COORDINATING KNOWLEDGE IN PERVASIVE ENVIRONMENTS

Lyndon Nixon, Robert Tolksdorf
Institute of Computer Science
Networked Information Systems
Free University of Berlin
Takustr. 9, 14195 Berlin, Germany
nixon, tolk@inf.fu-berlin.de

Alan Wood
Department of Computer Science
University of York
Heslington, York YO10 5DD, UK
wood@cs.york.ac.uk

Ronaldo Menezes
Department of Computer Sciences
Florida Institute of Technology
150 W. University Blvd., Melbourne, FL 32901, USA
rmenezes@cs.fit.edu

Abstract

A new application of tuple-space-based coordination systems is in knowledge communication and representation. This knowledge is being published on the Web (the so-called "Semantic Web") and could be concurrently accessed and used by large numbers of agents distributed across the world, such as in a global pervasive system. Present coordination models assume that the tuples contain plain data, but when knowledge is being coordinated some aspects of coordination systems such as Linda must be revised. In this paper, we focus on the issues of tuple-space views and the ability of agents to perform destructive retrieval of information from the tuple spaces. Knowledge is not something that is universally shared, rather different groups share different views of the world. It is also something that does not cease suddenly to exist, rather it tends to be forgotten as new knowledge replaces it. Hence we introduce the concepts of scoping and fading and describe how this can be applied to support knowledge-based multi-agent coordination on the emerging Semantic Web.

Keywords: Semantic co-ordination, semantic tuplespaces, RDF, scopes, fading

1 INTRODUCTION

The Semantic Web promises to solve many challenging and cost-intensive problems present in the current generation of Web applications. Ontologies, explicit representations of a domain based on logics, are formalized using Web-suitable representation languages such as Resource Description Framework RDF [6] or Web Ontology Language OWL [11] and are shared and reused across the Web. Web content can be described and human knowledge can be expressed in terms of ontological concepts. The Semantic Web intends to allow agents (whether human or computer) to handle the huge amount of heterogeneous data on the Web in a more flexible, automatable, and dynamic manner. This brings benefits also in the setting of pervasive and ubiquitous systems, where large numbers of heterogeneous agents must communicate knowledge about their context and environment with one another, and decision making is supported by such knowledge publication and exchange.

Semantic Web knowledge in pervasive and ubiquitous environments raises two ideas. The first relates to the issue of *coordination* that has been a central part of all distributed systems. The second is to place the machine-readable data of the Semantic Web into tuples for coordination in a tuple-space system. In [5] coordination is defined as "the process of building programs by gluing together active pieces". Here, the focus is given to the integration of heterogeneous components communicating through different concurrent processes as to produce a virtually unified application which operates like a single system, abstracted from its distribution, parallelism and internal heterogeneities.

The requirements for applying Semantic Web tech-

nologies in multi-agent systems have been discussed in [18]:

- a decentralized and distributed architecture, in order to allow agents to publish and retrieve information efficiently and effectively.
- scalability as a central issue because of the dimensions and the dynamics of the Web-based multi-agent scenario.
- a high-level of abstraction to cope with inherent heterogeneity problems.
- support for asynchronous interaction among agents and between agents and the middleware. Interaction should be uncoupled in space and time in order to allow agents to publish and retrieve information in a flexible manner.

We consider a tuplespace paradigm as well suited to meeting these requirements, as will be explained in the next subsection.

1.1 A coordination language and model

One of the first coordination languages, Linda [5], has its origins in parallel computing and was developed as a means to inject the capability of concurrent programming into sequential programming languages. It consists of coordination operations (*the coordination primitives*) and a shared data space (*the tuple space*) which contains data (*the tuples*). The tuple space is a shared data space which acts as an associative memory for a group of agents. The coordination primitives are a small, yet elegant, set of operations that permit agents to emit a tuple into the tuple space (operation *out*) or associatively retrieve tuples from the tuple space either removing those tuples from the space (operation *in*, also referred to as destructive read) or just getting a copy of it (operation *rd*). A tuple is an ordered list of typed fields and retrieval is governed by matching tuples against a template which contains both literals and typed variables. A match occurs when the template and the tuple are of the same length, the field types are the same and the value of constant fields are identical. Both retrieval operations are blocking, i.e. they continue only after a matching tuple is returned. In this way Linda combines synchronization and communication in an extremely simple model with a high level of abstraction.

Specifically, the benefits of tuplespace computing with respect to semantic co-ordination and pervasive environments have also been identified [14]:

- Tuplespaces uncouple interacting processes both in space and in time. In other words, the producer of a tuple and the consumer of that tuple do not need to know one another's location nor exist concurrently.
- Tuplespaces permit associative addressing, which means that data is accessed in terms of what kind of data is requested, rather than which specific data is referenced.
- Tuplespaces support asynchrony and concurrency as an intrinsic part of the tuplespace abstraction.
- Tuplespaces separate the coordination implementation from characteristics of the host platform or programming language.

1.2 Semantic Web

RDF is based on a simple triple model which can be represented in a three field tuple of form (subject,predicate,object). As foreseen by the RDF abstract model, the first three fields contain URIs (or, in the case of the object, literals). These URIs identify RDF resources presumingly available on the Web. Fields are typed by RDFS/OWL classes (URIs) and XML-based datatypes (literals). RDFS is a schema language which can define classes and properties as well as express logical relations such as subclasses and property ranges which are used in inferring new knowledge.

An example RDF triple could look like this:

```
fu:Anon foaf:mailbox "some@mail.address"
```

The three values form the triple, the first value - the subject - is the instance named Anon which exists in the namespace identified by the prefix ”fu”. Actually, the value when expanded is an URI but for readability we use QNames as in XML. The second value - the property - is the mailbox propery from the Friend of a Friend vocabulary [1]. The final value - the object - is a literal (a String in this case). Together, this forms the statement that some person Anon has a mailbox with the address ”some@mail.address”.

Previous work has recognized the possibility to map the simple triple data model of RDF into three fielded tuples [3]. One must additionally take into account issues of mapping the data model of RDF into tuples (as it includes these RDF-specific constructs: blank nodes, containers/collections, reification). Furthermore there is a conceptual difference in the coordination model when tuples are considered to contain statements of knowledge rather than raw (syntactic) data. Taking into account these differences requires rethinking the coordination model and language of tuplespaces and Linda.

[1] www.foaf-project.org/

1.3 Contribution

The contribution of this paper is to introduce two important concepts in co-ordination - scopes and fading - and apply them to the specific scenario of co-ordinating knowledge, as envisaged in a global, pervasive knowledge system. We argue that both concepts are important to capture the particular needs of knowledge co-ordination, which is an emerging field as pervasive and ubiquitous computing also begins to support Semantic Web technologies.

Scopes were introduced as a means for access to data in spaces orthogonical to the space organisation [10]. How to control this access was defined in Multicapabilities [21].

Recently, a new concept in Linda has been introduced where information stored as tuples may have the ability to be given less importance when compared to others [9]. This concept, called *fading*, argues that in some applications the retrieval of tuples may be probabilistic according to the importance the tuples have to the system — this importance is tracked based on the usage of tuples by processes in the application. In other words, tuples seem to be forgotten (or to fade) if they are not used in the application.

In this paper, we refine a Linda coordination model for knowledge further with the ideas of scoping and fading and show its application in a real semantic tuplespace scenario.

Firstly, in Section 2 we outline a concrete scenario to ground our work in knowledge co-ordination. In Section 3 we introduce a semantic tuplespace implementation called Semantic Web Spaces. Then we discuss in Section 4 how to model coordination of knowledge and explain the concepts of scopes, multicapabilities and fading. Section 5 describes the application of these concepts to Semantic Web Spaces and their use in our coordination scenario. Finally, we conclude with a review of this work and the outlook for the future.

2 ONTOLOGY REPOSITORY SCENARIO

As a concrete scenario, it is proposed that future pervasive and ubiquitous systems, using the exchange of semantic data to enable richer inference over the knowledge being generated by agents, would need to support an ontology repository [17] with concurrent alternative modellings of real world domains which we expect to change over time. By taking a tuple-space approach, all of the knowledge about the domain is available through a single access point, using contexts to partition the different changes being made to the same base ontology, and a standard simple API. Heterogeneity between emerging models such as the same concept being given different identifiers by different users can be resolved by extending the semantic matching algorithm of a semantic system to support means, whether manual or semi-automatic, to determine mappings between concepts in ontologies. Concurrency is handled explicitly in the Linda-based coordination model, including the 'copy' primitive [15] that copies all matching tuples found in the context of the query into a new, protected context, accessible only to the querying client. This protects the discovered knowledge from further changes, and the client can then feed the knowledge back into the space after processing, or destroy it. The blocking nature of the coordination primitives means that processes wait for the availability of matching tuples rather than immediately fail with an empty query response. As a result, the separate evolution of ontologies and the coordination of their access and changes made by multiple users is supported in a semantic tuple-space computing scenario.

As an example to further illustrate issues raised by coordinating knowledge, we consider a shared repository used by cooperating companies and their internal departments to model roles and skills in their organizations so that individual employees can annotate themselves with those roles and skills and support internally finding the correct contact person for some need semantically. To ground this in a concrete domain, consider an IT services provider that makes its services available to several companies. To streamline the process of allocating employees to the different IT tasks being delivered to the service provider, the provider stores annotations for each employee in a private (intranet) space where their roles and skills are expressed according to the ontologies for roles and skills being agreed upon among the participating companies. Hence, a company can expressly search the space for an IT expert with a particular role or skill, and can insert a task description into the space which the IT services provider will read and match to one of its employees. Let us consider a number of specific scenarios in this imaginary system to illustrate the particular needs of coordination and knowledge.

Firstly, we note that companies will be likely to differ in how they attach importance to roles within their industry and within their own departments. A single global ontology will be unable to capture such heterogeneities. Furthermore, over time roles will alter as the companies' short and long term strategies change, market conditions alter and so on. Hence, the importance of a particular IT services employee may increase or decrease according to their roles and skills as the im-

portance of those roles and skills vary for a particular company or department. It may be that some of the IT experts who were earlier very relevant for one company become gradually more relevant for another. Furthermore, some experts may become irrelevant for all companies as their role and skill set becomes irrelevant. However, in terms of the Semantic Web, the increasing or decreasing importance of some statements of knowledge can only be evaluated through (often computationally intensive) reasoning over the entire knowledge base, and as factors change such relevance measurements become only calculable when specific queries are executed over the knowledge. In other words, from the perspective of the semantic tuple space, it is not immediately visible if some statements of knowledge are more or less relevant to particular users or globally (for all users).

Secondly, we note that the skills required by companies or departments will change in importance as systems and technologies change. Hence, some knowledge can still be valid but is less important. An example of this would be in programming languages. An IT expert may be equally skilled in ALGOL and Java; the statements in the space are equally true. However, for most companies, that the expert is skilled in Java will be much more important than in a language which is probably not used at all in their present IT infrastructure. In the semantic tuple space, both statements would however be equally evaluated against a query, while one could argue that the evaluation of the statement regarding ALGOL is probably redundant for most queries (assuming ALGOL has little or no relevance to the company making the query). However, removing the statement is erroneous as then this knowledge, which is certainly true, may be valid to some other company.

Finally, assume this space is opened to other outsourcing companies which offer IT experts on their books to the participating companies in competition to our example IT service provider. In the semantic tuple-space platform, the descriptions of each IT expert are evaluated equally against one another in a query. However, we can expect participating companies, on the basis of many years of experience, to already trust those experts belonging to the original IT services provider, both in terms of the trustworthiness of their employees and the truthfulness of their descriptions. The other outsourcing companies may compete in the space for queries but actually companies will tend to want to prefer matching with descriptions from the original provider and attach a lower trust value to the descriptions for the new entrants in the space.

The problem encountered in these scenarios still is that knowledge can often not be simply marked as true and false, in fact humans can always differ on how they evaluate the truth of certain ways to understand the world around them. Other concepts such as relevance, importance and trustworthiness model our views on knowledge. It is not enough to present one "truth" and permit the deletion of any differing statements. In a shared system one group could then act to ensure that only their "truths" existed. To model the complexities of the real world, we need to rethink how shared knowledge can be coordinated in systems with concurrent access, e.g. in pervasive and ubiquitous computing.

3 SEMANTIC WEB SPACES

Some initiatives are emerging in realizing tuple-space systems for the coordination of Semantic Web data. The sTuples proposal [7] attempts to combine Semantic Web and tuple-space technology by extending JavaSpaces to permit a tuple field to contain an OWL individual. However, the underlying platform is not semantic, i.e. there is no consideration of how storing RDF/OWL into a tuple space affects the fundamental issues of tuple and tuple-space representation, Linda operations and matching. Triple Spaces [4], which focus on the use of RDF-enabled tuple spaces for Semantic Web Service operation, have been proposed as a communication mechanism for the WSMX platform [2]. A minimal architecture for Triple Spaces is also proposed [1] in which, however, many of the powerful and beneficial aspects of Linda and the tuple-space model have been removed.

Semantic Web Spaces [18, 19] is proposed as a generic Semantic Web middleware platform which can provide synchronization, coordination and mediation functionalities to intercommunicating Semantic Web agents. While in the above work it is not yet clear how the particular issues of (particularly RDF) semantics would be handled in tuple spaces, Semantic Web Spaces has been defined explicitly in terms of these issues. As we choose Semantic Web Spaces to illustrate our work, we examine its conceptual model in some more depth.

3.1 Conceptual model

The idea of using the Linda coordination model to support a tuple-space exchanging Semantic Web information, requires some extension to the traditional Linda model. These extensions can be divided into four categories as follows:

- **New types of tuples:** the representation of semantic data within tuplespaces requires new types of tuples which are tailored to standard Semantic Web languages (RDF(S) [6] and OWL [11]). Additionally, we choose to uniquely identify each

RDF statement by means of an ID field. In this way statements sharing the same subject, predicate and object can be addressed separately, which is consistent with the Linda model. The allocation of the IDs is coordinated by the tuplespace with the help of the tuplespace ontology (see below).

- **New co-ordination primitives:** the transition from data-centered tuplespaces to the new semantics-aware Semantic Web Spaces requires a revision of the meaning of the Linda coordination model. In Semantic Web Spaces we fundamentally distinguish between a data view and an information view upon the stored RDF tuples. In the data view all tuples are seen as plain data, without semantics, like in traditional Linda systems. In the information view we see the set of RDF tuples in the tuplespace as a RDF graph. This imposes consistency and satisfiability constraints w.r.t. the RDF semantics and to associated ontologies defined in RDFS or OWL. Hence, the traditional coordination model is revised in order to support this distinction. In the data view we make use of a Linda-compliant variation of the traditional `out`, `in` and `read` operations. They preserve the original semantics while operating on the structure of RDF triples. Handling Semantic Web knowledge in the information view requires however co-ordination primitives, which take into account the truth value of the underlying tuples and the ontologies the tuples might refer to. For this purpose we introduce the operations `claim`, `endorse` and `retract`. In addition, we have defined multiple tuple operations which output or read a set of RDF triples as a single request-response. The `excerpt` operation in particular uses a ”context” in the information view to contain a set of copies of the matched tuples in a private partition of the space (the reference is only passed to the agent excerpting the triples). This allows contexts to explicitly contain inferred tuples which are only implicit in the information view of the space and to allow agents to construct RDF models and continue interacting with those models without affecting the rest of the space (e.g. destructive reads).

  The “retract” primitive redefines the idea of destructive read so that it can be applicable to knowledge. In logics operating under the *closed-world assumption*, the deletion of a statement could be understood as negation. In the Semantic Web, which operates under the *open-world assumption*, the non-existence of a fact is interpreted rather as “unknowableness”. In Semantic Web Spaces, the current approach of “retract” is to replace the retracted triple’s fields with an empty value (“bottom”) which does not match any content of a Linda template (not even a wildcard) and only its ID is retained. Consequently, all inferable knowledge from that retracted tuple is lost, and only the reference (marking that once a statement existed) remains to be found in the space.

  Table 1 gives an overview of the co-ordination model within Semantic Web Spaces.

- **New matchings:** the standard Linda matching approach is extended in order to efficiently manage the newly defined tuple types. This applies for both the data and the information view. The former includes procedures to deal with the types associated with the subject, predicate and object fields of each RDF tuple.[2]. The latter additionally takes into account domain-specific types defined in external ontologies. Semantic matching techniques may then make use of this knowledge with the help of reasoning services in order to refine the retrieval capabilities of the tuplespace.

Orthogonal to these dimensions Semantic Web Spaces contain a **tuplespace ontology**, which is used as a formal description of the tuplespace, its components and properties. Using ontologies in this context allows a more flexible and efficient management of the tuplespace content and of the interaction between tuplespace and information providers and consumers (extendability, automatic inferencing etc.).

3.2 Realisation

A concept for the technical realization of Semantic Web Spaces was drawn up subsequent to the specification of the conceptual model (Figure 3.2).

From left to right Semantic Web Spaces can be divided into three major components, which are concerned with i). the publication of Semantic Web information, ii). its retrieval with the help of tuple matching heuristics, and iii). the security of the execution of the aforementioned activities, respectively. From top to bottom the architecture contains three layers: the first two layers correspond to the information and data view we mentioned in the previous section, while the third layer handles the persistent storage of the tuplespace information. Accordingly, from bottom to top, the tuplespace

[2]These types are pre-defined in RDF Schema (RDFS)

DATA VIEW	
outr: (Statement) → boolean	Insert a new RDF statement to the data view
outgr: (Model) → boolean	Insert a set of RDF statements extracted from a Jena model to the data view
rdr: (Triple or Node) → Statement	Read an RDF statement from the data view of the space using a three-fielded template of the form (s,p,o) or a Node containing a tuple id
rdgr: (Triple) → Model	As rdr but returns all matches as a Jena model
inr: (Triple or Node) → Statement	Destructively read an RDF statement from the data view of the space likewise with a template or tuple id
ingr: (Triple) → Model	As inr but destructively reads all matched RDF statements from the data view
INFORMATION VIEW	
claim: (Statement) → boolean	Assert a RDF statement in the information view of the space if consistent with the RDF Schema
endorse: (Triple) → Subspace	Read a RDF statement from the information view of the space
excerpt: (Triple) → URI	Read all matching RDF statements by copying them into a context and returning an URI identifying it
retract: (Triple) → Subspace	Deny the truth value of an RDF statement in the information view of the space (retained in the data view)

Table 1: Co-ordination model in Semantic Web Spaces

system manages raw data, syntactic virtual data (Linda tuples) and semantic virtual data (RDF tuples). In other words, as we build ever higher the architecture of Semantic Web Spaces the data representation becomes higher level.

With reference to the above concept for the technical realization, we have realized the Triple and RDFS I/O on top of the LighTS I/O [13], ontology and triple matching on top of Linda matching and views (subspaces, contexts and meta-model). Security and trust, as well as backend persistent storage are subject of future work.

A Java class diagram of the implementation is shown in Figure 3.2. For more details regarding the realisation and an evaluation of Semantic Web Spaces the reader is referred to [20].

4 COORDINATING WITH KNOWLEDGE

The multiple view (data and knowledge views) standpoint mentioned earlier is fundamental, and is able to be exploited due to the particular properties of the tuple-space based coordination model used in this paper. There are two aspects which we motivate and explain: multiple views and fading.

To emphasize this, consider the way in which human understanding of the physical universe has evolved. Penrose [12] identifies four different relativistic worldviews: Aristotelian, Galilean, Newtonian, and Einsteinian, each of which modifies the previous in terms of how space and time are to be construed. So we may say that the "knowledge" view of the Universe has changed over the millennia, even though the underlying "data" view has (probably) remained the same — the basic components of the Universe are what they were before Aristotle, but the relationships that we perceive or construct (the "knowledge") have altered. Consequently, some things "fade" (time is absolute, space is fixed) while others "grow" (time and space are relative), even if the underlying *facts* — whatever they are — remain constant.

In the less philosophically charged realm of Web middleware, these ideas correlate with the distinction between *publishing* and *mining* knowledge. In the former case, data is structured to reflect the intended semantics of the information; in the latter, raw data is *viewed* semantically — it is the view that provides the knowledge structure. The Web *exists*, and is filled with data. The purpose of the *Semantic* web, then, is to be able to *view* that data as knowledge — hence the need

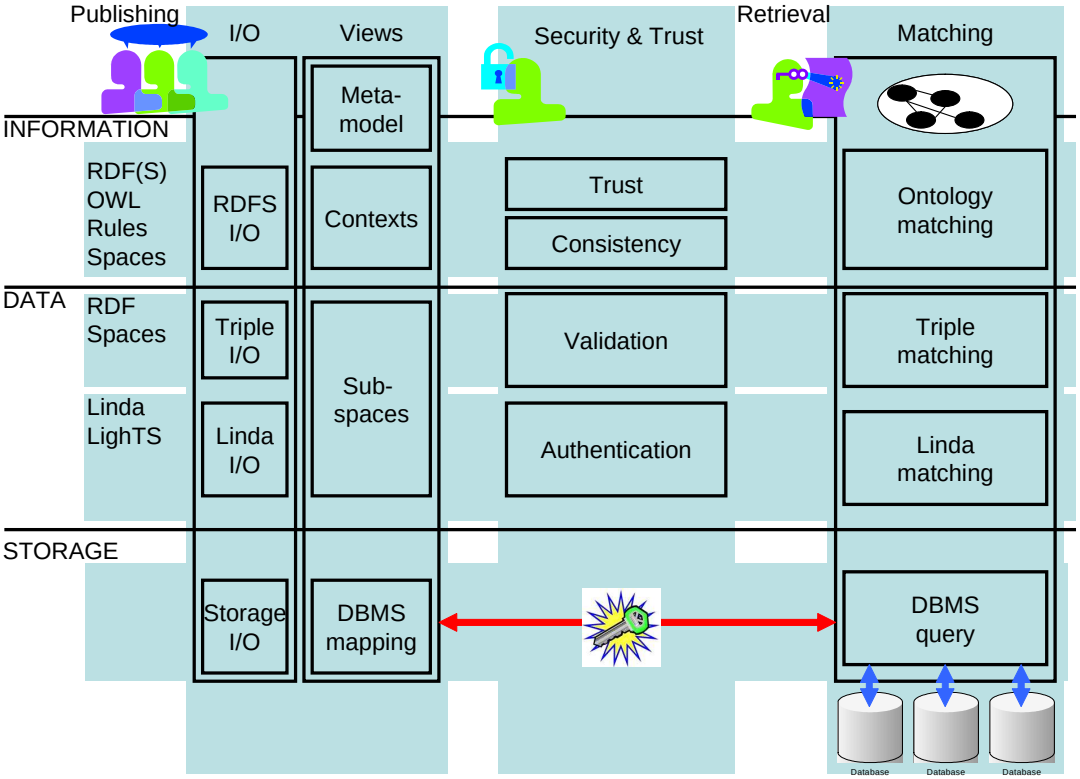

Figure 1: Concept for realization

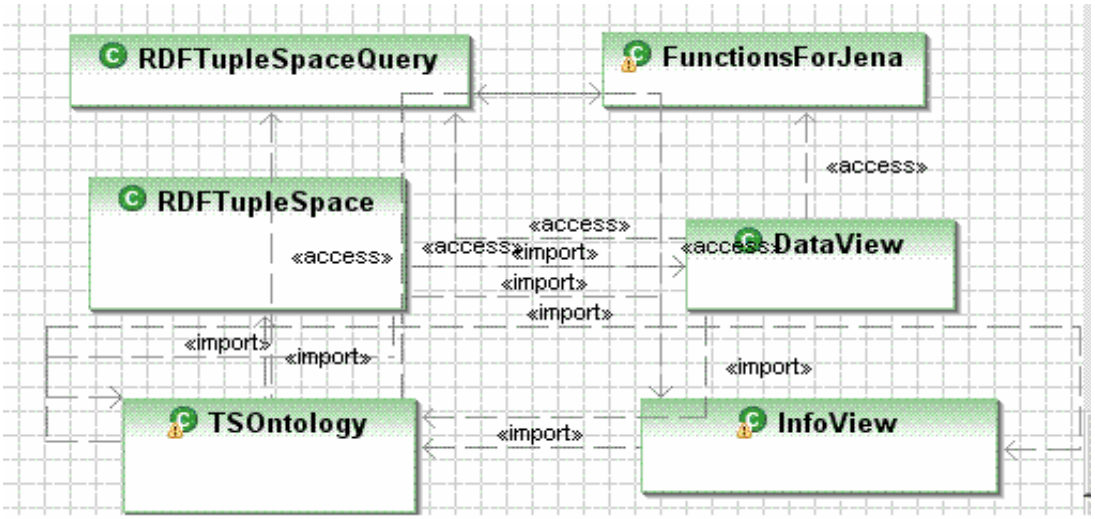

Figure 2: Implementation Class Diagram

to separate the two views. Once separated, it becomes clear that the way in which knowledge evolves is different from the way that data evolves. Knowledge reflects the current state of understanding of the relationships between the data. Thus, when the data-state of the Web changes, the knowledge view must evolve. At the coarsest level, this involves removing or *retracting* semantic relationships. However, this complete deletion is not necessarily the most appropriate course of action. Consider the evolution of the physical world-views mentioned above: although Newtonian physics is strictly no longer true, being superseded by the Einsteinian physics, it is *still* the one that is most often used in world in which we live. It has *not* been fully "retracted", rather it has "faded" as a universal truth. Therefore, we need to be able to model this fading of importance of knowledge — "retraction" becomes a *continuous*, rather than a binary process.

The question then arises as to *what* is being retracted? Clearly, by analogy with the physical world-views, it is not the *data* that is being faded since that merely exists, and that fact of its existence does not fade: it exists or ceases to exist. So it must be the *knowledge* that evolves, and since this is represented as a separate "view" it is possible to focus on how that evolution may be obtained without any of the practical, political or other problems of proposing that data that exists in individual web-users' file systems should be modified or deleted by alien middleware. The proposal then is that agents that wish to "play the game" would access knowledge-level views into the raw data, and it is in *these* views that retraction or fading will be instantiated.

4.1 Modelling views

In the *Scopes* model, tuple-spaces are replaced by scopes which are *defined* by the tuples that they contain. That is, a tuple is no longer *in* a unique tuple-space, but is *viewed* through a scope. This allows a tuple to be 'in' multiple scopes. Consequently it is possible for an agent to combine scopes — for instance by union or intersection — to form other scopes and to pass these to other agents. In essence, a scope represents a way of defining the *visibility* of tuples: it defines a sub-class of the universe of tuples. Multicapabilities are strongly related to Scopes: a 'standard' capability [16] is an object reference together with a set of 'rights'[3]; a *multi*capability replaces the object reference with a specification of an object 'type', thus conferring rights on the agent to operate on *any* object of the corresponding type. Thus, the capability idea is extended to *tuples* which, being accessed associatively, do not have references. Multicapabilities, like scopes, can be combined: for instance, if an agent holds two multicapabilities, they can be 'summed' to produce a multicapability that grants *both* sets of rights to the holder. The crucial point about multicapabilities that makes the concept appropriate for the domain of this paper is that a single multicapability represents a *region* in a tuple-space or Scope, defined by the set of tuples that are *potentially* accessible to a holder of that multicapability. These regions (which may, in fact, be distinct even when the types specified are identical: see [21] for details), together with the combination operations can then be used to refine the knowledge views into the raw tuple data. By manipulation of the set of rights associated with the multicapabilities, the middleware, and agents themselves, can modify the visibility of the knowledge, and thus its significance.

To support the semantic partitioning of views on knowledge in the Semantic Web Space, the concept of "contexts" is used. Both clients and tuples are associated with a set of contexts, and an agent can only see the tuples which exist in the same context. Contexts are based on scopes [10] and multicapabilities [21], which allow for a tuple space to be split into (overlapping) partitions. This provides a means to control client access to statements in the space and to split client operations into subsets of relevant statements. Contexts differ from scopes primarily in that they are applied to a tuple space of knowledge rather than data and follow Semantic Web principles, leading to a few conceptual differences: contexts are identified by URIs and every context is an instance of the RDF class Context which is defined in the tuple-space ontology. In order to support contexts in Semantic Web Spaces two more operations are added: a means for a client to construct a context, and a means to acquire a context from a matched tuple. Other context operations are handled by the system e.g. a client upon joining a space is allocated a certain context set. Its operations apply to the context set in which the client is currently active. Clients can add and remove contexts to and from their current context set so long as they know the identifier of the context, which is possible either through creating the context themselves or by having access to the metadata identifying the context(s) in which a certain tuple exists. Context identifiers are allocated by the system in order to guarantee uniqueness.

4.2 Fading and Its Implementation

As discussed in beginning of Section 4, the idea of fading seems to be quite natural in the real world when we consider concepts. But how can this be implemented?

[3] Corresponding to a set of methods on the object in OOP terms.

Swarm Intelligent (SI) systems offer us a good starting point for this concept. Most of those systems that can be said to be in this category (see [8] for a survey on such systems) implement an idea of a logical time keeper that drives the activity of the agents. This is better seen in the analogy of ants looking for food in the environment. Successful ants (ones that found food) return to the nest leaving a trail that can be used by other ants to reach the same food source. The information left takes the form of a pheromone that evaporates (fades) over time. However, the fading rate is not decided by the passing of an external (real clock) time but rather on environmental properties such as temperature and humidity. In addition, independent processes of reinforcement are superimposed on the evaporation — for instance, the more often ants use the trail, the more pheromone is added to the trail (reinforcing the information).

The implementation of a fading concept in a Semantic Web should be organized as the aforementioned idea in ants: concepts that are more used are more visible (less faded) than others. Yet, the implementation of fading may be realized in many different ways (as described later in this section). In reality the concept of fading is quite orthogonal to how it is implemented. We discuss some approaches here to demonstrate that proposed fading concept is implementable in a reasonably economic way.

Some assumptions have to be made about the structure of the Linda kernel (or in our case, Semantic Web Spaces), even though the generality of the model allows a very wide choice of implementation architectures. So in the following it is assumed that the kernel consists of many distributed communicating sub-kernels with any arbitrary connected network topology and no control hierarchy. The tuple spaces are likewise distributed over the (sub-)kernels, so that a particular sub-kernel will handle partitions of multiple tuple spaces. Processes interact with a single sub-kernel which may or may not reside on the same host as the process. A kernel will receive requests from its attached processes and neighboring kernels. It will then try to match the template of the request with the tuples in the (local partition of the) specified tuple space. It is at this point that the fading level associated to each tuple, $\Phi(t)$, can be updated according to an equation such as Equation 1 (introduced in [9]).

$$\Phi(t) \leftarrow (1-\rho)\cdot\Phi(t) + U(t) \tag{1}$$

where $U(t)$ is an indication of the usefulness of a tuple t and ρ represents the fade factor (i.e how much a tuple is forgotten each time the equation 1 is used).

Note that one of the crucial points on the fading of a tuple is the calculation of $U(t)$, its representation, and usage. The usage is the easiest to understand since $U(t)$ is used on adaptive retrieval of tuples; by indicating tuples that are more important, the Linda system can retrieve them adaptively according to this importance. The calculation may be represented as in Equation 2.

$$U(t) = \frac{1}{m}\sum_{z=1}^{m} s(t, k(z)) \tag{2}$$

which is an average over the number objects (or tasks) m referring to tuples, $k(z)$, similar to t — the similarity is calculated based on a function $0 < s(t, k(z)) \leq 1$ in which 1 refers to high similarity.

Last we need to look at the representation of $U(t)$. For example, m could represent the number of reads of tuples similar to t, or the number of processes that have a capability of handling tuples similar to t. This is what we call representation based on Tuple Usage. Below we discuss a few possible representations of the value of $U(t)$.

Tuple Usage: Tuple usage is probably the first and obvious way to control the value of $U(t)$ because usage may be a metric of how important a tuple is to processes/agents in the system. If a tuple is read frequently it means that the collective belief of processes is that the knowledge represented by the tuple is reliable. Hence, this usage can be measured and used when retrieving knowledge, where more important knowledge is given a higher probability of being retrieved. Note that this process should lead to the emergence of faded tuples (unreliable or unimportant knowledge). This usage may also be based on similarity to other tuples/templates in the system [9]. Here the argument may be that in a particular system, one may be interested on certain "kinds" of knowledge (e.g. knowledge about the stock market) making all other knowledge less important or more faded.

Tuple Distance: The system may be able to estimate the distance of a tuple (in terms of number of hops for instance) from where it is stored and the process requesting it. Given a set of candidate tuples, their retrieval may be adaptive and consider this distance as fading. The further a tuple is from the requesting process the less important it is for that process. In terms of knowledge representation, such an approach could indicate the clustering of knowledge according to their belief for a group of processes. The implementation of this does not necessarily require an attribute associated to each tuple. In fact, the distance of the tuple and consequently the time it takes to access it is an indication of its importance. However, to get this effect, the system should be capable of moving tuples between sub-kernels in response to process activity.

Process Activity: Another metric that can be uti-

lized is the pairing between processes and tuples. One may keep track of tuples according to the reliability of the process that stored it initially. Knowledge added by unreliable process may be considered less useful. As the process becomes more reliable so does the tuples it stored. Note that the implementation of this variant assumes a sub-system to control process reputation. The implementation here is probably the most complex and most expensive one because the attribute $U(t)$ may change according to the reputation of the process that was responsible for the inclusion of the knowledge.

In the simplest implementation, the search for a match will take place by trying all the stored tuples in (some) order. The extra expense in time for update of the fading levels would be that once a match were found, the remaining tuples would have to be accessed in order to update them, whereas in a non-fading Linda the search could stop on first match. However, since on average this simple search scheme needs to access half of the stored tuples to find a match, the fading version would only double the time.

5 FADING IN SEMANTIC WEB SPACES

The problem for the coordination of knowledge is that different users of the space may have differing, even contradictory, views about a certain domain, and hence it must be possible for users to be able to virtually separate the global body of knowledge into subsets which express the "truths" that they hold for them without invalidating the truthfulness of statements for other users. Furthermore, knowledge as opposed to data is evaluated not only on the basis of its truth value, which means on its existence in the space (which signifies that for at least one user it is considered true), but in notions of importance, relevance and trustworthiness. The usual Linda approach with non-deterministic retrieval does not offer any further guidance as to which view may possibly be "more" right (or at least shared by more other users).

By extending Semantic Web Spaces with the concept of fading, we provide an in-built means to allow a selectively deterministic form of knowledge retrieval from the space. In terms of the scenarios given in section 2 we will explain how fading-based retrieval can enable interactions with the semantic tuple space respecting our theory of coordination of knowledge. Referring to Equation 1, each tuple (statement of knowledge) in the space is associated with a fading level. This level will be constant for all newly inserted tuples (let us say 1.0). The space defines also a constant fading rate, i.e. all knowledge loses relevance in the same way. Note that this does not mean that statements of knowledge will in fact fade at the same rate as this will depend upon the accesses to knowledge made in the space. However it does mean if two statements of knowledge are accessed in precisely the same way, and all other variables are the same, they will fade to the same extent. The last variable to mention is the measure of usefulness, which will be calculated on the basis of some properties of the tuple in question, and will be expected to vary throughout the lifetime of the tuple. We will consider some calculations of usefulness in our explanations of the scenarios.

Menezes and Wood [9] present a proposal for adaptive retrieval through fading, giving as an example a LIFO approach where the last tuple placed in the system is the tuple with the higher chance of being chosen. We need to adapt this idea of adaptive retrieval to semantic tuple-space computing. Here, the order of tuple insertion is irrelevant for the truthfulness of a statement. The form of semantic queries made on the space however can be interpreted to represent the interest an agent has in certain truths, and the selection of a statement can be interpreted to be a "reinforcement" of that statement's truth. Alternatively, the destructive read (removal) of a statement in the space can be interpreted as a rejection of the truth of a statement by a particular agent. So agents can read knowledge from the space and if they find statements which they (as a result of some further processing) conclude to be erroneous (from their point of view) the "retract" operation acts as a form of invalidation, which fades the statement and hence makes it less likely to be retrieved in a subsequent query of the same form. Equation 1 would be used to fade the value of $\Phi(t)$ for the erroneous tuple.

To explain this approach, assume a Semantic Web Space consisting of RDF-based tuples of the form $\langle$s,p,o$\rangle$ (subject, predicate and object). Given a non-destructive read over the space of the form $\langle$?s,p,o$\rangle$ (where ? means a variable) we select all tuples matching this template and, in the case of a single read operation, return the tuple with the strongest fading level. We then fade all tuples matching $\langle$?s,p,!o$\rangle$ (where ! means NOT), i.e. we assume that statements about this property which contain some other object other than the one queried for are less relevant for the agent (and hence, to some extent, for the space as a whole). This approach for all templates is outlined in Table 5. We restrict this non-relevance to instances belonging to the same class as the concrete subject, property or object in the template. This includes subclasses but not superclasses, hence requesting people who like ice cream may fade statements about liking other desserts but not other statements of liking food and drink. Additionally, the extent to which these other statements fade will vary according to how

often they are being read by other processes, how distant they are from the reading processes and how trusted the supplying process is, as this will determine their usefulness quotient independent of the query which invoked the fade.

In combination with contexts, users are able to form virtual partitions of the space (apart from its actual structure) which group particular statements (the in-built "extract" primitive allows this by selecting all statements matching a particular semantic query). Agents can select which sets of contexts they read from and hence "prefer" some sets of knowledge over others. This provides for a way to apply the fading principle on "theories" (grouped sets of knowledge) as well as on individual statements. Agents are hence able to build views on their own knowledge, as when they apply reading operations within a context, the effects of fading on tuples apply to that context only, and the validity of those statements to other agents viewing the statements through other contexts are unaffected (while agents can also group themselves to share knowledge views by agreeing to act within the same contexts). Let us illustrate this adaptive retrieval approach in Semantic Web Spaces in terms of the three scenarios we outlined in section 2.

Firstly, we noted that companies will likely differ in how they attach importance to roles within their industry and within their own departments. The IT services provider places its employee descriptions in the space and companies will read employee descriptions relevant to the IT task requirements into their own contexts. If we consider a company A which is querying the space more often for IT specialists skilled in CORBA and a company B querying the space for IT specialists skilled in .NET, assuming both as instances of a Middleware class of objects, then knowledge about specialists with skills in other middleware technologies can be expected to fade more than those about CORBA and .NET skills. As this (at a larger scale) can represent for the space which middleware technologies in general are more relevant to others, we could expect a new company joining the space and asking for middleware specialists to consider CORBA and .NET skills more relevant than others as would be represented by the relative fading levels of the statements. However, this still does not help differentiate between different companies' interests. Adding the Tuple Distance factor however, and assuming that companies host knowledge in their context on more local space kernels, the relevant tuples will be positioned more locally than other facts (this acts as a sort of "cheap" self-organization — clearly we could build rules into the system to intuitively move tuples closer to the processes reading them). Then, even when another process from the company makes a different type of query — e.g. context-less and more general (e.g. which specialist has some skill in middleware) — it is more likely that a specialist with the relevant skills is returned as the knowledge about, say, CORBA specialists, is closer and hence — from the perspective of the requester — less faded than other knowledge that matches the same query.

Secondly, we note that the skills required by companies or departments will change in importance as systems and technologies change. For example, considering queries for IT specialists with skills in particular programming languages, those queries "reinforce" the statements about those languages as statements applying to other languages will be faded. If over time Java skills is commonly requested while ALGOL skills are almost never requested, it follows that the fading of statements about ALGOL skills is greater than those about Java skills. If we now add the Tuple Usage factor, we note that the actual reading of a tuple (as opposed to the difference to a query in the space) acts as a factor in the extent to which a statement fades. For example, rather than fade all statements which relate to skills in non-Java programming languages equally, it may be that some other statements about programming language skills are being read often by other processes and hence are still relevant, even if not relating to Java, and will fade less due to a higher usefulness quotient. Note that tuple usage strengthens the fading or non-fading of individual tuples. If a particular specialist is being returned when companies query for someone with Java skills, his tuple will fade less than the others when non-Java specialists are sought, making it more likely he will be returned to the next Java specialist query. As long as no-one retracts his tuple, the relevance of this statement is reinforced to other processes.

Finally, we expect participating companies to already trust those experts belonging to a IT services provider with whom they have many years of experience over newer entrants in the market who are also offering IT services. Here, the Process Activity factor could be added to recalculate the usefulness of tuples with respect to the process providing those tuples. Companies would wish to program the system so that the trusted services provider has a higher usefulness quotient as the other providers. As a result, the new services in the space begin (for fairness' sake) with the initial fading level but, as long as they do not establish themselves in the market through their descriptions being read by companies, not only do their statements fade but at a greater rate than others. They still have the chance to establish themselves by discovering a niche in the market (answering queries which other providers fail to answer) or by be-

Table 2: Table describing the templates used in a read operation and the template of tuple affected in the fading operation. Retract operates in the inverse, fading only those statements in the space matching the template passed in the operation.

Template:	**Fade tuples matching:**
s,p,o	!s,p,o OR s,p,!o
?s,p,o	?s,p,!o
s,?p,o	s,?p,!o OR !s,?p,o
s,p,?o	!s,p,?o
?s,?p,o	?s,?p,!o
?s,p,?o	?s,!p,?o
s,?p,?o	!s,?p,?o

ing retrieved by standard queries early on (before fading has made a significant difference) which are validated (i.e. not retracted by the process in an act of rejecting the services of this provider).

These few examples show how the scenarios we had outlined, and in particular the problems surrounding coordination of knowledge, with its associated problems of views, relevance and trustworthiness, may be resolved by a new coordination model incorporating multicapabilities and fading.

6 CONCLUSIONS

In this paper, we tackle the issue of coordinating knowledge on the Semantic Web through a tuple-space-based system. The Semantic Web effort is tackling the problem of representing and exchanging knowledge and the coordination systems community has done much work on the coordination of data in open distributed multi-agent systems. This work is also being applied to the Web, but the problem of coordination of knowledge is an issue that is not only still open, but barely tackled in current efforts, even those which attempt to merge knowledge and coordination in some Linda-based approach. This paper has taken the work on Semantic Web Spaces, which is the most advanced in terms of considering the implications of coordinating Semantic Web knowledge and drawn on innovative work from coordination systems research which we motivate as being highly relevant to knowledge coordination. We concentrated on the notion of knowledge retraction, which is more about invalidation than negation. We have argued that retraction is unnatural due to the open nature of any Semantic Web and that the concept of forgetfulness or fading of knowledge is more appropriate to the reality. We have described a coordination model for Semantic Web based on multicapabilities and the concept of fading, outlined generally its implementation in Semantic Web Spaces and given a number of examples of its application, demonstrating its relevance to our scenario.

Having motivated the value of fading as an approach to knowledge coordination, we plan to further specify fading in this direction and, deciding on concrete algorithms for both the fading and usefulness calculations, extend the Semantic Web Spaces system and evaluate the approach using real Semantic Web data based on the scenario outlined in this paper.

References

[1] C. Bussler. A minimal triple space computing architecture. In *Proc. of the WIW 2005 Workshop on WSMO Implementations*, 2005.

[2] C. Bussler, E. Kilgarriff, R. Krummenacher, F. Martin-Recuerda, I. Toma, and B. Sapkota. D21. v0.1 WSMX Triple-Space Computing, June 2005.

[3] D. Fensel. Triple-based Computing - WSMO Working Draft. http://www.wsmo.org/2004/tp-computing/, June 2004.

[4] D. Fensel. Triple-Space Computing: Semantic Web Services Based on Persistent Publication of Information. In *INTELLCOMM*, pages 43–53, 2004.

[5] D. Gelernter and N. Carriero. Coordination languages and their significance. *Commun. ACM*, 35(2):97–107, 1992.

[6] P. Hayes and B. McBride. RDF Semantics. Available at http://www.w3.org/TR/rdf-mt/, 2004.

[7] D. Khushraj, O. Lassila, and T. W. Finin. sTuples: Semantic Tuple Spaces. In *MobiQuitous*, pages 268–277, 2004.

[8] M. Mamei, R. Menezes, R. Tolksdorf, and F. Zambonelli. Case studies for self-organization in computer science. *Journal of Systems Architecture*, 52(8-9):443–460, 2006.

[9] R. Menezes and A. Wood. The Fading Concept in Tuple-Space Systems. In *SAC '06: Proceedings of the 2006 ACM Symposium on Applied Computing*, pages 440–444. ACM Press, 2006.

[10] I. Merrick and A. Wood. Coordination with scopes. In *Proceedings of SAC'00*, pages 210–217. ACM Press, 2000.

[11] P. F. Patel-Schneider, P. Hayes, and I. Horrocks. OWL Web Ontology Language Semantics and Abstract Syntax. Available at http://www.w3.org/TR/owl-absyn/, 2004.

[12] R. Penrose. *The Road to Reality : A Complete Guide to the Laws of the Universe*. Jonathan Cape, 2004.

[13] G. P. Picco, D. Balzarotti, and P. Costa. LIGHTS: A Lightweight, Customizable Tuple Space Supporting Context-Aware Applications. In *Proceedings of the 20^{th} ACM Symposium on Applied Computing (SAC05)*, Santa Fe (New Mexico, USA), Mar. 2005. ACM Press.

[14] D. Rossi, G. Cabri, and E. Denti. Tuple-based technologies for coordination. In A. Omicini, F. Zambonelli, M. Klusch, and R. Tolksdorf, editors, *Coordination of Internet Agents: Models, Technologies, and Applications*, chapter 4, pages 83–109. Springer Verlag, 2001. ISBN 3540416137.

[15] A. I. T. Rowstron and A. M. Wood. Solving the Linda multiple rd problem using the copy-collect primitive. *Sci. Comput. Program.*, 31(2-3):335–358, 1998.

[16] A. Tanenbaum. *Distributed Operating Systems*. Prentice Hall, 1995.

[17] R. Tolksdorf, L. Nixon, and E. Paslaru Bontas. Using Semantic Web Spaces to realize Ontology Repositories. Technical Report TR-B-05-15, Free University of Berlin, October 2005.

[18] R. Tolksdorf, L. Nixon, E. Paslaru Bontas, D. M. Nguyen, and F. Liebsch. Enabling real world Semantic Web applications through a coordination middleware. In *Proceedings of the ESWC05*, 2005.

[19] R. Tolksdorf, E. Paslaru-Bontas, and L. Nixon. A co-ordination model for the Semantic Web . In *Proceedings of the 21st ACM Symposium on Applied Computing, Track "Coordination Models, Languages and Applications"*, 2006.

[20] R. Tolksdorf, E. Paslaru-Bontas, and L. Nixon. Towards Semantic Tuplespace Computing: The Semantic Web Spaces System . In *Proceedings of the 22nd ACM Symposium on Applied Computing, Track "Coordination Models, Languages and Applications"*, 2007.

[21] N. I. Udzir, A. M. Wood, and J. L. Jacob. Coordination with Multicapabilities. In *Coordination Models and Langauges* , volume LNCS 3454, pages 79–108. Springer Verlag, 2005.