taxonomy. In *Proceedings of the International Conference on Computational Linguistics*, 1997.

[110] K. Janowicz. Sim-dl: Towards a semantic similarity measurement theory for the description logic alcnr in geographic information retrieval. In R. Meersman, Z. Tari, and P. Herrero et al., editors, *Proceedings of OTM Workshops at SeBGIS 2006*, volume 4278 of *Lecture Notes in Computer Science*, pages 1681 – 1692. Springer, 2006.

[111] R.A. Jarvis and E.A. Patrick. Clustering using a similarity method based on shared near neighbors. *Pattern Analysis and Machine Intelligence. PAMI*, C-22(11):1025–1034, 1973.

[112] T. Joachims. *Learning to Classify Text using Support Vector Machines.* Kluwer Academic Publishers, 2002.

[113] I. Jonyer, D.J. Cook, and L.B. Holder. Graph-based hierarchical conceptual clustering. *Journal of Machine Learning Research.*, 2:19–43, 2001.

[114] H. Kashima and A. Inokuchi. Kernels for graph classification. In *In ICDM Workshop on Active Mining*, 2002.

[115] H. Kashima and T. Koyanagi. Kernels for semistructured data. In C. Sammut and A. Hoffmann (Eds.), editors, *Proceedings of the 19th International Conference on Machine Learning.* Morgan Kaufmann, 2002.

[116] H. Kashima, K. Tsuda, and A. Inokuchi. Marginalized kernels between labeled graphs. In T. Fawcett and N. Mishra, editors, *Proceedings of International Conference on Machine Learning, ICML2003*, pages 321–328. AAAI Press, 2003.

[117] N. Kavantzas, D. Burdett, and G. Ritzinger. Web service choreography description language version 1.0, 27 April 2004. http://www.w3.org/TR/2004/WD-ws-cdl-10-20040427/.

[118] D. Kibler and D. Aha. Learning representative exemplar of concepts: An initial case study. In *Proceeding of the Fourth International Workshop on Machine Learning*, pages 24–30. Morgan Kaufmann, 1987.

[119] J.-U. Kietz. Learnability of description logic programs. In S. Matwin and C. Sammut, editors, *Proceedings of the 12th International Conference on Inductive Logic Programming*, volume 2583 of *LNAI*, pages 117–132, Sydney, 2002. Springer.

[120] B. King. Step-wise clustering procedures. *Journal of the American Statistical Association.*, 69:86–101, 1967.

[121] D. Koller, A. Levy, and A. Pfeffer. P-classic: a tractable probabilistic description logic. In *In Proceeding of AAAI 1997*, 1997.

[122] C. Krumhansl. Data: The interrelationship between similarity and spatial density. *Psycological Review*, 85(5):445–463, 1978.

[123] R. Küsters and R. Molitor. Approximating most specific concepts in description logics with existential restrictions. In F. Baader, G. Brewka, and T. Eiter, editors, *Proceedings of the Joint German/Austrian Conference on Artificial Intelligence, KI/ÖGAI01*, volume 2174 of *LNCS*, pages 33–47. Springer, 2001.

[124] R. Küsters and R. Molitor. Computing least common subsumers in $\mathcal{ALEN}$. In editor B. Nebel, editor, *In Proceeding of the International Joint Conference on Artificial Intel ligence, IJCAI2001*, pages 219–224, 2001.

[125] Ralf Küsters. *Non-Standard Inferences in Description Logics*, volume 2100 of *Lecture Notes in Computer Science*. Springer, 2001.

[126] J. Lee, M. Kim, and Y. Lee. Information retrieval based on conceptual distance in is-a hierarchies. *Journal of Documentation*, 2(49):188–207, 1993.

[127] C. Leslie, E. Eskin, and W. Noble. The spectrum kernel: A string kernel for svm protein classification. In *In Proceedings of the Pacific Symposium on Biocomputing*, pages 564–575, 2002.

[128] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady Akademii Nauk SSSR*, 10(8):707–710, 1966.

[129] L Li and I Horrocks. A software framework for matchmaking based on semantic web technology. In *Proceedings of the Twelfth World Wide Web Conference*, 2003.

[130] D. Lin. An information-theroretic defintion of similarity. In *Proceedings of the International Conference on Machine Learning*, pages 296–304. Morgan Kaufmann, 1998.

[131] J. W. Lloyd. *Logic for Learning.* Springer-Verlag, 2002.

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

[133] Alexander Maedche and Steffen Staab. Measuring similarity between ontologies. In Asunción Gómez-Pérez and V. Richard Benjamins, editors, *EKAW*, volume 2473 of *Lecture Notes in Computer Science*, pages 251–263. Springer, 2002.

[134] T. Mantay. Commonality-based ABox retrieval. Technical Report FBI-HH-M-291/2000, Department of Computer Science, University of Hamburg, Germany, 2000.

[135] A. B. Markman and D. Gentner. Splitting the differences: A structural alignment view of similarity. *Journal of Memory & Language*, 32:517–535, 1993.

[136] A. B. Markman and D. Gentner. Structural alignment during similarity comparisons. *Cognitive Psychology*, 25:431–467, 1993.

[137] A. B. Markman and D. Gentner. Commonalities and differences in similarity comparisons. *Memory & Cognition*, 24:235–249, 1996.

[138] D. Maynard, W. Peters, and Y. Li. Metrics for evaluation of ontology-based information extraction. In *Proceeding of the EON 2006 Workshop*, 2006.

[139] D.L. McGuinness and A. Borgida. Explaining subsumption in description logics. In *Proceedings of the 14th Int. Joint Conf. on Artificial Intelligence (IJCAI95)*, pages 816–821, 1995.

[140] Sheila A. McIlraith and David L. Martin. Bringing semantics to web services. *IEEE Intelligent Systems*, 18(1):90–93, 2003.

[141] J. McQueen. Some methods for classification and analysis of multivariate observations. In *In Proceeding of the 5th Berkley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[142] R. Michalski, R.E. Stepp, and E. Diday. Automated construction of classification: conceptual clustering versus numerical taxonomy. *Pattern Analysis and Machine Intelligence. PAMI*, 5:396–409, 1983.

[143] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell. *Machine Learning: An artificial Intelligence Approach.* Tioga, 1983.

[144] R.S. Michalski. Knowledge acquisition through conceptual clustering: A theoretical framework and an algorithm for partitioning data into conjunctive concepts. *Policy Analysis and Information Systems (A Special Issue on Knowledge Acquisition and Induction)*, 4(3):219–244, 1980.

[145] R.S. Michalski and R.E. Stepp. Concept-based clustering versus numerical taxonomy, 1981. Technical Report No. 1073.

[146] R.S. Michalski and R.E. Stepp. Automated construction of classifications: conceptual clustering versus numerical taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 5:219–243, 1983.

[147] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller. Fisher discriminant analysis with kernels. In in Y.H. Hu, J. Larsen, E. Wilson, and S. Douglas (Eds.), editors, *In Proceeding of the IXth Neural Networks for Signal Processing*, pages 41–48. Piscataway, NJ: IEEE, 1999.

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

[149] M. Minsky. A framework for representing knowledge., 1975.

[150] Tom M. Mitchell. *Machine Learning.* McGraw-Hill, 1997.

[151] R. Molitor. Structural subsumption for $\mathcal{ALN}$. Technical Report LTCS-98-03, LuFg Theoretical Computer Science, RWTH Aachen, Germany, 1998.

[152] K. R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Transaction on Neural Networks*, 2(2), 2001.

[153] F. Murtagh. A survey of recent advances in hierarchical clustering algorithms which use cluster centers. *The Computer Journal.*, 26:354–359, 1984.

[154] M.N. Murty and G. Krishna. A computationally efficient technique for data clustering. *Pattern Recognition*, 12:153–158, 1980.

[155] G. Nagy. State of the art in pattern recognition. *Proceeding of the IEEE*, 56(5):836–863, 1968.

[156] B. Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43:235–249, 1990.

[157] T. Di Noia, E. Di Sciascio, F.M. Donini, and M. Mongiello. A system for principled matchmaking in an electronic marketplace. In *WWW*, pages 321–330, 2003.

[158] G. Paass, E. Leopold, M. Larson, J. Kindermann, and S. Eickeler. Svm classification using sequences of phonemes and syllables. In In T. Elomaa, H. Mannila, and H. Toivonen (Eds.), editors, *In Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 373–384. Springer-Verlag, 2002.

[159] M Paolucci, T. Kawamura, T. Payne, and K.P Sycara. Semantic mathcing of web service capabilities. In *Proceedings of the Intern. Semantic Web Conf. (ISWC)*, page 333347, 2002.

[160] M Paolucci and K.P Sycara. Autonomous semantic web services. *IEEE Internet Computing*, 7(5):34–41, 2003.

[161] Chris Preist. A conceptual architecture for semantic web services. In *Proceedings of the 3rd Intern. Semantic Web Conf. (ISWC)*, volume 3298 of *LNCS*. Springer, 2004.

[162] W.V. Quine. *Ontological Relativity and Other Essays.* New York, Columbia University Press, 1969.

[163] R. Rada, H. Mili, E. Bicknell, and M. Blettner. Development and application of a metric on semantic nets. *IEEE Transactions on System, Man, and Cybernetics*, 19(1):17–30, 1989.

[164] L. De Raedt. Attribute-value learning versus inductive logic programming: the missing links. In D. Page, editor, *In Proceeding of the 8th International Conference on Inductive Logic Programming*, volume 1446 of *Lecture Note in Artificial Intelligence*, pages 1–8. Springer-Verlag, 1998.

[165] J. Ramon. Clustering and instance based learning in first order logic, 2002. PhD. Thesis.

[166] P. Resnik. Using information content to evaluate semantic similarity in a taxanomy. In *Proceeding of the International Joint Conference for Artificial Intelligence (IJCAI-95)*, pages 448–453, 1995.

[167] P. Resnik. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research*, 11:95–130, 1999.

[168] M. Andrea Rodríguez and Max J. Egenhofer. Determining semantic similarity among entity classes from different ontologies. *IEEE Transaction on Knowledge and Data Engineering*, 15(2):442–456, 2003.

[169] M.A. Rodríguez. *Assessing semantic similarity between spatial entity classes.* PhD thesis, University of Maine, 1997.

[170] E. Rosch. Cognitive representations of semantic categories. *Journal of Experimental Psychology.*, 104:192–233, 1975.

[171] S. Ross. *A First Course in Probability.* Macmillan, New York, 1976.

[172] C. Rouveirol and V. Ventos. Towards learning in CARIN-$\mathcal{ALN}$. In J. Cussens and A. Frisch, editors, *Proceedings of the 10th International Conference on Inductive Logic Programming*, volume 1866 of *LNAI*, pages 191–208. Springer, 2000.

[173] S. Saitoh. *Theory of Reproducing Kernels and Its Applications.* Longman, Harlow, U.K., 1988.

[174] G. Salton. Developments in automatic text retrieval. *Science.*, 253:974–980, 1991.

[175] A. Schaerf. Reasoning with individuals in concept languages. *Int. Journal of Data and Knowledge Engineering*, 13(2):141–176, 1994.

[176] M Schmidt-Schaußand G Smolka. Attributive concept descriptions with complements. *Int. Journal of Artificial Intelligence*, 48(1):1–26, 1991.

[177] B. Schölkopf. *Support Vector Learning.* Oldenbourg-Verlag, Munich, Germany, 1997.

[178] B. Schölkopf, C. J. C. Burges, and A. J. Smola. *Advances in Kernel MethodsSupport Vector Learning.* MIT Press, Cambridge, MA, 1999.

[179] B. Schölkopf, A. J. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computing*, 10:1299–1319, 1998.

[180] B. Schölkopf and A.J. Smola. *Learning with Kernels.* The MIT Press, 2002.

[181] J. Schürmann. *Pattern Classification: A Unified View of Statistical and Neural Approaches.* Wiley, New York, 1996.

[182] M. Sebag. Distance induction in first order logic. In Nada Lavrac and Saso Dzeroski, editors, *Proceeding of Inductive Logic Programming, 7th International Workshop, ILP-97*, volume 1297 of *Lecture Notes in Computer Science*, pages 264–272. Springer, 1997.

[183] M. Sebag and M. Schoenauer. A rule-based similarity measure. In S. Wess, K. D. Althoff, and M M. Richter, editors, *Selected papers of Topics in Case-Based Reasoning, First European Workshop, EWCBR-93*, volume 837 of *Lecture Notes in Computer Science*, pages 119–131. Springer, 1993.

[184] Evren Sirin and Bijan Parsia. Pellet: An owl dl reasoner. In *Proceeding of International Workshop on Description Logics*, volume 104 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2004.

[185] G. Smolka. A feature logic with subsorts. Technical Report 33, IWBS, IBM Deutschland, 1998.

[186] P.H. Sneath and R.R. Sokal. *Numerical Taxonomy.* Freeman, 1973.

[187] J. Sowa, editor. *Semantic Networks.* Wiley, New York, 1992.

[188] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques., 2000. Technical Report No. 00-034.

[189] M. Sussna. Word sense disambiguation for free-text indexing using a massive semantic network. In *Proceedings of the 2nd International Conference on Information Knowledge Management, CIKM'93*, pages 67–74, 1993.

[190] K. Sycara, R. Guttal, J. Koning, S. Narasimhan, and D. Navinchandra. Cadet: A case-based synthesis tool for engineering design. *International Journal of Expert Systems*, 4(2):157–188, 1992.

[191] L. Talavera and J. Bjar. Generality-based conceptual clustering with probabilistic concepts. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI).*, 23:196–206, 2001.

[192] B. Taush, C. d'Amato, S. Staab, and N. Fanizzi. Efficient service matchmaking using tree-structured clustering. In *Poster proceedings of 5th International Semantic Web Conference (ISWC)*, 2006.

[193] W. S. Torgerson. *Theory and methods of scaling.* Wiley, New York, 1958.

[194] W. S. Torgerson. Multidimensionsal scaling of similarity. *Psychometrika*, 30:379–393, 1965.

[195] D Trastour, C. Bartolini, and J. Gonzales-Castillo. A semantic web approach to service description for matchmaking of services. In *Proceedings of the First Semantic Web Working Symposium*, 2001.

[196] D Trastour, C. Bartolini, and C. Preist. Semantic web support for the business-to-business e-commerce lifecycle. In *Proceedings of the Eleventh Intern. Conf. on World Wide Web*, page 8998, 2002.

[197] A. Tversky. Features on similarity. *Psycological Review*, 84(4):327–352, 1977.

[198] S. Ullman. *High-level vision: object recognition and visual cognition.* MIT Press, London, 1996.

[199] V. N. Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag, New York, 1995.

[200] V. N. Vapnik. *Statistical Learning Theory.* Wiley, New York, 1998.

[201] V. N. Vapnik and A. Y. Chervonenkis. *Theory of Pattern Recognition.* Nauka, Moscow, Russia, 1974.

[202] M. M. Veloso. *Planning and Learning by Analogical Reasoning.* Springer-Verlag New York, Inc., 1994.

[203] E. Voorhees. *WordNet: An Electronic Lexical Database.* MIT Press, 1998.

[204] J.H. Ward. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association.*, 58:236–244, 1963.

[205] C. Watkins. Kernels from matching operations. Technical report, Department of Computer Science, Royal Holloway, University of London, 1999.

[206] D. Wettschereck. A study of distance-based machine learning algorithms, 1994. PhD thesis.

[207] D.R. Wilson and T.R. Martinez. Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, (6):1–34, 1997.

[208] C.T. Zahn. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on Computers.*, C-20:68–86, 1971.

[209] K. Zhang. Algorithms for the constrained editing distance between ordered labeled trees and related problems. *Pattern Recognition.*, (28):463–474, 1995.

Appendix A

The K-Nearest Neighbor Algorithm

The most basic instance-based method is the k-NEAREST NEIGHBOR algorithm [47, 70, 150] that is a *classifier* that inputs a query instance x_q and outputs a prediction for its class. A classifier's performance objective is to minimize *expected loss*, or *misclassification risk*, for each class $c_j \in C$.

In the classical setting, the k-NEAREST NEIGHBOR algorithm assumes each training instance $x = \{x_1, x_2, \dots, x_{|F|}\}$ to correspond to a point in multidimensional space defined by a feature set F, whose class c_j is a member of a set of classes C. Classification is performed, firstly, by selecting the nearest neighbors of an instance query x_q, namely by selecting the k most similar training examples to x_q. The similarity values are computed by the use of a distance/dissimilarity function (typically the Euclidean distance). Once that the neighbors have been selected, the algorithm assigns to x_q, the most common class value among the k nearest training examples. Formally, the algorithm can be described as follow.

Input: query instance x_q Output: the class $c_j \in C$ to which x_q belongs to

Training algorithm:

- For each training example $(x, h(x))$, where $h(x)$ represents the class to which x belongs to, add the example to the list *training-examples*

Classification algorithm:

- Given a query instance x_q to be classified,
  - Let $x_1, \dots, x_k$ denote the k instances from *training-examples* nearest to x_q, selected by means of a dissimilarity function d

– return

$$\hat{h}(x_q) \leftarrow \underset{c \in C}{\operatorname{argmax}} \sum_{i=1}^{k} \delta(c, h(x_i)) \tag{A.1}$$

where δ is the Kronecker delta and is defined: $\delta(a, b) = 1$ if $a = b$; $\delta(a, b) = 0$ otherwise.

Here, the target function to learn is given by $h : \Re^{|F|} \rightarrow C$ where C is the finite set $\{c_1, \dots, c_s\}$ of distinct classes for the classification. The k-NEAREST NEIGHBOR learning task consists in approximating the discrete-valued target function h. Such approximation $\hat{h}$ is given by computing the most common value of h among the k training examples nearest to x_q. Note that the values of h for the training examples are known because they represents the classes to which the training examples belong to.

As the k-NEAREST NEIGHBOR is a lazy learning algorithm, it never forms an explicit general hypothesis $\hat{h}$ regarding the target function h. It simply computes the classification of each new query instance by adopting a majority voting criterion with respect to the classes to which the selected k training examples belong to. The inductive bias of this algorithm corresponds to an assumption that the classification of an instance x_q will be most similar to the classification of the other instances that are nearby w.r.t. the chosen dissimilarity/distance function. Hence, the crucial aspects to ensure good accuracy of the k-NEAREST NEIGHBOR algorithm are: 1) the determined value for the parameter k; 2) the effectiveness of the used dissimilarity measure.

One obvious refinement to the presented algorithm is to weight the contribution of each of the k neighbors according to their distance to the query point x_q, giving greater weight to closer neighbors [150]. Hence, the presented algorithm can be simply modified by substituting eq. A.1 with: $\hat{h}(x_q) \leftarrow \operatorname{argmax}_{c \in C} \sum_{i=1}^{k} w_i \cdot \delta(c, h(x_i))$ where $w_i \equiv 1/d(x_q, x_i)^2$. This means to weight the vote of each neighbor according to the inverse square of its distance from x_q. To accommodate the case where the query point x_q exactly matches one of the training instances x_i (in which case $d(x_q, x_i) = 0$) it can be assign to $\hat{h}(x_q) := h(x_i)$. If there are several such training examples, the value assigned to $\hat{h}(x_q)$ will be the majority classification among them.

Appendix B

The Single and the Complete-Link Algorithms

The Single-Link and the Complete-Link clustering algorithms are hierarchical agglomerative clustering algorithms generally applied to feature-vector representations. They are called agglomerative because the clustering process begins considering each pattern in a distinct cluster. Successively clusters are merged until a stopping criterion is satisfied. In the following the algorithmic description of the Single-Link and Complete link is given.

Single-Link Algorithm
(1) Place each pattern in a cluster. Build a list of inter-pattern distances for all distinct unordered pairs of patterns, and sort it in ascending order
(2) Step through the sorted list, forming, for each dissimilarity value d_k a graph on the patterns where pairs of patterns closer than d_k are connected by a graph edge. If all the patterns are members of a connected graph, stop. Otherwise, repeat this step.
(3) The output of the algorithm is a nested hierarchy of graphs which can be cut at a desired dissimilarity level forming a partition (clustering) identified by simply connected components in the corresponding graph.

Complete-Link Algorithm
(1) Place each pattern a cluster. Build a list of inter-pattern distances for all distinct unordered pairs of patterns, and sort it in ascending order
(2) Step through the sorted list, forming, for each dissimilarity value d_k a graph on the patterns where pairs of patterns closer than d_k are connected by a graph edge. If all the patterns are members of a completely connected graph, stop.
(3) The output of the algorithm is a nested hierarchy of graphs which can be cut at a desired dissimilarity level forming a partition (clustering) identified by completely connected components in the corresponding graph.

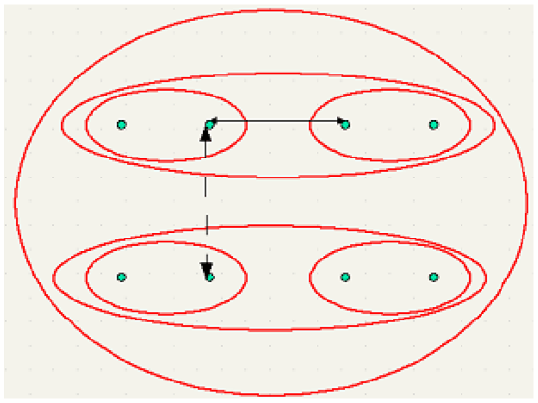
Figure B.1: Clustering process performed by the single-link algorithm. Cluster distances are given by the minimum distance among their elements.

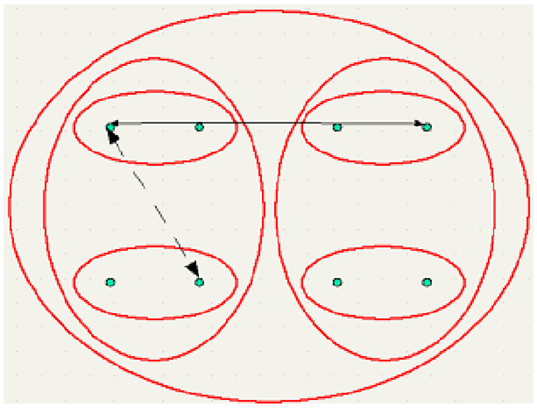
Figure B.2: Clustering process performed by the complete-link algorithm. Cluster distances are given by the maximum distance among their elements.

These two algorithms differ in the way they characterize the similarity between a pair of clusters. In the single-link algorithm, the distance between two clusters is the *minimum* of the distances between all pairs of patterns drawn from the two clusters (one pattern from the first cluster, the other from the second). Conversely, in the complete-link algorithm, the distance between two clusters is the *maximum* of all pairwise distances between patterns in the two clusters. This difference can be more clear by looking at Fig.B.1 and Fig. B.2. In both cases, two clusters are merged to form a larger one on the ground of the minimum distance criteria.

Besides of the way in which single-link and complete-link algorithms characterize the similarity between a pair of clusters, the other main difference between the two algorithms is given by the results they return. The complete link algorithm produces tightly bound or compact clusters (see [18] for more details) while the single link suffers from a chaining effect (see [155] for more details), it has a tendency to produce clusters that are straggly or elongated. This phenomenon is illustrated in Fig. B.3 and in Fig. B.4. Here, two clusters are separated by a "bridge" of noisy patterns. The clusters returned by the complete-link algorithm (Fig. B.4) are more compact than those returned by the single-link (Fig. B.3). The cluster labeled "1", obtained using the single-link, is elongated due to the noisy patterns labeled "*".

In general, the single link-algorithm is more versatile than the complete link. For example the single-link algorithm can extract concentric clusters has those ones shown in Fig. B.5, while the complete-link is not able to do this. On the contrary it has been observed that the complete link algorithm produces more useful hierarchies in many applications than the single-link algorithm [107]. Hence, the choice of the right algorithm strictly depends from the knowledge about the data distribution.

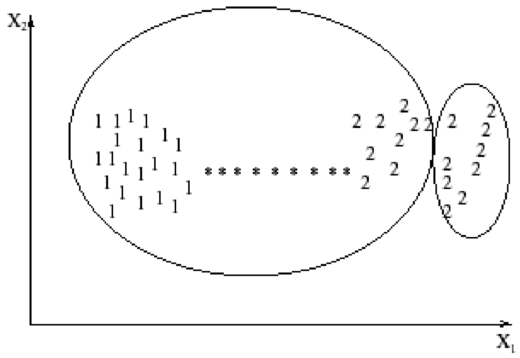

Figure B.3: A single-link clustering of a pattern set containing two classes (1 and 2) connected by a chain of noisy patterns (indicated by ”*”).

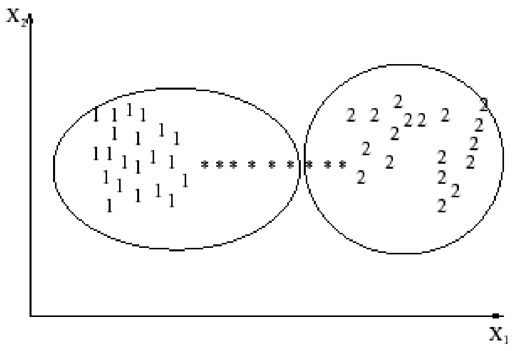

Figure B.4: A complete-link clustering of a pattern set containing two classes (1 and 2) connected by a chain of noisy patterns (indicated by ”*”).

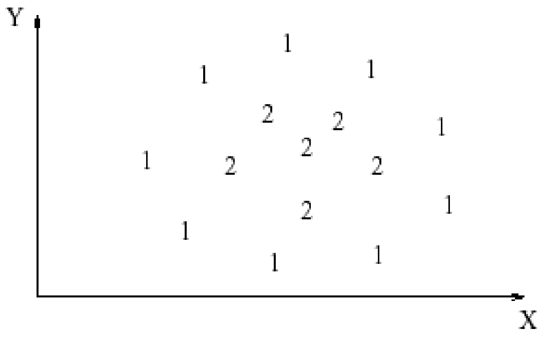

Figure B.5: Two concentric clusters