The Ties that Bind
XML, the Semantic Web, and Web Services

Harry Halpin

School of Informatics
Institution for Communicating and Collaborative Systems
University of Edinburgh
2 Buccleuch Place, EH8 9LW UK
h.halpin@ed.ac.uk

Abstract. The future success of service-oriented computing relies on a number of crucial connections being made between three distinct Web development initiatives: XML, the Semantic Web, and Web Services. However, it is unclear how they relate to each other. A model that views XML as a serialisation format for data, with a combination of XML Schema and the Semantic Web for typing, and Web Services as functions, provides a coherent vision for the future of the Web. If co-ordinated, these three ties allow the Web to transform from a universal information space to a universal computation space.

1 Introduction: Three Magic Pieces

Service-oriented computing is in need of clarification on how XML, XML Schemas, the Semantic Web, and Web Services fit together. At first glance, these developments, while often relying on each other, do not currently present a unified vision of how the Web should develop. The Web was primarily the result of a unified vision: a universal information space[1]. Yet these new initiatives seem to be competitors: XML vs. RDF for the transfer of data and Web Services vs. the Semantic Web for sharing data. However, these initiatives are upon closer inspection complementary and together they provide the three magic pieces that can make service-oriented computing possible.

2 XML as Serialisation Syntax

In the words of Don Box, "XML has replaced Java, Design Patterns, and Object Technology as the industry's solution to world hunger," yet Chris Maden took the stance that "XML is like violence. If it doesn't solve your problem, you're not using enough of it." There is confusion as regards to what XML actually is and what it does. In essence, XML is a mixture of ordered constraints and unordered constraints, allowing structured data to be mixed with unstructured data and nested to an arbitrarily depth[2]. It is a syntax for a model, semi-structured data, noted by the database

community before the XML syntax was developed[3]. Since such tree-structures can be used as a data format for almost anything, XML can be considered "the ASCII of the 21st century"[4]. XML is just a serialisation format for storing and reading arbitrarily complex semi-structured data. Since it allows custom domain-specific vocabularies to be made, it makes sense to use XML as an exchange and integration language for services, moving from application-specific syntax to a self-describing syntax more appropriate for exchange.

3 The Intentional Equivalence Problem

Just because data shares a common syntax does not mean merging it will be trivial: data integration is the *hard* problem of the Web. To show that two pieces of data are equivalent, you have to have a common semantics for them, and XML only provides a syntax. As argued for by B.C. Smith, we will distinguish between informal *world semantics* and formal *process semantics*[5]. World semantics rely on the philosophically mysterious phenomena of reference: given a piece of data, what is it about? For example, a XML document about my Amazon.com purchase may contain my name and maybe even address. One would assume it is about myself, not the string "Harry Halpin." However, if I move to a new address, or if someone has the same name as me, merging data while preserving the world semantics becomes difficult. Finding out if two pieces of data are about or refer to the same thing is a task that computers are notoriously incapable of doing automatically.

For example, imagine if we got an order in XML to our database from a "Robert Smith" living at "8 Oak Avenue," and in our database of past customers we have a "Robert Smith" living at "123 Maple Street" and an Alice Smith living at "8 Oak Avenue." Should we assume both Robert Smiths are the same person? While there is scant evidence for this in the database, in the world outside the database he recently married Alice Smith and they both moved to 8 Oak Avenue. Only Robert Smith has not ordered from our store since he moved, while Alice has. However, for us to identify these two orders as originated from the same person, we need a complex chain of inference that is currently not possible using only XML.

In contrast to world semantics, process semantics maps data to a precise mathematical model, as done in operational semantics and denotational semantics. XML has an abstract data model: the XML Infoset[6]. The Infoset is not about syntactic open and closed brackets, but it is about the abstract data model they define, which could just as easily be implemented on top of a binary format. The Infoset defines characteristics of XML such as elements, attributes, namespaces, and normalised values. The Infoset maps to the operational semantics as given by the XQuery Data Model[7]. When using XML, we are really transforming and exchanging XML Infosets.

4 Schemas for Syntax-Driven Typing

The XML Infoset does not provide any mechanism for typing, the most common of needs for data transfer in service-oriented computing. An entire XML Infoset, a simple component such as an element or attribute, and a complex component built from simple components can all have a type. All of these levels are dealt with by W3C XML Schema, with the "Part 1: Structures" dealing with a way to describe whole or complex parts of Infosets[8], and "Part 2: Data Types" dealing with the types of element and attribute values[9]. "Part 1: Structures" defines constraints on the syntactic structure of an XML document, and so can type a whole XML Infoset. "Part 2: Datatypes" defines the set of lexical representations of attribute and element content that are valid for a given type. These bindings are both extensible and based on regular expression types such as sequence, choice, repetition, and option. Elegant alternative well-understood formal models have been proposed for regular expression typing over XML such as XDuce[10]. A schema-validated XML document results in the Post-Schema Validation Infoset, which is at its essence a typed Infoset[11]. Unlike many tree grammar-based schemas, XML Schema provides deterministic data typing using a finite-state automaton[12]. In combination with a static type inference system such as the one used in XQuery, schemas can provide a full type system for XML.

5 The Semantic Web as Ontological Types

There is a crucial limitation purely schema-based typing: it only currently provides types such as "strings" and "integers," but it does not operate on the level of knowledge, providing types such as "Robert Smith", "address", and "city." The former is an encoding, such that "Robert Smith" is a string and "52" is an integer. However, in terms of knowledge Robert Smith is also, across all his appearances in various Infosets, a "person" who has a "name" known as "Robert Smith" and an "age" of "52." While XML Schema provides the *syntactic* typing (since it transforms the syntax of XML to a typed Infoset and deals with syntactic constraints), only with considerable human interpretation does it provide the second *ontological* typing. This typing is "ontological" since it is about what is out there in the world beyond the particular Infoset that contains the data, and models that world using logic. XML Schema provides no model-theoretic and machine-readable semantics for ontological statements. Luckily, providing a machine-readable "web of meaning" is precisely the mission of the Semantic Web[13].

If service-oriented computing is going to be used to merge data, we need to reach the level of "world semantics." This requires a much more rich notion of typing: one that can provide rich representations of human knowledge and bind that knowledge to machine-readable data. OWL (Web Ontology Language) provides such a knowledge representation language which can have as its "process semantics" a description logic that

is both decidable and tractable[14]. In terms of "world semantics," OWL ontologies used as knowledge representations can be considered as a second kind of typing: ontological typing (or "Semantic Web" typing to be more precise). Yet OWL ontologies are unlike traditional types since description logics are based on the open-world assumption. This is *crucial*, as all the types of data cannot be known in advance in a loosely-coupled service-oriented framework. In fact, one purpose of service-oriented computing could be to accumulate knowledge through inspecting the ontological types. Although traditionally in typing one defines both the necessary and sufficient conditions for a type, with ontological typing one can have necessary but not sufficient conditions for some data being a certain type, i.e. partial knowledge of the type. This can in turn be used by a reasoner with rules to merge heterogeneous data types, and these tools can also help provide humans with the necessary information to determine if the two data are actually about the same thing. XML Schema can use schema annotations to easily bind the data in a Post-Schema Validation Infoset directly to OWL[15]. Instead of having to write OWL in XML and convert documents between OWL and XML, one can simply semantically model the data in OWL, bind the OWL in the XML Schema, and read and write XML.

6 Web Services as Functions

While the Semantic Web is a *specification language*, people ultimately want *to do things* with data, and so need Web-accessible programs. Web Services began as a what was basically a cheap hack to get data across firewalls. Yet the truth is more subtle. Web Services, are in essence *functions* on the Web. As we can use XML Schema and the Semantic Web to provide typing information, Web Services are typed functions. Currently, service-oriented computing tasks are often composed using workflows. It is a well-known problem that workflows do not support constructs like iteration. One needs a full-scale and service-oriented functional programming language. Currently, functional programming languages such as Haskell have been used successfully to compose complex Web Services[16], and functional languages that have native support for XML like Scala are also providing Web Service support[17]. If such a language could be combined with the types provided by OWL, then one could imagine Web-scale type-inference being used to compose complex services in a natural and intuitive manner while building knowledge-bases that in turn could trigger further service invocation.

While XML gives us regular expression types for service-oriented computing. There is another tradition in type theory where functions themselves can be regarded as types: *higher order functions*, as in ML or Haskell. Higher order functions and algebraic types would allow for a type to be a tuple, a sum, a function, and so on. This has even been implemented using a type theory with native XML support by CDuce[18]. Expanding our definition of type beyond regular types would allow for polymorphism and flexible record types, letting Web Services be passed to Web Services as parameters of functions.

To complete our example, we can resolve the issue of intentional equivalence if we assume the relevant data we need is accessible on the Web. We receive the order through a Web Service, and parse it with a schema to receive an typed Post-Schema Validation Infoset augmented with Semantic Web typing. Our system adds an instance of the class "Person" named "Robert Smith" to the knowledge-base. A "Person" can have an "Address," and also have a "married" relationship with another "Person." In attempting to merge this "Person" to our existing database, we notice that his current "Address" is the same as the "Address" for a previous customer, "Alice Smith." We also note they live at the same "Address," and so we want to know if they are "married." Since the married relationship is unknown, a rule is triggered that launches another Web Service that returns whether or not a "marriage" relationship holds, by checking various marriage registries automatically. Indeed, Robert and Alice are married. However, there is also another "Robert Smith" in the database with a different address. The dates of the two orders are compared, and the first order took place considerably earlier than the most recent one. A rule holds that a customer may have only one current "Address," so when the order is made an enquiry is given to the user asking them if they have previously ordered from the store from the suspected previous address. When the user answers yes, this information is in turn fed back into the knowledge base for future reference. With this more complete picture of Robert Smith, whatever Web Services we can provide him are more likely to be to his liking, and perhaps we can use our knowledge to remind him not to forget his marriage anniversary next year.

7 The Future

Two tasks remain to be done: merging XML Schema data types with Semantic Web ontologies in order to build open-world types for XML, and also to build a coherent ontologically-aware framework for higher-order typing for Web Services. The simplest way to do the first task is to let both syntactic typing and ontological typing remain independent. The Infoset is typed *twice*, once using XML Schema and then using the Semantic Web. Second, in the context of Web Services in particular, XML Schema and the Post-Schema Validation Infoset needs to be augmented to allow it to recursively contain possibly unknown types, higher-order Web Services, and other algebraic type features. It should also be easy to extract ontologies from the Infoset. Then a functional programming language capable of static type inference, both using the Semantic Web and XML Schema, would make the creation of service-oriented computing elegant, safer, and more powerful. With XML for serialisation of data, schemas and ontologies for typing, and Web Services as functions, the ties that bind the standards for service-oriented computing together are clear. In fact, these three aspects: data, types, and functions are the crucial components of any computational system. With the advent of XML as a *universal syntax*, with XML Schemas and the Semantic Web providing *universal types*, and Web Services providing *universal functions*,

the Web transforms from a universal information space to a universal computation space.

References

1. Berners-Lee, T.: Weaving the Web. Texere Publishing, London (2000)
2. Bray, T., Paoli, J., Sperberg-McQueen, C.: Extensible Markup Language (XML) 1.0 (Third Editition) (2004) http://www.w3.org/TR/REC-xml/.
3. Buneman, P.: Semistructured data. In: Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of Database Systems. (1997)
4. Thompson, H.S.: Putting XML to Work (2001)
5. Smith, B.C.: The Correspondence Continuum. Technical Report SciDAC-SPA-TN-2003-0, Center for the Study of Language and Information (1987)
6. Cowan, J., Tobin, R.: XML Information Set (Second Edition) (2004) http://www.w3.org/TR/xml-infoset.
7. Draper, D., Fankhauser, P., Fernandez, M., Malhotra, A., Rose, K., Rys, M., Simeon, J., Wadler, P.: XQuery 1.0 and XPath 2.0 formal semantics (2004) http://www.w3.org/TR/xquery-semantics/.
8. Thompson, H.S., Beech, D., Maloney, M., Mendelsohn, N.: XML Schema Part 1: Structures Second Edition (2004) http://www.w3.org/TR/xmlschema-1/.
9. Biron, P., Malhotra, A.: XML Schema Part 2: Datatypes Second Edition (2004) http://www.w3.org/TR/xmlschema-2/.
10. Hosoya, H., Pierce, B.C.: XDuce: A typed XML processing language. ACM Transactions on Internet Technology **3** (2003)
11. Simeon, J., Wadler, P.: The Essence of XML. In: Proceedngs of Symposium on Principles of Programming Languages. (2003)
12. Thompson, H.S., Tobin, R.: Using finite state automata to implement W3C XML Schema content model validation and restriction checking. In: Proceedings of the XML Europe 2003. (2003)
13. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American (2001)
14. Patel-Schneider, P., Hayes, P., Horrocks, I., van Harmelen, F.: OWL Web Ontology Language Abstract Syntax and Semantics (2004) http://www.w3.org/TR/2004/REC-owl-semantics-20040210.
15. Krupnikov, A., Thompson, H.S.: Data Binding using W3C XML Schema Annotations. In: Proceedings of the XML Conference. (2003)
16. Ludascher, B., Altinas, I.: On providing declarative design and programming constructs for scientific workflows based on process networks. Technical Report SciDAC-SPA-TN-2003-0, Unviersitate Mannheim (2003)
17. Odersky, M.: An overview of the scala programming language. Technical Report IC/2004/64, EPFL Lausanne, Switzerland (2004)
18. Benzaken, V., Castagna, G., Frisch, A.: CDuce: An XML-centric general-purpose language. In: Proceedings of the ACM International Conference on Functional Programming. (2003)