The First International Semantic Web Conference (ISWC2002),
Sardinia, Italy, June 2002.

Building the Semantic Web on XML

Peter F. Patel-Schneider and Jérôme Siméon

Bell Labs Research, Murray Hill, NJ, U.S.A.
{pfps,simeon}@research.bell-labs.com

Abstract. The semantic discontinuity between World-Wide Web languages, e.g., XML, XML Schema, and XPath, and Semantic Web languages, e.g., RDF, RDFS, and DAML+OIL, forms a serious barrier for the stated goals of the Semantic Web. This discontinuity results from a difference in modeling foundations between XML and logics. We propose to eliminate that discontinuity by creating a common semantic foundation for both the World-Wide Web and the Semantic Web, taking ideas from both. The common foundation results in essentially no change to XML, and only minor changes to RDF. But it allows the Semantic Web to get closer to its goal of describing the semantics of the World Wide Web. Other Semantic Web languages (including RDFS and DAML+OIL) are considerably changed because of this common foundation.

1 Introduction

The Semantic Web [3, 4] is supposed to be an extension of the World-Wide Web where the meaning of data is available to and processable by computers. However, the current recommended Semantic Web languages (RDF [13] and RDFS [9]), as well as other Semantic Web languages such as DAML+OIL [8], are not built on top of the current World-Wide Web. We argue that, in order to describe the semantics of the World-Wide Web, the Semantic Web must be based on XML, the data format of the World-Wide Web.

RDF (the Resource Description Framework) and RDFS (the RDF Schema Specification), as well as other Semantic Web languages, do use the syntax of XML [16, 6]. However, they do not give this syntax the same meaning that it is given in XML. Thus there is a semantic discontinuity at the very bottom of the Semantic Web, interfering with the stated goal of the Semantic Web: If Semantic Web languages do not respect World-Wide Web data, then how can the Semantic Web be an extension of the World-Wide Web at all?

The reason that RDF and RDFS, as well as other Semantic Web languages, do not give the same meaning to XML syntax that XML does is that RDF and XML have different modeling foundations. XML, as evidenced in the XML Information Set [17] and the XQuery Data Model [19] is based on a tree model where edges have no labels and outgoing edges from a node have a total order. This kind of model has its roots in semi-structured data [1] and databases. RDF and RDFS are based on a directed graph model where edges do have labels, but are unordered. This kind of model has its roots in the model theory for standard logics, such as first-order logic, and in models for knowledge representation languages, including frames [12] and description logics [2].

Achieving the stated goals of the Semantic Web as an extension of the World-Wide Web requires the elimination of this discontinuity. This cannot simply be done by just using the XML foundation, as the Semantic Web needs more than just trees. Instead a new foundation must be created, augmenting the XML foundation with the capabilities needed by the Semantic Web. From this merged foundation a new Semantic Web can be created as a true extension of the World-Wide Web, able to employ any data from the World-Wide Web and give it computer-accessible and -manipulable meaning.

This document proposes a foundation for this new Semantic Web. Many aspects of this new Semantic Web are left for future elaboration, especially where they do not affect the broad outlines described herein.

2 Foundations for the World-Wide Web and the Semantic Web

The foundations of the World-Wide Web and the Semantic Web are the meanings of their major languages, XML (taken in the large) and RDF (including RDFS). The meanings of both XML and RDF are currently given in terms of data models, several for XML—including the XML Information Set [17] and the XQuery Data Model [19]—and the model specification for RDF [13].

2.1 The Current XML and RDF Foundations

It is, unfortunately, not so easy to pin down exactly what the information contained in XML documents is. The relevant XML recommendations, the XML Infoset and the XQuery Data Model, retain almost all of the bits of the document itself, including whitespace, comments, etc. This is *not* what is wanted for describing the information contained in a document. Therefore some abstraction of the information retained in these recommendations is needed to get to the real meaning of XML documents.

The basic espoused idea here is to ignore the information in an XML document that is specific to its status as a document and retain only the "data" in it. So, from the XQuery Data Model, one should ignore all processing instructions and comments, whitespaces, as well as the lexical form of typed data. Other aspects of the XQuery Data Model are also handled somewhat differently. For instance, an IDREF would be interpreted in a different way, more like a pointer than an actual child node, resulting in a directed acyclic node rather than a tree.

This results in a simplified, semantics driven, tree based, data model for XML. Some nodes in the tree correspond to XML elements. These nodes have labels that are the element names. Other nodes of the tree correspond to XML attributes. These nodes have labels that are the attribute names. Type information, derived from XML Schema documents, can also be present on these element nodes. Yet other nodes of the tree correspond to XML text nodes or XML attribute values (that are not typed by XML Schema or other means). These nodes have labels that are the text of the node or string-value of the attribute value. Yet other nodes of the tree correspond to typed text nodes or typed XML attribute values. These nodes have labels that are the typed value of the node or the typed value of the attribute. The outgoing edges from a node are equipped with a total order.

The RDF data model on the other hand is specified quite well in the RDF Model and Syntax Specification (RDF [13]), at least for the non-controversial aspects of RDF.

The data model underlying RDF is a directed graph. Nodes of the graph are labeled with identifiers, or literals (i.e., text), or are unlabeled. Edges of the graph are also labeled with identifiers. Some edges of the graph, those labeled with rdf:type, are have a built-in meaning. These edges link nodes to their types (a combination of element and attribute names and XML Schema types in XML parlance).

2.2 A New Foundation

Can these two data models be somehow merged together to form a common foundation for both XML and RDF? Several attempts in this direction have already been made. Melnik [11] created a version of RDF that can handle arbitrary XML, but uses extra information on how to handle the XML constructs that don't fit well into the RDF model. Boley [5] has a data model that can be used for both XML and RDF. However, his approach requires changes to XML to unify it with RDF. In [14], we propose our first attempt at a model theory for XML and RDF that allows to describe the semantics of both in a unified way.

Our approach is to create a data model that requires the minimal amount of change to both XML and RDF, and essentially no change to XML.

The difference between trees and directed graphs is easily reconciled. As directed graphs are a generalization of trees, we allow arbitrary directed graphs.

There are several ways to deal with the labels on edges in RDF graphs. One way would be to allow edges to optionally have labels, which would be very close to the RDF data model. However this would severely change the meaning of RDF documents viewed as XML document. Another way would be to move the XML Data Model node labels to the edges of the graph. This is an attractive alternative, but requries considerable change to both XML and RDF. Instead we use the XML data model directly, not allowing labels on edges. The result is that labeled edges in RDF graphs correspond to two unlabeled edges, with the label ending up on the node in the middle.

XML and RDF take different views of node labels as well. In XML some node labels are element names, which correspond to RDF types, not RDF labels. Other node labels are attribute names, which correspond to RDF edge labels. Yet other node labels are text or typed values, which correspond to RDF literals. RDF identifier labels have no corresponding label in the XML data model.

In our merge, nodes are of one of two kinds. One kind of node corresponds to text or value nodes from XML or literal nodes from RDF. These nodes are given labels that are their text or their typed value, as appropriate. The other kind of node corresponds to element or attribute nodes from XML or non-literal nodes or edge labels from RDF. These nodes are given two optional labels, one of which is the RDF identifier and the other of which contains the element or attribute names from XML or the rdf:type(s) or edge label from RDF. Two of these nodes in a graph cannot have the same RDF identifier.

To handle the edge ordering of XML, our data models partially order the outgoing edges from each node.

A final reconciliation that is needed is the difference in naming conventions between XML and RDF. XML uses qualified names, which are two-part names having a local part, whereas RDF uses URIs plus fragments. We propose to go with qualified names, abandoning direct use of URIs, although we generally treat qualified names as abbreviations for URIs plus fragments.

So we end up with a directed graph data model where edges have no label, text or value nodes have their text or value as their label, other nodes have two optional labels, both of which use Qnames, and there is a partial order on edges.

The data model for a document (XML or RDF) can now easily be determined. XML documents are treated just as before. The graph for an XML document in this data model is an ordered tree. The only change is in the node labels for non-text and non-value nodes, but this is only a cosmetic change. RDF documents are treated differently. In particular, they are treated as if they were XML documents, with two exceptions. First, if an RDF identifier occurs multiple times in a document, only one node is generated. Second, the outgoing edges resulting from an RDF document are unordered. This treatment of RDF documents does ignore some of the esoteric features of RDF, such as reification, and also does not handle some of the shorthand versions of RDF.

3 A New Semantic Web (Data Portion)

So what would data in this new Semantic Web look like?

At first glance, it would not be much different from the current Semantic Web. The content of most RDF data documents would not change,

```
<Organization rdf:ID="BUU">
  <purchase>
    <Item rdf:ID="po7" />
  </friend>
</Organization>
```

would still be a way to say that BUU is an organization that has made a purchase identified as "po7".

However, the role of the various languages would change to some extent.

The role of XML in this new Semantic Web would be quite different, and much stronger. Instead of just being a syntax-carrier, XML would be a major means of providing both syntax and semantics for the Semantic Web.

The meaning of XML documents in this new Semantic Web would be (very close to) their current XML meaning, instead of the different RDF meaning. So, as a Semantic Web document fragment,

```
<Organization>
  <purchase>
    <item />
  </purchase>
</Organization>
```

would have its current XML meaning of a tree fragment with three nodes.

The role of RDF in this new Semantic Web would also be quite different, and much weaker. RDF would no longer be the carrier of all information—that role would be at least partly taken over by XML. The existing RDF data model would not be a part of this new Semantic Web, being replaced by a data model much closer to the XML data model. So the RDF example above would not result in a two-node graph fragment with a labeled edge, plus extra edges for the type information, but instead would result in the XML-style three-node graph fragment. This implies a much more consistent treatment of the semantics XML and RDF. Nonethless, most RDF constructs would have equivalent meanings.

4 Going Beyond Data

So far we have concentrated on the data portion of the Semantic Web. However, the goal of the Semantic Web is to provide machine-accessible and -processable meaning for this data. There are several ways of providing this meaning employed in the World-Wide Web and the Semantic Web, one used by RDF and RDFS and another used by XML Schema.

In the RDF and RDFS method, the meaning for data is provided by adding more nodes to the data model. These extra nodes represent classes, or categories, and connections are made from the data nodes to these class nodes, just like connections are made between data nodes. Some portion of the data model (rdf:type edges in RDF) is given special meaning to account for the special status of these nodes.

In the XML Schema method, the meaning for data is provided by separate means. XML Schema uses schema documents for this purpose. These separate documents are given special meaning and are processed differently from documents that carry data. XML Schema schema documents use XML syntax internally, but the meaning of schema documents is *not* the data model that would result from processing them as XML documents.

The RDF and RDFS method looks promising, and works well for limited formalisms like RDF and RDFS, but ends up causing problems for more-powerful formalisms. For example, a formalism that wants to express disjunctions would have to end up bypassing RDF and RDFS meaning, because that RDF and RDFS treat a collection of data as the conjunction of its parts. There is no mechanism to turn off this treatment the component parts of, for example, a disjunction, as it is only possible to *add* to meanings.

If the Semantic Web is to grow in power it will need to have constructs that do not fit into the RDF and RDFS approach. Therefore we will avoid the RDF and RDFS approach in favour of an approach more like the one used by XML Schema. Our approach will be to make a separation between data and meaning for the data, although we will not solely use the XML Schema separation method.

There are several kinds of meaning that can be supplied for data, and RDF and RDFS differ from XML Schema in this area also.

XML Schema is (only) concerned with providing typing and structuring information for isolated chunks of data. This is perfectly fine for XML which, because of its tree-like data model has only isolated chunks of data, but is not completely adequate in the graph-like world of the Semantic Web. XML Schema also is generally used to restrict the kinds of information that can be specified, which again goes against open-ended nature of the Semantic Web. Note that in addition, XML Schema also provides limited additions to the data model, most notably: typed values and type annotations.

So we propose that XML Schema retain its current role of restricting the form of XML documents and providing typing and structuring information for them. We propose that the meaning portions of RDF and RDFS be taken over by a Semantic Web Ontology language, which we will call SWOL. SWOL would roughly fill the same role as currently filled by DAML+OIL, but would have a much closer relationship to XML and XML Schema. Finally, for the parts that is handled by both XML Schema and RDFS (notably type annotations and containment relationship), we believe there is some need for coordination between XML Schema and the Web Ontology language.

5 A New Semantic Web

So what will this new Semantic Web look like? Again, there would not be much difference from the current World-Wide Web, at least as far as XML and XML Schema are concerned. Documents would still be parsed and validated, much as they are now. The only change here would be that the meaning provided by XML and XML Schema would end up being the Semantic Web meaning of XML documents, instead of being supplanted by the RDF meaning.

The situation is somewhat different from the RDF and RDFS side. RDF is reduced from the main language of the Semantic Web to an extension of XML that really only provides the identifiers (rdf:ID) that tie the XML data model nodes together to make a graph. RDFS is completely gone, being replaced by our as-yet-unspecified Semantic Web Ontology Language, SWOL.

5.1 Semantic Foundations for the New Semantic Web

Unfortunately, the machinery we have introduced so far is not sufficient to provide a semantic foundation for this new Semantic Web. Data models, as above, only work well for providing meaning for simple, inexpressive languages. The main point of data models—that a single data model can be used to capture the meaning of a document—breaks down for more-expressive languages, such as a Semantic Web Ontology Language. Data models even have problems when applied to RDF and RDFS, resulting in the W3C RDF Core producing a different formal foundation for RDF and RDFS [15].

The meaning of documents that contain disjunctive or vague information, such as saying that either John or Jim is the author or a web page, where the exact state of affairs of the world is not known, cannot be captured in a single data model, where everything is fixed and certain. The usual solution to this problem is to have a document correspond to any one of several interpretations, each of which correspond to one of the ways the vague information can be resolved. So the above disjunction would give rise to two classes of interpretations, one class where John is the author of the web page and one where Jim is.

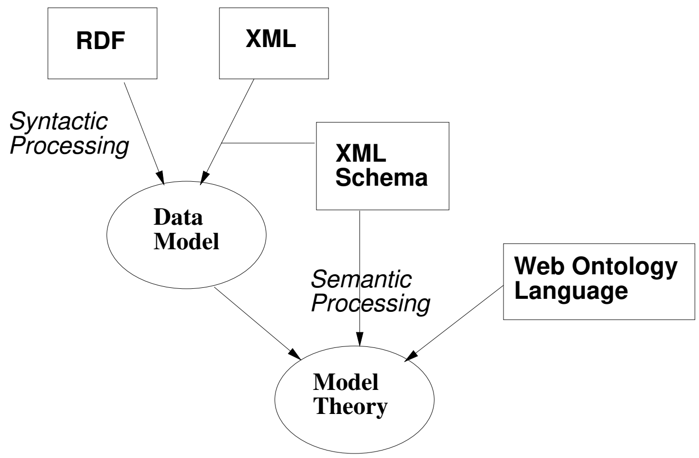

Fig. 1. From Documents to Model Theory

This way of giving meaning to documents (usually referred to as collections of statements) is called model theory. Model theories have been developed for most of the well-known representation formalisms, including propositional logics, first-order logics, and description logics. In data models the meaning of a document is a single data model, which corresponds to the portion of the world being considered. In model theory the meaning of a document is a collection of interpretations. Each of these interpretations corresponds to one of possible ways the world could be, given the information under consideration.

The formal definition of our models are given below.

Resources, Names, Data Values, and Datatypes This model theory assumes a universe of resources and data values. QNames are used as identifiers for resources.

Definition 1. *We call L the lexical space of strings, and U the value space of QNames, i.e., pairs of URIs and local parts. We call DT the subset of U corresponding to XML Schema primitive datatypes, and DV the union of the value spaces of the XML Schema primitive datatypes. In RDF elements of DV are generally called literals. The function $DTC : DT \to \mathcal{P}(DV)$, (where $\mathcal{P}$ is the powerset operator) maps XML Schema primitive datatypes to their value spaces and $DTS : DT \to (L \to DV)$, maps XML Schema primitive datatypes to their lexical to value maps. We define the union of the datatype mappings $XTS : L \to \mathcal{P}(DV)$, where $v \in XTS(l)$ iff $v = DTS(dt)(l)$ for some XML Schema datatype dt.*

Interpretations Interpretations are the essential component of our model theory. An interpretation corresponds to one possible way the world can be, hence encoding a certain meaning for the information manipulated by an application. Interpretations give information about resources and related resources through relationships and semantic constraints. We define a notion of interpretation that is suitable for both XML and RDF documents, through the XQuery data model.

Definition 2. *An* interpretation I *is a six-tuple,* $\langle R, E, EXT, CEXT, O, S\rangle$*, where:*

R is a set of resources,
E is a set of relationships,
$EXT : E \to R \times (R \cup DV)$ maps relationships to the resources they relate,
$CEXT : U \to \mathcal{P}(R \cup DV)$ maps classes to their extensions,
$O : R \to \mathcal{P}(E \times E)$ provides a local order on the relationships, and
$S : U \to R$ is a partial map from QNames to resources.

An interpretation must also satisfy the following conditions:

1. *$O(r)$ is a strict partial order.*
2. *If $\langle x, y\rangle \in O(r)$ then $EXT(x)$ and $EXT(y)$ have r as their first element.*
3. *If $d \in DT$ then $CEXT(S(d)) = DTC(d)$, provided that S is defined on d.*

An interpretation can be thought of as a multigraph with an ordering on the edges. Resources (R) form the nodes of the graph. Edges of the graph are formed from relationships (E) and EXT. For instance, a relationship $e1 \in E$ with $EXT(e1) = \langle John, p\rangle$ indicates that the resource $John$ is related to the resource p, and a relationship $e2 \in E$ with $EXT(e2) = \langle p, po7\rangle$ indicates that the resource p is related to the resource $po7$. We remark that there is no distinction at this point between $John$, usually thought of as an instance of a class, and p, which is usually thought of a property.

S provides a mapping between syntax (QNames) and their denotation (resources). S gives a means to identify these entities using QNames. There is no requirement that all resources have corresponding QNames, nor is there a requirement that QNames are all mapped to different resources.

$CEXT$ provides typing information for resources. For instance, if BUU is in $CEXT(Organization)$ then the resource BUU is of type $Organization$. Similarly, if p is in $CEXT(purchase)$ then the resource p is of type $purchase$. Loosely speaking, in RDF terms $CEXT$ serves for both property and class extensions. Or, considered another way, a property is presented as a type whose values and related tuples identify arcs in the traditional RDF graph structure.

Finally, O provides ordering information between the relationships that are related to a common resource. This information is not usually part of RDF model theories [15], but it is important to capture document order in XML documents.

5.2 SWOL

We now, finally, give some details on SWOL, our proposed web ontology language. SWOL is actually very close to a description logic [2], the only difference being a few minor changes to move away from edge-labelling.

The syntax of SWOL is unimportant, at least for now. We will introduce SWOL by means of an example that shows most of the relevant features of SWOL.

```
<swol:class name="Organization" defined="no">
  <swol:exists>
    <swol:class name="name"/>
    <swol:class name="xsd:String"/>
  </swol:exists>
  <swol:all>
    <swol:class name="purchase"/>
    <swol:class name="PurchaseOrderType"/>
  </swol:all>
</swol:class>
```

This SWOL document fragment contains the definition of a class, named `Organization`. Elements of this class, organizations, have a name, in the form of a string. Elements of the class also can have purchases, each of which must belong to the PurchaseOrderType.

For now, we can think of a SWOL document as a collection of several definitions of this sort, each with a name. There are actually many possibilities for SWOL, just as there are many description logics, varying in expressive power from frame-like formalisms up to very powerful ontology-definition formalisms. The particular expressive power of the W3C-recommended SWOL is under consideration by the W3C Web Ontology Working Group (although they are still working from an RDF base).

5.3 Models of XML and SWOL documents

Now that we have defined our notion of interpretation, we need to explain how documents correspond to sets of interpretations. Intuitively, each node in the data model graph resulting from document (either validated or unvalidated) is mapped to a resource in the interpretation, and EXT relationships are built according to the edges of the data model graph.

Definition 3. *An interpretation $I = \langle R, E, EXT, CEXT, O, S \rangle$ is a* model *for a data model graph G if S is defined on all names in G, and there is a mapping $M : N \to R \cup DV$ that maps the nodes of G into either resources or data values. Further, the interpretation and mapping have to satisfy the following conditions.*

1. *If $n \in N$ is the top-level element of an XML document with URL u, then $M(n) = S(u)$.*
2. *For each $n \in N$ a untyped text node, $M(n) \in DV$ and $M(n) \in XTS(string-value(n))$.*
3. *For each $n \in N$ a typed value node with typed value v, $M(n) = v$.*
4. *For each other $n \in N$*
   (a) *$M(n) \in R$.*
   (b) *If n has an identifier label id, then $M(n) = S(id)$.*
   (c) *For each name label of n, c, $\langle M(n), S(c) \rangle \in CEXT$.*

Definition 4. *An interpretation $I = \langle R, E, EXT, CEXT, O, S \rangle$ is a* model *for a SWOL ontology document O if S is defined on all names in O. Further, the interpretation has to satisfy the following conditions for each definition in O.*

1. *For each definition in O of the form*
   `<swol:class name="n" defined="yes"> d1 ... dn </swol:class>`
   $CEXT(S(n)) = I(d1) \sqcap \ldots \sqcap I(dn)$.
2. *For each definition in O of the form*
   `<swol:class name="n" defined="no"> d1 ... dn </swol:class>`
   $CEXT(S(n)) \leq I(d1) \sqcap \ldots \sqcap I(dn)$.

where $I(d)$ (d is called a description) is defined as

1. *If d is* `<swol:class name="n"/>`
   then $I(d) = CEXT(S(n))$.
2. *If d is* `<swol:intersect> d1 ...dn </swol:intersect>`
   then $I(d) = I(d1) \cap \ldots \cap I(dn)$.
3. *If d is* `<swol:union> d1 ...dn </swol:union>`
   then $I(d) = I(d1) \cup \ldots \cup I(dn)$.
4. *If d is* `<swol:complement> d1 </swol:complement>`
   then $I(d) = R - I(d1)$.
5. *If d is* `<swol:all> d1 d2 </swol:all>`
   then $I(d) = \{r \in R : \forall \langle r, s \rangle \in EXT, s \in I(d1) \rightarrow s \in I(d2)\}$.
6. *If d is* `<swol:exists> d1 ...dn </swol:exists>`
   then $I(d) = \{r \in R : \exists \langle r, s \rangle \in EXT, s \in I(di), 1 \leq i \leq n\}$.

More-powerful versions of SWOL would have more possibilities for definitions and descriptions, but would have still have their meaning defined in this way.

5.4 Dealing with Models

Documents no longer correspond to single data models, but instead correspond to a set of interprepretations, the models of the document. Thus it is no longer possible to just get the information contained in a document by performing operations on the data model. Instead, we have to find out this information indirectly.

The usual method for finding out this information in model theory is via an entailment relationship. Basically, information (in the form of a query document) is present in a document if all the models of the document are also models of the query document. What we are saying here is that all the possible ways the world can be and still be compatible with the document (the models of the document) are also compatible with the query. Another way of saying this is that entailment captures what information is implicit in a document.

Definition 5. *Given a document, D, or collection of documents, see below, and another document, E, D entails E if all models of D are also models of E.*

Entailment is the simplest semantic relationship. It is also possible to define more-powerful semantic relationships, including variants of querying.

5.5 Multiple Sources of Information

Our notions of models and entailment above are not restricted to single documents, or even documents all of one kind. In fact, most of the interesting information sources will consist of several documents

- one or more XML (or RDF) documents containing base facts,
- zero or more XML Schema documents, brought in by the XML documents, and
- zero or more SWOL Ontology documents, brought in by explicit directives.

The first two kinds of documents are processed into data models, which are then given meaning in the semantics, whereas the third is given meaning directly.

So, an interpretation is a model of a set of data model graphs and a set of SWOL Ontology documents if is a model of each each of the data model graphs and each of the ontology documents separately.

5.6 Giving Direct Meaning to XML Schema Documents

So far the only meaning we have given to XML Schema Documents is their effect on the creation of XQuery data models. However, it is possible (but not necessary) to also have XML Schema Documents have a direct relationship to the model theory.

Whether one wants to do this depends on one's view of the status of XML Schema. If XML Schema definitions only constrain the form of XML documents then there should not be a direct connection between XML Schema documents and the model theory. In this view the definitions in an XML Schema document are *local*, that is, their import should only be felt by XML documents that directly reference the XML Schema document. Here two different XML documents could use the same element names but give them different meaning, by using different XML Schema documents. So, for example, one XML document could use one XML Schema document for purchases and another XML document could use a different XML Schema document, with a different definition of purchases, even though they both used the same (qualified) element names.

On the other hand, one might want to require that all purchases have similar meaning, although maybe not similar form. In this view the XML Schema document that defines purchase would not just affect one (or more) XML documents, but would have a direct and global impact on the model theory.

XML Schema is a (very) large specification, so the details of interpretations can model XML Schema documents are beyond the scope of this paper but the general outline is clear.

Definition 6. *An interpretation* $I = \langle R, E, EXT, CEXT, O, S\rangle$ *is a* model *for a XML Schema ontology document* O *if* S *is defined on all names in* O*. Further, the interpretation has to satisfy the following conditions for each "definition" in* O*.*

1. *For each global complex type, element, or attribute definition in* O *with name* n, $CEXT(S(n))$ *contains only those resources that have the pieces in the definition, in the correct order, and with the correct characteristics, but also, possibily, other pieces.*

2. ...

Not all the components of an XML Schema document have direct model-theoretic implications. In particular, default information does not give rise to any conditions, although, of course it *does* have model-theoretic effects through its effects on the data model.

In this way XML Schema documents can be added to the inputs of SWOL and end up with very similar status to SWOL ontology documents. XML Schema documents and SWOL ontology documents can even refer to definitions from the other kind of document and everything still works well.

6 Example

A simple example that shows how all this fits together, and gives some hint as to the power of the scheme, can be constructed on top of the purchase order example in the XML Schema primer [18] (http://www.w3.org/TR/xmlschema-0/#po.xml).

XML Schema is designed to constrain and give typing information for single XML documents, but if we want to describe collections of XML documents, particularly if these documents have differing formats, then something more than XML Schema is needed.

We assume the existence of a collection of different purchaseOrders and PurchaseOrderTypes each defined in a different XML Schema document, with different URLs. We will assume that each of these documents have a namespace, pos-i. Without loss of generality, we will assume that the different XML Schema documents use the same internal name for their top-level components.

We can use SWOL to define the Organization class, containing resources that have purchases that belong to the PurchaseOrderType.

```
<swol:class name="Organization" defined="no">
  <swol:all>
    <swol:class name="purchase">
    <swol:class name="PurchaseOrderType">
  </swol:all>
  ...
</swol:class>
```

This PurchaseOrderType is then defined as a generalization of the various PurchaseOrderTypes via

```
<swol:class name="pos-i:PurchaseOrderType" defined="no">
  <swol:class name="PurchaseOrderType" />
</swol:class>
```

We can then create a document that ties together various purchase orders, again, each in its own document with its own name, here given as po-i.

```
<Organization rdf:ID="foo">
  <purchase rdf:ID="po-1:">
  <purchase rdf:ID="po-2:">
  ...
</Organization>
```

However, all we have so far is a collection of purchase orders with no combined way of accessing the information in them, because they each have different elements names (because of their differing namespaces). To unify these elements, we have to provide a generalization of the different element names, as in

```
<swol:class name="pos-i:shipTo" defined="no">
  <swol:class name="shipTo" />
</swol:class>
<swol:class name="pos-i:items" defined="no">
  <swol:class name="items" />
</swol:class>
...
```

Now the various fields of the different PurchaseOrderTypes are considered to be sub-categories of the combined PurchaseOrderType we have created.

So far, we have not done much more than could have been done with RDF Schema, if RDF Schema was modified to deal with XML data and XML Schema types. However, we *can* go further. For example, we can say that our PurchaseOrderType can only be one of the other PurchaseOrderTypes, and nothing else, via:

```
<swol:class name="PurchaseOrderType" defined="yes">
  <swol:union>
    <swol:class name="pos-1:PurchaseOrderType" />
    ...
    <swol:class name="pos-n:PurchaseOrderType" />
  </swol:union>
</swol:class>
```

Using this, and other, facilities from SWOL, we can take information from disparate XML documents, using disparate XML Schema types, and access it in a uniform manner, resulting in a Semantic Web version of the World-Wide Web.

There are, of course, some things that we cannot do with SWOL, as SWOL is only a limited ontology language. Organizations of XML documents that require arbitrary inference will have to wait for the next level of this vision of the Semantic Web.

7 Conclusion

So what have we achieved?

We have created a semantic foundation for the Semantic Web that unifies it with the semantic foundation for the World-Wide Web. This semantic foundation takes the semantic foundation of XML, node-labelled ordered trees, and adds in semantic notions from RDF, including node identifiers and graphs. We then moved from data models, where a document corresponds to a single data structure, to model theory, where a document singles out a collection of interpretations, so as to allow for disjuntive or vague information, as needed in ontologies. In the process we eliminated some of the semantic notions from RDF, like edge labels, to achieve a better relationship between the XML expression of RDF and XML itself.

This model theory accepts several information sources:

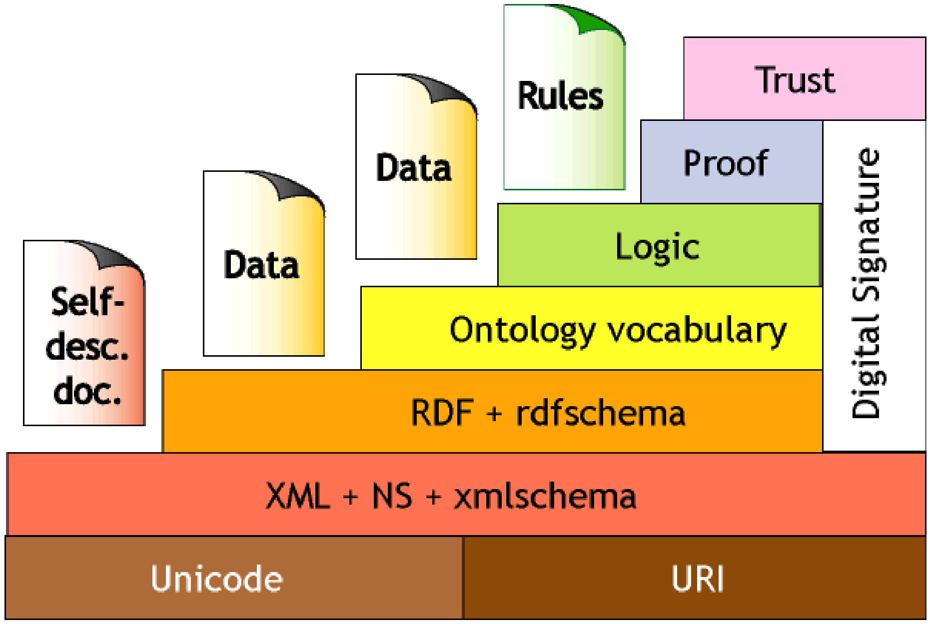

Fig. 2. Semantic Web Layers (figure by Tim Berners-Lee, http://www.w3c.org/2000/Talks/1206-xml2k-tbl/slide10-0.html)

1. XML documents, either validated or unvalidated;
2. XML Schema documents, either just as validations on XML documents or as input sources in their own right;
3. RDF documents containing RDF base facts; and
4. SWOL documents containing ontology definitions.

The details of SWOL and the details of how models are defined for XML Schema are not complete here. There are also several issues that remain to be finalized, in particular how to reconcile the URI view of names espoused by RDF against the QName view espoused by XML.

There are also several other ways to build the model theory. For example, it would be possible to go with the RDF view of models as having labels on edges. This would require changes to XML, but XML documents are generally compatible with this way of viewing the world.

The result of the new semantic foundation is a new vision of the Semantic Web as a natural extension of the World-Wide Web. In this new Semantic Web, XML is no longer just "the universal format for structured documents and data on the Web" (http://www.w3.org/XML/), but instead is the major source of semantic information for the Semantic Web. XML Schema documents still play their current role of constraining and typing XML documents, but, since XML plays a larger role in this vision of the Semantic Web, even this use of XML Schema has more utility. XML Schema documents can also be used as global definitions of types, a new role for XML Schema. Ontology information, that cannot be represented by XML Schema, is carried by a new ontology language SWOL.

With this view of the Semantic Web, the bottom levels of the Semantic Web tower (Figure 2) finally fit together, with no inappropriate semantic disconnects. Further, with the division of documents into different categories (XML, XML Schema, SWOL), no one level claims the entire possible syntax, as was true for RDF, and extensions can be easily given their own syntax and their own place in the Semantic Web.

References

1. Serge Abiteboul, Peter Buneman, and Dan Suciu. *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann, 1999.
2. Franz Baader, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, implementation, and applications*. Kluwer, to appear.
3. Tim Berners-Lee. *Weaving the Web*. Harper, San Francisco, 1999.
4. Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, May 2001.
5. Harold Boley. A web data model unifying XML and RDF. http://www.dfki.uni-kl.de/∼boley/xmlrdf.html, September 2001.
6. Jon Bosak and Tim Bray. XML and the second-generation web. *Scientific American*, May 1999.
7. Ronald J. Brachman and Hector J. Levesque, editors. *Readings in Knowledge Representation*. Morgan Kaufmann Publishers, San Francisco, California, 1985.
8. DAML+OIL language. http://www.daml.org/language/, 2001.
9. Resource description framework (RDF) schema specification 1.0. W3C Candidate Recommendation, 27 March 2000, http://www.w3.org/TR/rdf-schema, March 2000.
10. John Haugeland, editor. *Mind Design*. Bradford Books, Montgomery, Vermont, 1981.
11. Sergey Melnik. Bridging the gap between RDF and XML, December 1999. http://www-db.stanford.edu/∼melnik/rdf/fusion.html.
12. Marvin Minsky. A framework for representing knowledge. In Patrick Henry Winston, editor, *The Psychology of Computer Vision*, pages 211–277. McGraw-Hill, New York, 1975. An abridged version published in [10] and also published in [7].
13. Resource description framework (RDF): Model and syntax specification. W3C Recommendation, 22 February 1999, http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/, February 1999.
14. Peter F. Patel-Schneider and Jérôme Siméon. The yin/yang web: Xml syntax and rdf semantics. In *Eleventh International World Wide Web Conference*, May 2002.
15. RDF model theory. http://www.w3.org/TR/rdf-mt/, 2002.
16. Extensible markup language (XML) 1.0 (second edition). W3C Recommendation, http://www.w3.org/TR/REC-xml.
17. XML information set. http://www.w3.org/TR/xml-infoset/, October 2001.
18. XML Schema part 0: Primer. W3C Recommendation, 2 May 2001, http://www.w3.org/TR /xmlschema-0/.
19. XQuery 1.0 and XPath 2.0 data model. http://www.w3.org/TR/query-datamodel/, December 2001.