Institute for Parallel and Distributed Systems
Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 3516

Similarity Recognition for OWL-based Action Recipes in RoboEarth

Ting Sun

Course of Study:	Computer Science
Examiner:	Prof. Dr. rer. nat. habil. Paul Levi
Supervisor:	Dr. rer. nat. Oliver Zweigle
Commenced:	January 01, 2013
Completed:	July 01, 2013
CR-Classification:	I.2.9, H.2.m, H.3.3

Abstract

Sharing and integrating heterogeneous knowledge from different robots requires finding an agreement between the underlying abstract, hardware independent ontologies, saved in the RoboEarth cloud framework in the World Wide Web. A variety of methods from the literature may be used for this task, by means of similarity computation of two ontologies. While most of them come from other fields such as sequence alignment methods from the field of bioinformatics, they can be extended for ontology alignment purposes. While such methods basically perform a pair wise comparison of ontology entities, another approach called the OWL-Lite Alignment method uses a variety of different ontology alignment methods to integrate many ontology comparison techniques in one common framework. Both the sequence alignment based and OWL-Lite based solutions are presented and their suitability to conduct a similarity check for robot task descriptions upon an upload into the RoboEarth platform discussed.

Contents

List of Figures

List of Listings

1 Introduction

This work represents a fragment of the RoboEarth project. Hence a rough introduction over the project shall be given in this introduction to cover the basic idea of RoboEarth, followed by an outline of the motivation and an overview of this work.

1.1 Cloud Computing in General

The Cloud Computing technology emerged approximately around 2008 out of existing technologies like Grid Computing, Utility Computing and Web Services. It has since been heavily discussed and according to Gartner's 2011 Hype Cycle Special Report [Gar11] Clouds were still on the peak of expectations and most probably still are. With a widespread use followed by the introduction or norms and patterns [FLR+13], "clouds" have moved from a mere trend towards a commonly accepted technology.
Cloud Computing revolves around information infrastructures, platforms or applications provided dynamically on demand and usually via the Internet. Users only use whatever resources they want upon requirement and do not need to deal with the effort to set up and maintain their own computer centres, while often not even using them at full capacity. This saves them a lot of effort, hence also time as well as monetary costs. This is especially the case for users not well versed in hardware and software issues. The provided resources are also scalable. When more resources are needed, the users can simply upgrade their service plan and the cloud provider will supply them with the requested additional resources. So called Service Level Agreements (SLA) regulate the prices for resources, along with other service guarantees, such as High Availability or High Performance. The provider can be an IT company, who specializes in providing such systems, more likely to provide far better resources than when the user would have to maintain them themselves. To allow scalability of the resources virtualization and cluster technologies are employed to maximize the utilization of hardware resources in their computer centres. Since providers usually have more than just one customer, they can bundle all requested resources and make better use of less hardware. This lessens the working expenses and can be handed down to the customers. Another major advantage of clouds is the global availability. Since most of the cloud services are distributed via the internet, they are globally accessible, usually hardware independent and need no prior local installation.
Works such as "Cloud Computing: The Next Revolution in IT" [Ley], "Above the Clouds: A Berkeley View of Cloud Computing" [FGJ+09] and the "Open Cloud Manifesto" [Man10] among many others [MH10] [Com09] [Vos12] [MRV11] discuss the topic in general and more in depth. Here RoboEarth is used a a model implementation illustrating the usage and advantages

of "clouds".

1.2 RoboEarth: A Cloud for Robots

In robotics one major challenge is to create a robotic system capable of recognizing and interacting with human environments. When execution planning enables this, robots will be capable of safely coexisting with humans in complex indoor environments such as common domestic homes. They will be capable of carrying out tasks beneficial to society, such as simple household tasks help, supporting, for instance, the elderly or handicapped as a sort of care assistant. Additional merit can be gained in integrating such technology into other future products. Further knowledge such as manipulative, perceptive and communicative skills is essential for this purpose as robots need to act and react independently on situations and environments in a flexible way, while not merely relying on a narrow set of pre-programmed environment and object models. Programming a robust plan for a robot is often a tedious business and usually such plans cannot be reused for different situations and thus create a lot of avoidable workload.
One strategy to cope with these issues of uncertainty and dynamics is to have robots learn and afterwards share their memory of re-usable knowledge among robot peers in a sort of web-community. This is put into practice through a Web-style database for and by robots in the World-Wide-Web, realized by the RoboEarth Project. The project's interdisciplinary team researches new approaches on equipping robots with advanced perception and action capabilities as means to autonomously carrying out tasks within not previously planned surroundings. Learning Algorithms are reinforced and optimized. Other new approaches include automatic conversions from instructions in natural language into logic-based, formal representations in stead of the traditional approach of plans composed from atomic actions.
Such knowledge is then collected abstractly within one core component of the project, the RoboEarth platform. Any data is saved in the cloud computing database, which is based on Adobe Hadoop and globally accessible. It serves as a collective, worldwide memory which robots can access to gain necessary knowledge on how to deal with new environments. Upon performing a task a robot can download adequate high-level information on both the task itself and the environment in which the task is to be carried out. The robot will proceed to translate this abstract knowledge into local hardware specifications and configurations and the local system's functionality will ultimately improve by learning during the task. The newly gained knowledge is in turn shared with its robot peers again via an upload to the RoboEarth platform. Such reusable data is to be saved in a generic and open format to ascertain the global accessibility.
The project's goal is to prove that such a "global memory", realized through the World-Wide-Web, accelerates learning new tasks as well as adapting to new scenarios. [WBC+11] [DMTH+13] [AAV+08] [Bro10] [TNB10] [Sch10] [Sch10] [TKPB11]

1.3 Similarity Recognition

Underlying any action or recognition of a robot is a plan and therein task descriptions. A problem is given with the fact that the same issue can be described in subtly to slightly different manners. This includes the description's level of detail, little differences in task orders and usage of different nouns and verbs to accomplish the same result. For example "fetch a bowl and fork, smash first the egg yolk and beat the eggs" and "stir eggs with a spoon" essentially mean the same thing in probably the same context of, for instance, baking a cake. There is a difference in the level of detail in the instructions (smashing the egg yolk first is not essential for the action of beating the egg itself), the used tools (as both fork and spoon or even chopsticks can be used to beat eggs) and semantic similarity in the verbs used (beat, stir or whip the egg). There are many cases which contain similar parts, namely one or several subactions, while the context or overall action differs thoroughly. For instance both baking a cake and cooking an omelet need a bowl fetched to have the eggs stirred in it, but the results are two totally different dishes. This can go even further that the fetched bowl has an entirely different use, like being filled with fresh water to be delivered somewhere. While not all actions with similar parts may yield the same result, there may still be lot of indifferences to the results itself if the performance is not strictly linear, when no explicit order constraint is given. For example, in making a cake batter it does not matter, if first sugar or the baking powder is added to the flour before adding wet ingredients, while in case of baking a cake it would matter to first mix the batter and then bake.
Thereby robots would naturally accumulate semantically similar tasks description and increasingly be confronted with the problem of having to choose among multiple such descriptions. This is especially so, as one of the innovative approaches of the RoboEarth project consists of conversion of natural language instructions into logic-based, formal representations. [MHZF12] With natural language having many ways of formulating the same thing it is not unsurprising that such duplicates with only minor differences will occur. Another possible cause may for instance, come from the work on having robots observe humans demonstrating a task to them and the analysis thereof not yielding completely the same results.
The current situation creates a redundant information processing overhead, as the robots would eventually have to find out they describe the same task. It also creates ab information storage overhead in the database of the "global memory". As data space usage and traffic are main points in cloud services' SLA (Service Level Agreements) [MH10], i.e. scalation of how expensive a cloud service is, it is also of interest to reduce the overhead under economical aspects.

Tenorth, Ziegltrum and Beetz [TZB12] introduced the idea, to have the platform, in whose database the task instructions are assembled, match such descriptions. In this precise case, the RoboEarth platform should be able to detect and, if necessary, delete a redundant entry amongst the action descriptions. Furthermore, if several tasks share heavy commonalities, they are probably more used and thus more important. It would probably be expedient to make a reusable alternative version of the descriptions for such action sequences.

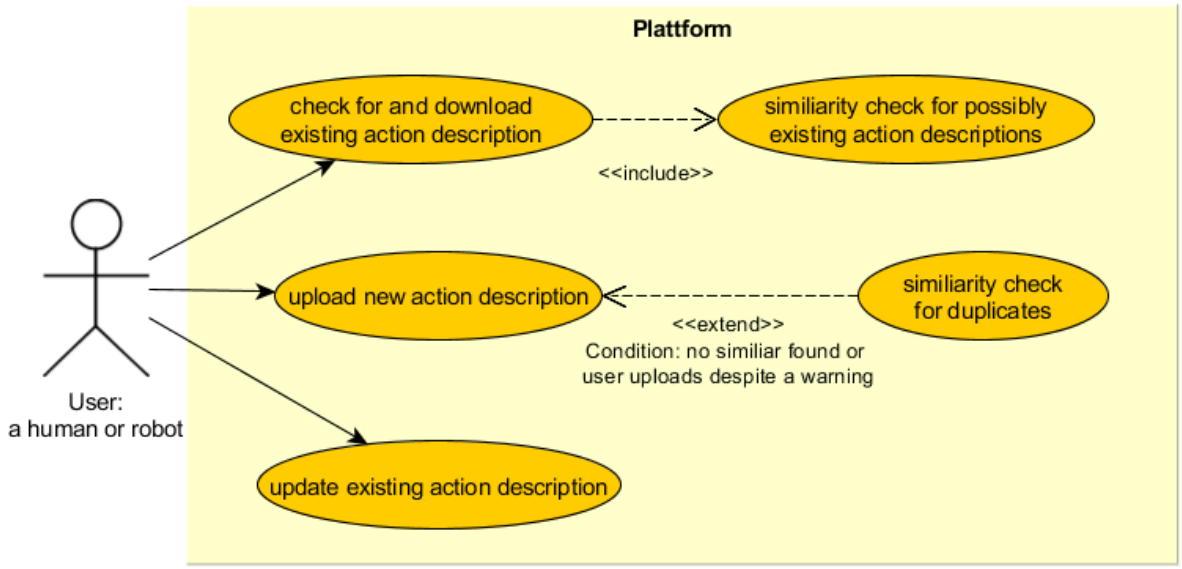

Figure 1.1: A use case diagram illustrating the use of the similarity check within the platform

Figure 1.1 shows the working field of the similarity check with some usage context. If the user wants to upload an action description the uploaded file will be checked against the database contents. If one or more similar descriptions exist, the user is given a similarity value and asked to check if the new upload may be redundant. The user is then free to refrain from the upload or to ignore the warning after which the file will be inserted into the database.
Currently human-assigned tags are used to narrow fields of topics down, but this solution is quite imperfect and can only serve as a temporary workaround. In this work some approaches for the similarity comparison will thus be presented.

1.4 Structure of this Work

Chapter 2 gives a general scheme on the background of this work, the RoboEath platform, ontologies in general and OWL in particular. Chapter 3 gives a quick overview of other related work dealing with ontology alignment and mapping. Based on Tenorth, Ziegltrum and Beetz' [TZB12] motivation to extend bioinformatical sequence alignment methods for action recipe alignment and inspiring this work, chapter 4 presents their ideas along with some further extensions. Chapter 5 presents the method OLA [EDLV04] in detail, a system using various methods to accomplish an optimal alignment. Finally chapter 6 gives a quick summary along with a conclusion on to what extend the two methods are adequate to solving the recipe alignment issue in the RoboEarth platform.

2 Background

2.1 Object and Action Recipes used in RoboEarth

For a "global memory" to play out its usefulness its knowledge needs to be shared on an abstract meta-level, independent of the robots original hardware platform and its configurations and specifics (i.e. impossibility to make assumptions about available capabilities of the heterogeneous robots for instance). It also has to be a description in well-defined, machine-understandable logical axioms. Ontologies serve this purpose quite well. This section will give a quick introduction to ontologies in general, action recipes written in OWL in particular and illustrate their composition on a simple action recipe example.

2.1.1 Ontologies

In the eighties of the last century ontologies had been used in bigger research projects as Cyg and Wordnet for the first time. These two projects remain model examples for large scale ontologies to this day. Ever since then ontologies have gradually become more popular. Much attention is attributed to them within the Semantic Web, which is somewhat associated to the "Cloud Revolution". Both of them provided the necessary technologies to employ the already existing basic concepts which originated from philosophical and linguistic fields. Figure 1.1 illustrates the model: The world's real object, called "Referent", is associated with linguistic sign called "Symbol" and logics based on what one would expect the referent's attributes, called "Thought or Reference". For instance a rabbit with the linguistic sign "English Lop" has "two hanging ears and 4 legs" as its logic. Ontologies as one way of data processing formalize the "Thought or Reference", generally also better known as concepts (or classes). Ontologies formally and explicitly specify concepts, which can be commonly used. They organize information by providing a structural framework and find usage in various fields, such as the already mentioned Semantic Web, Artificial Intelligence or Information Architecture. [Stu09] [Bat08]

The ontology used in the RoboEarth platform is the Web Ontology Language (OWL). Like many other ontology languages it is based on first-order logic and therefore benefits from reusing or integrating already existing technologies and paradigms from the field of logic programming (for instance PROLOG being the most known example) to formalize ontologies in a practical and efficient manner. More precisely OWL is a description logic, which is a distinct subset of the first order logic. This offers the advantage of relatively efficient inference methods while maintaining the unambiguous characteristics of first-order logic. In recent

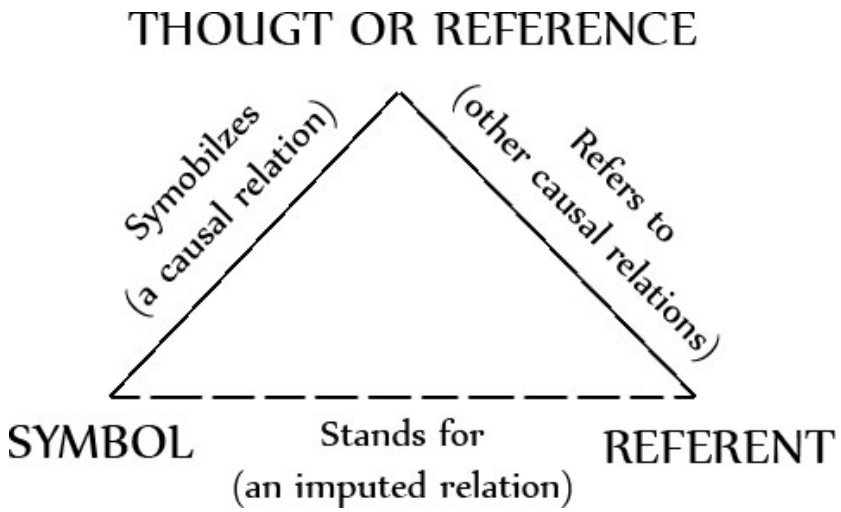

Figure 2.1: The triangle of reference [OPR23]

years OWL has gained more importance, as it has been standardized (the latest standard being OWL 2 from 2009 [W3C]) and become a W3C recommendation in 2004 [MVH+04]. It provides the foundation of the Semantic Web and has therefore become the most used ontology language. Among description logics OWL also stresses the importance of the element of relations within the ontology, which enables complex connections between relations, such as subsumption, equivalence and mathematical relations. [Stu09] OWL is built with RDF (Resource Description Framework) as a base. RDF allows description of resources through specification of properties and property values. OWL semantically extends RDF and adds other desirable capabilities. Among others these include cardinality constraints, specifications of transitivity, inversion, negation or the ability to describe new classes through combining existing classes. [OWLa]
OWL components consists of classes, properties and individuals. Classes are distinguished from instances of them. Properties can be attributed to only an instance or the whole class. The classes built up a hierarchical structure and represent the ontology as such.
OWL comes with three flavours: OWL-Full, its subset OWL-DL and OWL-Lite, again subset of OWL-DL. OWL-Lite was built to express the same semantics of OWL-DL, supporting users who primarily need a classification hierarchy with simple constraints. In consequence OWL-Lite is a less complex language than OWL-DL through further syntactic restrictions. Any OWL-Lite ontology is automatically an OWL-DL ontology and can be migrated quickly to the more expressive derivatives of the OWL mother language. [MVH+04]
For a more detailed elaboration on ontologies, numerous works have are available, such as [Stu09] or [HKR+04].

2.1.2 Action Recipes in the RoboEath System

Action recipes are hardware-agnostic task descriptions for the robots and as this work centers upon them, an overview of RoboEath's architecture in general and action recipes in particular shall be given.

As illustrated in figure 2.2 RoboEarth implements a three-layer architecture. The server layer (figure 2.2 (a)) is the foundation and holds the RoboEath database, storing global world models, environments and actions, while providing basic reasoning Web services. The database services are available through interfaces of the architecture, accessible via common Web interfaces. Figure 2.2 (b) illustrates the generic middle layer, whose main purpose is to allow a robot to interpret RoboEarth' action recipes. The layer is hardware-independent and also provides several functionalities and communicates with robot-specific skills, which the third layer (figure 2.2 (c)) implements. The third layer also provides a generic interface to the robot's specific, hardware-dependent functionalities though an abstraction layer. [WBC+11]

The RoboEarth platform uses the RoboEarth language, which is based on description logic and uses OWL for implementation in form of an extension to the KnowRob [TB09] knowledge base. KnowRob itself is written in OWL and derived from the OpenCy Ontology [MCWD06]. The RoboEarth Language includes object models, environments maps and action recipes. Reasoning methods are used to make sure that execution of an action recipe is possible on a robot, find adequate information within the RoboEarth database and provide basic abstract descriptions on the robot. Descriptions of information are done in shareable, abstract manner. Any data is annotated with an ontology description to allow semantic and taxonomic queries. [DMTH+13]
The ontology for actions is a part of the KNOWRob ontology and more than a hundred everyday activity actions' class descriptions are currently available. Action recipes have a specialized hierarchy and tasks composition. These currently over 130 class descriptions serve as a foundation block for building up more complex task descriptions in the future. There are declarative and procedural descriptions and involved objects and locations are included, as well as dependencies both from inherited OWL classes or specially defined restrictions, e.g. ordering of actions or pre- and post-conditions. Figure 2.3 illustrates the structure of such an OWL file on a simple daily life chore, while figure 2.1 gives a short excerpt of the OWL file.

The action recipes store information such as a list of included subaction recipes (e.g. the GraspBottle class in figure 2.3), skills and ordering constraints as well as further properties in form of action parameters (e.g. grasp types in figure 2.3) necessary for execution. The subactions are partially ordered, which allows both sequential and parallel task execution order. [DMTH+13]

Action recipes are being linked to object models. Reasoning methods search for actions that can be performed with or on the object. Objects are in turn included in the semantic map. As presenting all components would go beyond the scope of this world, please refer to [WBC+11], [DMTH+13], [PT11], [SJ12], [Sch10], [TPLB12], [TNB10], [TB09], [TPLB12], [TB13], [ZMdH09], [SHZ10], [TZB12], [Sch10] and [PT11] for further reading.

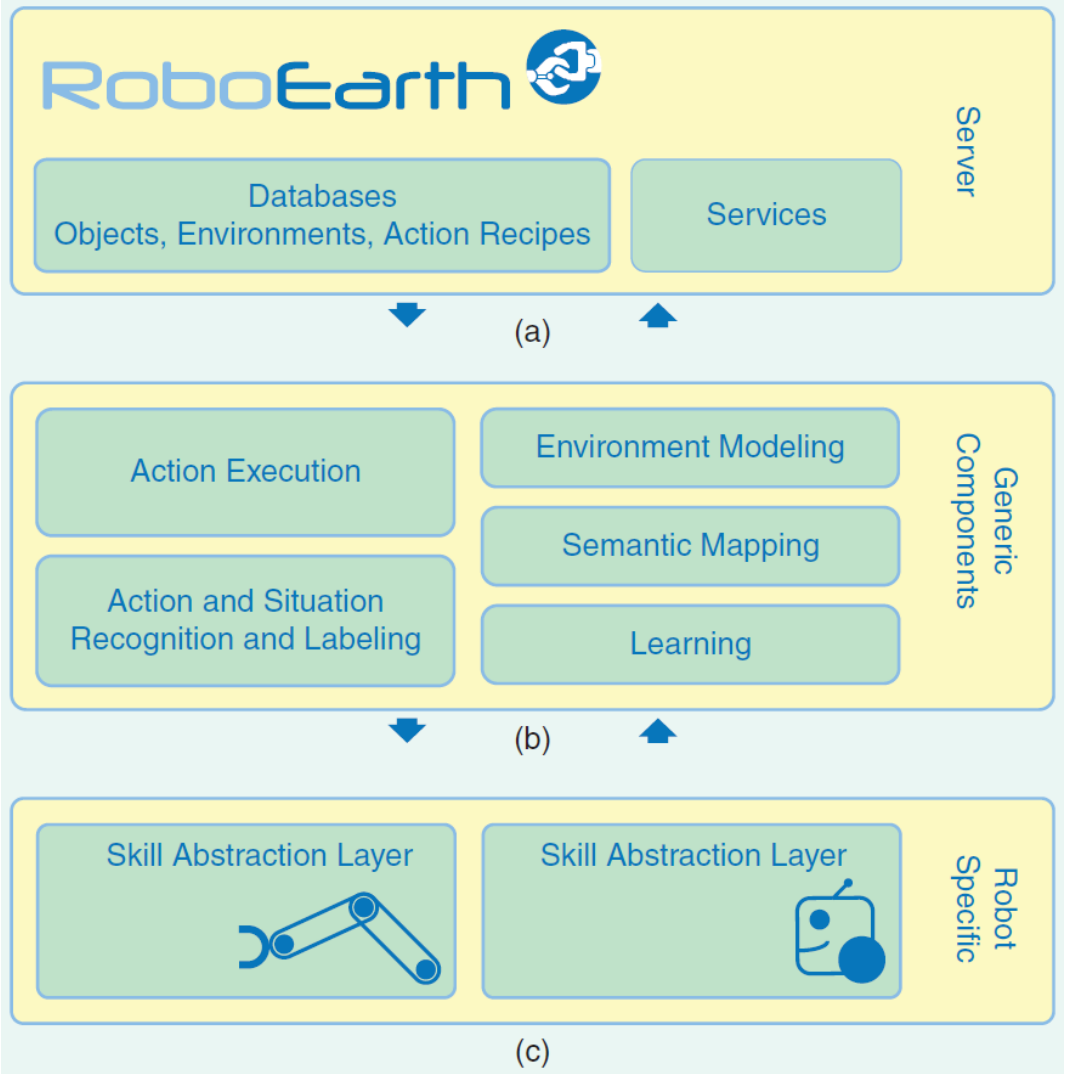

Figure 2.2: Three-layer architecture of RoboEarth [WBC+11]

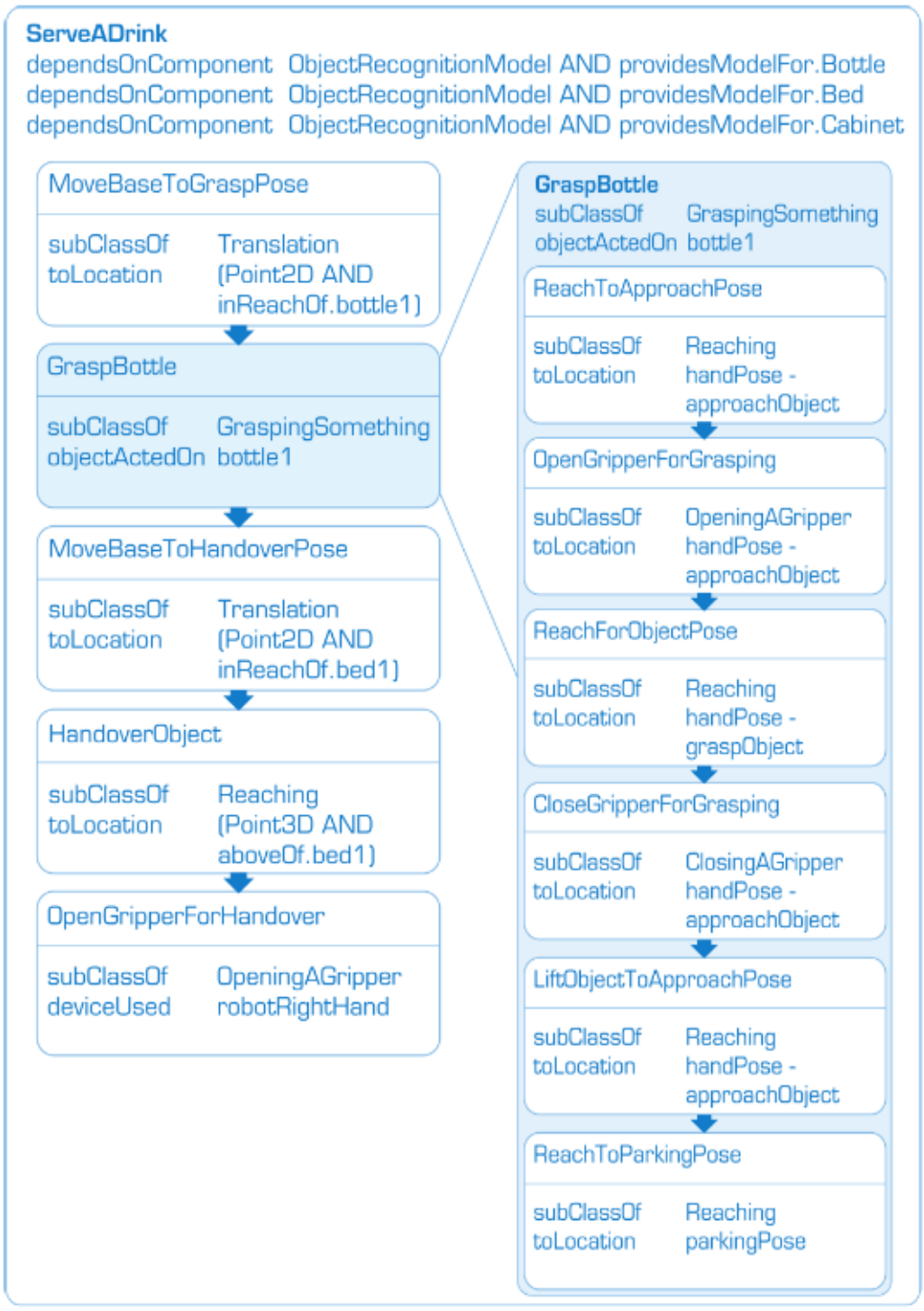

Figure 2.3: Class diagram for a sample action recipe [WBC+11]

Listing 2.1 Action recipe excerpt: Subaction and partial ordering specification [DMTH+13]

```
Class: MoveBaseToHandoverPose
       EquivalentTo:
              knowrob:toLocation some
                  robotPose-handover1)
       SubClassOf:
              roboearth:Translation-LocationChange

...

       Individual: ServeADrinkOrder10
           Types:
                  knowrob:PartialOrdering-Strict
           Facts:
                  knowrob:occursBeforeInOrdering
                         MoveBaseToGraspPose,
                  knowrob:occursAfterInOrdering
                         GraspBottle
```

3 Quick Overview of Related Work

Ontologies have become very common, often even considered to be the backbone of the Semantic Web. Integrating heterogeneous knowledge has consequently been an issue as well. There are many other approaches than the two presented within this work. This chapter shall give a quick general overview of other related work to the topic.

[ont] collects papers from the yearly OAEI (Ontology Alignment Evaluation Initiative) campaign since 2004. Ontology matching consists of not just similarity computation, but outputs correspondences of semantically related entities of ontologies, which may serve for other purposes as well, such as ontology merging or data translation. The OAEI workshop of 2007 [þES07] [OAE07] received 26 submission dealing with the topic along with experimental results, including OLA. Within those implementations many of them are frameworks, basically accumulations of other methods included. WordNet, which is presented in this work as well, is also often used. Aside terminological, structural and/or extensional methods, methods from the Semantic Web like Semantic Matching through trees or schema matching are also often introduced as an alternative. Examples include [GYM07] or with atop of it the use of Semantic Search Engines [GLd+07]. [GYS07] also gives an overview of semantic matching algorithms.
In [AP07] a set of semantic ontology similarity measures are presented as well. In [WIVDMS07] multi-concept alignment are focused on, i.e. a set of concepts aligned to another set.
In [Ehr07] Ehrig also presents a general formal overview of the common ontology alignment problem and gives a quick summery over methods addressing them. Its focus is also more on the usage within the Semantic Web including ontology merging and mapping. Among others, OLA is also mentioned there.
In [AKTKVH06] a case study on selected ontology alignment tools is done, such as FOAM [ES05] (also part of the OAEI 2007 campaign), Falcon-AO [JHCQ05] [HQ08] [HCZ+06] (part of the OAEI 2007th predecessor 2006 campaign) and S-Match [GSY04].
In [MS02] a set of similarity measures and empirical evaluation is presented. Its focus is more on finding similar ontologies for adequate data querying and retrieval.
Maedche and Staab [MS01] propose a methodology to measure the extent to which two ontologies overlap and fit with each other at various semiotic levels. The method also contains a syntactic, as well as a semantic comparison level through taxonomy and template comparison.
In [SM01] proposes a method called FCA–MERGE for merging ontologies through a bottom-up approach which offers a structural description of the merging process.
In [GS03] a complete prover is used for deciding subsumption or equivalence between classes given initial equivalence of some classes and analysis of the relationships in the taxonomy.
In [SE05] Shvaiko and Euzenat did a survey on recent ontology and schema matching techniques within the context of proposing their own new classification system.

[RB01] and [MBR01] deal with schema matching from the database area, whose methods can and are often translated to the ontology alignment problem.
[MLQ09] propose a method with translation into RDF triples, dynamic adjustment of the semantic weight of OWL constructors to calculate the similarity.
[DMDH04] presents a system called GLUE, which employs learning techniques to semi-automatically create semantic mappings between ontologies.
In [Sun08] Sunna also gives an introduction to the general problem of heterogeneous ontologies and the presentation of a multi-layered approach to ontology alignment, which has also been part of the OAEI 2007 campaign.
[LDK07] presents method for mapping entities of ontologies written in OWL DL/Lite. The so called ASCO3 algorithm searches through the two ontologies' maximal common subgraph, while searching the maximal clique of their association graph. [MGMR02] proposes another graph-based method and is addressed to not just OWL, but a variety of data structures. Their method relies on the similarity of adjacent concepts on the graphs, which are similar, when the two given ontologies are similar.
[KC08] uses description logic and a vector model-based weight between concepts.
[DH98] uses a conceptual graph formalism for ontology comparison.
[Bun97] discusses graph edit distances in a more general scope.
[GANJ06] propose a similarity measure specialized on OWL-S annotated web services. [GQ08] proposes one as well.
[DK10] presents a Protege plug-in to including various methods to align ontologies.
[Ghi12] deals with OWL-Lite ontology alignment, including a survey on existing methods an presenting a new algorithm, which is specially compared to OLA. Another OWL-Lite centered method is presented in [ZNK+07] using graphs.
RiMOM (Risk Minimization based Ontology Mapping) [TLL+06] is a system intending to combine various strategies, including both linguistic and structural comparison.
OWLDiff [KSK11] is a system using syntactic, explanation-based and semantic comparison and offers merge functionalities.
AnchorPROMPT [NM01] is another graph implementation, using anchors, which are either user defined related terms or automatically identified by lexical matching.
[LSM06] focuses on less traditional requirements, such as multiple ontology matching instead of pairwise, mapping of ontologies not close to each other and efficiency instead of only matching quality.
Chapter 4 of [SG12] also discusses the ontology alignment problem and a brief overview of several selected methods in the context of discussing applied Semantic Web technologies.
[DMDH04] presents a method using machine learning techniques to make emerge good alignments and finding class similarity from instances.
[Euz94] gives an overview of a tropes taxonomy building tool using T-trees to infer dependencies between classes (bridges) of different ontologies sharing the same set of instances based only on the "extension" of classes
Other work include proposals to extend OWL itself for improvement of the representation of OWL entities. [LJP10]

4 Similarity Recognition via Extensions to Sequence Alignment Methods

Tenorth, Ziegltrum and Beetz [TZB12] motivated the use and extension of bioinformatical sequence alignment methods for comparison of action recipes in RoboEarth. Similarity Recognition of sequences is a very important field in the field of bioinformatics. Nuclear acids need to be matched for comparison purposes of DNA strings or amino acid sequences of proteins.
Figure 4.1 shows a sample alignment quite common in the bioinformatics. Note however that this sample is only a minimal fraction of how long such a sequence usually is.

There are several differences for the OWL Alignment purposes to consider: DNA and protein sequences are usually considered atomic. For example, DNA sequences only have the four bases Adenine, Cytosine, Guanine and Thymine (A, C, G, T) and a DNA sequence would be represented by combination of only these letters and can take on extreme lengths. Robot action sequences usually are rather short, but contain complex structures with the related action classes, objects handled with or upon and locations among others. Most importantly, they are hierarchically structured, as an robot action and in consequence also its action recipe is a composition of subactions. These additional attributes require an extension of the existing algorithms from the bioinformatic field, before they can be applied to action recipes. [TZB12]

This chapter will therefore introduce several sequence alignment methods from the bioinformatic field followed up with a presentation of extensions to deal with the more complex action recipe structure adequate for employment in the robot knowledge base RoboEarth.

Listing 4.1 Sample multiple alignment of a portion of the cytochrome b protein from various organisms. [KYB03]

```
PGNPFATPLEILPEWYLYPVFQILRVLPNKLLGIACQGAIPLGLMMVPFIE
PANPFATPLEILPEWYFYPVFQILRTVPNKLLGVLAMAAVPVGLLTVPFIE
PANPMSTPAHIVPEWYFLPVYAILRSIPNKLGGVAAIGLVFVSLLALPFIN
PANPLVTPPHIKPEWYFLFAYAILRSIPNKLGGVLALLFSILMLLLVPFLH
PANPLSTPAHIKPEWYFLFAYAILRSIPNKLGGVLALLLSILVLIFIPMLQ
PANPLSTPPHIKPEWYFLFAYAILRSIPNKLGGVLALLLSILILIFIPMLQ
IANPMNTPTHIKPEWYFLFAYSILRAIPNKLGGVIGLVMSILIL..YIMIF
ESDPMMSPVHIVPEWYFLFAYAILRAIPNKVLGVVSLFASILVL..VVFVL
IVDTLKTSDKILPEWFFLYLFGFLKAIPDKFMGLFLMVILLFSL..FLFIL
```

4.1 Bioinformatical Sequence Alignment Methods Overview

In bioinformatical sequence comparison the order, type and properties of the atomic sequence elements are of importance. When comparing two sequence elements, the possible outcome is either a match (the elements are equal up to the current position), a mismatch (they are not equal) or an insertion or gap (one sequence contains an element the other does not).
For example [AE86], three possible alignments for the Sequences $CGGA$ and CGA are:

$$A_1 = \begin{cases} CGGA \\ CG-A \end{cases} \qquad A_2 = \begin{cases} CGGA \\ C-GA \end{cases} \qquad A_3 = \begin{cases} CGG-A \\ -CG-A \end{cases}$$

A common cost specification would be 0 for a match, 1 for a mismatch and alignment of an element with a *null* (shown as "-" in the alignment example) through a point insertion or deletion is charged with an arbitrary *null* cost, for instance 2. In this case, alignment A_1 and A_2 would cost 2 while A_3 costs 7 in total with three deletions and one mismatch.

Sequence comparison methods assign a certain cost to each alignment with the goal to minimize the overall cost by means of finding a solution that makes minimal use of necessary atomic transformations to transform one sequence into another. Atomic transformations include inserting, deleting an element or replacing an element. The latter case, which results from a mismatch is usually allocated a greater cost than inserting or deleting an element in occurrence of a gap.
In bioinformatics, where alignment problems occur very commonly, efficient algorithms and tools have been developed for the last few decades. There are multiple algorithms for comparison of just two sequences as well as alignment of multiple sequences for computing alignment of two sequences at minimal cost. 2-alignment algorithms are mainly based on dynamic programming, while multiple alignment algorithms mostly base upon and include such 2-alignment algorithms. As diving into the field of multiple alignment methods would go far beyond the scope of this work and no real further merit in contributing to the problem solution, the presentation of possible algorithms shall be restricted to a few more common 2-alignment algorithms. For further read on multiple alignment methods, please refer to [CWC92] and [CL88] for a quick overview.

4.1.1 Needleman-Wunsch Algorithm

The Needleman-Wunsch algorithm, published in 1970 [NW70], was the first algorithm to propose a method for sequence alignment of amino acid sequences found in proteins. It remains one of the main algorithms to this day with numerous derivations to optimize the performance for more specialized problems.
The Needleman-Wunsch algorithm performs a global alignment of the two complete sequences.

It uses a dynamic programming approach to find the optimal global alignment of two sequences progressively through first computing the alignments of sub-sequences.
Given two sequences x and y, let m and n be their respective length, with x_i being the ith symbol in x and y_j the jth symbol in y. The Needleman-Wunsch algorithm will sequentially compute the optimal alignment for all prefixes of both sequences in a matrix F. The cell $F(i, j)$ will always contain the optimal solution for the alignment of the prefix subsequences $x_0...x_i$ and $y_0...y_j$. The algorithm exits, when both prefixes of the iteration consist of the full sequence and leaves the optimal alignment score in the cell $F(m, n)$ of the score matrix.
The method consists of three main steps: Initializing the matrix, recursive computation of alignment scores and a traceback step:

4.1.1.1 Matrix Initialization

The matrix is initialized with a first row and column. They correspond to the alignment of sequence x to y with gaps in the latter and vice versa. (Basically aligning one sequence of the prefix of the other consisting of an empty string). The first element $F(0, 0)$ (align "nothing" to "nothing") is initialized with zero and the other fields of the first row and column follow with the cost of deleting/inserting every element of the prefix of the sequence. In short:

$$F(0,0) = 0 \tag{4.1}$$

$$F(i,0) = i \cdot d \tag{4.2}$$

$$F(0,j) = j \cdot d \tag{4.3}$$

with d being the cost associated to gaps (deleting an element).

4.1.1.2 Recursive computation of alignment scores

Upon computing the following alignment scores, the prior scores in the neighbouring cells are looked upon and the optimal alignment is computed by adding the alignments of the preceding sequences, adding the costs of the necessary next steps to align the one element longer current prefix and choosing the least expensive global alignment of the current prefix. There are three possibilities to choose the best from, which are based on three prior neighbouring elements $F(i_1, j_1)$ (diagonally left), $F(i-1, j)$ (on top) and $F(i, j-1)$ (left). (See also figure 4.1.) In short:

$$F(i,j) = max \begin{cases} F(i-1,j-1) + S(x_i, y_i) \\ F(i-1,j) - d \\ F(i,j-1) - d \end{cases} \tag{4.4}$$

with $S(x_i, y_i)$ being the cost to match x_i to y_i.

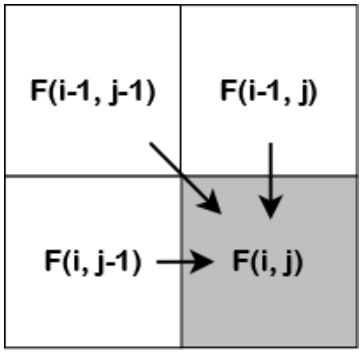

Figure 4.1: Computation of an alignment score using its neighbouring matrix elements [TZB12]

4.1.1.3 Traceback

There is either a second matrix or the alignment score matrix' cells also contain an additional pointer to the cell, from which it has been reached. Once the alignment scores are completed, the optimal alignment is reconstructed by tracing the path back with $F(m, n)$ being the starting cell and $F(0, 0)$ being the destination cell.

4.1.2 Smith-Waterman Algorithm

The Smith-Waterman algorithm was published in 1981 [SW81], for "Identification of common molecular subsequences". In contrast to the Needleman-Wunsch algorithm, which aligns globally, this algorithm performs a local alignment. The major difference is that in local alignment only subsequences are searched for the optimal alignment. Findings of non-matching parts before or after the subsequence are of no importance and ignored. The method was meant to find subalignments of interest, as an alignment with length 20 and 2 mismatches is often considered more interesting than a subalignment of length 5 with no mismatches.
Essentially the Smith-Waterman algorithm can be transformed into the Needleman-Wunsch algorithm and vice versa [AE86]. Chosen the appropriate parameters both of them are even directly equal. The difference of the computation is within the additional condition of $F(i, j) = 0$ if all three alignment scores to choose from happen to be negative values:

$$F(i,j) = max \begin{cases} 0 \\ F(i-1, j-1) + S(x_i, y_i) \\ F(i-1, j) - d \\ F(i, j-1) - d \end{cases} \tag{4.5}$$

again, with $S(x_i, y_i)$ being the cost to match x_i to y_i. Reasonably $S(x_i, y_i)$ is defined to have a match increase the similarity, while adding a mismatch will decrease it.

The Needleman-Wunsch and Smith-Waterman methods among others define $S(x_i, y_i)$ as a linear function, due to algorithmic considerations on finding the local optimal subalignments fast. There are approaches to use other measures as well. [AE86] for instance proposes a method with a nonlinear, logarithmic measure for subalignment similarity for a definition of "good subalignments" that does not focus on speed, but on the probability which alignment is less likely to occur.

4.2 Further Sequence Alignment Comparison Methods

While there exist plenty more specialized algorithms, they consider more particular characteristics and are usually calculated in a more complex manner. The profits of employing these appears rather questionable given the current size of the RoboEarth platform's database [rob] and the general problem specification of the similarity recognition of action recipes, as there are more characteristics to consider than just string similarity comparisons done on labels and descriptions. The Needleman-Wunsch or Smith-Waterman should be sufficient while having the advantage of simplicity. A very short introduction of some further methods shall be given nonetheless, along with an overview of further literature in the bioinformatical sequence alignment field.

4.2.1 Molecular Sequence Comparison with Levenshtein Distances

Levenshtein [Lev66][THK91] already gave early definitions of distances by a function of insertions and deletions. One way to define it is as a minimum effort to change a sequence a to a sequence b. I.e. it may be the number of insertions, deletions and substitutions to transform the sequence. Another definition is the number of insertions and deletions (with no substitution allowed) to transform sequence a to sequence b. (See figure 4.2 for a visual example.) The minimum distance $D(a, b)$ is defined as the smallest possible weighted sum of insertions, deletions and substitutions to change sequence a to sequence b. $D(a, b)$ doesn't necessarily be the minimum number of changes in the sequence. So $D(a, b)$ can either just be a listing distance or an alignment distance, depending what type of analysis is chosen. A listing gives the steps of the transformation, an alignment gives a visual matching of the two sequences.

The listings method, simply takes the number of necessary steps to change the sequence. The steps can be weighted and if the steps of figure 4.2 (left) are each assigned weight 1, the distance is simply two, regardless from the order of steps. The alignment distance is the sum of weighted distances associated to the alignment. Taking the examples from figure 4.2 (right) with matchings getting a weight of 0, substitutions a weight of 2 and insertions/deletion a weight of 1, the first example would get the value of 2, the second a 4 and the third would be 6. Through the distances, alterations to the Needlemann-Wunsch algorithms can be done, which

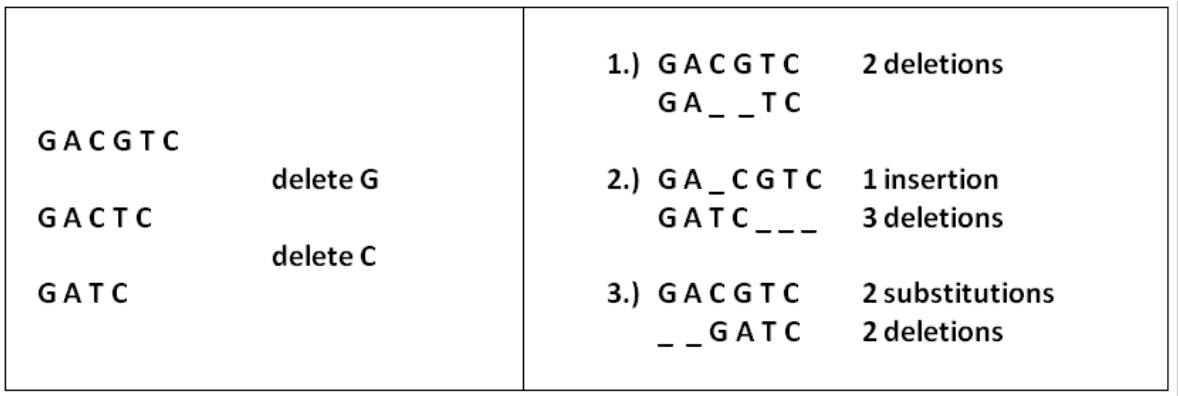

Figure 4.2: Left: example of listing, right: example of alignments [THK91]

happens in numerous occasions. [THK91] explores some methods with formal definitions and some variations employing these distances.

4.2.2 FASTA/LFAST/FASTP

These algorithms from Pearson and Lipman have been developed for rapid database searching. FASTA and LFASTA are used to search nucleic acid databases, whereas FASTP is used to search for proteins. They are similar to another quite famous sequence alignment algorithm called Wilbur-Lipman algorithm, which has a focus on rapid database searches. [THK91]
The algorithms locate matches and rank the database sequences. They only perform alignments which are highly ranked sequences instead of using the alignment to rank the sequences. At first each sequence is converted to a numeric sequence based on a k-tuple value. For exact matches a lookup table is used. Based on the number of consecutive matches and the distance between, each diagonal is scored. The process happens in $O(M)$, but depends on the value of k. With k rising, the number of k-tuples and thus the time required to locate all matches decreases. The ten highest scoring diagonals are found and restored in the next step of FASTP and FASTA using a scoring matrix.
Protein sequences restoration is often done using an amino acid substitution matrix. (Like the PAM250 matrix that organizes amino acids by biochemical similarity.) A subregion with maximal score is located and called an initial region within each diagonal.
In a third step FASTP ranks the sequences using the single best scoring initial region in each database sequence. FASTA checks whether various initial regions can be joined for producing an optimized alignment of the initial regions. Similar to a gap penalty, a joining penalty is assessed. This optimized alignment's score is the penalties subtracted from the sum of the initial region scores. The sequences are then ranked using this optimized score. Alignments are hence not done before the sequences are ranked, which reduces the time required for comparing each sequence enormously.
In the final step of FASTP and FASTA use a modification of the Needleman-Wunsch and Smith-Waterman algorithm to align the highest scoring sequences. They consider only alignments

within a window about the diagonal of the highest scoring initial region. LFASTA is similar that it uses the same first two steps but saves all diagonal regions with a score above a threshold. From the end of each initial region onwards an alignment optimization is performed in the reverse direction until all scores have become zero. Thereafter an optimization is performed in the forward direction starting at the maximal optimized score. Working backwards from the forward optimal maximum the alignment is traced.
There is also a fourth algorithm, TFASTA, which can be used to compare a protein sequence to a DNA library through translating the DNA sequences in six reading frames while the comparison is done.

4.2.3 Further Related Work

In the past several reviews and surveys [THK91], [CWC92], [FJD85], [Gus93], [Wat84] on sequence alignment, sequence comparison and similarity scoring algorithms have been done. Among others, there are for example methods using Finite State Machines [Smi88] or lattice graphs in [RH83]. Numerous further proposals of algorithms, mostly specifications (focusing on quick database searches) or expansions for particularly special amino acid or proteins have been done. Usually they are based on Needleman-Wunsch, but there are also regex or finite automats based methods. A very short selection of such further methods contains: [OU92], [S$^+$79], [BY91], [MM89], [SSA$^+$09], [WP84] [Ukk85] and [OB84]. Another very famous tool in bioinformatics is the Basic Local Alignment Tool (Blast), which also has several variants. [AGM$^+$90] [KYB03] [MZMN07] [Spo89] [KR08] [AMS$^+$97]
There has also been a good deal of literature dealing with metrics for defining distances to assign scores, such as [WSB76], [Alt93], [Sel74a] [Hen96], [Lev66], [Hir97] and [Sel74b].

Since spelling checkers and corrections (as found in any somewhat more elaborate text editor or office program or browser or email plugin) or other word auto completion tools (as found for instance in any mobile phone or tablet keyboard) originate from bioinformatical sequences comparison, this field may provide even more possible methods. For instance n-gram (usually trigram) similarity measures [AFW83] [Sal89] [ZPZ81] or a token matcher [Por80] can be employed.

4.3 Extensions to Account for Ontology Features

As already elaborated in the beginning of this chapter, action recipes written in OWL have considerable differences to bioinformatical string sequences containing atomic elements which use methods whose focus are mainly on processing huge amounts of DNA information quickly and efficiently.
Furthermore, as discussed in chapter 1 action sequences with similar portions of subactions can but do not necessarily perform semantically the same result action. A local alignment can align the similar subactions, ignoring the global context. Naturally an adaptation would also be needed to deal with this issue, as to avoid yielding similarity values, when a simple

match or mismatch could be decided when looking at it globally. A modification to deal with these more complex non-atomic structures of the actions descriptions is discussed in this section.

4.3.1 Weighting Complex Sequence Elements

Two actions may only differ in the descriptive name assigned to a location or an object acted upon or with. (e.g. "tea" instead of "drink" or "water"). Some of these objects or locations may also be more important than others (e.g. "fetch a glass of Tea" and "fetch a glass of water" will have the "glass" more importantly than its liquid content). A weighted computation of the individual matches will address this problem.
Assume a_i is a type that describes an action act_i along with a set of n role-value pairs:

$$act_i = \langle a_i, \langle p_i^1, o_i^1 \rangle, ... \langle p_i^n, o_i^n \rangle \rangle \tag{4.6}$$

The similarity of two actions is computed through the weighted similarity of the action types plus the weighted sum of the similarities between the respective roles p_i^k and values o_i^k (which are for instance objects and locations):

$$sim(act_i, act_j) = \alpha \cdot sim(a_i, a_j) + \sum_{k=1}^{n} \cdot \beta_k sim(p_i^k, p_j^k) + \gamma_k \cdot sim(o_i^k, o_j^k) \tag{4.7}$$

α, β_k and γ_k serve as weighting parameters to determine the influence of the individual components. For example, the role p_i^k are each often correlated with the type of the action a_i^k (e.g. "take" and "from"). They should hence have less influence. The similarity results are then normalized to the range [-1, 1], as to be compatible with the costs for match (1) and mismatch (-1) case. [TZB12]

4.3.2 Semantic Similarity Between Two Concepts

Semantic similarities take the depth of classes into account. So if one action happens to have the same classes, or even be a subset of the action to be compared to, this will affect the ultimate similarity score. Also if two classes derive from the same super class, for instance a general "Translation-LocationChange" class, they will share more similarities than with another class derived from for example a "GraspingSomething" super class.
Ngan, Hang and Goh proposed in [NHG06] a method to account for class matches, as did Tenorth, Ziegltrum and Beetz in [TZB12]. They shall be presented both.

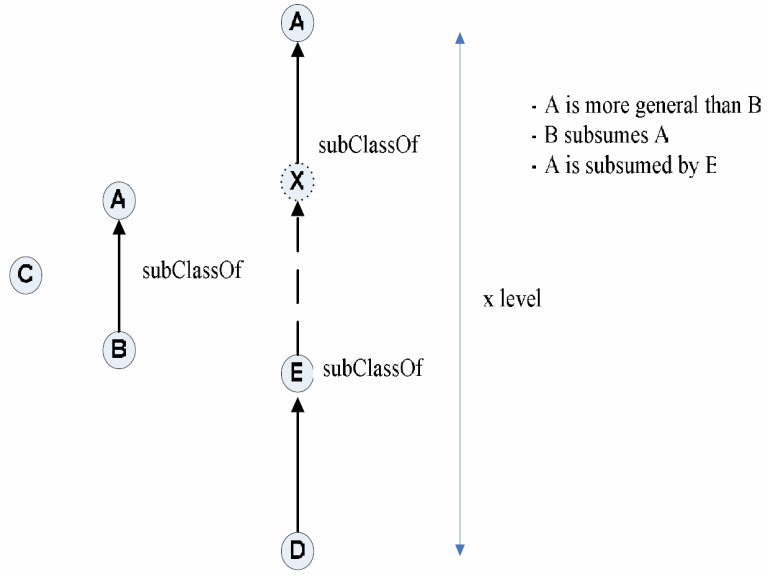

Figure 4.3: Possible relations of concepts (classes) [NHG06]

4.3.2.1 Six Degrees of Matching for Semantic Similarity

Ngan, Hang and Goh [NHG06] define a class in an ontology as a "concept". Such a class usually has a name, perhaps a short description and zero or more properties. These properties describe relationships. An object or individual may have a relationship with another object or individual, but can also have a relationship to a data value by having it as a property. Such properties may have sub or super properties. Properties have a name and description and they limit and restrict the values of an individual or object. OWL also has a vocabulary for describing properties and classes, relations between classes (e.g. disjointness), cardinality (e.g. "exactly one"), equality, richer typing of properties and characteristics of properties (e.g. symmetry), as well as enumerated classes.

Assume that A, B, C, D, E, and X are concepts (classes) of an ontology. Figure 4.3 describes the possible relationships between these concepts.

There are six degrees of matching between two concepts within the same ontology:

- Exact match (A, A): is a most accurate match with the highest similarity that is to happen when the two descriptions are semantically equivalent. The similarity degree for this match is given a value 1.
- Subsumes match (A, B): The case when B is subsumed by A. A being a direct super class of B, makes it more general than B. This match is less accurate than exact match, so the degree for this match is given a value of 0.75.

- General match (A, D): A is more general than D but A is not a direct super class of D. Assuming x levels between A and D, the similarity degree for this match is assigned a $0.3 + 0.3 * 2^{x-2}$.
- Invert-Subsumes match (B, A): B is more specific than A and B is a direct sub class of A. The similarity degree for this match is 0.3.
- Specific match (D, A): The inverse case of the general match. A is more general than D but A is not a direct super class of D. Assuming x levels between A and D, the similarity degree for this match is assigned a $0.3 + 0.3 * 2^{x-2}$.
- Fail match (A, C): A and C are not related in the ontology. The similarity degree for this match is 0. The pair (B, C) in figure 4.3 also accounts to a fail match.

4.3.2.2 WUP similarity for Semantic Similarity

Tenorth, Ziegltrum and Beetz in [TZB12] originally motivated the use of the "WUP similarity" proposed by Wu and Palmer [WP94] in 1994. Two actions may act upon the same or very similar object, but only differ slightly in what level of details or method to employ the action (e.g. "stir" or "whip" eggs, the eggs being either white or brown). This case should not be counted as a complete mismatch. The "WUP similarity" yields a similarity score that is computed based on the closeness of the respective concepts i.e. classes in the action ontology. The similarity of two concepts in an ontology is defined in the interval [0; 1]. The depth of the concepts and the depth of their lowest common super concept (LCS) is being taken into account. Formally the WUP similarity is defined as:

$$wupSim(C_1, C_2) = \frac{depth(LCS(C_1, C_2))}{\frac{1}{2}(depth(C_1) + depth(C_2))} \tag{4.8}$$

and the reflexive case

$$wupSim(C, C) = 1 \tag{4.9}$$

Figure 4.4 gives a simply example on a snippet of a common action recipe ontology. Concepts in an ontology can have multiple super-concepts. Typically the minimum of all these distances are computed by the WUP similarity since it considers the lowest common super-concept (LCS). With this the ontology's structure will contribute to the similarity values. In [TZB12] experiments have been done with the KNOWROB ontology and claimed to have yielded very reasonable results. They used the *wupSim* to the power of three to increase the weight of higher similarities upon computing the alignment costs.

The WUP similarity thus gives another computation method to compute semantic similarity based class derivation depth and super concept similarity. It basically addresses the same issues as Ngan, Hang and Goh's six degrees of concept matching in their CS algorithm, as

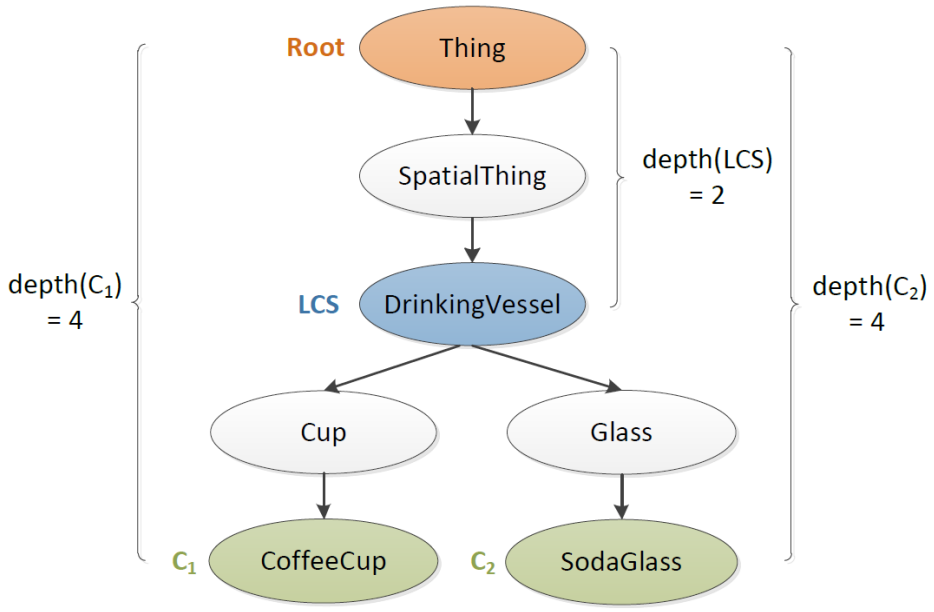

Figure 4.4: Computation of the WUP similarity metric. The similarity of the concepts $wupSim$(CoffeeCup and SodaGlass) is computed as 0.5. [TZB12]

presented in 4.3.2.1. While these six degrees account for more explicit cases, the computation of the $wupSim$ seems still slightly preferable, as it is more simple and does not use as many exponential calculations and would probably yield results faster. On the other hand, given the current size of the RoboEarth platform's database [rob], such speed considerations may yet be disproportionate. In fact, when speed is of no major issue, both methods could also be combined and an average used in the final weighting of all similarity components.

4.3.3 WordNet for Syntactic Similarity

A concept in an ontology has a label (concept name) and a short descriptive text (concept description). A syntactic similarity computation compares the similarity between two concept names and each their descriptions. Words or a set of words define a concept name in OWL. String similarity computations as described in the first half of the chapter can be used, but these are limited to strings of characters in the word and their semantic relationship is ignored. For English language WordNet [Mil95] [Mil98] can be used to overcome this issue. WordNet is a large lexical database of English vocabulary. Word types such as nouns, verbs, adjectives and adverbs are grouped into sets of synonyms (synsets) that express about the same semantic concept. These synsets are interlinked through conceptual-semantic and lexical relations. WordNet stores a root word in its database. To use WordNet for comparison, the words need to be preprocessed by finding their roots. In case both concept names consist of just a single word, WordNet can be used directly for similarity comparison. In case of multiple words

the most similar word pair is computed. In case WordNet fails to have the word pair in its database, common string comparison techniques like those presented previously can be used. Ngan, Hang and Goh proposed the usage of Jaro. Jaro is a method to compare two strings by using insertions, deletions and transpositions processes from the field of record linkage (also referred to as information retrieval) and uses a bigram computation method. [PW+97]
The same process for non-single word comparison is also applied on descriptions of two concepts. The final syntactic similarity value is a weighted average of concept names and concept descriptions similarity.

This method also addresses the potential trouble of homonyms. For instance, the word "bow" can either be a weapon or a ribbon and thus not be in the same synset, while a mere string comparison of the labels would yield a full score. WordNet also allows adding own vocabulary to the existing synsets. [NLH12]. This is especially interesting, as the user can define synonyms usually not used in a common lexicographical language database. For instance, "water" or "tea" or a list of other drinks, including even copyrighted brands (which would never appear in a dictionary) can be defined under a synset with "drink" as a root word. This way, the system could even include some sort of self learning. If e.g. two action recipes "fetch a glass of water" and "fetch a glass of tea" are almost identical with the drink types being the only difference and thus yield a very high final similarity score, due to the other similarity score components, "tea" and "water" could likely be grouped into a synset.

4.3.4 Property Similarity

A concept may have one or more properties. Similar to the concept itself, a property has a name and description as well. On top of that it contains a range and cardinality. Reasonably all the information of the two properties in question should be matched for the property similarity computation. Property name and description similarity are essentially computed the same way as the syntactic similarity computation. The range of a property is either a primitive data type or another concept. In case both ranges are primitive data types, the similarity between two ranges is defined as shown in the table of figure 4.5. A similarity of 0 is assigned for the obviously incompatible case of one range having a primitive type and the other having a concept as property. If both range properties are concepts the matching will be carried out recursively with the full algorithm containing the five components.

A property's cardinality specifies the exact number of elements in a relation. Ngan, Hang and Goh simply define them as:

- Cardinality similarity = 1: the cardinality of two properties are equal
- Cardinality similarity = 0: the cardinality of two properties are different

Hence the property similarity is a combination of syntactic, range, and cardinality similarity.

		Range of Property 1				
		Integer	Long	Float	Decimal	String
Range of Property 2	Integer	1	0.9	0.8	0.7	0.3
	Long	0.9	1	0.8	0.7	0.3
	Float	0.8	0.8	1	0.7	0.3
	Decimal	0.7	0.7	0.7	1	0.3
	String	0.3	0.3	0.3	0.3	1

Figure 4.5: Similarity values for ranges with primitive data types [NHG06]

4.3.5 Context Similarity

Also known as a domain of the ontology, context is very important. Take an easy example: An ontology about pets may contain the concept (class) "rabbit" and another ontology about food also contains a "rabbit" concept. Both of them would be the same, but obviously a "rabbit" as a pet and a "rabbit" as an ingredient for food are entirely different and thus should have a low similarity rate. It would therefore not make sense to have them match with a value 1, which they would, if the context is ignored.
For context similarity, the similarity of roots of the two ontologies is computed. The root represents the context (domain) of the ontologies and is a special concept in an ontology, which does not have super concepts.
This is also implicitly contained in the WUP-Similarity from section 4.3.2.2, since the depth of the lowest common super concept (LCS) would the the root class at depth 0, thus making the whole similarity 0.
There are also other approaches to include the ontology's context structure into the similarity computation. In the OLA ontology alignment method to be discussed in chapter 5 takes this even further, a graph is built in accordance to the ontology structure and similarities are computed upon the edges of the graph, which represent the relationship between different entities of the ontology, i.e. their context.

4.3.6 Neighbourhood Similarity

Neighbourhood similarity takes, as the name already indicates, the neighbourhood of the two concepts (classes) in question to account. To be more precise the neighbourhood consists of their super concepts and sub concepts. Figure 4.6 illustrates a matching of two concepts

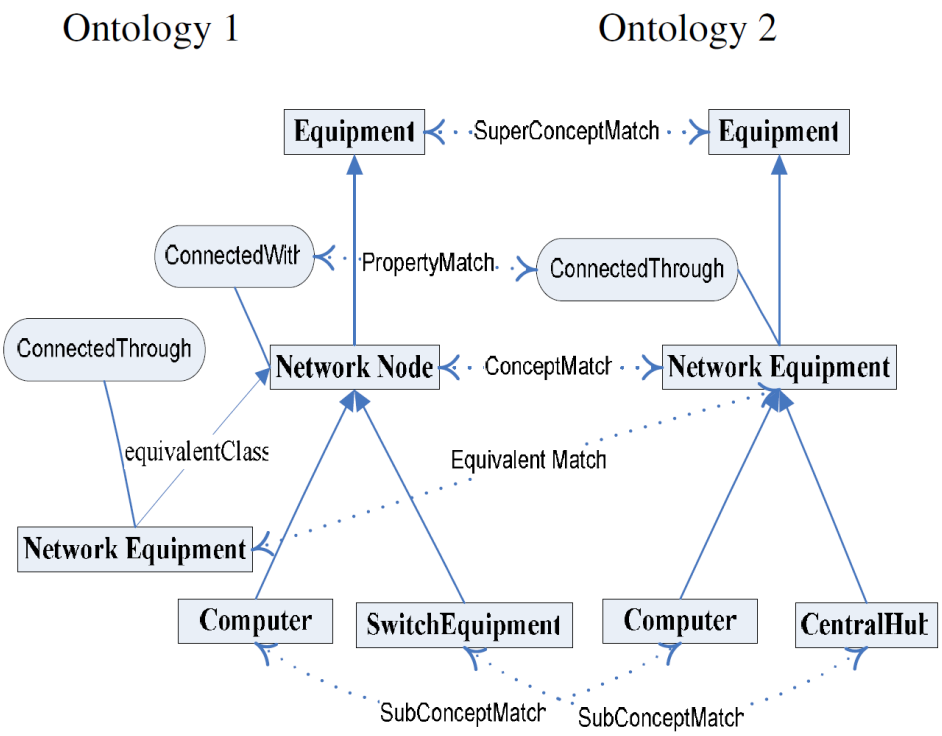

Figure 4.6: Comparison of two NetworkNode concepts (classes) [NHG06]

Network Node and Network Element. Both concepts have as super concepts Equipment; their sub concepts are Computer, Switch Equipment, and Computer, Central Hub, respectively.

The neighbourhood similarity is calculated through:

$$\texttt{neighborSim} = \sqrt{\texttt{supSim * subSim}} \tag{4.10}$$

with `supSim` being super concept similarity and `subSim` being sub concept similarity.

A concept may also have more than one super concepts. To compute the super concept similarity the following equation is used:

$$supSim(C_P, C_R) = \frac{allSupSIM(C_P, C_R)}{n} \tag{4.11}$$

with n being the number of matches super concepts pairs and C_P and C_R being concepts from the two different ontologies. $allSupSim$ is the similarity of the matched super concepts. As equation 4.12 illustrates, the super concepts are matched one to one, so the average super concept match is maximized. A recursive formula is used for this purpose:

$$\begin{aligned} allSupSim(CP, CR) = Max(allSupSim(CP - SupP, CR - SupR)) \\ + conceptSimilarity(SupP, SupR)) \end{aligned} \tag{4.12}$$

That way, multiple super concepts are all taken into account. The computation of subSim is calculated the same way as supSim.

4.3.7 Equivalent Concepts Similarity

In OWL, *sameConceptAs* and *equivalentClass* relationships indicate that the concepts in question have the same meaning. Since these relationships already exist, it is expedient to consider these as well. The "equivalent relationship" named as such by Ngan, Hang and Goh [NHG06] encompasses these two relationships. Taking figure 4.6 as an example again, the concepts Network Node and Network Element shall have a similarity comparison done on. The value of the two concepts which is computed by the combination of four components is 0.89. But the ontology 1 defines the Network Node concept as having equivalentClass relationship with Network Element concept. The Network Element concept in ontology 1 and Network Element concept in ontology 2 are exactly the same; so the similarity of two Network Element concepts from ontology 1 and 2 gets the value 1. With this explicit equivalent statement, the similarity of Network Node and Network Element from the two ontologies should be 1 instead of 0.89.
More formally, assume that similarity between two concepts A and B are to be computed. A has equivalent relationship with C, and B has equivalent relationship with D. First the concept similarity between: A-B, A-D, B-C, and C-D are computed. Assume that the similarity values are SA-B, SA-D, SB-C, and SC-D, respectively. The similarity between A and B is the maximum value of SA-B, SA-D, SB-C, and SC-D. The method to compute the similarity between A-B, A-D, B-C, and C-D is based on the four components described above.

4.3.8 Weighting Similarities Between Two Different Ontologies

Going further to a complete comparison of two concepts of different ontologies, Ngan, Hang and Goh presented the concept similarity (CS) algorithm. The algorithm includes five main components: syntactic similarity, properties similarity, context similarity, neighbourhood similarity, and equivalent concept similarity. When the two concepts have equivalent relationship, the equivalent concept similarity is used. For all the other cases the final matching is computed through the average of the sum of the four components: syntactic, properties, context, and neighbourhood similarity. Formally:

$$CS = \frac{w_s * synSim + w_p * proSim + w_c * conSim + w_n * neiSim}{w_s + w_p + w_c + w_n} \tag{4.13}$$

with w_s, w_p, w_c, w_n being user-defined weights.

4.4 Summary

In the first half of this chapter an overview of sequence alignment in the bioinformatical field was given, along with the presentation of some most common methods in more detail,

followed by an overview of related work in the sequence alignment field. The second half of the chapter introduced several methods to account for ontology class characteristics not to be found in bioinformatical sequences. Complex sequence elements can now be weighted upon their importance, a semantic similarity between two concepts detected through both semantic similarity of the descriptive labels or common super concepts. Property, context and neighbourhood similarity is also taken into account as is equivalence of two concepts. Finally all of these similarity scores are weighted and contribute to a final similarity score of two concepts.

5 Ontology Alignment with OWL-Lite Alignment (OLA)

The previous chapter addressed an alignment method on the basis of [TZB12] as proposed by Tenorth, Ziegltrum and Beetz as well as [NHG06] by Ngan, Hang and Goh. The focus of the discussion of the first half was more on the underlying string comparison algorithms and in the second half on comparisons of concepts i.e. classes of the ontology. The discussion about the necessary extensions to expand the bioinformatical string alignment methods already elaborated on why those methods can not be adopted one to one and need to account for the complex structure of ontologies. This chapter will focus on a whole different approach of ontology alignment, introduced by Jérôme Euzenat (INRIA Rhône-Alpes) and Petko Valtchev (Université de Montréal) in 2003-2004. [EV03]
OLA has also participated in the OAEI workshop of 2007 [þES07].

5.1 General Overview

Ontologies were introduced in the Semantic Web to avoid heterogeneous information representation of resources that are necessarily distributed and heterogeneous. They serve as a common approach for homogeneous data description to lessen the work on integrating the resources. The problem that springs forth from this approach is the heterogeneity of the underlying ontologies themselves. The first chapter already illustrated with some examples the problems of separate ontologies with generally the same content with RoboEarth action recipes as a use case. Ontology conformity generally contributes to semantic interoperability.

The methods the previous chapter discussed are basically string-based with extension to structural comparison of classes. There are many other approaches, such as some being rooted in the classical problem of schema matching or being graph based for structural comparison. (Please refer to chapter 3 for an overview of related work). These methods essentially use partial entities of the ontology and conduct a pair-wise comparison and pick the most similar pair out. The OLA method relies on the classical similarity-based paradigm for entity comparison and is supposed to take advantage of not just some entities, but the ontology as a whole. It serves as a common framework, into which many ontology comparison methods are integrated. To add on those, a further technique is added to account for one-to-many relations and circularities in the similarity definitions.

Jérôme Euzenat and Petko Valtchev [EV03], [EV04] describe this 'ontology alignment' problem as follows: If given two ontologies, which each describe a set of discrete entities such as classes, properties, rules, predicates, discover the relationships between the entities (e.g. equivalence or subsumption). Alignment results can serve different purposes such as displaying the similarities, creating a set of bridge axioms between the ontologies or transforming one source into another. Generally the ontology alignment method focuses on automatic and autonomous alignment, but more interactive scenarios can be built over it, such as using the result as a suggestion to the user (which would fill the use case of the specific problem addressed in this work) or just completing a partial alignment. The method also addresses overcoming circularities and existence of external data types.

The general functional specifications of OLA offer these services: [EDLV04]

- parsing and visualization of (pairs of) ontologies,
- automated computation of similarities between entities from different ontologies,
- automated extraction of alignments from a pair of ontologies,
- manual construction of alignments,
- initialization of automated alignment construction by an existing alignment,
- visualization of alignments,
- comparison of alignments.

In particular they focus on these characteristics: Universality, automation, uniform comparison and comparability of similarity results. Given ontologies, all knowledge about their entities should be used and the alignment should be done automatically. No user interventions are required in intermediate steps, albeit the alignment process may be done on a semi-automated basis. The user can set parameters and the tool may ask for better ones at the end and automation is also used of learning optimal parameters. The ontology's entities are divided into categories and only entities in the same category are to be compared using the same similarity function on the same space. This ensures that the same similarity considerations are used for each pair of a given category and their contribution to the overall similarity depends only on their category. The similarity measure's values are also normalized, so that a comparison can be done between different alignment tasks. This is done within the iterative computation process through an appropriate function definition.

5.2 OWL-Lite

One peculiarity of this method is that it limits the comparison of ontologies on those described and built in the knowledge representation language OWL-Lite. OWL-Lite is one of the three major sub languages of the OWL description language, while taking advantage of most of the language's features. [MVH+04]

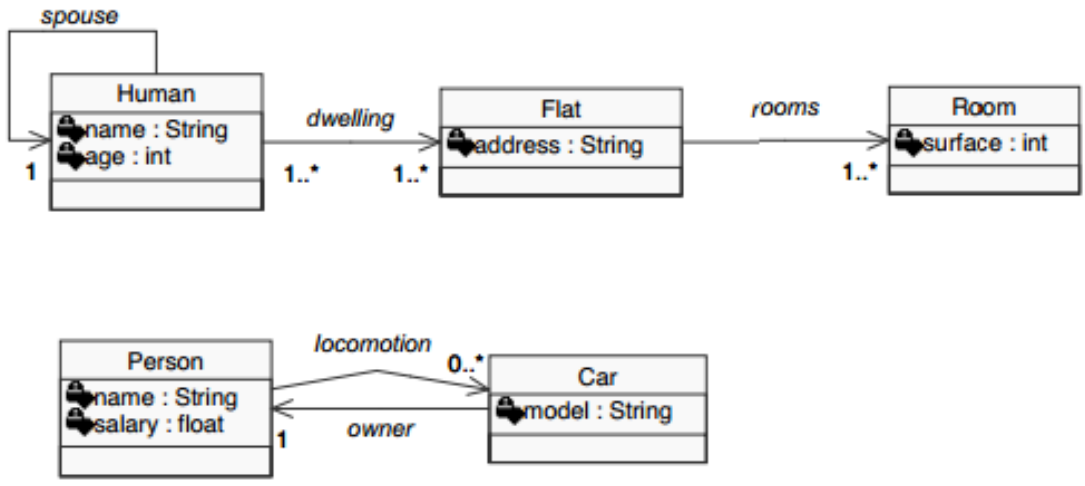

Figure 5.1: Classes of two example ontologies drawn as UML class diagrams [EV04]

OWL-Lite is also based on various features such as classes and subsumption, properties and type constraints, etc. and primarily supports the need of a classification hierarchy and simple constraints. For instance and contrast to its relatives it supports cardinality constraints, but only permits cardinality values of 0 or 1. OWL-Lite can also be migrated quickly to the more expressive derivatives of the OWL mother language. Some constructs of OWL-DL are also already supported and full coverage of the OWL-DL language is planned as a long term goal.

5.3 Background Work Used Within Ontology Alignment

Aside from the method elaborated on in the previous chapter, various other methods exists, some of which are integrated into this ontology method, among which are discrete mathematics for matching graphs and trees [HK73], databases for reconciling and merging schemas [RB01] and machine learning for clustering compound objects described in a restricted First Order Logic [Bis92]. Aligning consists itself here as defining a distance between a pair of entities and finding the minimal distance (the best match) among them. As the quantity of such alignment algorithms suggest, there is multiple different ways to compute such a distance. Jérôme Euzenat and Petko Valtchev [EV03] classify these methods into five groups:

- **terminological (T)** methods compare the entities' labels. This presents itself in two flavours: The **string-based (TS)** and the **terminological with lexicons (TL)**. The first computes the string structure (dis)similarity like editing distance. Please refer to the previously chapter for an introduction of some sample methods. The latter matches

terminologically modulo the relationships to be found in a lexicon (i.e. take a synonym as equivalent or a hyponym as subsumed into consideration)

- **internal structure comparison (I)** methods contemplates on the internal structure of entities such as their value range or cardinality of attributes
- **external structure comparison (S)** methods compare relations between entities. This also comes in two flavours: **taxonomical structure (ST)** methods are about the position of the entities within a taxonomy, while **external structure comparison with cycles (SC)** are robust to cycles.
- **extensional comparison (E)** compares known extensions of entities, i.e. the set of other entities that are attached to them (in general instances of classes)
- **semantic comparison (M)** compares the models of the entities

5.4 Representation as a Graph

The method does not compute the similarity on base of the OWL-Lite Syntax, but defines a dedicates typed graph representation of the language and compute upon the graph based syntax. This built graph will have classes (C), objects (O), relations (R), properties (P), property instances (A), data types (D), data values (V), property restriction labels (L) as node categories, concentrating the necessary information for computing the similarity between OWL entities. Also differences are made for data type relations (R_dt), object relations (R_o) and property restriction labels (P_dt) Among those nodes, different sorts of relationships are possible according to [EV03]:

- **specialization:** `rdfs:subClassOf` **($\mathcal{S}$)** between two classes or two properties,
- **instantiation: rdf:type ($\mathcal{I}$)** between objects and classes, property instances and properties, values and data types;
- **attribution: ($\mathcal{A}$)** between classes and properties, objects and property instances;
- **restriction: owl:Restriction ($\mathcal{R}$)** expressing the restriction on a property in a class;
- **valuation ($\mathcal{U}$)** of a property in an individual

The relation symbols are used as set-valued functions ($\mathcal{F}(x) = x; \exists y; \langle x, y \rangle 2 \in \mathcal{F}$). Furthermore every node can become attached annotations via an URI reference identified by $\lambda : C \cup O \cup R \cup P \cup D \cup A \rightarrow URIRef$. Knowledge encoded in relation types should also be on the object level as to provide a most complete basis on the comparison. One way to do this is to insert an additional edge between objects, whose edge or a pair of edges is reverse, symmetric or transitive. Such relation types can be handled either by saturation of the graph or these simple methods: for

- **owl:TransitiveProperty** add transitivity arcs
- **owl:SymmetricProperty** add symmetric arcs

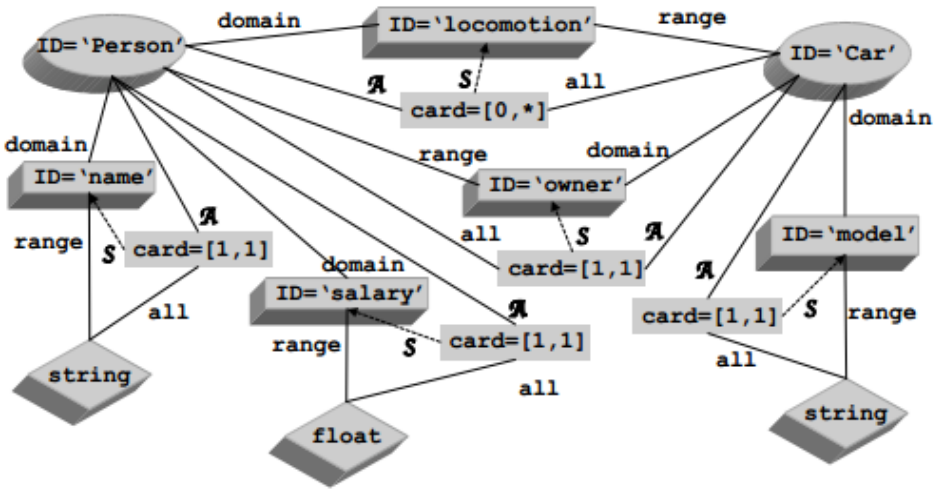

Figure 5.2: The OL-graph of the second set of classes from figure 5.1 [EV04]

- **owl:inverseOf** add the reverse arcs (both in generic and individual descriptions)
- **owl:FunctionnalProperty** add a cardinality constraint
- `owl:InverseFunctionnalProperty` will not be accounted for at that stage

Figure 5.1 illustrates such a graph.

[EV04] claims, that the relationships between language elements get more explicit through the graph structure. For instance, if a class c refers to c' `owl:allValuesFrom restriction`, this will also show in the OL-Graph via a path between the corresponding nodes. Also can OL-graphs record other information possibly supporting comparison, such as descriptive knowledge inherited from nodes of the same or related categories. So the similarity of an entity pair will depend on their neighbour pairs, whose members describe the initial entities.
It should also be noted however, that only descriptive knowledge is taken into account, due to efficiency considerations. So no inference is done on the ontologies and inheritance in particular to expand descriptions of entities. [EDLV04]
OLA uses an extension of the JGraph API [jgr] to perform a graph representation of the ontologies. [olaa]

5.5 Similarity: Measurement and Computation

As alignment boils down to finding the best correspondence between of two ontologies' entities a definition of similarity between a pair of entities is required and elaborated on in section 5.5.1. Processing relationships are thus essential. Accompanying them are also their "surrounding" entities, which should be taken into account as well. To avoid trouble with circularity an effective computation mechanism is needed, which will be introduced in section 5.5.3.

5.5.1 Integrative Similarity Measure

The OWL-Lite file will be represented graphically. The different entity categories of the ontology are highlighted and linked. In a dedicated similarity measure for each node category in the OL-Graph, entities from one ontology will be mapped to the most similar ones of the other one. The measure will rank the pair of entities in a real number in the range of $[0, 1]$, whereby 0 (1) signifies entirely different (similar) entities. There are two basic key assumptions:

- any component of an entity category has a priori relevance to the similarity evaluation. The relative relevance can be adjusted by weighting them among each other. Section 5.5.3 will go deeper on that.
- any entity within each category is processed in the same manner, while computation methods for different categories may differ

Thus given two nodes from category X, the similarity depends on:

- similarity of designators (labels, Names, URIs, etc.)
- similarity of pairs of neighbour nodes in the respective OL-Graphs liked by edges with the same relationship (e.g. similarity of a class node depends on similarity of super classes, property restrictions and member objects)
- similarity of other category specific local descriptive features (e.g. cardinality intervals, property types)

In short, an entity category such as a class is assigned a specific measure. Related entity categories like a property, a sub-class, etc. are computed with their respective measures and a function on all the results make up the specific entity category measure. For simplicity's sake, the different components in this function are aggregates through a weighted sum, in which the importance of each component can be emphasized or even ignored by reducing the weight down to zero.

For any given category X with $\mathcal{N}(X)$ the set of its relationships, the similarity measure $Sim_X : X^2 \to [0, 1]$ is formally defined as follows: [EDLV04]

$$Sim_X(x, x') = \sum_{\mathcal{F} \in \mathcal{N}(X)} \pi_{\mathcal{F}}^X MSim_Y(\mathcal{F}(x)\mathcal{F}(x') \tag{5.1}$$

and the function is normalized with:

$$1 = \sum_{\mathcal{F} \in \mathcal{N}(X)} \pi_{\mathcal{F}}^X \tag{5.2}$$

So for example in case of two classes c, c' this weighted normalized similarity function would be computed as follows: [EV04]

$$
\begin{aligned}
Sim_C(c, c') = \pi^C_L sim_L(\lambda(c), \lambda(c')) \\
+ \pi^C_{\mathcal{I}} Sim_O(\mathcal{I}(c)\mathcal{I}'(c')) \\
+ \pi^C_{\mathcal{S}} Sim_C(\mathcal{S}(c)\mathcal{S}'(c')) \\
+ \pi^C_{\mathcal{A}_{dt}} Sim_P(\mathcal{A}_{dt}(c)\mathcal{A}'_{dt}(c')) \\
+ \pi^C_{\mathcal{A}_O} Sim_P(\mathcal{A}_O(c)\mathcal{A}'_O(c'))
\end{aligned} \tag{5.3}
$$

with its similarity scores calculated for the nodes of the graph: classes (C), objects (O) and properties (P).

And due normalization:

$$1 \quad = \quad \pi^C_L + \pi^C_{\mathcal{I}} + \pi^C_{\mathcal{S}} + \pi^C_{\mathcal{A}_{dt}} + \pi^C_{\mathcal{A}_O} \tag{5.4}$$

So for instance, in case of the two ontologies in figure 5.1, the similarity of `Flat` and `Person` (see also figure 5.2) would be: [EV04]

$$
\begin{aligned}
Sim_C(\texttt{Flat, Person}) = \pi^C_L sim_L(\texttt{'flat'}, \texttt{'person'}) \\
+ \pi^C_{\mathcal{A}_{dt}} MSim_P(\{rooms\}, \{locomotion\}) \\
+ \pi^C_{\mathcal{A}_O} MSim_P(\{address\}, \{name, salary\})
\end{aligned} \tag{5.5}
$$

with $\pi^C_{\mathcal{A}dt}$ and $\pi^C_{\mathcal{A}_O}$ being components of $\pi^C_{\mathcal{A}}$ assigned to data type and object property parts of Sim_C(`Flat, Person`) respectively. Any missing $\pi^C_{\mathcal{F}}$ means that $\mathcal{F}$ contribution are ignored in the measure computation.

The Table in figure 5.3 shows the other function decompositions in detail.

5.5.2 Similarity-based Matching of Entity Sets

The generic set similarity function $MSim$ averages the components' similarities, i.e. two sets of nodes in the same category are measured for similarity by combining all pair similarities into one unique value. ($MSim$ thus averages a limited subset of the product $S_1 \times S_2$ that represents a matching optimizing the total similarity. [Val99]) The function ensures equity between the factor of function $Sim_X(n_1, n_2)$. $MSim$ is formally defined as:

$$MSim_C(S, S') \quad = \quad \frac{\sum_{\langle c,c' \rangle \in Pairing(S,S')} Sim_C(c, c')}{max(|S|, |S'|)} \tag{5.6}$$

with S and S' being two sets of entities of the same category Y and the respective measure Sim_Y. $Pairing(S, S')$ is the mapping of elements S to elements S', that maximizes the $MSim_C$ similarity. Therefore the average of the matched pair's values is the similarity between

Function	Node	Factor	Measure
Sim_O	$o \in O$	$\lambda(o)$	sim_L
		$a \in A, (o,a) \in \mathcal{A}$	$MSim_A$
Sim_A	$a \in A$	$r \in R, (a,r) \in \mathcal{R}$	Sim_R
		$b \in O \cup V$	$MSim_V / MSim_O$
Sim_V	$v \in V$	value literal	type dependent
Sim_C	$c \in C$	$\lambda(c)$	sim_L
		$p \in P, (c,p) \in \mathcal{R}$	$MSim_P$
		$c' \in C, (c,c') \in \mathcal{S}$	$MSim_C$
sim_D	$d \in D$	$\lambda(r)$	XML-Schema specific
Sim_R	$r \in R$	$\lambda(r)$	sim_L
		$c \in C, (r, \texttt{domain}, c) \in \mathcal{R}$	$MSim_C$
		$c \in C, (r, \texttt{range}, c) \in \mathcal{R}$	$MSim_C$
		$d \in D, (r, \texttt{range}, d) \in \mathcal{R}$	Sim_D
		$r' \in R, (r,r') \in \mathcal{S}$	$MSim_R$
Sim_P	$p \in P$	$r \in R, (p,r') \in \mathcal{S}$	Sim_R
		$c \in C, (p, \texttt{allValuesFrom}, c) \in \mathcal{R}$	$MSim_C$
		$n \in \{0,1,+\infty\}, (p, \texttt{cardinality}, n) \in \mathcal{R}$	equality

Figure 5.3: Decomposition of the different similarity functions (`card = cardinality` and `all = allValuesFrom`)

the two sets.
For illustration of the function [EV04], using the previous example calculation of Sim_C(`Flat, Person`) of equation 5.5 and assuming that Sim_P(`address, name`) = 0.64 and Sim_P(`address, salary`) = 0.34. This will have {(`address, salary`)} be the best matching. Hence

$$MSim_P(address, \{name, salary\}) = 0.64/2 = 0.32 \tag{5.7}$$

The eventual similarity values will depend on the similarities between data types, values and URIRef as well as how these are translated through the relationships in the graphs. An abstract data type definition should accompany measurements of data types and values. URIRef comparison can be conducted by a string similarity or by an equality predicate.

5.5.3 Similarity Computation

One problem that is to be avoided are cyclic dependencies in the similarity weight. The above formula $Sim_C(c, c')$ itself depends on the computation of a computation of Sim_C of other classes. This could lead to a simple deadlock as $Sim_C(c_1, c_2)$ depend on $Sim_C(c_3, c_4)$, while the latter depends on the first again. Therefore, similarities should not be recursive, but only be expressed as equations. For this, a variable is introduced, that corresponds to the similarity

of an entity pair. A system of equations is formed by substituting all similarity occurrences with the corresponding variables:

$$\begin{cases} x_{1,1} = Sim_C(c_1, c_1) & y_{1,1} = Sim_C(p_1, p'_1) \\ x_{1,2} = Sim_C(c_1, c'_2) & y_{1,2} = Sim_C(p_1, p'_2) \\ ... \end{cases} \tag{5.8}$$

When the program is given some similarity values, some similarity or dissimilarity assertions as an input, the assertion of the similarity between the objects can replace the corresponding equation. In case of only one entity being compared to another, each $MSim$ would be deterministic and the system be directly solvable, as all the variables would be only degree one. This would be the easiest case, but is not very realistic, since the system is not very likely to be linear, as multiple candidate pairs for a best match are possible in OWL-Lite. Such a system can still be computed iteratively by simulating computation of the fixed point of a vector function (see [Bis92] for details and proof). Essentially, the $MSim$-measures are first being approximated. The system is then solved and the approximations replaced by the newer solutions computed in the iteration and the process is repeated. The initial values approximating the $MSim$-measures are the maximal similarity found for a pair, while ignoring the dependency of the equation. The values following are computed with the complete similarity formula and the system's computed solutions as an input. The system always converges eventually. The function consists of a steady part, which does not change in any iteration. The dependencies can therefore only propagate their own increase, but the similarity function is normalized and thus has an upper bound 1, which no value can exceed, because none of their components can (inductively). The iteration stops, when the values does not increase by more than an ϵ value compared to the previous iteration's value. In theory there may be a different global solution, which this iteration may miss, as it will get stuck on a local optimum. To avoid this, some matchings could be randomly changed. The result value will approximate the similarity between entities from opposite ontologies. These similarity values will be used as a basis for ranking entity pairs for the final goal of a satisfactory ontology mapping.

5.5.3.1 Example

[EV04] gives an computation example to illustrate this method:
Using the two ontologies from figure 5.1 and figure 5.2, (`Flat, Person`) and (`rooms, locomotion`) are two pairs of nodes. It may easily happen, that they are each other's contributor as is the case which these two nodes, resulting in the recursive dependency of Sim_C(`Flat, Person`) being dependent on Sim_P(`rooms, locomotion`) and vice versa. As [Bis92] shows, an equation system can be built to account for the need of a non standard computation method addressing the cyclic dependency problem. Therein, each pair of "alignable" nodes, i.e., in C (classes), R (relations) and, possibly, O (objects), is assigned a variable x_i, y_j, and z_k, respectively, representing each similarity. Using the similarity as described in section

5.5.1 and the guidelines listed in the table of figure 5.3, the contributor similarities are replaced by the corresponding variables. For the sample purposes, take Sim_C(`Flat, Person`) using the following weights for the C (classes) and P (properties) categories:

π^C_L	$\pi^C_{\mathcal{I}}$	$\pi^C_{\mathcal{S}}$	$\pi^C_{\mathcal{A}_O}$	$\pi^C_{\mathcal{A}_d}$	π^P_L	π^P_{card}	$\pi^P_{\mathcal{A}}$	$\pi^P_{\mathcal{R}}$
.4	0.	.1	.25	.25	.25	.15	.4	.2

(The categories p and R are merged due to space limitations in the similarity computation. But the only difference is that property name and domain are directly included in Sim_P instead of having an influence via Sim_R.)

The variable substitutions are like this:

$$x_1 = Sim_C(\texttt{Flat, Person}) \qquad x_2 = Sim_C(\texttt{Room, Car})$$
$$x_3 = Sim_C(\texttt{Human, Person}) \qquad x_4 = Sim_C(\texttt{Flat, Car})$$
$$x_5 = Sim_C(\texttt{Room, Person}) \qquad x_6 = Sim_C(\texttt{Human, Car})$$
$$x_1 = Sim_R(\texttt{rooms, locomotion}) \qquad y_2 = Sim_R(\texttt{address, name})$$
$$x_3 = Sim_R(\texttt{address, salary}) \qquad y_4 = Sim_R(\texttt{surface, model})$$

In the following example calculation, identity function is set for data type similarities and cardinality measure is 1 if both limits correspond, 0.5 if only one does ($*$ is ignored), 0.35 if there is no match, but still inclusion, and 0 otherwise. Also assume the (arbitrary chosen) label similarity values are assumed:

$$Sim_L(\texttt{flat, person}) = .4 \qquad Sim_L(\texttt{room, car}) = .5$$
$$Sim_R(\texttt{rooms, locomotion}) = .25 \qquad Sim_R(\texttt{address, name}) = .7$$
$$Sim_R(\texttt{address, salary}) = .3 \qquad Sim_R(\texttt{surface, model}) = .35$$

For composing the equations define $choice(S)$ as a simulation of set matchings underlying S (`rdfs:subClassOf` two properties). In other words, it assigns 0/1 weights to each variable in the set:

$$x1 = .16 + .25 * choice(\{y1\}) + .125 * choice(\{y2, y3\})$$
$$y1 = .115 + .4 * choice(\{x1\}) + .2 * choice(\{x2\})$$
$$x2 = .2 + .125 * choice(\{y4\})$$
$$y2 = .525 + .4 * choice(\{x1\})$$
$$y3 = .225 + .4 * choice(\{x1\})$$
$$y4 = .238 + .4 * choice(\{x2\})$$

The system would be a directly solvable, i.e. a linear system, if each choice could be established beforehand. OWL-Lite however usually produces non-linear systems, so choices can not be established beforehand. In such cases, an iterative process that produces the nearest reachable fixed point of a vector function can be used to produce a sequence of approximations getting more precise each iteration. The initial similarity values are based exclusively on 0-th level contributors. Using the similarities of the 1st-level contributors from step n (including matching re-calculations) the values at step $n + 1$ are computed.
Doing so will yield the following solution for the entire system as corresponding to figure 5.1 after six steps:

$$\begin{array}{lllll} x_1 = .293 & x_3 = .566 & x_5 = .156 & y_1 = .290 & y_3 = .342 \\ x_2 = .288 & x_4 = .492 & x_6 = .370 & y_2 = .642 & y_4 = .353 \end{array}$$

5.5.4 Lexical similarity measures

OLA uses WordNet 2.0 [Mil95] to compare identifiers. WordNet is a large lexical database of English vocabulary. Word types such as nouns, verbs, adjectives and adverbs are grouped into sets of synonyms (synsets) that express about the same semantic concept. These synsets are interlinked through conceptual-semantic and lexical relations. It thus gives a measure of "relatedness" between two terms. This is done by retrieving the synsets for each term and applying a normalized Hamming distance to them. A default similarity value for identifier pairs is established by using a variant of substring distance. With default mechanism identifiers that are not entries in WordNet, e.g., compound identifiers or abbreviations can be processed in a sensible way.

5.5.5 Implementation of OLA

OLA is implemented in Java and relies in the OWL API for parsing OWL files. A complete subsystem deals with constructing the OL-Graphs based on the information from parsing the OWL ontologies. A set of further components offer similarity computation services such as substring distances, edit distances or Hamming distance. Another component specially extracts similarity values from the limited WordNet interface provided by the JWNL library [Did04]. Entity set comparison is done by another component. The similarity and matching mechanisms are integrated into the subsystem producing the alignment. This supports the entire iterative computation process.
The implementation in a 2.0.1 Version is available as a complete zipped archive with source codes, libraries and an executable jar file at [olab].

5.5.6 Strength and Weaknesses of the Comparison Method

According to [EDLV04], based on several initial tests, the method performs well, if the structures of the compared ontologies are close to identical. In 2007, at the Ontology Workshop

[ont] where various ontology methods have been tested, among them an improved OLA in version 2, further results on numerous benchmark tests have been published. [DKEV+07] For language variations it returned a mean precision of 1, while alteration of names and/or suppression of comments yield a mean precision of 0.92 and synonyms or foreign languages a value of 0.9. The algorithm relies heavily on similarity of names or paths of entities. It scored poorly on those tests with entities playing similar roles within their ontology graph and shows weaknesses in semantic similarity and due to a lack of language translators and the fact that in version 2 the WordNet engine was removed. According to [DKEV+07] the problem of semantic similarity will be addressed in future improvements. Overall the results are nonetheless satisfying and yielded already considerably better results than its previous version 1.

A major drawback however is the limitation to OWL-Lite by OLA. Action recipes are mostly written in OWL-DL, of which OWL-Lite is a subset. OWL-Lite was built to express the same semantics of OWL-DL, supporting users who primarily need a classification hierarchy with simple constraints. In consequence OWL-Lite is a less complex language than OWL-DL through further syntactic restrictions. Any OWL-Lite ontology is automatically an OWL-DL ontology. [MVH+04] The opposite is however reasonably not the case. While there are ways to convert OWL-DL to OWL-Full or RDF Graphs (as OWL-Full is a RDF Abstract Syntax) [owlb], there is close to none about OWL-Lite to OWL-DL conversion, let alone a directly usable package providing such functionality. This also makes sense, as OWL-DL has a many more constructs added atop OWL-Lite [MVH+04], among which `rdfs:subClassOf` is used excessively in action recipes. Therefore writing all future action recipes directly in OWL-Lite language constructs does not seem feasible either. Not to mention the additional effort it would take to rewrite all existing recipes manually.
While the target language has changed from OWL-Lite to OWL-DL since Version 2 [DKEV+07] and is already partially usable on OWL-DL, sample tests have shown the already implemented packages of OLA to be not yet compatible to the action recipes used in RoboEarth as an input. Full OWL-DL support is as of yet only planned as a future work. [EDLV04]

5.6 Summary

OLA aligns ontologies written in OWL-Lite and measures their object based similarity. This solution considers multiple ontology alignment methods, as it deals successfully with external data types, the inner structure of classes provided by their properties and constraints, as well as their external structure as given by their relationships to other classes. This solution has the advantage that is does not make use of just a subpart of the language features, but uses an integrates similarity definition atop the linear computation of similarity of entities, making those interact with each other while coping with unavoidable circularities within the ontologies. A major drawback is however the limitation on OWL-Lite and a few OWL-DL constructs.

6 Conclusion

In chapter 1 a general overview of Cloud Computing and the RoboEarth platform as a sample implementation was given and the issues that springs forth from heterogeneous action recipe ontologies discussed. Chapter 2 proceeded to elaborate in more detail about ontologies the RoboEarth platform and action recipes written in OWL used therein. Chapter 3 gave a general overview of related literature and in chapter 4 and chapter 5 two selected methods have been discussed in more detail.
Upon comparing the methods described in chapter 4 and chapter 5, OLA seems at first clearly preferable. Tenorth, Ziegltrum and Beetz in [TZB12] mainly consider string similarities of labels and description with some weighting on the ontology entities' importance to the overall action recipe. Semantic similarity is comparably included only in a rudimentary manner, as only class depths are taken into account. Ngan, Hang and Goh [NHG06] who use five different similarity scores to compare class similarities between two ontologies address far more of the problematic issues, but ultimately the method only compares classes. An ontology however is traditionally seen as a set of entities and a set of relationships between those. This is especially true for action recipes, when subclasses (i.e. subactions) get used in other classes within the ontology and explicit ordering constraints are sometimes specified. Action recipes also contain more entities than just the classes. There are data types, annotation and object properties, individuals or other designators among others. Such entities are not taken into proper consideration with these two methods. OLA however includes them as nodes in the built OL-graph and the relationships among the entities are included into the similarity score as well. The context similarity of this method is not just limited to comparing the top level concepts (classes), but also takes the term context in a broader sense by comparing further possible relationships of the ontology entities.
OLA also has the advantage of having a downloadable alpha version implementation of the proposed methods. Compared to the methods of chapter 4, which would have yet to be implemented, OLA could just be integrated, which saves a good deal of time and effort. Since it is written in Java, integration should also not be a problem, since the OWL API [owlc] already employed in the RoboEath platform is written in Java as well.
However OLA's current limitation to OWL-Lite language constructs creates a major disadvantage, as action recipes in the RoboEath are usually written in OWL-DL. While there is already partial support of OWL-DL, it nonetheless does not support action recipes as an input. Full OWL-DL support is planned, as stated in [DKEV$^+$07], but has yet to be implemented. Conversion from OWL-DL seems hardly possible, especially since action recipes excessively utilize the OWL-DL `rdfs:subClassOf` construct. Writing all future action recipes in OWL-Lite is therefore also not feasible.
So in conclusion, using OLA for the action recipes alignment purpose will highly depend on further support of OWL-DL. An estimation of effort on a self-done expansion of OLA to this

purpose can hardly be given in scope of this work, as it would amount to be a separate work on its own. Meanwhile, the methods described in chapter 4 will have to do or other alternatives explored. As action recipes mainly consists of classes, this should also yield reasonable results nonetheless, despite not taking any external information into account.

Bibliography

[AAV+08] T. Asfour, P. Azad, N. Vahrenkamp, K. Regenstein, A. Bierbaum, K. Welke, J. Schroeder, R. Dillmann. Toward humanoid manipulation in human-centred environments. *Robotics and Autonomous Systems*, 56(1):54–65, 2008. (Cited on page 7)

[AE86] S. F. Altschul, B. W. Erickson. A nonlinear measure of subalignment similarity and its significance levels. *Bulletin of mathematical biology*, 48(5):617–632, 1986. (Cited on pages 19, 21 and 22)

[AFW83] R. C. Angell, G. E. Freund, P. Willett. Automatic spelling correction using a trigram similarity measure. *Information Processing & Management*, 19(4):255–261, 1983. (Cited on page 24)

[AGM+90] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, D. J. Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, 1990. (Cited on page 24)

[AKTKVH06] Z. Aleksovski, M. Klein, W. Ten Kate, F. Van Harmelen. Matching unstructured vocabularies using a background ontology. In *Managing Knowledge in a World of Networks*, pp. 182–197. Springer, 2006. (Cited on page 16)

[Alt93] S. F. Altschul. A protein alignment scoring system sensitive at all evolutionary distances. *Journal of molecular evolution*, 36(3):290–300, 1993. (Cited on page 24)

[AMS+97] S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, D. J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic acids research*, 25(17):3389–3402, 1997. (Cited on page 24)

[AP07] R. Araújo, H. S. Pinto. Towards semantics-based ontology similarity. *Ontology Matching*, p. 37, 2007. (Cited on page 16)

[Bat08] J. Bateman. Einführung in die Computerlinguistik I Winter Semester 2008, Uni Bremen. 2008. URL http://www.fb10.uni-bremen.de/anglistik/ling/ws08/intro-CL-materials/quick-intro-to-logic-and-semantics.pdf. (Cited on page 10)

[Bis92] G. Bisson. Learning in FOL with a similarity measure. In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, pp. 82–82. Citeseer, 1992. (Cited on pages 36 and 42)

[Bro10] J. v. Bronswijk. Robotics and the changing work force: Examples from the housing domain. *Gerontechnology*, 9(2):72, 2010. (Cited on page 7)

[Bun97] H. Bunke. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, 18(8):689–694, 1997. (Cited on page 17)

[BY91] R. A. Baeza-Yates. Searching subsequences. *Theoretical Computer Science*, 78(2):363–376, 1991. (Cited on page 24)

[CL88] H. Carrillo, D. Lipman. The multiple sequence alignment problem in biology. *SIAM Journal on Applied Mathematics*, 48(5):1073–1082, 1988. (Cited on page 19)

[Com09] C. Computing. Web-basierte dynamische IT-Services, von C. Baun, M. Kunze, J. Nimis, S. Tai, 2009. (Cited on page 6)

[CWC92] S. Chan, A. Wong, D. Chiu. A survey of multiple sequence comparison methods. *Bulletin of mathematical biology*, 54(4):563–598, 1992. (Cited on pages 19 and 24)

[DH98] R. Dieng, S. Hug. Comparison of" personal ontologies" represented through conceptual graphs. In *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI 98)*, pp. 341–345. Citeseer, 1998. (Cited on page 17)

[Did04] J. Didion. The Java WordNet Library. 2004. URL http://jwordnet.sourceforge.net/. (Cited on page 44)

[DK10] W. E. Djeddi, M. T. Khadir. XMAP: A novel structural approach for alignment of OWL-Full ontologies. In *Machine and Web Intelligence (ICMWI), 2010 International Conference on*, pp. 368–373. IEEE, 2010. (Cited on page 17)

[DKEV+07] J.-F. Djoufak-Kengue, J. Euzenat, P. Valtchev, et al. OLA in the OAEI 2007 evaluation contest. pp. 188–195, 2007. (Cited on pages 45 and 46)

[DMDH04] A. Doan, J. Madhavan, P. Domingos, A. Halevy. Ontology matching: A machine learning approach. In *Handbook on ontologies*, pp. 385–403. Springer, 2004. (Cited on page 17)

[DMTH+13] D. Di Marco, M. Tenorth, K. Häussermann, O. Zweigle, P. Levi. Roboearth action recipe execution. In *Frontiers of Intelligent Autonomous Systems*, pp. 117–126. Springer, 2013. (Cited on pages 5, 7, 12 and 15)

[EDLV04] J. Euzenat, M. T. D. Loup, P. Valtchev. Ontology Alignment with OLA. *Proceedings of the 3rd EON Workshop, 3rd Intl. Semantic Web Conference, Hiroshima (JP)*, 2004. (Cited on pages 9, 35, 38, 39, 44 and 45)

[Ehr07] M. Ehrig. *Ontology alignment: bridging the semantic gap*, volume 4. Springer Science+ Business Media, 2007. (Cited on page 16)

[ES05] M. Ehrig, Y. Sure. FOAM-framework for ontology alignment and mapping-results of the ontology alignment evaluation initiative. In *Workshop on integrating ontologies*, volume 156, pp. 72–76. 2005. (Cited on page 16)

[þES07] J. r© þme Euzenat, P. Shvaiko. *Ontology matching*. Citeseer, 2007. (Cited on pages 16 and 34)

[Euz94] J. Euzenat. Brief overview of T-tree: the Tropes taxonomy building tool. In *Proc. 4th ASIS SIG/CR workshop on classification research, Columbus (OH US)*, pp. 69–87. 1994. (Cited on page 17)

[Euz04] J. Euzenat. An API for ontology alignment. pp. 698–712, 2004.

[EV03] J. Euzenat, P. Valtchev. An integrative proximity measure for ontology alignment. In *Proc. ISWC-2003 workshop on semantic information integration, Sanibel Island (FL US)*, pp. 33–38. Citeseer, 2003. (Cited on pages 34, 35, 36 and 37)

[EV04] J. Euzenat, P. Valtchev. Similarity-based ontology alignment in OWL-Lite. In *Proc. 15th ECAI, pp333–337, Valencia (ES),*. 2004. URL http://ola.gforge.inria.fr/pdf/align-ECAI04-FSub.pdf. (Cited on pages 5, 35, 36, 38, 39, 40, 41 and 42)

[FGJ+09] A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica. Above the clouds: A Berkeley view of cloud computing. *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS*, 28, 2009. (Cited on page 6)

[FJD85] D. Feng, M. Johnson, R. Doolittle. Aligning amino acid sequences: comparison of commonly used methods. *Journal of Molecular Evolution*, 21(2):112–125, 1985. (Cited on page 24)

[FLR+13] C. Fehling, F. Leymann, R. Retter, W. Schupeck, P. Arbitter. *Cloud Computing Patterns: Fundamentals to Design, Build, and Manage Cloud Applications*. Springer, 2013. (Cited on page 6)

[GANJ06] Y. Ganjisaffar, H. Abolhassani, M. Neshati, M. Jamali. A similarity measure for OWL-S annotated web services. In *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pp. 621–624. IEEE Computer Society, 2006. (Cited on page 17)

[Gar11] Gartner. Gartner's 2011 Hype Cycle Special Report. 2011. URL http://www.gartner.com/it/page.jsp?id=1763814. (Cited on page 6)

[Ghi12] P. Ghislandi. *eLearning–Theories, Design, Software and Applications*. ERIC, 2012. (Cited on page 17)

[GLd+07] J. Gracia, V. Lopez, M. d'Aquin, M. Sabou, E. Motta, E. Mena. Solving semantic ambiguity to improve semantic web based ontology matching. 2007. (Cited on page 16)

[GQ08] J. Ge, Y. Qiu. Concept similarity matching based on semantic distance. In *Semantics, Knowledge and Grid, 2008. SKG'08. Fourth International Conference on*, pp. 380–383. IEEE, 2008. (Cited on page 17)

[GS03] F. Giunchiglia, P. Shvaiko. Semantic matching. 2003. (Cited on page 16)

[GSY04] F. Giunchiglia, P. Shvaiko, M. Yatskevich. S-Match: an algorithm and an implementation of semantic matching. In *The semantic web: research and applications*, pp. 61–75. Springer, 2004. (Cited on page 16)

[Gus93] D. Gusfield. Efficient methods for multiple sequence alignment with guaranteed error bounds. *Bulletin of Mathematical Biology*, 55(1):141–154, 1993. (Cited on page 24)

[GYM07] F. Giunchiglia, M. Yatskevich, F. McNeill. Structure preserving semantic matching. 2007. (Cited on page 16)

[GYS07] F. Giunchiglia, M. Yatskevich, P. Shvaiko. Semantic matching: Algorithms and implementation. In *Journal on Data Semantics IX*, pp. 1–38. Springer, 2007. (Cited on page 16)

[HCZ+06] W. Hu, G. Cheng, D. Zheng, X. Zhong, Y. Qu. The results of Falcon-AO in the OAEI 2006 campaign. *Ontology Matching*, p. 124, 2006. (Cited on page 16)

[Hen96] S. Henikoff. Scores for sequence searches and alignments. *Current opinion in structural biology*, 6(3):353–360, 1996. (Cited on page 24)

[Hir97] D. Hirschberg. Serial computations of Levenshtein distances. 1997. (Cited on page 24)

[HK73] J. E. Hopcroft, R. M. Karp. An n^5/2 algorithm for maximum matchings in bipartite graphs. volume 2, pp. 225–231. SIAM, 1973. (Cited on page 36)

[HKR+04] M. Horridge, H. Knublauch, A. Rector, R. Stevens, C. Wroe. A Practical Guide To Building OWL Ontologies Using The Protégé-OWL Plugin and CO-ODE Tools Edition 1.0. *University of Manchester*, 2004. (Cited on page 11)

[HPS08] M. Horridge, P. F. Patel-Schneider. Manchester syntax for OWL 1.1. *OWL: Experiences and Directions, Washington, DC*, 2008.

[HQ08] W. Hu, Y. Qu. Falcon-AO: A practical ontology matching system. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(3):237–239, 2008. (Cited on page 16)

[jgr] URL http://www.jgraph.com/. (Cited on page 38)

[JHCQ05] N. Jian, W. Hu, G. Cheng, Y. Qu. Falcon-ao: Aligning ontologies with falcon. In *Proceedings of K-CAP Workshop on Integrating Ontologies*, pp. 85–91. 2005. (Cited on page 16)

[KC08] S.-K. Kim, H.-J. Choi. Decision of semantic similarity using description logic and vector weight between concepts. In *Networked Computing and Advanced Information Management, 2008. NCM'08. Fourth International Conference on*, volume 2, pp. 345–350. IEEE, 2008. (Cited on page 17)

[KR08] V. Kundeti, S. Rajasekaran. A local structural alignment algorithm with Variable Length Alignment Fragment Pairs. In *BioInformatics and BioEngineering, 2008. BIBE 2008. 8th IEEE International Conference on*, pp. 1–7. IEEE, 2008. (Cited on page 24)

[KSK11] P. Kremen, M. Smid, Z. Kouba. OWLDiff: A practical tool for comparison and merge of OWL ontologies. In *Database and Expert Systems Applications (DEXA), 2011 22nd International Workshop on*, pp. 229–233. IEEE, 2011. (Cited on page 17)

[KYB03] I. Korf, M. Yandell, J. Bedell. *Blast : (an essential guide to the basic local alignment search tool)*. O'Reilly, 2003. (Cited on pages 5, 18 and 24)

[LDK07] B. T. Le, R. Dieng-Kuntz. A graph-based algorithm for alignment of owl ontologies. In *Web Intelligence, IEEE/WIC/ACM International Conference on*, pp. 466–469. IEEE, 2007. (Cited on page 17)

[Lev66] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. In *Soviet physics doklady*, volume 10, p. 707. 1966. (Cited on pages 22 and 24)

[Ley] F. Leymann. Cloud Computing: The Next Revolution in IT. (Cited on page 6)

[LJP10] I. Lera, C. Juiz, R. Puigjaner. Owl-m extension for semantic representations of ontology alignments. In *Complex, Intelligent and Software Intensive Systems (CISIS), 2010 International Conference on*, pp. 956–961. IEEE, 2010. (Cited on page 17)

[LSM06] V. Lopez, M. Sabou, E. Motta. Powermap: Mapping the real semantic web on the fly. In *The Semantic Web-ISWC 2006*, pp. 414–427. Springer, 2006. (Cited on page 17)

[Man10] O. C. Manifesto. Open cloud manifesto, 2010. (Cited on page 6)

[MBR01] J. Madhavan, P. A. Bernstein, E. Rahm. Generic schema matching with cupid. In *Proceedings of the International Conference on Very Large Data Bases*, pp. 49–58. 2001. (Cited on page 17)

[MCWD06] C. Matuszek, J. Cabral, M. J. Witbrock, J. DeOliveira. An Introduction to the Syntax and Content of Cyc. In *AAAI Spring Symposium: Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering*, pp. 44–49. Citeseer, 2006. (Cited on page 12)

[MGMR02] S. Melnik, H. Garcia-Molina, E. Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pp. 117–128. IEEE, 2002. (Cited on page 17)

[MH10] Meir-Huber. *Cloud Computing: Praxisratgeber und Einstiegsstrategien*. Entwickler. Press, 2010. (Cited on pages 6 and 8)

[MHZF12] C. Matuszek, E. Herbst, L. Zettlemoyer, D. Fox. Learning to parse natural language commands to a robot control system. In *Proc. of the 13th Int'l Symposium on Experimental Robotics (ISER)*. 2012. (Cited on page 8)

[Mil95] G. A. Miller. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995. (Cited on pages 28 and 44)

[Mil98] G. Miller. WordNet: An Electronic Lexical Database. *The MIT Press*, May 15, 1998. (Cited on page 28)

[MLQ09] X. Min, Z. Luo, X. Qianxing. Semantic similarity between concepts based on OWL ontologies. In *Knowledge Discovery and Data Mining, 2009. WKDD 2009. Second International Workshop on*, pp. 749–752. IEEE, 2009. (Cited on page 17)

[MM89] E. W. Myers, W. Miller. Approximate matching of regular expressions. *Bulletin of mathematical biology*, 51(1):5–37, 1989. (Cited on page 24)

[MRV11] C. Metzger, T. Reitz, J. Villar. *Cloud Computing: Chancen und Risiken aus technischer und unternehmerischer Sicht*. Hanser, 2011. (Cited on page 6)

[MS01] A. Maedche, S. Staab. *Comparing ontologies-similarity measures and a comparison study*. AIFB, 2001. (Cited on page 16)

[MS02] A. Maedche, S. Staab. Measuring similarity between ontologies. In *Knowledge engineering and knowledge management: Ontologies and the semantic web*, pp. 251–263. Springer, 2002. (Cited on page 16)

[MVH+04] D. L. McGuinness, F. Van Harmelen, et al. OWL web ontology language overview. *W3C recommendation*, 10(2004-03):10, 2004. (Cited on pages 11, 35 and 45)

[MZMN07] T. Milledge, G. Zheng, T. Mullins, G. Narasimhan. SBLAST: Structural basic local alignment searching tools using geometric hashing. In *Bioinformatics and Bioengineering, 2007. BIBE 2007. Proceedings of the 7th IEEE International Conference on*, pp. 1343–1347. IEEE, 2007. (Cited on page 24)

[NHG06] L. Ngan, T. Hang, A. Goh. Semantic similarity between concepts from different OWL ontologies. pp. 618–623, 2006. (Cited on pages 5, 25, 26, 30, 31, 32, 34 and 46)

[NLH12] J. Niemi, K. Lindén, M. Hyvärinen. Using a bilingual resource to add synonyms to a wordnet: FinnWordNet and Wikipedia as an example. In *Proceedings of the 6th international global Wordnet conference (GWC 2012)*, pp. 227–231. 2012. (Cited on page 29)

[NM01] N. F. Noy, M. A. Musen. Anchor-PROMPT: Using non-local context for semantic matching. In *Proceedings of the workshop on ontologies and information sharing at the international joint conference on artificial intelligence (IJCAI)*, pp. 63–70. 2001. (Cited on page 17)

[NW70] S. B. Needleman, C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453, 1970. (Cited on page 19)

[OAE07] The Second International Workshop on Ontology Matching. Collocated with the 6th International Semantic Web Conference. 2007. (Cited on page 16)

[OB84] B. C. Orcutt, W. C. Barker. Searching the protein sequence database. *Bulletin of mathematical biology*, 46(4):545–552, 1984. (Cited on page 24)

[olaa] URL http://ola.gforge.inria.fr/visualization.html. (Cited on page 38)

[olab] OLA (OWL-Lite Alignment) SCM Repository. URL https://gforge.inria.fr/scm/?group_id=271. (Cited on page 44)

[ont] Workshop: Ontology Matching. URL http://ontologymatching.org/. (Cited on pages 16 and 45)

[OPR23] C. K. Ogden, J. P. Postgate, I. A. Richards. *The Meaning of Meaning. A Study of the Influence of Language Upon Thought and of the Science of Symbolism... With an Introduction by JP Postgate... and Supplementary Essays by B. Malinowski... and FG Crookshank*. 1923. (Cited on pages 5 and 11)

[OU92] M. Ohya, Y. Uesaka. Amino acid sequences and DP matching: A new method for alignment. *Information sciences*, 63(1):139–151, 1992. (Cited on page 24)

[OWLa] URL http://www.co-ode.org/resources/tutorials/intro/slides/OWLFoundationsSlides.pdf. (Cited on page 11)

[owlb] URL http://www.w3.org/2007/OWL/wiki/Mapping_to_RDF_Graphs. (Cited on page 45)

[owlc] OWL API. URL http://owlapi.sourceforge.net/documentation.html. (Cited on page 46)

[Por80] M. F. Porter. An algorithm for suffix stripping. *Program: electronic library and information systems*, 14(3):130–137, 1980. (Cited on page 24)

[PT11] A. Perzylo, M. Tenorth. *RoboEarth - Workshop on Knowledge Engineering, European Robotics Forum 2011, Vasteras, Sweden*. 2011. (Cited on page 12)

[PW+97] E. H. Porter, W. E. Winkler, et al. Approximate string comparison and its effect on an advanced record linkage system. In *Advanced Record Linkage System. US Bureau of the Census, Research Report*. Citeseer, 1997. (Cited on page 29)

[RB01] E. Rahm, P. A. Bernstein. A survey of approaches to automatic schema matching. volume 10, pp. 334–350. Springer, 2001. (Cited on pages 17 and 36)

[RH83] A. Rosenfeld, P. D. Hyde. Parallel string acceptance using lattice graphs. *Pattern Recognition Letters*, 1(4):237–243, 1983. (Cited on page 24)

[rob] RoboEarth Web Interface. URL http://api.roboearth.org/. (Cited on pages 22 and 28)

[S+79] P. H. Sellers, et al. Pattern recognition in genetic sequences. *Proc. Natl. Acad. Sci. USA*, 76(7):3041, 1979. (Cited on page 24)

[Sal89] G. Salton. *Automatic Text Processing: The Transformation, Analysis and Retrieval of Information by Computer*. Addison-Wesley, 1989. (Cited on page 24)

[Sch10] B. Schießle. *RoboEarth - Entwicklung einer Basisplattform zu verteilten Speicherung von Roboterdaten*. 2010. (Cited on pages 7 and 12)

[SE05] P. Shvaiko, J. Euzenat. A survey of schema-based matching approaches. In *Journal on Data Semantics IV*, pp. 146–171. Springer, 2005. (Cited on page 16)

[Sel74a] P. H. Sellers. An algorithm for the distance between two finite sequences. *Journal of Combinatorial Theory, Series A*, 16(2):253–258, 1974. (Cited on page 24)

[Sel74b] P. H. Sellers. On the theory and computation of evolutionary distances. *SIAM Journal on Applied Mathematics*, 26(4):787–793, 1974. (Cited on page 24)

[SG12] V. Sugumaran, J. A. Gulla. *Applied semantic web technologies*. Auerbach Pub, 2012. (Cited on page 17)

[SHZ10] B. Schießle, K. Häussermann, O. Zweigle. Deliverable D6. 1: Complete specification of the RoboEarth platform. Technical report, Technical report, December 1, 2010. http://www.roboearth. org/wp-content/uploads/2011/03, 2010. (Cited on page 12)

[SJ12] A. P. Shakya, G. K. Jha. *Learning of Robots by using & Sharing Their Experiences*. 2012. (Cited on page 12)

[SM01] G. Stumme, A. Maedche. FCA-Merge: Bottom-up merging of ontologies. In *International joint conference on artificial intelligence*, volume 17, pp. 225–234. LAWRENCE ERLBAUM ASSOCIATES LTD, 2001. (Cited on page 16)

[Smi88] R. Smith. A finite state machine algorithm for finding restriction sites and other pattern matching applications. *Computer applications in the biosciences: CABIOS*, 4(4):459–465, 1988. (Cited on page 24)

[Spo89] J. L. Spouge. Speeding up dynamic programming algorithms for finding optimal lattice paths. *SIAM Journal on Applied Mathematics*, 49(5):1552–1566, 1989. (Cited on page 24)

[SSA+09] M. S. Sadi, A. Z. M. Sami, I. U. Ahmed, A. B. M. Ruhunnabi, N. Das. Bioinformatics: Implementation of a proposed upgraded Smith-Waterman algorithm for local alignment. In *Computational Intelligence in Bioinformatics and Computational Biology, 2009. CIBCB'09. IEEE Symposium on*, pp. 87–91. IEEE, 2009. (Cited on page 24)

[Stu09] H. Stuckenschmidt. *Ontologien*. Springer DE, 2009. (Cited on pages 10 and 11)

[Sun08] W. G. Sunna. *Multi-layered approach to aligning heterogeneous ontologies*. ProQuest, 2008. (Cited on page 17)

[SW81] T. F. Smith, M. S. Waterman. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197, 1981. (Cited on page 21)

[TB09] M. Tenorth, M. Beetz. KnowRob—knowledge processing for autonomous personal robots. pp. 4261–4266, 2009. (Cited on page 12)

[TB13] M. Tenorth, M. Beetz. Exchanging Action-Related Information among Autonomous Robots. In *Intelligent Autonomous Systems 12*, pp. 467–476. Springer, 2013. (Cited on page 12)

[THK91] E. C. Tyler, M. R. Horton, P. R. Krause. A review of algorithms for molecular sequence comparison. *Computers and biomedical research*, 24(1):72–96, 1991. (Cited on pages 5, 22, 23 and 24)

[TKPB11] M. Tenorth, U. Klank, D. Pangercic, M. Beetz. Web-enabled robots. *Robotics & Automation Magazine, IEEE*, 18(2):58–68, 2011. (Cited on page 7)

[TLL+06] J. Tang, J. Li, B. Liang, X. Huang, Y. Li, K. Wang. Using Bayesian decision for ontology mapping. *Web Semantics: Science, Services and Agents on the World Wide Web*, 4(4):243–262, 2006. (Cited on page 17)

[TNB10] M. Tenorth, D. Nyga, M. Beetz. Understanding and executing instructions for everyday manipulation tasks from the world wide web. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 1486–1491. IEEE, 2010. (Cited on pages 7 and 12)

[TPLB12] M. Tenorth, A. C. Perzylo, R. Lafrenz, M. Beetz. The roboearth language: Representing and exchanging knowledge about actions, objects, and environments. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 1284–1289. IEEE, 2012. (Cited on page 12)

[TZB12] M. Tenorth, J. Ziegltrum, M. Beetz. Automated Alignment of Specifications of Everyday Manipulation Tasks, 2012. (Cited on pages 5, 8, 9, 12, 18, 21, 25, 27, 28, 34 and 46)

[Ukk85] E. Ukkonen. Algorithms for approximate string matching. *Information and control*, 64(1):100–118, 1985. (Cited on page 24)

[Val99] P. Valtchev. *Construction automatique de taxonomies pour l'aide à la représentation de connaissances par objets*. Ph.D. thesis, 1999. (Cited on page 40)

[Vos12] G. von Vossen. Cloud Computing für Unternehmen-Technische, wirtschaftliche, rechtliche und organisatorische Aspekte. 2012. (Cited on page 6)

[W3C] OWL 2 Web Ontology Language Document Overview. W3C recommendation, W3C. (Cited on page 11)

[Wat84] M. S. Waterman. General methods of sequence comparison. *Bulletin of Mathematical Biology*, 46(4):473–500, 1984. (Cited on page 24)

[WBC+11] M. Waibel, M. Beetz, J. Civera, R. D'Andrea, J. Elfring, D. Galvez-Lopez, K. Haussermann, R. Janssen, J. Montiel, A. Perzylo, et al. RoboEarth – A World Wide Web for Robots. *Robotics & Automation Magazine, IEEE*, 18(2):69–82, 2011. (Cited on pages 5, 7, 12, 13 and 14)

[WIVDMS07] S. Wang, A. Isaac, L. Van Der Meij, S. Schlobach. Multi-concept alignment and evaluation. In *Second International Workshop on Ontology Matching, ISWC*. 2007. (Cited on page 16)

[WP84] M. S. Waterman, M. D. Perlwitz. Line geometries for sequence comparisons. *Bulletin of Mathematical Biology*, 46(4):567–577, 1984. (Cited on page 24)

[WP94] Z. Wu, M. Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pp. 133–138. Association for Computational Linguistics, 1994. (Cited on page 27)

[WSB76] M. S. Waterman, T. F. Smith, W. A. Beyer. Some biological sequence metrics. *Advances in Mathematics*, 20(3):367–387, 1976. (Cited on page 24)

[ZMdH09] O. Zweigle, R. van de Molengraft, R. d'Andrea, K. Häussermann. RoboEarth: connecting robots worldwide. pp. 184–191, 2009. (Cited on page 12)

[ZNK+07] S. Zghal, E. M. Nguifo, K. Kamoun, S. Ben Yahia, Y. Slimani. A new alignment method for OWL-Lite ontologies using propagation of similarity over the graph. In *Database and Expert Systems Applications, 2007. DEXA'07. 18th International Workshop on*, pp. 524–528. IEEE, 2007. (Cited on page 17)

[ZPZ81] E. Zamora, J. J. Pollock, A. Zamora. The use of trigram analysis for spelling error detection. *Information Processing & Management*, 17(6):305–316, 1981. (Cited on page 24)

All links were last followed on June 28, 2013.

Declaration

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschift