Semantic Web Reasoning by Swarm Intelligence

Kathrin Dentler, Christophe Guéret, and Stefan Schlobach

Department of Artificial Intelligence, Vrije Universiteit Amsterdam, de Boelelaan 1081a, 1081HV Amsterdam, The Netherlands

Abstract. Semantic Web reasoning systems are confronted with the task to process growing amounts of distributed, dynamic resources. This paper presents a novel way of approaching the challenge by RDF graph traversal, exploiting the advantages of swarm intelligence. The nature-inspired and index-free methodology is realised by self-organising swarms of autonomous, light-weight entities that traverse RDF graphs by following paths, aiming to instantiate pattern-based inference rules. The method is evaluated on the basis of a series of simulation experiments with regard to desirable properties of Semantic Web reasoning, focussing on anytime behaviour, adaptiveness and scalability.

1 Introduction

Motivation It is widely recognised that new *adaptive* approaches towards *robust* and *scalable* reasoning are required to exploit the full value of ever growing amounts of dynamic Semantic Web data.[8] Storing all relevant data on only one machine is unrealistic due to hardware-limitations, which can be overcome by distributed approaches. The proposed framework employs twofold distribution: the reasoning task is distributed on a number of agents, i.e. autonomous micro-reasoning processes that are referred to as beasts in the remainder, and data can be distributed on physically distinct locations. This makes reasoning fully parallelisable and thus scalable whenever beasts do not depend on results of other beasts or data on other locations. In most use-cases, co-ordination between reasoning beasts is required, and this paper explores the application of swarm intelligence to achieve optimised reasoning performance.

A second problem of current reasoning methods that focus on batch-processing where all available information is loaded into and dealt within one central location, is that the provenance of the data is often neglected and privacy-issues are risen. An interesting alternative is local reasoning that supports decentralised publishing as envisioned in [19], allowing users to keep control over their *privacy* and the ownership and dissemination of their information. Another advantage of decentralised reasoning compared to centralised methods is that it has the potential to naturally support reasoning on constantly changing data.

Adaptiveness, robustness and scalability are characteristic properties of swarm intelligence, so that its combination with reasoning can be a promising approach. The aim of this paper is to introduce a swarm-based reasoning method and to provide an initial evaluation of its feasibility and major characteristics. A model

of a decentralised, self-organising system is presented, which allows autonomous, light-weight beasts to traverse RDF graphs and thereby instantiate pattern-based inference rules, in order to calculate the deductive closure of these graphs w.r.t. the semantics of the rules. It will be investigated whether swarm intelligence can contribute to reduce the computational costs that the model implies, and make this new reasoning paradigm a real alternative to current approaches.

Method In order to calculate the RDFS or OWL closure over an RDF graph, a set of entailment rules has to be applied repeatedly to the triples in the graph. These rules consist of a precondition, usually containing one or more triples as arguments, and an action, typically to add a triple to the graph. This process is usually done by indexing all triples and joining the results of separate queries. Swarm-based reasoning is an index-free alternative for reasoning over large distributed dynamic networks of RDF graphs.

The idea is simple: an RDF graph is seen as a network, where each subject and each object is a node and each property an edge. A path is composed of several nodes that are connected by properties, i.e. edges. The beasts, each representing an active reasoning rule, which might be (partially) instantiated, move through the graph by following its paths. Swarms of independent light-weight beasts *travel* from *RDF node to RDF node* and from *location to location*, checking whether they can derive new information according to the information that they find on the way. Whenever a beast traverses a path that matches the conditions of its rule, it locally adds a new derived triple to the graph. Given an added transition capability between (sub-)graphs, it can be shown that the method converges towards closure.

Research questions The price for our approach is redundancy: the beasts have to traverse parts of the graph which would otherwise never be searched. It is obvious that repeated random graph traversal of independent beasts will be highly inefficient. The trade-off that needs to be investigated is thus, whether the overhead can be reduced so that the method offers both adaptive and flexible, as well as sufficiently efficient reasoning. The main research question of this paper is whether *Swarm Intelligence* can help to guide the beasts more efficiently, so that the additional costs are out-balanced by a gain in adaptiveness. More specifically, the following research questions are going to be answered:

1. Does a swarm of beasts that is co-ordinated by stigmergic communication perform better than the same number of independent beasts?
2. Does adaptive behaviour of the population lead to increased reasoning performance?
3. How does the reasoning framework react to a higher number of locations?

Implementation and Experiments To prove the concept, a prototypic system has been implemented based on AgentScape[15]. Each beast is an autonomous reasoning agent, and each distributed graph administered by an agent that is referred to as dataprovider and linked to a number of other dataproviders. Based

on this implementation, the feasibility and major characteristics of the approach are evaluated on the basis of simulation experiments in which beasts calculate the deductive closure of RDF graphs[2] w.r.t. RDFS Semantics[11]. To study the properties of the approach in its purest form, the focus is explicitly restricted to the most simple instantiation of the model of swarm-based reasoning. This means that the answers to the research questions are preliminary.

Findings The experiments described in this paper have two goals: proof of concept and to obtain a better understanding of the intrinsic potential and challenges of the new method. For the former, fully decentralised beasts calculate the semantic closure of a number of distributed RDF datasets. From the latter perspective, the lessons learned are less clear-cut, as the results confirm that tuning a system based on computational intelligence is a highly complex problem. However, the experiments give crucial insights in how to proceed in future work; most importantly on how to improve attract/repulse methods for guiding swarms to interesting locations within the graph.

What to expect from this paper This paper introduces a new swarm-based paradigm for reasoning on the Semantic Web. The main focus is to introduce the general reasoning framework to a wider audience and to study its weakness and potential on the most general level.

We will provide background information and discuss related work in the next section 2, before we define our method in section 3. We present our implementation in section 4 and some initial experiments in section 5, before we discuss some ideas for future research and conclude in section 6.

2 Background and Related Work

In this section, we provide a brief overview of RDFS reasoning and Swarm Intelligence and discuss existing approaches towards distributed reasoning.

The challenge we want to address is to deal with truly decentralised data, that each user keeps locally. Bibliographic data is an example that would ideally be maintained by its authors and directly reasoned over. We will use this scenario throughout the paper to introduce our new Semantic Web reasoning paradigm.

2.1 Semantic Web Reasoning

To simplify the argument and evaluation, we focus on RDF and RDFS (RDF Schema), the two most widely used Semantic Web languages.[1]

Listing 1.1 shows two simple RDF graphs in Turtle notation about two publications `cg:ISWC08` and `fvh:SWP` of members of our Department, maintained

[1] It is relatively straightforward to extend our framework to other rule-based frameworks, such as OWL-Horst reasoning, which is currently the most widely implemented form of Semantic Web reasoning.

separately by respective authors and linked to public ontologies `pub` and `people` about publications and people[2], and reasoned and queried over directly.

```
cg:ISWC08
  pub:title "Anytime Query Answering in RDF through Evolutionary Algorithms" ;
  pub:publishedAs pub:InProceedings ;
  pub:author people:Gueret ;
  pub:author people:Oren ;
  pub:author people:Schlobach ;
  pub:cites fvh:SWP .

fvh:SWP
  pub:title "Semantic Web Primer" ;
  pub:publishedAs pub:Book ;
  pub:author people:Antoniou ;
  pub:author people:vanHarmelen .
```

Listing 1.1. Two RDF graphs about publications

These two graphs describe two publications `cg:ISWC08` and `fvh:SWP` by different sets of authors and are physically distributed over the network. The statements contained in the graphs are extended with schema-information as shown in listing 1.2 and defined in the two respective ontologies, for example with the information that `pub:InProceedings` are `pub:Publications`, `people:Persons` are `people:Agents`, or that `pub:author` has the range `people:Person`.

```
pub:InProceedings rdfs:subClassOf pub:Publication
people:Person rdfs:subClassOf people:Agent
pub:author rdfs:range people:Person
```

Listing 1.2. Some RDFS statements

Given the standard RDFS semantics, one can derive that `cg:ISWC08` is a publication, and that authors are also instances of class `people:Person`, and thus `people:Agent`. The formal semantics of RDFS and OWL enable the automation of such reasoning. The task addressed in the experiments is to calculate the RDF(S) deductive closure, i.e. all possible triples that follow implicitly from the RDF(S) semantics[11]. Table 1 lists some examples of these entailment rules, where the second column contains the condition for a rule to be applied, and the third column the action that is to be performed.[3]

2.2 Swarm Intelligence

As our alternative reasoning method is swarm-based, let us give a short high-level introduction to the field of Swarm Intelligence. Inspired by the collective behaviour of flocks of birds, schools of fish or social insects such as ants or bees, Swarm Intelligence investigates self-optimising, complex and highly structured systems. Members of a swarm perform tasks in co-operation that go beyond the

[2] In the experiments, SWRC and FOAF are used, but to simplify the presentation of the example, ontology names are generic.

[3] The rules follow the conventions: p denotes a predicate, i.e. a URI reference, s a subject, i.e. a URI reference or a blank node, and o refers to an object (which might also be a subject for an ongoing triple), i.e. a URI reference, a blank node or a literal.

Rule	If graph contains	Then add
rdfs2	p rdfs:domain o_1 . and s p o_2 .	s rdf:type o_1 .
rdfs3	p rdfs:range o . and s_1 p s_2 .	s_2 rdf:type o .
rdfs4a	s p o .	s rdf:type rdfs:Resource .
rdfs7	p_1 rdfs:subPropertyOf p_2 . and s p_1 o .	s p_2 o .
rdfs9	s_1 rdfs:subClassOf o . and s_2 rdf:type s_1 .	s_2 rdf:type o .

Table 1. RDFS entailment rules

capabilities of single individuals, which function by basic stimulus → response decision rules. Swarms are characterised by a number of properties, most importantly *lack of central control*, *enormous sizes*, *locality* and *simplicity*. Those properties result in advantageous characteristics of swarms, such as adaptiveness, flexibility, robustness, scalability, decentralisation, parallelism and intelligent system behaviour. These are also desirable in distributed applications, making swarms an attractive model for bottom-up reverse engineering.

Formicidae Ant (Formicidae) colonies appear as super-organisms because cooperating individuals with tiny and short-lived minds operate as a unified entity. Large colonies are efficient due to the self-organisation and functional specialisation of their members. Another characteristic property of ant colonies is their ability to find shortest paths by indirect communication based on chemical pheromones that they drop in the environment. The pheromones act as a shared extended memory and enable the co-ordination of co-operating insects. This mechanism is known as stigmergy[4].

2.3 Related Work

Scalability issues that are risen by the growing amount of Semantic Web data are addressed by projects such as the Large Knowledge Collider[9], a platform for massive distributed incomplete reasoning systems. Calculating the deductive closure with respect to RDFS entailment rules is a standard problem on the Semantic Web and has been addressed extensively. Relevant is the recent work on distributed reasoning. The most prominent paradigms are based on various techniques to distribute data [14,12,1,6] towards several standard reasoners and to combine the results. This is different to the swarm-based methodology, where reasoning is distributed over data that remains local at distributed hosts, so that only small bits of information are moved in the network. Probably closest to this methodology is the distributed resolution approach proposed in [16], but it requires more data to be exchanged than our light-weight swarm-approach.

Biologically inspired forms of reasoning are an emerging research area that aims at modelling systems with intelligent properties as observed in nature. Semantic Web reasoning by Swarm Intelligence has been suggested in [3], that proposes the realisation of "knowledge in the cloud" by the combination of modern "data in the cloud" approaches and Triplespace Computing. Triplespace

Computing[7] is a communication and co-ordination paradigm that combines Web Service technologies and semantic tuplespaces, i.e. associative memories as embodied in Linda[10]. This combination allows for the persistent publication of knowledge and the co-ordination of services which use that knowledge. The envisioned system includes support for collaboration, self-organisation and semantic data that can be reasoned over by a swarm which consists of micro-reasoning individuals that are able to move in and manipulate their environment. The authors suggest to divide the reasoning task among co-operating members of a semantic swarm. Limited reasoning capabilities of individuals and the according reduction of required schema information result in the optimisation of the reasoning task for large-scale knowledge clouds.

3 Semantic Web Reasoning as Graph Traversal

In this section, we introduce our new reasoning methodology, which is based on autonomous beasts that perform reasoning by graph traversal. Given a possibly distributed RDF graph and corresponding schemata, beasts expand the graph by applying basic inference rules on the paths they visit.

3.1 Reasoning as Graph Traversal

When a beast reaches a node, it chooses an ongoing path that starts at the current node. This decision can be taken based on pheromones that have been dropped by previous beasts, or based on the elements of the triples, preferring triples which correspond to the pattern of the inference rule (best-first or hit-detection). If the chosen triple matches the beast's pattern, the rule will be fired and the new inferred triple added to the graph.

Given three infinite sets I, B and L respectively called URI references, blank nodes and literals, an *RDF triple* (s, p, o) is an element of $(I \cup B) \times I \times (I \cup B \cup L)$. Here, s is called the subject, p the predicate, and o the object of the triple. An *RDF graph* G (or graph or dataset) is then a set of RDF triples.

Definition 1 (Reasoning as Graph Traversal). *Let G be an RDF graph, N_G the set of all nodes in G and $M_G = (L \cup I) \times \ldots \times (L \cup I)$ the memory that each beast is associated with.* RDF graph traversal reasoning *is defined as a triple (G, B_G, M_G), where each $b \in B_G$ is a transition function, referring to a* (reasoning) beast *$rb : M_G \times G \times N_G \rightarrow M_G \times G \times N_G$ that takes as input a triple of the graph, moves to an adjacent node in the graph, and depending on its memory, possibly adds a new RDF triple to the graph.*

RDFS reasoning can naturally be decomposed by distributing complementary entailment rules on the members of the swarm, so that each individual is responsible for the application of only one rule. Therefore, we introduce different types of beasts, one type per RDF(S) entailment rule containing schema information. If a concrete schema triple of a certain pattern is found, a corresponding reasoning beast is generated. Regarding for example the rule rdfs3

from Table 1, which deals with range restrictions: whenever in the schema an axiom `p rdfs:range x` is encountered, which denotes that every resource in the range of the property `p` must be of type `x`, a beast responsible for this bit of schema information is initialised as a function $rb3$ that is associated to memory $\{p, x\}$. Table 2 lists the RDFS entailment rules, omitting blank node closure rules rdf2 and rdfs1, with the pattern that is to be recognised in column 2 and the reasoning beast with its memory requirement in column 3.

Entailment rule	**Pattern of schema triple**	**Beast: memory**
rdfs2	p rdfs:domain x .	rb2: p x
rdfs3	p rdfs:range x .	rb3: p x
rdfs5	p_1 rdfs:subPropertyOf p . p rdfs:subPropertyOf p_2 .	rb7: p_1 p_2
rdfs6	p rdf:type rdf:Property .	rb7: p p
rdfs7	p_1 rdfs:subPropertyOf p_2 .	rb7: p_1 p_2
rdfs8	c rdf:type rdfs:Class .	rb9: c rdfs:Resource
rdfs9	c_1 rdfs:subClassOf c_2 .	rb9: c_1 c_2
rdfs10	c rdf:type rdfs:Class .	rb9: c c
rdfs11	c_1 rdfs:subClassOf c . c rdfs:subClassOf c_2 .	rb9: c_1 c_2
rdfs12	p rdf:type rdfs:ContainerMembershipProperty .	rb7: p rdfs:member
rdfs13	s rdf:type rdfs:Datatype .	rb9: s rdfs:Literal

Table 2. Schema-based instantiation of reasoning beasts

Table 3 shows the beasts needed for RDFS reasoning with their pattern-based inference rules. Underlined elements correspond to the memory. Reasoning beasts $rdf1b$, $rb4a$ and $rb4b$ are schema-independent and do not require any memory. They infer for each predicate that it is of `rdf:type rdf:Property` and for each subject and object that it is of `rdf:type rdfs:Resource`. Reasoning beasts $rb2$ and $rb3$ apply the semantics of `rdfs:domain` and `rdfs:range`, while beasts $rb7$ and $rb9$ generate the inferences of `rdfs:subPropertyOf` and `rdfs:subClassOf`. From now on, they are being referred to as domain-beast, range-beast, subproperty-beast and subclass-beast.

Beast : memory	**If pattern**	**Then add**
rdf1b : $\emptyset$	s p o .	p rdf:type rdf:Property .
rb2 : {p,x}	s p o .	s rdf:type x .
rb3 : {p,x}	s p o .	o rdf:type x .
rb4a : $\emptyset$	s p o .	s rdf:type rdfs:Resource .
rb4b : $\emptyset$	s p o .	o rdf:type rdfs:Resource .
rb7 : {p_1,p_2}	s p_1 o .	s p_2 o .
rb9 : {c_1,c_2}	s rdf:type c_1 .	s rdf:type c_2 .

Table 3. Inference patterns of reasoning beasts

Let us assume that a range-beast arrives at node o from a node s via an edge (s, p, o). Because p corresponds to its remembered property, it writes the triple `(o, rdf:type, x)` to the graph. As it can walk both directions of the directed

graph, it will output the same triple when it arrives at node s from node o via edge (o, p, s). Finally, it moves on to a new destination n via a property p_i, where $(o, p_i, n) \in G$.

There are many design decisions in the creation of beasts: in our prototypical implementation, all schema triples are retrieved and pre-processed, calculating the subclass and subproperty closure, before the beasts are created. For this reason, there is no one-to-one mapping between RDFS rules and beasts. For example, the transitivity of `rdfs:subClassOf` is first exhausted, so that rule rdfs11 is encoded with several beasts of type $rb9$. The more generic approach would have been to introduce a special transitivity-beast, which also writes new subclass triples to the graph and then to have beasts picking up the schema-information that they are to apply. This option is to be investigated in future work. An advantage of the proposed mechanism is that all reasoning patterns require only one matching triple for the rule to be fired, so that inferences do not depend on remote data.

Example Let us consider again the two RDF graphs from our previous example. Fig. 1 shows the RDF graph for the first publication. Dashed arrows denote implicit links derived by reasoning.

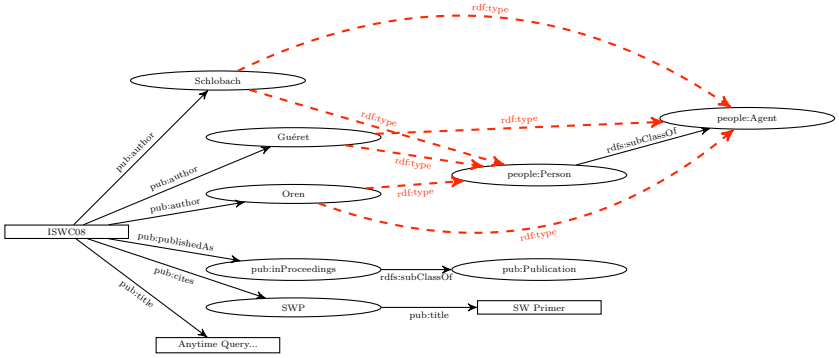


Fig. 1. An exemplary RDF graph

For the three schema axioms of the previous example, beasts are created. For the range-triple `pub:author rdfs:range people:Person`, a range-beast $rb3_1$ is created with memory `pub:author` and `people:Person`. For the subclass triple `people:Person rdfs:subClassOf people:Agent`, a beast $rb9_1$ is created which is instantiated with the memory bits `people:Person` and `people:Agent` (the other subclass-beast is generated accordingly). In this example, only one beast per instantiated type is created, in practise there will be more. The beasts are randomly distributed over the graph, say $rb3_1$ to node `fvh:SWP`, and similarly the other two beasts. Beast $rb3_1$ has now two options to walk. Moving to "SW Primer" will lead it to a cul-de-sac, which means it needs to walk back via

`cg:ISWC08` towards, e.g. `person:Oren`. At node `person:Oren`, the walked path is `cg:ISWC08 pub:author person:Oren` which means $rb3_1$'s pattern matches the walked triple, and it will add a triple `person:Oren rdf:type people:Person` to the graph. When, after walking other parts of the graph, the subclass beast $rb9_1$ chooses to follow the new `rdf:type` link from `person:Oren` to `people:Person`, it finds its memory condition matched, and will add the triple `person:Oren rdf:type people:Agent` to the graph, and so forth. This example highlights the obvious challenges faced by the approach, most importantly, that unnecessary paths need to be investigated and that the order of visiting beasts is important ($rb3_1$ had to be at `person:Oren` first, before $rb9_1$ could find anything).

Completeness The closure C^* over a dataset G contains all triples that follow from the RDF(S) semantics. In our framework, entailment rules are instantiated by the schemata and embodied by beasts. Let $b_1, \dots b_n$ be a swarm with at least one individual per type. The complete closure C^* is derived when the union of the beast-outputs $b_1(c_1) \cup \dots \cup b_n(c_n) \equiv C^*$.

Proposition 1 (Completeness). *Reasoning as graph traversal converges towards completeness.*

Proof. Sketch: To prove that the method converges towards completeness, it has to be shown that all elements of C^* are inferred eventually, i.e. that each beast b_m infers the complete closure $b_m(c^*_m)$ of the rule it incorporates. Given the beast-function as defined above, a beast infers c^*_m when it visits all triples of the graph that match its inference-pattern. This can be achieved by complete graph traversal, which is trivially possible and can be performed according to different strategies, such as random walk, breadth- or depth- first. It has to be performed repeatedly, as other beasts can add relevant triples. C^* is reached when the swarm performed a complete graph traversal without adding a new inference to the graph. Given random jumps to other nodes within the graph, which also prevents beasts from getting stuck in local maxima, the same holds for unconnected (sub-)graphs. When a swarm consists of s members $b^1_m, \dots b^s_m$ per type, the individuals of one type can infer $b_m(c^*_m)$ collectively.

The proposed method is sound but redundant, as two beasts can derive the same inference by the application of different rules and superfluous schema information can lead to inferences that are already present in the data. Beasts produce new inferences gradually, and as those are added to the corresponding graph and not deleted, the degree of completeness increases monotonically over time, so that our methodology shows typical anytime behaviour.

Jumping distributed graphs For the time being, our examples were not distributed. Our framework is more general as beasts can easily move to other locations in the network to apply their inference rules there. Challenging is the case when reasoning requires statements of two unconnected (sub-)graphs that are physically distributed over a network (which is not the case for the proposed

RDFS reasoning method but would be for distributed OWL-Horst[17] reasoning). In our current implementation, this case is covered by a simple routing strategy using Bloom filters [1].

Movement Control The reasoning beasts operate on an energy metaphor: moving in the graph costs energy and finding new inferences, which can be interpreted as food, is rewarding. This is modelled via a happiness-function. When the beasts are created, they have an initial happiness-value. Then, at each step from RDF node to RDF node, one happiness-point is subtracted. When a beast receives a reward due to an inference that is new (not-new inferences are worthless), its happiness increases. With this simple mechanism, a beast adapts to its environment: while it is successful and thus happy, it will probably find even more applications of its rule, whereas for a beast that did not find a new triple for a long time, it might be little reasonable to continue the search. When a beast is unhappy or unsuccessful (i.e. it did not find anything new for a given number of node-hops), it can change its rule instantiation, its type, the location or die.

3.2 Distributed Reasoning by Beasts in a Swarm

An advantageous property of populations that are organised in swarms is their ability to find shortest paths to local areas of points of interest. Inspired by ant colonies that lay pheromone-paths to food sources, beasts choose ongoing paths based on pheromones. The environment provides beasts with the information who has been at their current location before, which can be utilised by the swarm to act as a whole, to avoid loops and to spread out evenly, which is useful because the swarm has no gain if members of the same rule-instantiation traverse the same path more than once.

When a beast reaches a node and chooses the next ongoing edge, it parses its options, and while no application of its inference-pattern is found, which would cause it to choose the corresponding path and fire its rule, it applies a pheromone-based heuristic. It categorizes the options into sets of triples: the ones which are promising because no individual of its rule-instantiation has walked them before and the ones that already have been visited. If promising options are available, one of them is chosen at random, otherwise the ongoing path is chosen probabilistically, preferring less visited triples. Only pheromones of beasts with the same rule-instantiation are considered, other possibilities, such as preferring pheromones of other beasts to follow them, are subject to further research. Let τ_j denote the pheromones on a path j and n the number of paths. The probability p_i to choose path i is determined by equation 1. It is between 0 and 1 and higher for paths that have been walked less in the past. This formula has been inspired by Ant Colony Optimization[5], where the probability to choose a path i is τ_i, i.e. the pheromones on that path, divided by the sum of the pheromones on all options.

$$p_i = \frac{\frac{\sum_{j=0}^{n} \tau_j - \tau_i}{n-1}}{\sum_{j=0}^{n} \tau_j}. \tag{1}$$

4 Implementation

The implementation of our model is based on AgentScape [15], a middleware layer that supports large-scale agent systems. Relevant concepts are locations, where agents can reside, and agents as active entities that are defined according to the weak notion of agency [18], which includes autonomy, social ability, reactiveness and pro-activeness. Agents can communicate by messages and migrate from one location to another.

Both the environment and the reasoning beasts of our model are implemented as AgentScape agents. The environment consists of a number of agents that hold RDF (sub-)graphs in Jena [13] models and are called dataprovider. We assume that each dataprovider resides on a distinct location and does not migrate. Beasts do migrate and communicate directly with the dataprovider that resides on their current location. Reasoning beasts migrate to the data, perform local calculations and move on to the next location, so that only the code of the beasts and results that can lead to inferences on other locations are moved in the network and not the data. Other set-ups would be possible, such as beasts querying dataproviders from the distance, without being physically on the same location. Beasts communicate indirectly with each other by leaving pheromone-traces[4] in the environment. They operate on a plain N3 text representation.

5 Research Questions and Corresponding Experiments

This section presents a series of experiments. The focus is not on fine-tuning parameters, but on testing basic strategies to generate first answers to the research questions, and to provide a clear baseline for further experiments. The experiments are based on a number of publication.bib files of members of our Department. The files have been converted to RDF, following the FOAF and the SWRC ontologies.

Beasts are instantiated by our beast-creation mechanism as presented above.[5] Ignoring schema-triples that contain blank nodes, 195 beasts are generated from the employed FOAF and SWRC schemata: 11 subproperty-beasts, 87 subclass-beasts, 48 domain-beasts and 49 range-beasts, each of them with a unique rule-instantiation. In all experiments, swarms of 5 beasts per rule-instantiation are traversing the graphs. Each beast starts at a random node at one of the locations.

Each dataset is administered by a dataprovider that is residing on a distinct location, which is named after the corresponding graph. The initial happiness on each dataset and the maximum number of requests that do not lead to any new

[4] Pheromones are stored as reified triples.

[5] For the sake of simplicity, the experiments are restricted to RDFS simple reasoning. This means that RDFS entailment rules which have only one single schema-independent triple-pattern in the precedent (rdf1, rdfs4a and rdfs4b, rdfs6, rdfs8, rdfs10, rdfs12 and rdfs13) and blank node closure rules are deliberately omitted. These inferences are trivial and can be drawn after the RDFS simple closure has been derived, in case they are required.

inference, is set to 100 and the maximum number of migrations between locations and their corresponding dataproviders is set to 10. A beast's happiness increases by 100 when it inferred a triple that was new indeed. In all experiments, beasts ideally die when the closure is deduced. Then, they cannot find new inferences any more and eventually reach their maximum unhappiness after a number of unsuccessful node- and location-hops.

To evaluate the obtained results, the inferences of the beasts are compared to the output of Jena's RDFS simple reasoner.[6] All graphs show how the degree of completeness (i.e. the percentage of found inferences) per dataset is rising in relation to the number of sent messages. A message is a request for ongoing options from a beast to the dataprovider.

5.1 Baseline: random walk

To generate a baseline for the comparison of different strategies, the beasts in the first experiment choose between their options randomly, even if one of the options contains a triple that leads to an inference.

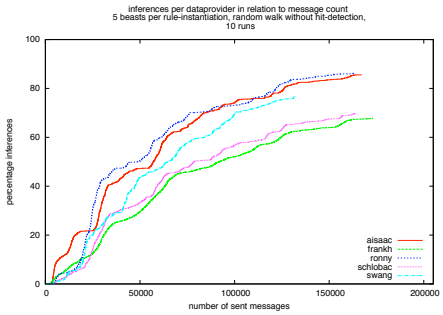


Fig. 2. Random walk without hit-detection

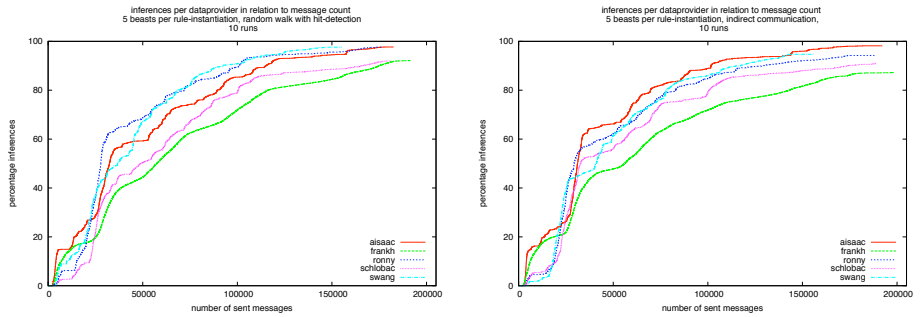
Fig. 2 visualizes the percentage of found inferences over time for each of the datasets. At the end of the experiment, 75.59% of the complete closure have been reached. The results demonstrate typical anytime behaviour: for longer computation times, more inferences are generated.

5.2 Does a swarm of beasts that employs stigmergic communication outperform the same number of independent beasts?

To investigate the question whether stigmergic communication accelerates convergence, a random walk strategy with hit-detection (so that the beasts prefer

[6] In contrast to the beasts, Jena did not generate any inference stating that a resource is of type Literal, so those triples have been added to the output of Jena.

triples that match their reasoning pattern) is compared to a strategy that employs indirect communication by repulsing pheromones, as described in section 3.2. Fig. 3(a) and Fig. 3(b) visualize the results.



(a) Random walk with hit-detection (b) Indirect communication

Fig. 3. Random walk with hit-detection compared to stigmergic communication

A first result is that compared to the random walk without hit-detection, both methods have a positive impact on the reasoning performance of the beasts. Secondly, in contrast to the hypothesis that stigmergic communication would lead to a faster convergence, the graphs are almost alike in the number of sent messages and also in their convergence-behaviour and reached percentage of the closure (94.52% for the random walk and 92.85% for the stigmergic communication).

In the start-phase of the experiments, new inferences are found easily, the difficult task is to detect the last remaining inferences. The assumption was that repulsing pheromones would guide the beasts to the unvisited sub-graphs where the remaining inferences are found. These experiments provide interesting insights into how to proceed in tuning the swarm behaviour: especially by the employment of attracting pheromones to lead fellow beasts to regions with food.

5.3 Does adaptive behaviour of the population increase the reasoning-performance?

To answer the question whether adaptive behaviour of the individuals is beneficial, a dynamic adaption of rule-instantiations in case of unsuccessfulness is tested. Note that this is only one possibility out of many for the beasts to adapt to their environment. Here, each beast is provided with a set of possible rule-instantiations instead of remembering only two arguments. For example, a subclass-beast does not only receive one subclass and one superclass to instantiate its rule, but several subclasses with the same superclass. The beasts switch their rule-instantiation to another random instantiation after each 50^{th} unsuccessful message-exchange. Fig. 4 shows the results of the adaptive beasts:

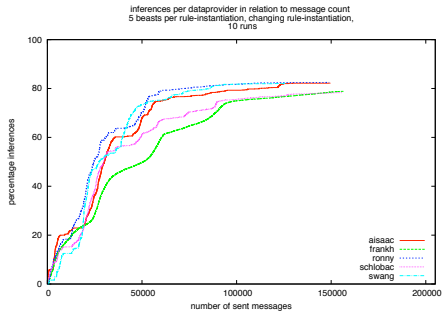


Fig. 4. Beasts that change their rule instantiation

The resulting graph shows clearly that the intuition that dynamic adaptions should lead to increased reasoning-performance did not apply to this experiment. On average, the swarms found 80.67% of the complete closure. In the future, more intelligent approaches have to be investigated, replacing e.g. the randomness in the changes, so that beasts change into successful rule-instantiations. Care has to be taken that not all beasts change to the most successful instantiations, because inferences that occur seldom also need to be found.

5.4 How does the reasoning framework react to a higher number of locations?

An important question regarding the potential scalability of our model is to study what happens when more datasets are added, so we compared the set-up with 5 datasets as shown in figure 3(b) to the same set-up with 10 datasets. We employed the same number of beasts as in the previous experiments.

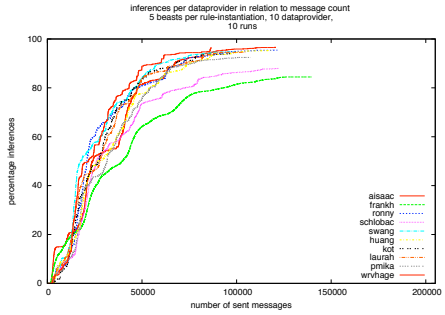


Fig. 5. 10 datasets

Fig. 5 demonstrates that the percentage of the complete closure (93.2%) per dataset is nearly as high as for a set-up with 5 datasets. This is because beasts

that can infer more new triples are increasingly happy and thus can send more messages to the dataproviders. The figure clearly indicates that more data on distributed locations lead to more activity, resulting in more inferences. This is, although not unexpected, a nice result.

6 Conclusion

We have presented a radically new method for Semantic Web reasoning based on Swarm Intelligence. The major feature of this idea is that many light-weight autonomous agents collectively calculate the semantic closure of RDF(S) graphs by traversing the graphs, and apply reasoning rules to the nodes they are currently located on. The advantage of this approach is its adaptiveness and its capability to deal with distributed data from dynamic sources. Such a reasoning procedure seems ideal for the Semantic Web and might help in setting up a decentralised publishing model that allows users to keep control over their personal data.

A metaphor for the way that the proposed paradigm envisages future Semantic Web reasoning is the *eternal adaptive anthill*, the Web of Data as decentralised accessible graphs, which are constantly traversed and updated by micro-reasoning beasts. Based on this vision it can be claimed that swarm-based reasoning is in principle more adaptive and robust than other Semantic Web reasoning approaches, as recurrently revisiting beasts can more easily deal with added (and even deleted) information than index-based approaches.

Our experiments show a proof of concept: over an (admittedly small) decentralised environment of our departmental publications we show that this idea works in principle, which gives us motivation to continue to improve the current framework in future research. Furthermore, first experiments have given clear guidelines for where more research is needed: first, the implementation framework based on AgentScape and Jena might not be ideal, as the dataprovider might be too central to the current implementation and thus become a bottleneck. However, through distribution of the reasoning, scaling is in principle straightforward. As usual, scaling comes at a price, in our case that the distribution will make it difficult for the beasts to find triples they can reason on. Initial experiments with Swarm Intelligence indicate that this might be a way forward, although it is well known that tuning such highly complex systems is very difficult (as our experiments confirm).

Ideas for future work are abundant: improving the implementation of the general model, routing strategies, security issues, dealing with added and deleted statements, trust etc. All these issues can relatively elegantly be represented in our framework, and some solutions are straightforward. Most interesting at the current stage will be to investigate more swarm strategies, such as scout models to guide beasts of similar types towards areas of interest, cloning successful beasts and much more. This paper is a first step.

References

1. M. Cai and M. Frank. RDFPeers: a scalable distributed RDF repository based on a structured peer-to-peer network. In *Proceedings of the 13th international conference on World Wide Web*, pages 650–657, 2004.
2. J. J. Carroll and G. Klyne. Resource Description Framework (RDF): Concepts and Abstract Syntax. *W3C recommendation*, 2004.
3. D. Cerri, E. Della Valle, D. De Francisco Marcos, F. Giunchiglia, D. Naor, L. Nixon, D. Rebholz-Schuhmann, R. Krummenacher, and E. Simperl. Towards Knowledge in the Cloud. *Proceedings of the OTM Confederated International Workshops and Posters on On the Move to Meaningful Internet Systems: 2008 Workshops: ADI, AWeSoMe, COMBEK, EI2N, IWSSA, MONET, OnToContent+ QSI, ORM, PerSys, RDDS, SEMELS, and SWWS*, pages 986–995, 2008.
4. M. Dorigo, E. Bonabeau, G. Theraulaz, et al. Ant algorithms and stigmergy. *Future Generation Computer Systems*, 16(9):851–871, 2000.
5. M. Dorigo and T. Stützle. *Ant Colony Optimization*. MIT press, Cambridge, MA, 2004.
6. Q. Fang, Y. Zhao, G. Yang, and W. Zheng. Scalable Distributed Ontology Reasoning Using DHT-Based Partitioning. In *The Semantic Web*, volume 5367, pages 91–105. Springer, 2008.
7. D. Fensel. Triple-space computing: Semantic Web Services based on persistent publication of information. *LNCS*, 3283:43–53, 2004.
8. D. Fensel and F. van Harmelen. Unifying Reasoning and Search to Web Scale. *IEEE Internet Computing*, 11(2):96–95, 2007.
9. D. Fensel, F. van Harmelen, Andersson, and et al. Towards LarKC: a platform for web-scale reasoning. In *Proceedings of the International Conference on Semantic Computing*, pages 524–529, 2008.
10. D. Gelernter. Generative communication in Linda. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 7(1):80–112, 1985.
11. P. Hayes and B. McBride. RDF semantics. *W3C recommendation*, 2004.
12. Z. Kaoudi, I. Miliaraki, and M. Koubarakis. RDFS Reasoning and Query Answering on Top of DHTs. In *LNCS*, volume 5318, pages 499–516, 2008.
13. B. McBride. Jena: Implementing the rdf model and syntax specification. In *Proc. of the 2001 Semantic Web Workshop*, 2001.
14. E. Oren, S. Kotoulas, G. Anadiotis, R. Siebes, A. ten Teije, , and F. van Harmelen. Marvin: A platform for large-scale analysis of Semantic Web data. In *Proceedings of the International Web Science conference*, 2009.
15. B. J. Overeinder and F. M. T. Brazier. Scalable Middleware Environment for Agent-Based Internet Applications. *LNCS*, 3732:675–679, 2006.
16. A. Schlicht and H. Stuckenschmidt. Distributed Resolution for ALC. In *Description Logics*, 2008.
17. H. J. ter Horst. Combining RDF and part of OWL with rules: Semantics, decidability, complexity. In *Proc. of ISWC*, pages 6–10. Springer, 2005.
18. M. Wooldridge and N. R. Jennings. Intelligent Agents: Theory and Practice. *Knowledge Engineering Review*, 10(2):115–152, 1995.
19. C. A. Yeung, I. Liccardi, K. Lu, O. Seneviratne, and T. Berners-Lee. Decentralization: The Future of Online Social Networking. Available at: `http://www.w3.org/2008/09/msnws/papers/decentralization.pdf`, 2008.