Contexts for the Semantic Web

R.Guha[2], R.McCool[1] and R.Fikes[1]

[1] Knowledge Systems Lab., Stanford University, USA
[2] IBM Research, USA

Abstract. A central theme of the semantic web is that programs should be able to easily aggregate data from different sources. Unfortunately, even if two sites provide their data using the same data model and vocabulary, subtle differences in their use of terms and in the assumptions they make pose challenges for aggregation. Experiences with the TAP project reveal some of the phenomena that pose obstacles to a simplistic model of aggregation. Similar experiences have been reported by AI projects such as Cyc, which has lead to the development and use of various context mechanisms. In this paper we report on some of the problems with aggregating independently published data and propose a context mechanism to handle some of these problems. We briefly survey the context mechanisms developed in in AI and contrast them with the requirements of a context mechanism for the semantic web. Finally, we present a context mechanism for the semantic web that is adequate to handle the aggregation tasks, yet simple from both computational and model theoretic perspectives.

1 Introduction

The ease with which web sites could link to each other doubtless contributed to the rapid adoption of the World Wide Web. It is hoped that as the semantic web becomes more prevalent, programs will be able to similarly weave together data from diverse sites. Indeed, the data model behind RDF was significantly motivated by the fact that directed labelled graphs provided a simple, but effective model for aggregating data from different sources.

Unfortunately, while the languages of the semantic web (RDF, RDFS, OWL, etc) provide a method for aggregation at the data model level, higher level differences between data sources sometimes make it inappropriate to directly merge data from them. Just as with the human readable web, semantic web publishers make assumptions and use the same term in subtly different ways. On the human readable web, the human consumers of web pages are able to use their common sense to reconcile these differences. On the semantic web, we need to develop the mechanisms that allow us to explicitly represent and reason with these assumptions and differences. This will enable the programs that consume data from the semantic web to reconcile these differences, or at least, avoid the problems that arise by applying an overly simple aggregation model to such data sources.

In the past, AI researchers have encountered similar issues when aggregating structured knowledge from different people or even the same person at different times. To handle these, mechanisms such as contexts and micro-theories have been proposed and implemented in projects such as Cyc [17]. While the kind of phenomena encountered in those projects are substantially more intricate and unlikely to be encountered in the near future on the semantic web, the scale and federated nature of the semantic web pose a new set of challenges. We believe that a context mechanism that is similar in spirit to the earlier context mechanisms will be not only useful, but required to achieve the semantic web vision. However, the differences between AI systems and the semantic web also mean that a context mechanism for the semantic web will have substantial differences from the AI context mechanisms. In this paper, we discuss the motivation for some of the basic requirements and present some possibilities for a context mechanism for the semantic web.

We begin by recounting some of the problems in aggregating data from different sources that were encountered on the TAP [20] project. These examples provide the motivation for the capabilities required of a context mechanism for the semantic web. We then present a simple context mechanism for handling these problems. After that, we discuss the model theory extensions required to incorporate this mechanism into RDF. Finally, we discuss related work and semantic web issues and constructs related to contexts.

2 Overview of TAP

The TAP project [20], [18] has over the last three years attempted to create platforms for publishing to and consuming data from the Semantic Web. Using this platform, we have built a number of applications [19] both to validate our assumptions and to help bootstrap the semantic web.

Building anything more than the simplest toy applications requires a substantial amount of data. Unfortunately, the semantic web, in its current stage, has little to offer in this regard. On the other hand, we do have a very large body of unstructured knowledge in the human readable World Wide Web. So, both to solve our problem and to bootstrap the semantic web, we have created a large scale knowledge extraction and aggregation system called onTAP. onTAP includes 207 HTML page templates which are able to read and extract knowledge from 38 different high quality web sites. The HTML template system has currently read over 150,000 pages, discovering over 1.6 million entities and asserting over 6 million triples about these entities. The system that aggregates this data can run in either a dynamic mode, in response to a query from an application, or in a batch mode. In batch mode, this aggregator is used to create a classification index for keyword-based queries and for scanning of documents for referenced entities.

onTAP also includes a system to read political news articles from Yahoo! News. The articles in this news feed are matched against natural language patterns to extract entities, attributes of and relationships between these entities.

Work is currently underway to expand this system to read more news sources. This system has read 46,113 articles to discover 6,052 unique entities and 137,082 triples about them.

Most of the content of TAP is obtained via these templates that draw on HTML pages that are targeted at humans. However, we believe that observations regarding the contextual phenomena associated with this data can be extrapolated to the case where the data is directly published by the sites in a structured form. Most of the sites that TAP uses do have their content in a structured form in a database. Front end processors query the database and format the data into HTML. When a site such as Amazon makes its data available in a structured form, they publish the same data. That is, the format and markup language used are different, but the assumptions, approximations and other contextual phenomena that are in their HTML pages are also present in the structured data that they publish.

3 Contextual Phenomena

In this section we discuss some of the contextual phenomena that we observed in the process of building the TAP knowledge base that cause problems in data aggregation. The examples of contextual phenomena we observed can be classified into a small number of varieties of contexts. Here we describe each of these varieties along with some examples that we encountered. We also discuss how we would like to handle each of these cases during aggregation.

Class Differences Different sites often use a particular class in slightly different ways. Sites may differ on the meaning of seemingly unambiguous concepts such as *Person*. For example, the site Who2 classifies C3PO (the robot in Star Wars) as a person, whereas most other sites classify it as a robot. During aggregation, we should map Who2's use of *Person* to something more general. Alternately, we can believe what Who2 has to say about some resource, unless what it says contradicts what some other site says.

Propositional Attitude A related phenomenon is that of a site having an implicit propositional attitude. For example, many sites providing reviews of television shows specify that Josiah Bartlet (a character who plays the role of the President of the US in the television series 'West Wing') is the President of the United States. During aggregation, the propositional attitude of these statements should be made explicit.

Property Type Differences A common source of differences between sites is that property types such as *capacity* and *price* are used differently. An example is the capacity of nuclear power plants. These plants have two different kinds of capacities: a design capacity and an actual capacity. Some sites specify the design capacity while others specify the actual capacity, but in most cases they just refer to it as the capacity. When aggregating, we need to either create a generalization of the two capacities or determine which capacity a particular site is referring to and map the *capacity* on that site to the appropriate version of capacity in the aggregate.

Point of View More substantial differences occur when there are conflicting points of view. Is Taiwan a country of its own, or a province of China? The answer depends very strongly on which site is queried. Similarly, different sites classify Hamas as a terrorist organization, a freedom fighting organization and a humanitarian organization. This kind of subjective data is often mixed in with more objective data (like the head of Hamas or the capital of Taiwan). When aggregating data from these sites, we would like to either make the subjectivity explicit (through the use of propositional attitudes) or only selectively import those facts that are not contentious.

Implicit Time Sites often publish a piece of data that is true at the time of publication, with the temporal qualification being left implicit. Equally often, this data does not get updated when it no longer holds. There are a number of sites that list Bill Clinton as the President of the US, that refer to Yugoslavia as a country, etc. Unfortunately, such implicitly temporally qualified data is often mixed in with data that is not temporally qualified. For example, it is still true that Bill Clinton graduated from Yale and the latitude and longitude of Sarajevo have not changed. When aggregating data from these sites, we would like to either make the time explicit or only selectively import those facts that are not likely to have changed.

A similar phenomenon is that of a site leaving the unit of measure associated with an attribute value implicit. So, instead of specifying the price as US$40, they might simply say 40. In such cases, we need to either make the unit of measure explicit or perform the appropriate conversion.

Approximations Approximations are another source of differences between sites. For example, the CIA World Factbook provides approximate values for the population, area, etc. of all countries. More accurate numbers are typically available from the governments of each of these countries, only some of which are online. We like to be able to combine data from the CIA World Factbook and data from the governments, preferring the government data when it is available.

We recognize that these differences could be because the TAP data was obtained by extracting structured data from HTML pages that were intended for human consumption. However, these phenomena are not an artifact of the information being published in an unstructured format. Most of the information on these sites is drawn from structured databases and these phenomena manifest themselves in these databases as well. Consequently, we believe that these problems will persist even when the data is made available in a machine readable form.

These kinds of differences between sites pose problems when data from these sites is aggregated. The problem is not that some of these sites are not trustworthy or that all of their data is bad. In fact, sources of data such as the CIA Factbook and Who2 are rich and useful repositories that should part of the semantic web. What we need is a mechanism to factor the kinds of differences listed above as part of the data aggregation process. Various formalizations of context mechanisms have been proposed in the AI literature to handle this process of factoring differences in representations between knowledge bases. In the next section we first briefly review some of the context related concepts from AI, note the differences between those and our requirements for the semantic web and then, in the following section, propose a simplified version to solve our problem for the semantic web.

4 Contexts in AI

Contexts as first class objects in knowledge representation systems have been the subject of much study in AI. KR systems as early as KRL [2] incorporated a notion of contexts. The first steps towards introducing contexts as formal objects were taken by McCarthy ([14], [15]) and Guha [10] in the early 1990s. This was followed by a number of alternate formulations and improvements by Buvac [6] [4], Fikes [5], Giunchiglia [8], Nayak [16] and others. Contexts/Microtheories are an important part of many current KR systems such as Cyc [17]. Contexts remain an active area of research in AI and Philosophy.

Contexts have been used in AI to handle a wide range of phenomena, a categorization of which can be found in [9]. They have been used in natural language understanding to model indexicals and other issues that arise at the semantic and pragmatic processing layers. They have found extensive use in commonsense reasoning systems where they are used to circumscribe the scope of naive theories. These systems use nested contexts, with the system being able to *transcend* the outermost context to create a new outer context. Both common sense and natural language systems also have a class of contexts that are ephemeral, that might correspond to a particular utterance or to a particular problem solving task. The ability to easily introduce a new context and infer attributes of that context adds substantial complexity to these context mechanisms. Contexts have also been used in model based reasoning [16] to partition models at different levels of abstraction.

The scope and complexity of the AI problems that contexts have been employed for is substantially more than anything we expect to encounter on the semantic web. The primary role for contexts on the semantic web is to factor the differences (like the ones described earlier) between data sources when aggregating data from them. Consequently, we do not need nested contexts, ephemeral contexts and the ability to transcend contexts.

On the other hand, the expected scale and highly distributed nature of the semantic web is in contrast to AI systems, most of which are much smaller and centralized. So, while we don't need the level of functionality provided by the AI formulations of contexts, we do place stronger constraints on the computational complexity and ease of use of the context mechanism.

In the next section, we develop a context mechanism for the semantic web. In the following section, we discuss the model theory extensions required for this mechanism.

5 Contexts for the SW

We present our context mechanism in the following setting. We have semantic web content (in RDFS or one of the OWL dialects) from a number of different URLs. The content from each URL is assumed to be uniform, but there may be differences like the ones described earlier between the content from the different URLs. We would like to create an internally consistent aggregate of the data from these different sites. The aggregate should be maximal in the sense that it should incorporate as much of the data from the different URLs as possible.

Each data source (or collection of data sources) is abstracted into a context. Contexts are first class resources that are instances of the class *Context*[3]. We define a PropertyType (*contextURL*) whose domain is *Context* that specifies the location(s) of the data source(s) corresponding to the context. The contents of the data source(s) are said to be true in that context. For the sake of keeping the description simple, for the remainder of this paper, unless otherwise specified, we will assume that the context has a single data source and that the URL of the context is that of the data source. So, for example, if a chunk of RDF is available at the URL *tap.stanford.edu/People.rdf*, we can have a context corresponding to this URL and the contents of this URL are said to be true in that context. Since the context is a resource, like any other resource on the semantic web, other chunks of RDFS/OWL can refer to it.

We are interested in defining contexts that aggregate data from other contexts. In keeping with the spirit of the semantic web, we would like to do this by declaratively specifying these new *Aggregate Contexts*. The different mechanisms that may be used in specifying these aggregate contexts define our design space. We start by examining the very general mechanism used in [10], [15] which has been followed by others in the AI community. We then present a much simpler, though less expressive variant that might be adequate for the semantic web.

Guha [10] and McCarthy [15] introduced a special symbol *ist* and the notation $ist(c_i, \varphi)$ to state that a proposition φ is true in the context c_i. Further, these statements can be nested, so that statements like $ist(c_i, ist(c_j, \varphi))$ can be used to contextualize the interpretation of contexts themselves. The system is always *in* some context. The system can enter and exit contexts. At any point in time, there is an outermost context, which can be *transcended* by creating a new outer context. A symbol can denote different objects in different contexts and the domains associated with different contexts can be different. Since contexts are first class objects in the domain, one can quantify over them, have functions whose range is a context, etc. All this allows one to write very expressive formulae that **lift** axioms from one context to another. While this is very convenient, it also makes it quite difficult to provide an adequate model theory and extremely difficult to compute with.

Nayak [16], Buvac [3] and others have tried to simplify this general formulation by introducing restrictions. Nayak considers the case where no nesting is

[3] We will drop namespace qualifiers, etc. for the sake of readability. Terms in *this* font refer to RDF resources.

allowed, contexts may only appear as the first argument to *ist* and are not first class objects (i.e., cannot be quantified over, etc.). Under these constraints, a classical modal logic like S5 can be used to provide a model theory for contexts. Buvac considers the case where a symbol is restricted to denote the same object in all contexts. For the purposes of the semantic web, Nayak's restrictions seem rather severe, but Buvac's may be acceptable. Both assume the Barcan formula:

$$\forall(x) ist(c, \varphi(x)) \leftrightarrow ist(c, \forall(x)\varphi(x)) \tag{1}$$

While these restrictions allow them to define a clean model theory, they are not enough to give us computational tractability. In fact, it is easy to show that all of these logics are not even semi-decidable.

Giunchiglia [8] and other researchers at Trento have used the notion of a bridge rule to formalize contexts. Much of their work is in the propositional realm and hence not directly relevant to the semantic web.

A general theme behind all these approaches is the introduction of the single interpreted symbol *ist*. *ist* is the only new symbol for which the underlying logic provides an interpretation. This is in contrast with RDF, RDFS, etc. in which a number of symbols (e.g., *Class*, *PropertyType*, *subClassOf*, ...) are all interpreted by the logic. We now extend this approach to handle contexts.

An aggregate context is a context whose contents (i.e., the statements that are true in that context) are lifted from other contexts. That is, they are imported into the aggregate context from other contexts after appropriate normalization/factoring. We now introduce a number of vocabulary elements that can be used to specify this lifting. This list is not exhaustive, but is adequate to cover the most common types of contextual differences that we discussed in section 3. We give informal descriptions and axiomatic definitions of these terms here and in the next section, outline the approach for defining the model theory for these constructs. We will use *ist* as a predicate in the meta-theory for the axiomatic definitions, even though it is not part of the base language.

- **AggregateContext**: This subclass of contexts corresponds to aggregate contexts. Since the semantic web allows anyone to make statements about any resource, there can be complications when different sites provide different definitions for a particular aggregate context. More specifically, allowing other contexts to specify what should be imported into a context, while safe in simple languages like RDF/S, opens the doors to paradoxes in more expressive languages like OWL. Even with RDF/S, it is important that the lifting process be simple to execute. To achieve this, we constrain the URL of an aggregate context to contain the full specification of what it imports. In other words, a lifting rule for importing into a particular context is true only in that context. We will later consider a semantic constraint for enforcing this.
- **importsFrom**: This is a property type whose domain is *AggregateContext* and whose range is *Context*. If c_1 *importsFrom* c_2, then everything that is true in c_2 is also true in c_1.

$$ist(c_2, p) \wedge ist(c1, importsFrom(c_1, c_2)) \rightarrow ist(c_1, p) \tag{2}$$

We do not allow cycles in the graph defined by *importsFrom*. *importsFrom* is the simplest form of lifting and corresponds to the inclusion of one context into another. The more sophisticated forms of lifting require us to create a resource for the lifting rule.

- **LiftingRule**: This class has all the lifting rules as its instances. Each *LiftingRule* must have a single *AggregateContext* as the value for *targetContext* and single *Context* for *sourceContext*. We have subclasses of *LiftingRule* for the different kinds of lifting we would like to perform. An *AggregateContext* may have any number of lifting rules that lift into it.

  Lifting rules generally ignore some of the triples in the source context, import some of the triples without any modification, transform and then import some other set of triples and optionally add some new set of triples into the aggregate context. The specification of a lifting rule involves specifying which triples to import or which triples to add and how to perform the necessary transformations. Some lifting rules may also specify a preference for one source over another for some class of triples. Our goal here is not to specify an exhaustive set of transformations, but to cover some of the important ones and provide a flavor for the general approach.

  An important factor that impacts the representation of these *LiftingRule*s is whether the aggregation process is restricted to be monotonic. If the process is allowed to be non-monotonic, the addition of a new LiftingRule to an aggregate context may cause certain triples to no longer hold. Non-monotonic lifting rules have the ability to say that everything not explicitly specified to be ignored or modified is to be imported. Consequently, they are easier to write, but do have the disadvantages of non-monotonicity. We describe the monotonic version and then suggest how it might be made more terse by introducing a non-monotonic construct.
- **Selective Importing**: These lifting rules explicitly specify the triples that should be directly imported from the source to the destination. Each triple may be specified by the property type or the first argument or the second argument. Optionally, these importing rules can specify the constraints on the first/second argument or combinations of property type and constraints on first/second argument etc. Examples: import *capitalCity* and *area* from the CIA Factbook. Import everything for instances of *Book* and *AudioCD* from Amazon. Import *manufacturer* for instances of *ElectronicsProduct* from Amazon.

$$
\begin{aligned}
&ist(c_i,\ targetContext(lr,c_i) \wedge sourceContext(lr,c_j) \wedge sourceFilter(lr,sc) \wedge \\
&\qquad targetFilter(lr,tc) \wedge propFilter(lr,p) \wedge type(lr, SelectiveImportLiftingRule)) \\
&ist(c_j,\ type(x,sc) \wedge type(y,tc) \wedge p(x,y)) \rightarrow \\
&ist(c_i,\ p(x,y))
\end{aligned} \tag{3}
$$

- **Preference Rules**: In many cases, a number of sources have sparse data about a set of entities, i.e., each of them might be missing some of the attributes for some of the entities they refer to and we might want to mitigate this sparcity by combining the data from the different sources. However, we might a preference for one source over another, if the preferred source has the data. A preference rule can either specify a total preference ordering on list of sources or simply that one particular source is preferred over another. As with Selective Importing lifting rules, a preference rule can be constrained to apply to only a particular category of triples. Example: Who2 has more detailed information about more celebrities than IMDB, but IMDB's data is more accurate. This class of lifting rule allows us to combine Who2 with IMDB, preferring IMDB over Who2 if both have values for a particular property type for the same individual.

A more sophisticated (but substantially more computationally complex) version of preference lifting rules is in terms of consistency, i.e., if an inconsistency is detected in the target context, triples from the less preferred context are to be eliminated first (to try and restore consistency).

Preference Lifting rules are non-monotonic across contexts in the sense that the addition of a new triple in one of the source contexts can cause another triple in the target context to be removed. However, they do not induce non-monotonicity within a context.

$$\begin{aligned}
&ist(c_i, targetContext(lr, c_i) \wedge sourceContext(lr, c_j) \wedge sourceContext(lr, c_k) \wedge \\
&\qquad propFilter(lr, p) \wedge sourceFilter(lr, sc) \wedge targetFilter(lr, tc) \wedge \\
&\qquad preferred(lr, c_j) \wedge type(lr, PreferenceLiftingRule)) \wedge \\
&ist(c_j, p(x, y) \wedge type(x, sc) \wedge type(y, tc)) \wedge \\
\neg &ist(c_k, (\exists(z) p(x, z) \wedge type(x, sc) \wedge type(z, tc) \wedge (z \neq x)) \rightarrow \\
&ist(c_i, p(x, y))
\end{aligned} \tag{4}$$

– **Mapping Constants**: One of the most common transformations required is to distinguish between slightly different uses of the same term or to normalize the use of different terms for the same concept. These lifting rules specify the source term and the target term. As with selective importing lifting rules, we can constrain the application of these mappings to specific categories of triples. Example: Many different sites use the term *price*, some referring to price with tax, some to price with tax and shipping, etc. This class of lifting rules can be used to distinguish between them in the aggregate context. The earlier example of nuclear power plant capacity also falls into this category.

$$\begin{aligned}
&ist(c_i, targetContext(lr, c_i) \wedge sourceContext(lr, c_j) \wedge \\
&\qquad propMapTo(lr, p_2) \wedge sourceFilter(lr, sc) \wedge \\
&\qquad propMapFrom(lr, p_1) \wedge type(lr, TermMappingLiftingRule)) \wedge \\
&ist(c_j, p_1(x, y) \wedge type(x, sc) \wedge type(y, tc)) \rightarrow ist(c_i, p_2(x, y))
\end{aligned} \tag{5}$$

– **Mapping more complex graph patterns**: All the above lifting rules deal with cases where the target graph is isomorphic to (portions of) the source graph. Sometimes, this constraint cannot be satisfied. For example, if the source leaves time implicit and the target has an explicit model of time, the source and target graphs are not likely to be isomorphic.

Assuming we don't use an explicit *ist* in the base language, we can introduce a special construct for each phenomenon, such as making implicit time explicit. With this approach, we would introduce a property type ($contextTemporalModel$) to specify the model of time used by a context (implicit, Situation Calculus, Histories, etc.). In the case where the context used implicit time, we use another property type ($contextImplicitTime$) to specify the implicit time. Using a reified version of the Situation Calculus to represent time, the following axiom defines these property types.

$$\begin{aligned}
&ist(c_i, contextTemporalModel(c_i, ImplicitStaticTime) \land contextImplicitTime(c_i, t_i)) \land \\
&ist(cj, targetFilter(lr, tc) \land contextTemporalModel(c_j, SitCatModel) \land \\
&\qquad type(lr, SitCalLiftingRule) \land propFilter(lr, p) \land sourceFilter(lr, sc) \land \\
&ist(c_j, type(x, sc) \land type(y, tc) \land p(x, y)) \rightarrow \\
&ist(c_i, (\exists z) type(z, SitProp) \land time(z, t_i) \land \\
&\qquad sitProp(z, p) \land sitSource(z, x) \land sitTarget(z, y))
\end{aligned} \tag{6}$$

A more general solution is to identify common transformation patterns (as opposed to particular phenomenon) and introduce vocabulary to handle these. For example, a common pattern involves reifying a particular triple to make something explicit. Examples of this include making time and propositional attitudes. Another common pattern involves transforming a literal into a resource. A common example of this is to make the unit of measure or language explicit.

- **Default Lifting**: The constructs described above are monotonic. In practice, it is often convenient to be able to say that all of the contents of one context should be included in the aggregate context without any modification unless one of the other lifting rules applies. To do this, we introduce a property type analogous to $importsFrom$, called $defaultImportsFrom$ that specifies this.

While not exhaustive, we believe this vocabulary and associated set of lifting rules are sufficient to solve current and many future issues that arise in data aggregation on the semantic web. More importantly, this functionality can be incorporated into the semantic web with fairly small and simple additions to the existing standards.

6 Model Theory

In the last section we discussed some potential alternatives for introducing contexts into the semantic web. In this section, we discuss issues related to the model theory. Of course, the particular alternative used has significant impact on the model theory. But there are some basic changes that are required, independent of the approach used. We discuss these first and then consider the impact on the model theory corresponding to the different approaches.

We restrict our attention to the model theory for RDFS. It should be possible to provide a similar treatment for at least the simpler versions of OWL.

The most basic change introduced by the addition of contexts is that the denotation of a resource is not just a function of the term and the interpretation (or structure), but also of the context in which that term occurs. We will assume that the denotation of literals is not affected by the context. This context dependence can be incorporated as follows. The definitions of interpretation and satisfaction (as given in the RDF Model Theory [11]) are changed as follows.

A simple interpretation I of a vocabulary V is defined by:

1. A non-empty set IR of resources, called the domain or universe of I. C is a subset of IR corresponding contexts.
2. A set IP, called the set of properties of I.
3. A mapping IEXT from IP into the powerset of IR x IR i.e. the set of sets of pairs $< x, y >$ with x and y in IR .
4. A mapping IS from the power set URI x URI in V into (IR union IP). The power set URI x URI corresponds to resource, context pairs. So, under this modified definition, IS maps resources to objects in the domain *in a context*. This is the only non-trivial change.
5. A mapping IL from typed literals in V into IR.
6. A distinguished subset LV of IR, called the set of literal values, which contains all the plain literals in V

The denotation of a ground graph is now with respect to the context it occurs in, which manifests itself as the second URI in the argument to IS in (4) above. We update and extend the definition of satisfaction so that instead of defining satisfaction for a graph, we define it for a set of graphs, each in a context. The updated definition of satisfaction is as follows:

1. if E is a plain literal "aaa" occurring in c in V then I(E,c) = aaa
2. if E is a plain literal "aaa"@ttt in V occurring in c then I(E,c) = $< aaa, ttt >$
3. if E is a typed literal in V occurring in c then I(E,c) = IL(E)
4. if E is a URI reference in V occurring in the context c, then I(E,c) = IS(E,c)
5. If E is a ground triple s p o in the context c then I(E,c) = true if c, s, p and o are in V, IS(p,c) is in IP and $< IS(s, c), IS(o, c) >$ is in IEXT(IS(p,c)). Otherwise I(E,c)= false.
6. if E is a ground RDF graph in the context c then I(E,c) = false if I(E',c) = false for some triple E' in E, otherwise I(E,c) =true.
7. if $< E_i, c_i >$ are a set of ground graphs occuring each occurring in the corresponding context, then I($< E_i, c_i >$) = false if I(E_i, c_i) = false for some grap E_i associated with the context c_i, otherwise I($< E_i, c_i >$) =true.

Finally, the definition of entailment is updated so that a ground graph G_1 in a context C_1 is entailed by a set of graph-context pairs $< G_i, C_i >$ if $< G_1, C_1 >$ is true under every interpretation under which ($< G_i, C_i >$) is true.

It is easy to see that the only significant change is the addition of the context argument to the interpretation. The change in the definition of statisfaction is that we can no longer simply merge graphs without regard to where they occur. This is of course the the point of this whole exercise.

The other changes to the model theory depend on whether we have a predicate/modal like *ist* in the vocabulary and the constraints we impose on quantifying over contexts, quantifying into contexts, whether we want the Barcan formula, etc. Since this approach has been discussed quite exhaustively in the literature ([3], [16], Giunchiglia [8], [7]) and since the use of an explicit *ist* is not very appropriate for the semantic web, we will not go into the details of that approach here.

The less expressive approach which eschews the use of an explicit symbol like *ist* in favour of a suite of specialized interpreted symbols (such as $importsFrom$) involves a less substantial, though more cumbersome change to the model theory. Following the pattern of sections 3 and 4 of [11], we can provide interpretations for the different context specific vocabulary items introduced in the last section. For example, the interpretation for the term $importsFrom$ is: *if I(c1 importsFrom c2, c1) is true and I(E,c2) is true then I(E, c1) is true.* The interpretations for the other vocabulary terms, though more verbose, are still straightforward model theoretic equivalents of the axiomatic definitions given earlier.

Finally, we need to add a semantic constraint so that a $LiftingRule$ for specifying what to lift into a particular context is true only in that context. So, *if I(c1, c2)* $\neq$ *I(c2, c2) then I(c2 targetContext lr, c1) is false.*

It is important to note that all the theorems and lemmas of [11] continue to hold. Overall, it appears that a restricted context mechanism that does not use an explicit *ist* can be accommodated into the model theory without substantial perturbations.

7 Related Work and Discussion

The issue of contextuality and contexts as first class objects has been a topic of much discussion since the early days of RDF. In this section, we first discuss the relation between the context mechanism presented in this paper and reification (and related proposals). We then discuss the issue of non-monotonicity that is raised by contexts.

7.1 Reification

RDF provides reification as a mechanism for making statements about statements. There are significant differences between reification and contexts both in what they are intended for and in their structure. Reification is intended to enable statements about potential statements (which may or may not be true). They can be useful for making statements about provenance. Since they have no coupling with the truth of the triple that has been reified, they cannot be used to relate the truth of a triple in one graph to its truth in another graph. Consequently, it is hard to see how reification can be used to mediate aggregation.

In contrast, the primary goal of the context mechanism presented here is to enable the aggregation while factoring in the differences between the data sources being aggregated. It is only incidental that the context mechanism may be used to make other kinds of statements about graphs. Since the goal is to aggregate triples that are true in the graphs being aggregated, contexts and truth are very closely related. Contexts and reification are also very different in their structure and in where they appear in the model theory. Because of the close coupling between truth and contexts, they cannot be a posteriori introduced at the RDF Vocabulary level. They appear in the guts of the model theory, in the definition of an interpretation. This is in contrast to the reification which does not play any significant role in the model theory.

In addition to the reification mechanism in RDF, the system/language CWM/N3 supports a construct called contexts. As far as we can make out, this notion of context is not substantially different from reification.

The shortcomings of reification have lead to numerous alternative proposals, the most substantial of which are those by Klyne [13], [12]. Indeed some of the constructs presented there, drawn from earlier work by Guha [10] and McCarthy [15], are very similar in spirit to those presented here. The extension to the model theory presented here is simpler. Unlike that work we avoid the use of an *ist* construct and significantly reduce the complexity. Further, our focus in this paper is on data aggregation, not on introducing contexts as a general piece of functionality into the semantic web. The general problem of contextual dependence is much too complex to tackle head-on (at least for the semantic web). We believe that we need to be driven by concrete tasks such as data aggregation to achieve progress.

7.2 Non-monotonicity

The context mechanism discussed in this paper leads to non-monotonic aggregation in the following sense. A graph G_1 might imply φ, but because of of selective lifting, preferential lifting or transformation during lifting, the aggregation of this graph with zero or more other graphs might not imply φ. It has been argued [1] that the semantic web should be based purely on classical monotonic logics for reasons of scalability. While monotonicity might be desirable at the local level, it is neither desirable nor feasible at the level of the semantic web as a whole. Further, the computational intractability associated with non-monotonic logics in AI are not because of non-monotonicity itself, but because of the particular mechanisms used to introduce it.

What we need is the ability to say that a particular theory T_1 implies φ without the fear that additions to T_1 cause φ to no longer be true. However, T_1, when aggregated with other theories T_2, T_3, ... might lead to a superior theory which does not imply φ. The examples given in section 3, especially the example of the CIA factbook combined with census data from particular countries is a good example of why this is desirable. The issue is not that of the CIA factbook not being trustworthy. It has a lot of good data that we would like to be able to use. It would be undesirable if we had to choose between it and data from the census bureaus. In the past, lacking a proper context mechanism, we have been unable to associate a granularity with the monotonicity. With the context mechanism, we are able to control the points of non-monotonicity. With the context mechanism, we are able to distinguish between non-monotonicity within a context versus non-monotonicity across contexts. We remain monotonic within contexts, but can be non-monotonic across contexts.

8 Conclusions

As shown by the experience with TAP and many other projects that aggregate independently created data, differences in assumptions, use of terms, etc. lead to problems during aggregation. In this paper, we present an adaptation of the context mechanism from AI to solve these problems. We avoid the general *ist* construct and the associated complexity but provide enough support to solve the contextual problems that arise in aggregation.

References

1. Tim Berners-Lee. Semantic Web Architecture. `http://www.w3.org/DesignIssues/ Rules.html`.
2. Daniel G. Bobrow and Terry Winograd. On overview of krl, a knowledge representation language. *Cognitive Science*, 1:3–46, 1997.
3. Sasa Buvač. Quantificational logic of context. In Howard Shrobe and Ted Senator, editors, *AAAI 1996*, pages 600–606, Menlo Park, California, 1996. AAAI Press.
4. Saša Buvač, Vanja Buvač, and Ian Mason. Metamathematics of contexts. *Fundamenta Mathematicae*, 23(3), 1995. Available from http:// www-formal.stanford.edu/ buvac.
5. Saša Buvač and Richard Fikes. A declarative formalization of knowledge translation. In *Proceedings of the ACM CIKM: the Fourth International Conference in Information and Knowledge Management*, 1995. Available from http:// www-formal.stanford.edu/ buvac.
6. Saša Buvač and Ian Mason. Propositional logic of context. In *AAAI*, pages 412–419, 1993.
7. Valeria de Paiva. Natural deduction and context as (constructive) modality. In *CONTEXT 2003*, pages 116–129, 2003.
8. Fausto Giunchiglia and Chiara Ghidini. Local models semantics, or contextual reasoning = locality + compatibility. In Anthony G. Cohn, Lenhart Schubert, and Stuart C. Shapiro, editors, *KR'98: Principles of Knowledge Representation and Reasoning*, pages 282–289. Morgan Kaufmann, San Francisco, California, 1998.
9. R. Guha and John McCarthy. Varieties of contexts. In *CONTEXT 2003*, pages 164–177, 2003.
10. Ramanathan V. Guha. Contexts: a formalization and some applications. Technical Report STAN-CS-91-1399, Stanford CS Dept., Stanford, CA, 1991.
11. Pat Hayes. RDF Semantics. `http://www.w3.org/TR/rdf-mt/`.

12. Graham Klyne. Contexts and the semantic web. `http://public.research.mimesweeper. com/RDF/ RDFContexts.html`.
13. Graham Klyne. Contexts for the semantic web. `http://www.ninebynine.org/RDFNotes/ UsingContextsWithRDF.html`.
14. John McCarthy. Generality in artificial intelligence. In Vladimir Lifschitz, editor, *Formalizing Common Sense: Papers by John McCarthy*, pages 226–236. Ablex Publishing Corporation, Norwood, New Jersey, 1990.
15. John McCarthy. Notes on formalizing contexts. In Ruzena Bajcsy, editor, *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 555–560, San Mateo, California, 1993. Morgan Kaufmann.
16. P. Pandurang Nayak. Representing multiple theories. In B. Hayes-Roth and R. Korf, editors, *AAAI-94*, pages 1154–1160, Menlo Park, CA, 1994.
17. R.Guha and D. Lenat. Context dependence of representations in cyc. In *Colloque ICO*, 1993.
18. R.Guha and R. McCool. Tap: A semantic web platform. *Computer Networks*, 42:557 – 577, 2003.
19. R.Guha, R. McCool, and E. Miller. Semantic search. In *WWW 2004, Budapest, Hungary*, 2003.
20. R.Guha and Rob McCool. Tap: Towards a web of data. `http://tap.stanford.edu/`.