



Workshops der wissenschaftlichen Konferenz
Kommunikation in Verteilten Systemen 2011
(WowKiVS 2011)

Towards Swarm-based Federated Web Knowledgebases

Philipp Obermeier, Anne Augustin and Robert Tolksdorf

12 pages

Towards Swarm-based Federated Web Knowledgebases

Philipp Obermeier¹, Anne Augustin² and Robert Tolksdorf²

¹Digital Enterprise Research Institute, National University of Ireland, Galway

philipp.obermeier@deri.org

²Freie Universität Berlin, Institut für Informatik, AG Netzbaasierte Informationssysteme,

Königin-Luise-Straße 24-26, D-14195 Berlin, Germany

aaugusti@inf.fu-berlin.de, tolk@ag-nbi.de

Abstract: Internet knowledgebases are more and more described using ontological vocabularies. However, efficient coherent solutions for both federated storage and reasoning upon widely distributed web repositories are rarely explored. We present a self-organized, distributed storage and reasoning approach based on swarm intelligence exploiting the strong applicability of swarm algorithms to distributed environments. While centralized reasoning has problems in scalability, swarm intelligence has the potential to reach Web scalability. Our concept comprises two layers of swarm algorithms - storage and reasoning layer - for information storage and inference of new statements, respectively. We present for the former our concepts with first evaluation results and give for the latter a general swarm-based concept for forward-chaining assertional reasoning in the description logic *ALC*.

Keywords: Selforganization, Semantic information, Reasoning, Swarming

1 Introduction

In contrast to the present internet-scale distribution of data today's RDF (Resource Description Format) storage solutions rarely offer efficient querying and reasoning support across multiple, widely distributed knowledgebases. The scalability of existing systems is often hampered by the need of centralized components to coordinate and control storage management and reasoning tasks, e.g. centralized indices and routing facilities. To this end we suggest a storage and reasoning infrastructure fully based on swarm algorithms, which is completely self-organized and highly applicable in distributed environments. On the surface our swarm-based system offers two major functionalities - federated distributed storage and forward-chaining assertional reasoning for a distributed knowledgebase defined by assertional statements and terminological axioms. In this connection, we assume, that the employed data model is RDF [W3C04b] and the used ontology language is based on a subset of description logics e.g. RDFS, OWL-Lite (Web Ontology Language) and OWL-DL (OWL Description Logics) [W3C04c, W3C04a]. Although, our swarm-based reasoning concept temporarily only supports *ALC* (Attributive Language with Complements), we strive to achieve *SHOIN* (basis for OWL logic) reasoning.

Storage and reasoning layer completely rely on swarm intelligence, whose individuals act in a fully self-contained, probabilistic way. Therefore, our system offers a completely decentralized, self-adaptive mode of operation without the need of any central control entity. As trade-off for this kind of flexibility, we cannot guarantee completeness for querying and reasoning, though, our

previous experiments on swarm-based RDF stores showed a high level of recall. Furthermore, ants in storage and reasoning layer are created according to a type describing the ants' behavior.

Section 2 addresses work related to this paper and Section 3 our overall concept. Sections 4 and 5 detail the storage and reasoning layers. We conclude with open problems and future work.

2 Related Work

Linda [Gel85] is a coordination language where processes exchange data via a shared, associative memory, the so-called tuple space. The *out*-operation puts a *tuple* which is an ordered sequence of typed data objects like $\langle 4, \text{"hello"}, 1.2 \rangle$ into that space. To retrieve the data, the *in*-operation is used which searches the tuple space for data matching a *template* which is an argument to the operation. For example $in(\langle 4, ?string, ?float \rangle)$ would retrieve the above tuple since the template contains the same value in the first field and the others have the datatype requested with the tuple. The *rd*-operation is the same as the *in*-operation but returns a copy of a matching tuple instead of removing it. In several projects the Linda Model has been extended to contain semantic information, whereas the tuples were replaced by RDF triples, see for example [NSKM08], [SKN07] and [TNS08]. In SwarmLinda [MT03] the idea is to completely decentralize the tuplespace and to establish self organization mechanisms to make it scalable and dynamic enough for large open systems. Tuple distribution and retrieval are performed by autonomic entities that use algorithms for foraging and brood sorting found in natural ant-colonies [BDT99]. In [TA09] and [Kos09] the ant colony algorithms of SwarmLinda were adopted to realize a distributed storage for RDF triples. Both approaches are the basis for the storage layer of our presentation in section 4.

In regard to scalable distributed RDF query processing and reasoning, *RDFPeers* [CF04] introduces a spanning of RDF repositories over a peer-to-peer network. Execution of disjunctive and conjunctive queries over distributed storages are supported, but no reasoning. A more recent peer-to-peer approach is *MarVIN* [AKO⁺08], which enables parallel computing of RDFS and OWL closures across multiple networked node machines. First evaluation results within a local grid setup supports the authors' claim of high scalability in terms of number of nodes. A first swarm-based reasoning proposal was introduced by Kathrin Dentler et al. [DGS09], in which ants move along the RDF graph structure and apply RDFS axioms to matching assertions. Additionally, an exemplary draft for an extension of this algorithm by supporting SWRL rules is given in [Den09]. In contrast to our approach, Dentler et al. use scents solely to find seldom-visited parts of the RDF graph. Furthermore, a centralized bloom filter for ant routing is employed, which conflicts with our goal of total decentralization.

3 Concept

We aim at a distributed system for RDF data storage and ontology-based reasoning, that is extensively based on swarm intelligence. The system consists of a storage and reasoning layer which are realized by two different classes of ants. The storage ants manage the system's data to facilitate efficient querying and the reasoning ants infer new knowledge from data.

The storage layer stores and queries triples in a network of server nodes. The primitive $out\langle triple \rangle$ stores a triple in the storage and $rd\langle template \rangle$ and $in\langle template \rangle$ query a triple from

the storage where *in* removes the concerning triple. The *out*-operation is performed by out-ants which build clusters of similar triples and leave trails to these so that future ants can locate them easily. Section 4 introduces two different clustering approaches used by the out-ants. One is based on a syntactical similarity measure that compares the URIs of the triples' resources. The second accounts for the taxinomial similarity of the resources. The *in*- and *rd*-operations are performed by in- and rd-ants. They carry templates and search the network for matching triples. The template can be a simple triple pattern with constant and variable fields like (s,p,?o). We also allow templates that have concept fields, which have to be replaced by a resource of that concept, like (<C>,p,?o). It matches all triples that have a resource of the concept *C* as subject and *p* as predicate. (a,p,o) matchesx this template if there is a statement (a, rdf:type, C).

In the reasoning layer ants implement forward assertional reasoning based on the knowledge held in the storage layer. More precisely, for every terminological axiom added to the storage layer the system defines a new ant type. Additionally, it creates a number of ant instances for this type in proportion to the scale of the distributed system. Each of these ants applies the newly inserted terminological axiom to matching assertions within the knowledgebase and ,thus, creates new assertions, which are in turn placed in the storage. This method of reasoning infers intensional knowledge in a *brave* way, i.e., assertions may be entailed iff they are in the closure of some minimal Herbrand model for the knowledge base. Over time a subset of the brave inference closure with regard to minimal Herbrand interpretations is computed. In general, though, it cannot be guaranteed that at a certain point of time the complete closure is computed, since storage and reasoning ants act in a probabilistic way on several occasions, e.g., detection or placement of assertions. Moreover, inferred statements can be retrieved in the same way as pre-existing data via the *rd* operation at the storage layer. In sequel, we will further investigated swarm-based reasoning and related issues at Section 5.

The following example depicts the underlying notion for the system's reasoning concept: Our system should be initialized with a distinct knowledge base *KB*. As mentioned before, for each TBox axiom *KB* our system creates a distinct type of ants. The TBox holds the DL axiom $ta := professor \sqcap researcher \sqsubseteq seniorresearcher$, encoded as OWL ontology in Listing 1. An ant of *ta*'s type begins its inference procedure by looking for scent trails that lead to the con-

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix : <http://ag-nbi.de/data/> .
:researcher rdf:type owl:Class .
:professor rdf:type owl:Class .
# intersection of classes 'researcher' and 'professor'
_:x rdf:type owl:Class ,
    intersectionOf _:l1 ,
    rdfs:subClassOf seniorresearcher .
_:l1 rdf:first :researcher ,
    rdf:rest _:l2 .
_:l2 rdf:first :professor ,
    rdf:rest rdf:nil .
```

Listing 1: TBox axiom given in OWL

cepts *professor* and *researcher*. Without loss of generality, we assume that the ant detects a

scent of the *professor* cluster, follows it, and finds and memorizes a resource *:peter* belonging to *professor*. It then follows a scent trail for *researcher* and searches *researcher(:peter)*. A successful search creates a new assertion *seniorresearcher(:peter)* and deposits it in the *seniorresearcher* cluster while acting like a storage ant. Finally, the ant clears its local memory and restarts this procedure. Besides that, the ant not only resolves the “proposition” of *ta* as described above, but also the “contraposition” of *ta*, i.e., $\neg\text{seniorresearcher}$ implies $\neg(\text{professor} \sqcap \text{researcher})$.

4 Storage Layer

Applying an $\text{out}\langle(s,p,o)\rangle$ to the system *three* out-ants are generated each of which walks in the network of nodes and tries to build clusters based on one of the triple’s fields respectively. We call that field *cluster resource* which can be subject, predicate or object and we refer to the corresponding out-ants as *subject ant*, *predicate ant* and *object ant*. For example the subject ant would determine the subject *s* as cluster resource and therefore would place the triple into a cluster where are triples with resources that resemble *s*. Roaming the network to find a suited place for the triple the ants follow scents that resemble the cluster resource. When an out-ant drops a triple it emits the scent of the triple’s cluster resource on the concerning node and in weaker quantity on its neighbors in order to attract future ants following these scents. To query triples from the storage template ants are generated carrying triples where one or two fields are missing. For example template $(?s, \text{http://d.org/o\#authored}, \text{http://d.org/books/orientexpress})$ is used to find a triple that informs about the authorship of a specific book. In this case two ants are generated that look in two resource-clusters where are resources similar to <http://d.org/o\#authored> and <http://d.org/books/orientexpress> respectively. The one arriving back first carries the actual result, the triple found by the second ant is ignored.

The triple search for the in-ant is the same as for the rd-ant, but when it finds a matching triple it has to remove all three copies. Therefore *locking*-ants are generated which try to locate the other two copies in the respective other resource-clusters. If they succeed in locking them, all three copies can be removed. The ants’ decisions for the path selection are probabilistic and influenced by the amount of pheromones on the neighbor nodes and their similarity to the cluster resource. Also the out-ant’s decision to drop the assigned triple on its current location is probabilistic and depends on the number of resources and their similarity to the cluster resource. All ant types have an aging mechanism which prevents them from wandering around endlessly. So when an ant cannot perform its task in a certain timespan it dies. In the case of the out-operation the triple is dropped on the current location. In the case of the rd- and in-operation the query remains without response and has to be repeated after a certain time-out. Detailed descriptions of the algorithms for the different types of ants can be found in [TA09] and [Kos09].

4.1 Clustering by Syntactic Similarity

An ant acting in a completely decentralized manner in general does not have any ontological knowledge about the regarded resources. If one triple is $(\text{penguin, colored, blackwhite})$ and another is $(\text{canary, colored, yellow})$, the ant cannot determine that the two are related because some ontology states that both *penguin* and *canary* are subconcepts of concept *bird*.

Therefore in [TA09] we have presented a similarity measure that is based on the URIs' syntactical structures only. The basic idea is to assume that URIs from similar namespaces are similar in regard to their related concepts. For the above example, we would expect that the *penguin*- and *canary*-URI would look like *http://birds.org/onto.rdf#penguin* and *http://birds.org/onto.rdf#canary*. They are identical in the host and path parts but differ only in their fragments. In order to measure the similarity between two regarded URIs we first split them into host- and path-components and then compare these separately. After that the results of the comparisons are weighted. The weighted sum of both results forms the namespace similarity sim_{URI} between the two URIs which is in the range of 0 and 1. The value is 1 if the URIs are equal and 0 if they are entirely different. In order to compare the hosts of two URIs we consider their “.”-separated domain labels ([BMM94], sec 3.1). Starting with the hierarchical highest label, we compare them pairwise applying a weighting function, so that a path segment on a higher level in the hierarchy gets twice a weight. Along these lines the path similarity is computed by comparing the path segments of the URL-paths pairwise. For the example above we would firstly compare the hosts of the two URIs which are equal and therefore have similarity 1, secondly we would compare the paths which differ only in their fragments (to simplify the comparison the fragment is assigned to the path). The first path segment gets twice a weight of the fragment, so the path similarity is $1 \cdot \frac{2}{3} + 0 \cdot \frac{1}{3} = 0,6$, as the the first path segments are equal and the fragments are different. Weighting the host with 0,9 and the path with 0,1 we get a overall similarity of $0,9 \cdot 1 + 0,1 \cdot 0,6 = 0,96$. The resulting namespace similarity between the two URIs is quite high, so that it is very probable that they are placed in the same cluster. The general formulas for computing the namespace similarity between different types of URIs can be found in [TA09]

The syntactical approach was implemented as a simulation using NetLogo [TW04]. First results of an evaluation on a network of 50 nodes with data from DBPedia [ABK⁺07] and LUBM [GPH05] show that the similarity of the resources and triples on the nodes increases when they are distributed by our algorithms in comparison with a random distribution. Also the rd-operations perform much better after triple distribution by the out ants. That seems to indicate that the ants successfully build clusters of similar triples and the trails work properly.

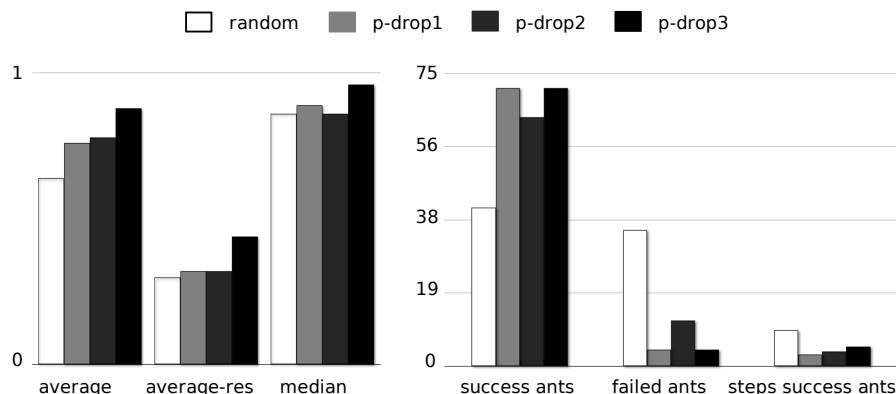


Figure 1: Evaluation of out (left) and rd (right)

The average results from five test runs are shown in figure 1. Three slightly different drop prob-

abilities $p_{drop1}, p_{drop2}, p_{drop3}$ that were used by the out-ants to decide whether a triple should be dropped on its current location were compared. **average** denotes the average similarity of the triples on the nodes in the network. Therefore the average similarity of the triples on each node was calculated and the average value of all nodes was determined. **average-res** was calculated similarly by comparing the resources on the nodes instead of the triples. In order to determine **median** the median similarity of the triples on the node was calculated and averaged. The detailed formulas used in the evaluation can be found in [TA09]

success ants denotes the number of *rd*-ants which found a matching triple, **failed ants** is the number of ants which did not find a triple and **steps success ants** is the average number of steps that a successful in-ant took before finding a matching triple.

The advantage of the syntactical clustering approach is that the ants do not need to have any ontological knowledge which is distributed over the network. Clusters are located easily without search for ontological information so that triples can be retrieved efficiently. In this case, extreme decentralization is leading to scalability. Adding n ants to the system the processing amount of a single ant increases with the factor n . In our implementation we realized the retrieval of single triples only, but theoretically it is possible when applying a *rd*-operation to retrieve all triples to a given template that are found in a certain timespan with no guarantees for completeness.

4.2 Clustering by Formal Concept Similarity

While the syntactical clustering approach seems to allow efficient distributed triple storage and retrieval, there are still reasons for clustering triples that are semantically related by an ontology. First of all semantically related resources can have completely different namespaces and can be in different clusters which are far away from each other. For example if we want to query all statements about all five-star hotels ($(\langle \text{fivestarhotel} \rangle, ?p, ?o)$ whereas *fivestarhotel* is a concept, which has to be replaced by a resource of this concept), the URIs of the hotels could differ entirely, and the concerning triples would be spread over the network. Thus such queries would be possible but quite inefficient, as after having queried all resources of a concept the ants would have to search in different parts of the network. For such queries it would be worthwhile if resources that are assigned to the same concept would be not too far from another. Also for many reasoning tasks it is advantageous if semantically related triples are close together. For example (C, p, o) and $(a, \text{rdf:type}, C)$ can be found in the same concept cluster C . From the two statements we can infer the triple (a, p, o) that also fits in this cluster. Thus the distances that the reasoning ants have to cover are reduced. In our second approach which is based on [Kos09] we cluster by concept similarity. Here the idea is to cluster triples with resources that are taxinomically related. For example penguin statements are similar to canary statements as they are statements about birds. So they would be close together in the network and also nearby statements about birds. The goal is to get clusters of triples with resources that are assigned to the same concept to which ants are attracted by concept scents. As in the beginning an ant that performs an *out*, *rd* or *in* generally does not have any ontological knowledge, it starts by querying the assigned concept. In [Kos09] this is solved by an additional syntactical cluster layer where the *rdfs:subClassOf*-, *rdfs:subPropertyOf*- and *rdf:type*-Statements are clustered using the syntactical strategy described above. In the following we will refer to these triples as *taxinomial information*. In the beginning an ant retrieves the necessary information by specialized *rd*-ants

with template (*cluster resource*, `rdf:type`, `?o`) which search in the syntactical cluster. It is sufficient to either retrieve one superior concept or - if the resource itself is a concept - to find out that it is a concept. From the moment that the concept is known the ant can start roaming the network following scents that lead to clusters of that concept. When it gets into this concept cluster it will learn more about the taxonomy and can decide based on measures for semantical similarity where its triple fits best. It is sufficient that the triples are stored once in the syntactical layer (instead of three times) and are clustered by subject as we always search for `rdf:type`-statements (or `rdfs:subClassOf`-, `rdfs:subPropertyOf`-statements) where the subject is given.

In theory, the ant could also look in the `rdf:type` cluster to retrieve the necessary taxinomial information, but as every ant has to locate it there would be much network traffic in this cluster (even if it is distributed over a huge amount of nodes). This would not be consistent with our idea of decentralization and would result in bottlenecks. However, introducing an additional syntactical cluster layer where the `rdf:type`- and `rdfs:subclass`- statements are clustered by subject has the advantage that the information is to some extent evenly distributed over the network.

In a first implementation [Kos09], that was also based on NetLogo [TW04], the ants had a global knowledge of the taxinomial information to all resources. This is a possible scenario where few concepts are used and the concept taxinomies are interlinked. Then it makes sense to replicate the taxinomial information on each node. Figure 2 shows how the similarity on the nodes increased with the time while the ants were storing triples using Lin's information theoretic similarity measure [Lin98], whereas initially there were randomly distributed triples on the nodes. For a detailed description of the measuring and the evaluation we refer to [Kos09].

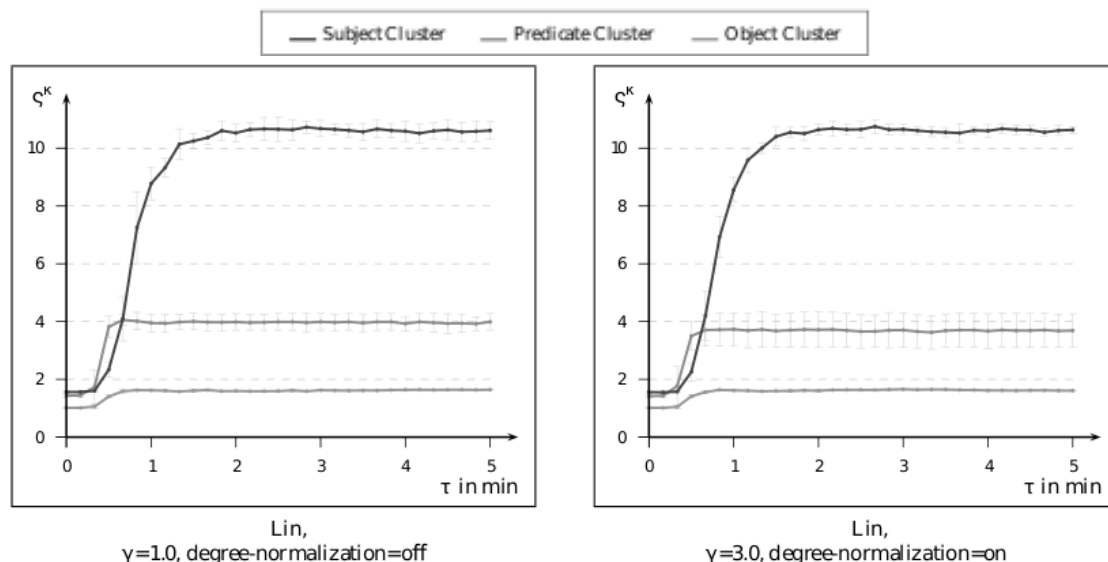


Figure 2: Measure local similarity gain

The evaluation in [Kos09] showed that the algorithms are capable of congregating triples that are similar in concept and forming semantic neighborhoods to a certain degree. Also it could be shown that the triple retrieval performs better than using a random walk for searching in the net-

work. In a next step we want to investigate how we can reach this with distributing the taxonomy over the network. In future work we want to implement and evaluate the suggested approach to add a syntactical layer where the *rdfs:subClassOf*-, *rdfs:subPropertyOf*- and *rdf:type*-Statements are clustered syntactically [Kos09]. This would have the effect that they would be distributed evenly over the network. Therefore it would be a convenient solution for huge concept taxonomies where one would not store the whole taxinomial information on each node.

5 Reasoning Layer

In general we assume a familiarity with the basic concepts of logic programming [Llo84], description logics [BCM⁺03], RDF [W3C04b], RDFS [W3C04c] and OWL[W3C04a].

For a knowledgebase KB given in a description logic dialect we denote its set of terminological axioms with KB_T and its set of assertions with KB_A . We refer with \mathcal{L}^2 to the FOL subset with equality, that only uses two different variables, constants, and monadic and dyadic predicates, but no functions. An *extended disjunctive logic program* over a first order language \mathcal{L} consists of logical inference rules of the form $L_1 \text{ or } \dots \text{ or } L_k \leftarrow L_{k+1}, \dots, L_m, \text{ not } L_{m+1}, \dots, \text{ not } L_n$, where each L_i is a literal A or $\neg A$ in \mathcal{L} . Furthermore, *not* refers to negation-as-failure and \neg to *classical (explicit or strong) negation*. Informally, the latter is a stronger negative statement than negation-as-failure, since $\neg A$ explicitly guarantees that the negation of A succeeds. For an extended disjunctive logic program \mathcal{P} we denote its Herbrand basis by $HB(\mathcal{P})$. In terms of *answer set semantics* [GL91] an *answer set* of \mathcal{P} is defined as: \mathcal{P} is transformed to a disjunctive logical program \mathcal{P}' without classical negation, by replacing each literal $\neg A = \neg p(t_1, \dots, t_n)$ with a positive literal $A' = p'(t_1, \dots, t_n)$ with a new predicate symbol p' . Each answer set AS of \mathcal{P} is defined by a stable model S for \mathcal{P}' , which itself a set of ground literals: $AS := \{A \in HB(\mathcal{P}) \mid A \in S\} \cup \{\neg A \in HB(\mathcal{P}) \mid A' \in S\}$; . If AS contains a pair of complementary literals A and $\neg A$, then AS is the set of ground literals over \mathcal{L} . In our context, classical negation succeeds, if we find an explicit negation of A , i.e., a literal $\neg A$. Thus, for each concept or role A in a knowledgebase KB we extend KB by default to $KB' := KB \cup \{\neg A \mid A \text{ is a role or concept in } KB\}$. Moreover, our swarm based reasoning layer uses *partial answer sets* [Prz91] and *brave inference* as semantic foundation: A ground atom can only be entailed from KB , if it lies in some answer set for KB .

5.1 From Terminological Axioms to Reasoning Ant Types

We now describe our swarm-based reasoning layer. We introduce the employed logic languages and model semantics, and give a description about the creation and behavior of reasoning ants.

Before all, we want to clarify, that at this very early stage of our research on swarm based reasoning we give an initial proof-of-concept, which, at least for the moment, consciously omits solutions for a number of well-known severe problems related to reasoning with description logics. With this in mind, we limit our swarm reasoning approach to \mathcal{ALC} , which in comparison to \mathcal{SHOIN} lacks of qualified number restrictions and transitive and inverse role descriptions. Our ultimate goal for the future, though, lies in the provision of reasoning support for \mathcal{SHOIN} . Furthermore, we disregard problems in conjunction with cyclic terminologies and, hence, assume as input only recursion-free TBoxes. We are aware of the fact, that the

Herbrand model semantics below imply UNA, which cannot be guaranteed for the general case.

Multiple concurrent changes of the ABox and, with much lower frequency, of the TBox are in general adherent to the nature of distributed knowledgebase systems. However, for the sake of simplicity, we here omit measures to prevent or mitigate the potentially emerging inconsistencies related to these manipulations. Thus, we expect for the moment, that ABox and TBox are not changed as long as the reasoning layer is activated.

In order to facilitate swarm-based reasoning upon a knowledgebase KB we reduce the assertional inference process for KB , i.e., application of axioms in KB_T against KB_A , to the behavior of a distinct set of ants. More precisely, the reduction translates KB_T into a set of ant types S_{inf} . These define behavior patterns for ants to implement our forward-chaining ABox reasoning process. Moreover, these ants probabilistically apply inference rules, which are disjunctive extended rules derived from KB_T , to the assertions of KB_A in the storage layer. Here, as model theoretic underpinning, we use partial answer set models and brave inference. Furthermore, our reasoning process is based on an open-world-assumption with explicit negation only. That is, in each rule all negation symbols are interpreted as explicit negation.

We now explain the three single steps of the translation from KB_T to S_{inf} in detail. First, we translate KB_T to FOL according to [Bor96]. More precisely, all \mathcal{ALC} formulas of KB are translated into \mathcal{L}^2 . For a terminological axiom ta we denote its corresponding \mathcal{L}^2 formula with $\mathcal{L}^2(ta)$, and we denote the set of formulas generated for KB_T with $\mathcal{L}^2(KB_T)$.

Secondly, we transform the formulas of $\mathcal{L}^2(KB_T)$ into prenex-normal-form with a matrix in conjunctive-normal-form. Apparently, we need now more than two different variables and, therefore, use general FOL with equality at this stage. Specifically, for each terminological axiom ta its formula $\mathcal{L}^2(ta)$ is transformed to a formula $PCNF(ta) := Q_1x_1 \dots Q_nx_k. \mathcal{C}_1 \wedge \dots \wedge \mathcal{C}_l$ where $Q_i \in \{\exists, \forall\}$, \mathcal{C}_i is a clause, and x_j is a variable occurring in \mathcal{C}_i . Under the pretense that all existential quantifiers are substituted by universal quantifiers each clause $\mathcal{C}_i = L_1 \vee \dots \vee L_n$ can be interpreted as two *extended disjunctive rules*

$$r_1(\mathcal{C}_i) := L_1 \text{ or } \dots \text{ or } L_m \leftarrow \overline{L_{m+1}}, \dots, \overline{L_n}, \quad (1)$$

$$r_2(\mathcal{C}_i) := \overline{L_{m+1}} \text{ or } \dots \text{ or } \overline{L_n} \leftarrow L_1, \dots, L_m, \quad (2)$$

under answer set semantics; there L_i is a literal A or $\neg A$ for an atom A , \neg denotes explicit negation, and $\overline{L_i} = \neg A$ or $\overline{L_i} = A$, respectively. We refer with $EDLP(ta)$ to the set of extended disjunctive rules for the clauses in $PCNF(ta)$. Despite our modification of $PCNF(KB_T)$ for creating $EDLP(KB_T)$, by replacing all existential quantifiers with universal ones, we will later see, that our probabilistic swarm-based approach still derives a (continuously growing) subset of all ground facts derivable from $PCNF(KB_T)$.

Eventually, we assign a reasoning ant type $\tau(ta)$ for each terminological axiom $ta \in KB_T$. An ant instance of $\tau(ta)$, intuitively, infers new assertions by finding assertions matching to the bodies of the rules in $EDLP(ta)$. More precisely, it tries to find a constant substitution θ for all variables in $EDLP(ta)$ such that for each clause \mathcal{C}_i occurring in $PCNF(ta)$ either $r_1(\mathcal{C}_i)$ or $r_2(\mathcal{C}_i)$ is true. For the determination of θ the ant searches the distributed storage, i.e., KB_A , for constants that appear in ground instances matching the body of $r_1(\mathcal{C}_i)$ or $r_2(\mathcal{C}_i)$ for every \mathcal{C}_i . To this end the ant efficiently tracks down relevant concepts, roles and their instances by behaving like a storage ant exploiting the the scents of concept and URI clusters in the storage layer while

performing a `rd` operation as described in Section 4. Furthermore, the ant embodies a local memory to store the adequate ground atoms found during this exploration.

If a substitution θ is found, the ant creates a new assertion for each \mathcal{C}_i : depending, if either the body of $r_1(\mathcal{C}_i)$ or $r_2(\mathcal{C}_i)$ under application of θ has been verified with respect to KB_A in the previous step, the ant creates a new ground atom $H_j\theta$ by randomly choosing a literal H_j from $r_1(\mathcal{C}_i)$'s or $r_2(\mathcal{C}_i)$'s head. After that, the ant behaves like a storage ant which puts $H_j\theta$, i.e., its encoding RDF triples, into the appropriate clusters by multiple `out` operations.

After we have learned, that a reasoning ant $\tau(ta)$ applies $EDLP(ta)$ only for a single matching substitution θ at a time, it seems apparent, that, despite the earlier replacement of existential qualifiers with universal ones in $PCNF(KB_T)$ in order to generate $EDLP(ta)$, only ground atoms derivable from $PCNF(KB_T)$ are entailed by reasoning ants. This has to be verified by a formal proof in the future. As an example for the introduced transformation consider a knowledgebase terminology KB_T comprising the axioms

$$\begin{aligned} \text{professor} \sqcup \text{assistant} &\sqsubseteq \text{researcher}, \\ \text{researcher} \sqcap \neg \text{professor} &\sqsubseteq \text{assistant}, \\ \text{assistant} &\sqsubseteq \exists \text{supervisedBy} . \text{researcher}. \end{aligned}$$

The corresponding $PCNF(KB_T)$ is

$$\begin{aligned} \forall x. ((\neg \text{professor}(x) \vee \text{researcher}(x)) \wedge (\neg \text{assistant} \vee \text{researcher}(x))), \\ \forall x. (\neg \text{researcher}(x) \vee \text{professor}(x) \vee \text{assistant}(x)), \\ \forall x. \exists y. ((\neg \text{assistant}(x) \vee \text{supervisedBy}(x, y)) \wedge (\neg \text{assistant}(x) \vee \text{researcher}(y))). \end{aligned}$$

The corresponding $EDLP(KB_T)$ is

$$\begin{aligned} \text{researcher}(x) \leftarrow \text{professor}(x); \neg \text{professor}(x) \leftarrow \neg \text{researcher}(x); \\ \text{researcher}(x) \leftarrow \text{assistant}(x); \neg \text{researcher}(x) \leftarrow \neg \text{assistant}(x); \\ \text{assistant}(x) \text{ or } \text{professor}(x) \leftarrow \text{researcher}(x); \\ \neg \text{researcher}(x) \leftarrow \neg \text{assistant}(x), \neg \text{professor}(x); \\ \text{supervisedBy}(x, y) \leftarrow \text{assistant}(x); \neg \text{assistant}(x) \leftarrow \neg \text{supervisedBy}(x, y); \\ \text{researcher}(y) \leftarrow \text{assistant}(x); \neg \text{assistant}(x) \leftarrow \neg \text{researcher}(y). \end{aligned}$$

Finally, we briefly compare our approach to the other recent swarm intelligence algorithm for ontology-based reasoning by Dentler et al. [DGS09]. Both offer swarm-based forward-chaining on RDF datasets with ontological background knowledge. However, in Dentler's approach the reasoning ants follow the RDF graph structure, whereas in our case they follow cents of concepts and roles. Also, Dentler's approach supports reasoning in RDFS while we support \mathcal{ALC} .

6 Conclusion and Future Work

We have developed a swarm-based concept of federated, decentralized storage and reasoning for RDF data combined with ontological information. In addition, the evaluation of already implemented swarm-based storage techniques yielded promising results with respect to scalability and adaptability. Moreover, in the reasoning layer we introduced a first swarm-based concept for assertional reasoning with \mathcal{ALC} vocabularies.

We plan to further extend our concepts for swarm-based storage and reasoning. Further investigations will be how the ants can get the taxinomial information without an extra query avoiding additional network traffic. One idea is to add the taxinomial information to the scents, where it is needed. This would have to be maintained by ants again. Furthermore we have to face the problem that resources can be assigned to different concepts that are not related by some ontology. How to deal with that fact has also to be investigated in future work. Methods to prevent or mitigate inconsistency problems inherent to distributed reasoning have to be added. Additional, our rule language and ant behavior patterns have to be extended to support assertional reasoning in *SHOIN*. Apparent problems with DL-reasoning, e.g. cyclic TBoxes, have to be addressed.

We will evaluate the combined implementation with representative benchmark tests for querying and reasoning. In particular, scalability in terms of number of node machines will be verified.

Acknowledgment This work has been partially supported by the “DigiPolis” project funded by the German Federal Ministry of Education and Research (BMBF) under the support code 03WKP07B. This work is partly funded by Science Foundation Ireland (SFI) project Lion-2 (SFI/08/CE/I1380) and an IRCSET postgraduate scholarship.

Bibliography

- [ABK⁺07] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, Z. Ives. DBpedia: A Nucleus for a Web of Open Data. In *In 6th Int'l Semantic Web Conference, Busan, Korea*. Pp. 11–15. Springer, 2007.
- [AKO⁺08] G. Anadiotis, S. Kotoulas, E. Oren, R. Siebes, F. van Harmelen, N. Drost, R. Kemp, J. Maassen, F. J. Seinstra, H. E. Bal. MaRVIN: a distributed platform for massive RDF inference. <http://www.larkc.eu/marvin/btc2008.pdf>, 2008.
- [BCM⁺03] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, P. F. Patel-Schneider (eds.). *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
- [BDT99] E. Bonabeau, M. Dorigo, G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Santa Fe Institute Studies in the Sciences of Complexity Series. Oxford Press, July 1999.
- [BMM94] T. Berners-Lee, L. Masinter, M. McCahill. RFC 1738: Uniform Resource Locators (URL). Dec. 1994.
- [Bor96] A. Borgida. On the Relative Expressiveness of Description Logics and Predicate Logics. *Artif. Intell.* 82(1-2):353–367, 1996.
- [CF04] M. Cai, M. Frank. RDFPeers: a scalable distributed RDF repository based on a structured peer-to-peer network. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*. Pp. 650–657. ACM, New York, NY, USA, 2004.
- [Den09] K. Dentler. Semantic Web Reasoning by Swarm Intelligence. Master's thesis, Vrije Universiteit Amsterdam, 2009.

- [DGS09] K. Dentler, C. Gueret, S. Schlobach. Semantic Web Reasoning by Swarm Intelligence. In *Proc. of Nature inspired Reasoning for the Semantic Web, ISWC*. 2009.
- [Gel85] D. Gelernter. Generative communication in Linda. *ACM Transactions on Programming Languages and Systems* 7:80–112, 1985.
- [GL91] M. Gelfond, V. Lifschitz. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing* 9:365–385, 1991.
- [GPH05] Y. Guo, Z. Pan, J. Heflin. LUBM: A benchmark for OWL knowledge base systems. *Web Semantics: Science, Services and Agents on the World Wide Web* 3(2-3):158–182, October 2005.
- [Kos09] S. Koske. Swarm Approaches for Semantic Triple Clustering and Retrieval in Distributed RDF Spaces. Technical report B-09-04B, FU Berlin, Inst. für Inf., 2009.
- [Lin98] D. Lin. An information-theoretic definition of similarity. In *Proc. 15th Int. Conf. on Machine Learning*. Pp. 296–304. Morgan Kaufmann, S.Francisco, CA, 1998.
- [Llo84] J. W. Lloyd. *Foundations of logic programming*. Springer-Verlag New York, Inc., New York, NY, USA, 1984.
- [MT03] R. Menezes, R. Tolksdorf. A New Approach to Scalable Linda-systems Based on Swarms. In *Proceedings of ACM SAC 2003*. Pp. 375–379. 2003.
- [NSKM08] L. J. B. Nixon, E. P. B. Simperl, R. Krummenacher, F. Martín-Recuerda. Tuplespace-based computing for the Semantic Web: a survey of the state-of-the-art. *Knowledge Eng. Review* 23(2):181–212, 2008.
- [Prz91] T. C. Przymusiński. Stable Semantics for Disjunctive Programs. *New Generation Computing* 9:401–424, 1991.
- [SKN07] E. P. B. Simperl, R. Krummenacher, L. J. B. Nixon. A Coordination Model for Triplespace Computing. In Murphy and Vitek (eds.), *COORDINATION*. Lecture Notes in Computer Science 4467, pp. 1–18. Springer, 2007.
- [TA09] R. Tolksdorf, A. Augustin. Selforganisation in a storage for semantic information. *Journal of Software* 4(8):798–807, 2009.
- [TNS08] R. Tolksdorf, L. Nixon, E. Simperl. Towards a tuplespace-based middleware for the Semantic Web. *Web Intelligence and Agent Systems: An International Journal* 6(3):235–251, 2008.
- [TW04] S. Tisue, U. Wilensky. NetLogo: A simple environment for modeling complexity. In Minai and Bar-Yam (eds.), *Proceedings of the Fifth International Conference on Complex Systems ICCS 2004*. Pp. 16–21. 2004.
- [W3C04a] W3C. OWL Web Ontology Language Overview, W3C Recommendation. Online <http://www.w3.org/TR/owl-features/>, February 2004.
- [W3C04b] W3C. *RDF Primer*. W3C Recommendation. World Wide Web Consortium, February 2004. <http://www.w3.org/TR/rdf-primer/>.
- [W3C04c] W3C. RDF Vocabulary Description Language 1.0: RDF Schema. Available online at <http://www.w3.org/TR/2004/REC-rdf-schema-20040210/>, February 2004.