A data-intensive lightweight semantic wrapper approach to aid information integration.

Dave Braines[1] **Yannis Kalfoglou, Paul Smart, Nigel Shadbolt**[2] and **Jie Bao**[3]

Abstract. We argue for the flexible use of lightweight ontologies to aid information integration. Our proposed approach is grounded on the availability and exploitation of existing data sources in a networked environment such as the world wide web (instance data as it is commonly known in the description logic and ontology community). We have devised a mechanism using Semantic Web technologies that wraps each existing data source with semantic information, and we refer to this technique as SWEDER (Semantic Wrapping of Existing Data Sources with Embedded Rules). This technique provides representational homogeneity and a firm basis for information integration amongst these semantically enabled data sources. This technique also directly supports information integration though the use of context ontologies to align two or more semantically wrapped data sources and capture the rules that define these integrations. We have tested this proposed approach using a simple implementation in the domain of organisational and communication data and we speculate on the future directions for this lightweight approach to semantic enablement and contextual alignment of existing network-available data sources.

1 Introduction

A plethora of data is available in structured forms today, either in existing Semantic Web encodings such as OWL/RDF or, more likely, in more traditional formats such as XML, CSV, HTML or relational databases. This data is available to the consumer today, usually via URIs resolving to network or local addresses, but may also be sourced from directly referenced files and other non-URI referenced resources. The data itself can take any form, but we propose that it can be relatively easily semantically wrapped through a lightweight application of OWL/RDF to represent the data in the form of entities (classes), attributes (data properties) and relationships (object properties). The purpose of this lightweight semantic wrapping is to provide representational homogeneity for these existing data sources, thereby providing a firm semantic basis for any downstream consumption in potentially unknown contexts.

The details regarding how this semantic wrapping is achieved are peripheral to the scope of this paper which is to focus on how to leverage and capitalize on the end result: *semantically wrapped data sources*. Our focus in this paper is mainly on the subsequent creation of context ontologies to specifically capture the alignments between these semantically wrapped data sources, and we assume this representational homogeneity as a pre-requisite for our data sources. In practical terms this semantic wrapping is usually achieved through manual design and construction of a simple ontology, or reuse of an existing published ontology. Then the corresponding instance data is generated through the application of simple transformations of existing data-sources to the corresponding RDF/OWL representations. A suitable existing pattern for this work is that of RDFa[4], microformats[5] (and GRDDL[6]) which are popular techniques for semantically enriching existing web data sources today, albeit within existing web markup languages rather than as stand-alone ontology instance data as we are proposing.

We elaborate on making use of this semantically wrapped data in the next section where we introduce the notion of a lightweight form for representing the alignments or relationships between these existing data sources: *context ontologies* (section 2). These are used in a principled manner which we describe in section 2.1, and we apply in an example case in section 3. We go on to discuss proposed extensions to our work in section 3.1, related work in section 4, and conclude this paper in section 5.

2 Context ontologies

We adopt a dynamic notion of context which is not common to the formal notions presented in the AI literature (see, for example, the seminal work in [4] on formalizing contexts as first class objects). Our aim is to use context dynamically in order to capture and define each purpose for which data is used, specifically enabling it to be used by a consumer application. We do not take into account the initial context of existing data, as all data exists for a specific reason and with a specific format, but we treat this originating context as a precursor to our interest: how to enable seamless processing of many data sources by a variety of consumer applications in different contexts. In the simplest scenario, a consumer application will merely consume a single data source for further processing, and in this extremely simple case one could argue that the consumer application has added no additional context to the original data. This is an unlikely scenario for a consumer application of any real value, and it is more likely that a consumer application will consume multiple data sources and fuse them or otherwise make use of both data sources and the relationships between these sources. In this scenario we argue that the consumer application does add a context to these data sources, not least because a specific combination of multiple data sources has been selected to fulfil a particular need. This context is

[1] Emerging Technology Services, IBM United Kingdom Ltd., Hursley Park, Winchester, SO21 2JN, UK, email: dave_braines@uk.ibm.com
[2] School of Electronics and Computer Science, University of Southampton, UK, email: {y.kalfoglou,ps02v,nrs}@ecs.soton.ac.uk
[3] Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY12180, USA, email: baojie@cs.rpi.edu

[4] http://www.w3.org/2006/07/SWD/RDFa/
[5] http://microformats.org/
[6] http://www.w3.org/2004/01/rdxh/spec

captured through the creation of a context ontology which specifically integrates the concepts from any semantically wrapped data sources that it references. This context ontology is then able to be easily used by the consumer application, thereby reading the various semantically wrapped data sources, processing the instance data and executing embedded rules to derive further information or alignments. The result of this could be published as instance data conforming to the context ontology and then made available for further consumption by unknown downstream consumer applications. For example, a context ontology may be created which aligns concepts from two semantically wrapped data sources containing geographic feature data and person location data. A consumer application can then use this new context ontology, execute the embedded rules to fuse these two data sources in specific ways, and infer the interesting intersection of these as defined within the context ontology. In this example case, the list of people attending specific geographic features of interest. This inferred additional data can then be used by the consumer application and can additionally be published as new instance data conforming to the context ontology.

Of course the fusion of two data sources by an application to achieve the results above could easily be achieved with existing technology, and does not require semantic representation. As a matter of fact, one could view the current trend of mashups[7] as a successful (usually non-semantic) form of such integrations. The specific benefits of semantically enabling the data sources and capturing the alignment representation and rules in a context ontology as defined in our approach lie in the representational homogeneity achieved through this approach, the self-defining and portable nature of the context ontologies and their embedded rules, and the ease with which consumer applications can use these context ontologies along with the appropriate semantically wrapped data sources. Further important capabilities are also enabled through the use of this approach, most notably the support for referencing common definitions via URIs to enable more rapid understanding and information integration.

A key aspect of our proposal is facilitating the creation, representation and consumption of information integration rules within these context ontologies, and this is something that existing OWL based solutions do not readily support. There are emerging standards in this area, notably SWRL[8] and potentially RIF[9], but for various reasons we have chosen a lightweight, pragmatic approach and use SPARQL[10] construct clauses to define these rules and store them as instances within our context ontology. This allows any SPARQL enabled endpoint to execute the rules and instantiate the inferred results directly from the construct clause held in the embedded rule without the need for any specific additional rule execution engine.

We store each actual SPARQL construct clause rule as instance data directly in the context ontology to which it applies, thus enabling these rules to be passed to the consumer application as part of the context ontology itself. In further iterations of this work we plan to introduce richer representation formats (such as SWRL, RIF) as these representations could be used to generate SPARQL construct clauses which would be executed as per our current solution. The use of these richer representation languages would expose the semantics of these information integration rules to consuming applications rather than the current solution which simply records the text of the SPARQL construct clause without providing any semantic representation of the rule which the SPARQL implements.

[7] http://en.wikipedia.org/wiki/Mashup_(web_application_hybrid)
[8] http://www.w3.org/Submission/SWRL/
[9] http://www.w3.org/2005/rules/
[10] http://www.w3.org/TR/rdf-sparql-query/

2.1 A conceptual model

Conceptually and practically we use a two-tier model to represent each of our ontologies, both for the simple semantic wrappings of existing data sources, and for the subsequent context ontologies which are created to capture the alignments of ontologies. This two-tier approach allows for a clear separation between the representation of the model and the capture of any associated rules. An example of this two-tier doughnut shaped model is shown in figure 1. The model ontology is at the centre, and it is comprised of traditional ontology modeling concepts: entities, attributes and relationships, as described earlier. This ontology is imported into the outer ontology, which simply adds support for rules to be defined against the model. In our current implementation this takes the form of an import to a generic information integration rules ontology which enables the SPARQL construct based rules to be represented as instances of simple entities. The separation of these two aspects of our ontologies enables the rules to be captured separately to the model, thus offering us a flexible way in which to improve the rule representation solution in the future without affecting the model ontology, and it also enables us to easily use existing ontologies and wrap them with our rules. We label this technique SWEDER (Semantic Wrapping of Existing Data Sources with Embedded Rules).

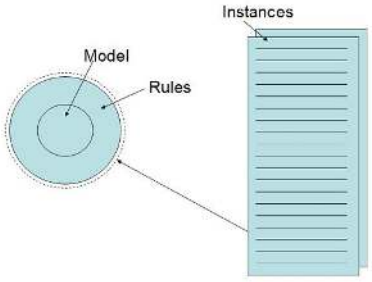

Figure 1. A doughnut shaped two-tier ontology model and associated instances.

The final aspect of our solution is the capture of context information for multiple ontologies, which we achieve via the creation of additional lightweight ontologies. These are the context ontologies we referred to previously and are built according to the same two-tier approach.

The context ontology defines any additional entities, attributes or relations which are relevant to the current context (in the inner model ontology), and also defines any instances of rules which are able to populate these additional items (in the outer rules ontology). The context ontology also imports any required source ontologies (which may of course be context ontologies themselves) and the new context ontology therefore captures the representation of this specific new context and embodies it in a semantic format consistent with the source ontologies. We visualize this approach in figure 2, and it should be noted that the imported ontologies can either be normal ontologies that exist already, or can be the semantically wrapped data source ontologies that we describe in this paper.

The final step is for a suitable consumer application to consume the context ontology and any associated instance data for the source ontologies. Since all of the integration rules are contained within the context ontology this consumer application simply invokes a standard process to extract all these rules (which are stored as SPARQL construct clause text), then executes them against the instance data

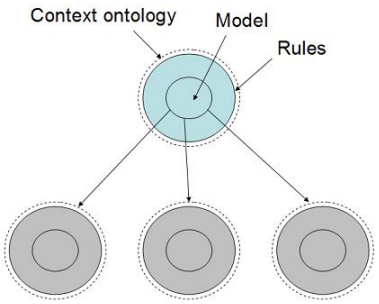

Figure 2. A typical context ontology.

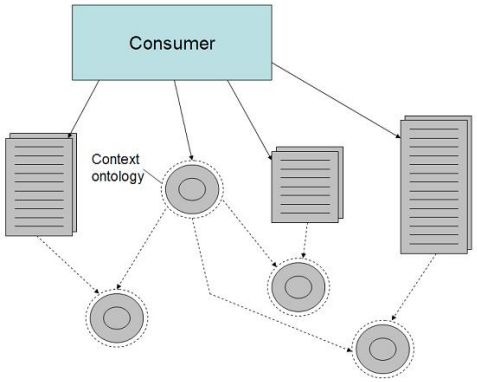

Figure 3. A consumer application interacts with context ontologies.

using an appropriate SPARQL endpoint. The results of these rule executions are that new instance data are created within the context ontology, and this can then be saved, published or further processed by the consumer application. The consumer application actually executes all the rules multiple times, until the set of all rule executions results in no further data being inferred. We depict diagrammatically the interaction with a consumer application in figure 3.

3 An example consumer application

In order to test our proposed technique we have applied SWEDER in the context of organisational and communication data. We used a variety of source data from existing applications, converting this to OWL/RDF based on simple ontologies defined in Protege[11]. The consumer application is built using the Jena framework from Hewlett Packard Labs[12] and the ARQ SPARQL processor for Jena. The source ontologies constructed in our example were:

- **Email** - this is a simple semantic representation of email data extracted from an email application. Includes `Email`, `EmailAddress` and `Tag` entities with multiple attributes and relations between them. (See figure 6);
- **Person** - this is a semantic representation of instant messaging system contacts and their groups. Includes `Person`, and `Relationship`. We could also extract this data from many sources such as FOAF[13], social network sites, etc.;
- **Organisation** - this is a semantic representation of basic organisation information such as *name*, *email suffix*, *homepage*, etc. The instance data for this was created specifically for the purposes of our experiment but could easily come from a CRM system or similar;
- **Project** - this a semantic representation of basic project information such as `project name`. This instance data for this was specifically created for the purposes of our experiment but could come from a DOAP[14] dataset or an application used to record project information.

It is noteworthy to point out that each of the above ontologies is completely stand-alone and requires no knowledge or understanding of data in the other. Conceptually these could have each been defined and created by different authors at different times, although for the purposes of our exercise we created each of these ourselves, but carefully ensured that each ontology was entirely separate from the others in terms of the constituent data and conceptual representation. Each of these ontologies makes use of appropriate RDF/OWL representations such as dependencies between properties, inverse relationships, etc. A single context ontology was then created to align the appropriate aspects of these source ontologies. Our approach supports multiple context ontologies, each for a specific alignment, but in our example case we used only one as we simply wished to demonstrate the value of these context ontologies and the recording of rules within them. The alignments we produced were captured using SPARQL construct rules that implement the following information integration tasks:

```
PREFIX context_m: <ContextModel.owl#>
PREFIX email_m: <EmailModel.owl#>
PREFIX org_m: <OrganisationModel.owl#>
PREFIX fn: <http://www.w3.org/2005/xpath-functions#>
CONSTRUCT
{
        ?o context_m:originatesEmailAddress ?ea .
}
WHERE
{
        ?ea email_m:processedEmailAddress ?easp .
        ?o org_m:emailSuffix ?eas .
        FILTER (fn:contains(fn:lower-case(?easp), fn:lower-case(?eas)))
}
```

Figure 4. A SPARQL example rule relating a person's email address to organisation.

- *EmailAddress to Person*: within the `Email` ontology (an excerpt of which is shown in figure 6), `EmailAddress` instances are created for each unique email address that is involved in sending or receiving an email. Each `EmailAddress` entity has a `rawEmailAddress` attribute containing the email address string which is that email address. Within the `Person` ontology a `Person` can have one or more values for the `emailAddress` attribute which contain their email address string(s). The rule we execute simply matches any `Person` with an `emailAddress` value which is identical to the `rawEmailAddress` of any `EmailAddress` entity. The construct clause populates a new relationship (object property) named `hasEmailAddress` on this `Person` and `hasPerson` on this `EmailAddress` to record the new inferred relationship between these two entities. We give an example of this SPARQL construct based rule in figure 5. From this we can subsequently infer a relationship between `Person` entities and `Email` entities and can now easily identify all emails that a `Person` has sent or received.

[11] Available from: http://protege.stanford.edu/
[12] Available from: http://jena.sourceforge.net/
[13] http://www.foaf-project.org/
[14] http://trac.usefulinc.com/doap

- *EmailAddress to Organisation*: within the `Organisation` ontology, `Organisation` instances are created, and each is populated with an `emailSuffix` string. Each `EmailAddress` entity has a `rawEmailAddress` attribute as described previously. The rule we execute matches any `Organisation` with an `emailSuffix` which is the same as the end of any `EmailAddress` entities `rawEmailAddress` attribute. The SPARQL construct clause populates a new relationship (object property) named `originatesEmailAddress` on this `Organisation` to record this inferred information.
- *Person to Organisation*: this builds on the previous rule, and identifies any `EmailAddress` which has a `Person` (`hasPerson`) and which has an `Organisation` (`hasOrganisation`). The rule then populates a new relationship (object property) named `employsPerson` on `Organisation` with a link to that `Person`. This rule relies on the previous two rules correctly instantiating `EmailAddress` to `Person` and `EmailAddress` to `Organisation` relationships. An example of such a rule is shown in figure 4. When we enable multiple rule executions this rule may infer additional information when it is run after the prerequisite rules.
- *Person to Project*: the `Email` ontology has multiple `Email` instances, many of which have already been tagged according to the name of the project that they relate to. This enables another rule to identify any `Email` with a `hasTag` text which is the same as any `Project name` or `alternativeName` attribute. This rule populates a new relationship (object property) named `relatedEmail` on `Project` and `relatedProject` on `Email`.
- *Project to Email* and *Project to Organisation*: these final two rules build on the same principles as before and identify each `Project` that has a relationship to a `Person` (or `Organisation`) and the `Email` which that `Person` (or `Organisation`) is involved with via the related `EmailAddress` entities. The results of these two rules are instantiated in the new `hasEmail` relationships (object properties) on `Project` and `Organisation`, and in the `hasProject` and `hasOrganisation` relationships on `Email`.

```
PREFIX context_m: <ContextModel.owl#>
PREFIX email_m: <EmailModel.owl#>
PREFIX person_m: <PersonModel.owl#>
PREFIX fn: <http://www.w3.org/2005/xpath-functions#>
CONSTRUCT
{
        ?p context_m:hasEmailAddress ?ea .
}
WHERE
{
        ?p person_m:emailAddress ?ease .
        ?ea email_m:processedEmailAddress ?easp .
        FILTER (fn:matches(?ease, ?easp, "i"))
}
```

Figure 5. A SPARQL example rule relating an email address to person.

The rules listed above are clearly very simple examples of the sorts of rules that may be desired by consumer applications and our proposed use of SPARQL construct clauses to represent these rules builds a flexible base against which richer and more complex rules can be written, limited only by the expressivity of SPARQL construct clauses.

In our simple demonstration we have built the consumer application to allow user navigation around this fused set of separate data sources, allowing the user to immediately see which emails relate to which projects, what organisations and people they are working with in the context of projects and so on. The instance data for this context ontology is also published out for potential further consumption simply through persisting it to an RDF/OWL file via the Jena API and making the URI of that file available to other consumers.

```
Email
  subject (String, functional)
  body (String, functional)
  emailType (String, functional)
  notesUrl (String, functional)
  universalID (String, functional)
  postedDate (dateTime, functional)
  hasResponse (Email)
  inResponseTo (Email, functional)
  hasTag (Tag)
  emailAddress (EmailAddress)
    emailAddressRecipient (EmailAddress)
      emailAddressTo (EmailAddress)
      emailAddressCc (EmailAddress)
      emailAddressBcc (EmailAddress)
    emailAddressFrom (EmailAddress, functional)

EmailAddress
  processedEmailAddress (String)
  rawEmailAddress (String)
  email (Email)
  receivesEmail (Email)
  receivesEmailDirect (Email)
  receivesEmailOnCc (Email)
  receivesEmailOnBcc (Email)
  sendsEmail (Email, functional)

Tag
  tagsEmail (Email, functional)
  tagName (String, functional)
  tagSource (String, functional)
```

Figure 6. The hierarchy of `Email` ontology entities and their attributes and relationships.

3.1 Extensions

We recognise some shortcomings in our current solution and aim to carry out further investigation into a number of specific areas to address these, most notably:

- The use of reification to record whether each instance data triple is stated or inferred. At the moment any inferred instance data triples are simply instantiated as a result of the SPARQL construct execution, and can then not easily be differentiated from the instance data triples originating in the source ontologies (other than by looking at the namespace into which they are persisted). Using a reification technique we would be able to record relevant provenance data such as the rule(s) which instantiated the instance data triple, the time, the application, etc.
- As the W3C[15] standardization work on rules languages and interoperability continues to mature we plan to extend the notion of context ontologies to support the representation of rules written in SWRL, RIF or another appropriate richer representation. These richer representations of rules would allow semantic information about the composition of the rules themselves to be conveyed, and would be used to generate the required SPARQL construct clauses.
- We also plan to use a flexible approach for disseminating the results of the information integration rules we presented in the previous section. In [3] we propose a novel mechanism for sharing and distributing ontology alignment information, POAF (Portable Ontology Aligned Fragments). POAF is agnostic as to what the alignment format is or to the type of data source used. In that sense, we could deploy a variant of the POAF solution to share and distribute the rules described in the previous section and even the context ontologies they operate on.

We also observe some aspects which loosely relate to this work, and which we will review further in our ongoing work:

- Our approach enables a limited form of distributed reasoning as it allows each instance of a consuming application to consume different data and publish their results. In some cases the results of these distributed consumer applications can then be collected and further analysed as appropriate.

[15] http://www.w3.org/

- This approach can be used to efficiently publish summary information about potentially private data when appropriate. In the example case we see a scenario where employee email data is processed locally to identify interesting contextual information, and in some cases this contextual information may be able to be then published to a wider audience whereas the actual email data is not.
- Finally, our work so far has identified that the SPARQL construct technique for building rules can be used to implement some of the standard RDF-S/OWL entailments. We specifically demonstrate this for `rdfs:subPropertyOf`, `owl:SymmetricProperty` and `owl:inverseOf`. This is a pragmatic solution to these specific RDF-S/OWL entailments where we use the RDF-S/OWL semantics to define occurrences of these in our model ontologies in the normal way, but we generate SPARQL construct clauses from rule templates to specifically instantiate each actual rule occurrence. This has two main benefits from our pragmatic perspective: firstly, there is no need to use a reasoner in addition to the SPARQL end point processing, and secondly, that we can use the same technique to instantiate and persist the resulting data. We do not propose that this SPARQL construct clause based implementation of these standard entailments should be used in preference to the capabilities offered by existing reasoners, but we note it here as a further capability for this rule representation and execution technique that we have described here.

4 Related work

Different notions of contexts have been proposed and investigated in the past. For example in [5] the authors argue for different types of contexts that contribute information relevant to natural language understanding. Each context is used to serve a different purpose, similar to our work where we adopt a dynamic notion context that is closely related and dependent on the use of source data. Our work uses source data and a set of semantic wrappers to elicit context and represent it in lightweight ontologies. Similarly, context has been used in [2] to aid in ontology elicitation whereby certain features of context dictate the primitive ontological constructs that will form up an ontology.

Information integration, the driver behind our work with semantically wrapped data and context ontologies, is also the focus of [6] but the authors deploy different means to achieve that: they propose to use a special kind of context knowledge, namely assumption knowledge, which refers to a set of implicit rules about assumptions and biases that govern the source data. This is similar to our notion of rules that integrate information from semantically wrapped data (section 3) but we apply them at a later stage. A number of existing information integration solutions are being researched and implemented, and are often referred to as ontology alignment solutions. The INRIA alignment API and server[16] is a good example of such an ontology alignment API for expressing and sharing alignments. Our work on SWEDER is currently focused at a far simpler and pragmatic level than existing efforts such as these, but our use of SPARQL construct does enable rich expressivity when it comes to information integration rule construction.

Another interesting angle we investigate with the use of context ontologies is deploying rules to capture the dependencies between properties. This is similar to the work of [1] where the authors elaborate on a naming convention scheme which is based on a loose ontology that represents the notions of kind and superkind. Their aim is to ease data usability by providing a naming scheme that allows for classification of source data. In our work we use properties and super properties found in the context ontologies to aid information integration and grouping.

[16] Available from: http://alignapi.gforge.inria.fr/

5 Conclusion

We presented SWEDER: Semantic Wrapping of Existing Data Sources with Embedded Rules. A pragmatic approach to semantically enable existing sources of data and then utilise multiple semantically enabled sources of that data through the creation of context ontologies to capture the specific rules and any new entities, relationships or attributes arising from the new context. This technique allows us to store rules directly within the ontologies in such a way that they can be easily extracted and executed by common capability within any consuming application, specifically through the use of SPARQL construct clauses.

Details of a simple example application in the domain of organisation and collaboration information were given, with a description of some simple rules that have been written to integrate these separate semantically wrapped data sources into the new context, taking advantages of inherent relationships between the data in those sources. Finally, we described our planned future work in this area which involves, amongst other things, the use of reification techniques to capture the provenance of any data inferred as a result of rule execution, and the desire to user a richer representation format to capture the semantics of our rules in the future.

ACKNOWLEDGEMENTS

This research was sponsored by the US Army Research laboratory and the UK Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, the U.S. Government, the UK Ministry of Defence, or the UK Government. The US and UK Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

REFERENCES

[1] N. Cohen, P. Castro, and A. Misra, 'Descriptive naming of context data providers', in *Proceedings of the CONTEXT'05 Context Representation and Reasoning (CRR'05), Paris, France*, (July 2005).

[2] P. DeLeenheer and A. deMoor, 'Context-driven disambiguation in ontology elicitation', in *Proceedings of the AAAI'05 Workshop on Contexts and Ontologies (AAAI'05/W1), USA*, (July 2005).

[3] Y. Kalfoglou, P. Smart, D. Braines, and N. Shadbolt, 'POAF: Portable Ontology Aligned Fragments', in *Proceedings of the ESWC'08 International Workshop on Ontologies: Reasoning and Modularity (WORM'08), Tenerife, Spain*, (jun 2008).

[4] J. McCarthy, 'Notes on formalizing contexts', in *Proceedings of the 5th National Conference on Artificial Intelligence, Los Altos, CA, USA*, pp. 555–560, (1986).

[5] R. Porzel, H-P. Zorn, B. Loos, and R. Malaka, 'Towards a separation of pragmatic knowledge and contextual information', in *Proceedings of the 2nd International Workshop on Contexts and Ontologies (C&O 2006), Riva del Garda, Italy*, (August 2006).

[6] H. Zeng and R. Fikes, 'Extracting assumptions from incomplete data', in *Proceedings of the CONTEXT'05 Context Representation and Reasoning (CRR'05), Paris, France*, (July 2005).