Compositionality, elaboration and XML document semantics

Henry S. Thompson

28 November 2007

1 Acknowledgements

My interest in this topic traces back to a discussion with Richard Tobin and Tim Berners-Lee. Berners-Lee's thoughts on the questions considered here can be found in [2]. My thanks to both of them—needless-to-say they are not responsible for where I've gone since our conversation.

2 Compositional semantics

The traditional definition of compositionality comes from linguistic theory, and its simplest expression is just "the meaning of the whole is a function of the meaning of the parts". This might seem to be vacuous, but once we a) apply it recursively and b) note that there is an implicit 'only' after 'function', then it has some teeth. It is often understood to apply in situations where there is a distinction between syntax and semantics and where syntactic forms are trees with labelled nodes. In this case we can say that a compositional semantics is one which can be expressed as follows, where we write $I(n)$ for the (sc. semantic) interpretation of a tree node n, $n.l$ for the label of node n and $n.c_i$ for the i^{th} child of node n:

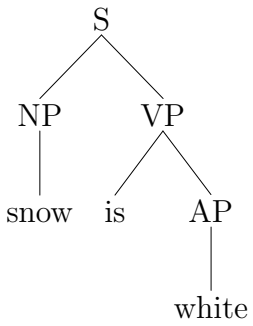
$$I(n) \equiv M(n.l)(I(n.c_1)...I(n.c_m)) \tag{1}$$

This definition embodies a particularly simple form of compositionality, because by writing $M(n.l)$ we imply that the *meaning* of the node depends solely on its label. Richer forms exist, particularly the so-called *rule-to-rule* approach, where the meaning of a node depends on the grammar rule which allows it. But in all cases, the meaning of a node is a function from the interpretations of the node's children to the interpretation of the node itself.

Consider the following context-free phrase-structure grammar for a (famous) trivial fragment of English:

$$\begin{aligned} \text{S} &\rightarrow \text{NP VP} \\ \text{NP} &\rightarrow \text{snow} \\ \text{NP} &\rightarrow \text{tar} \\ \text{VP} &\rightarrow \text{is AP} \\ \text{AP} &\rightarrow \text{white} \\ \text{AP} &\rightarrow \text{black} \end{aligned}$$

For the sentence "snow is white", this grammar yields the following analysis:

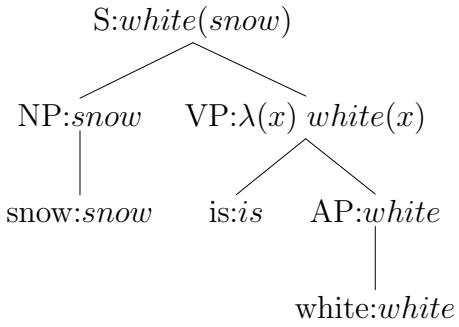


If in addition we assign the following meanings to the node labels from the above grammar:

$$\begin{aligned} M(\text{S}) &\equiv \lambda(s,p)\ p(s) \\ M(\text{NP}) &\equiv \lambda(n)\ n \\ M(\text{VP}) &\equiv \lambda(v,a)\ \lambda(x)\ a(x) \\ M(\text{AP}) &\equiv \lambda(a)\ a \\ M(\text{snow}) &\equiv \mathit{snow} \end{aligned}$$

and similarly for all the other leaf labels (that is, tar, black etc.)

Then we can construct the interpretation $\mathit{white}(\mathit{snow})$ for "snow is white" by recursive application of the composition rule (1), illustrated by annotating each node in the parse tree with its interpretation:



3 Quotation

The recursive operation of the interpretation function sometimes needs to be forestalled. Consider a sentence such as " 'La plume de ma tante' is French" or a Lisp S-expression such as `(cons 3 (quote (2 1)))`.

Although this problem is not commonly addressed in the linguistic theory literature, it can be handled in the example framework developed above by providing for the recursive application of the interpretation function to be controlled by the node label:

$$I(n) \equiv M(n.l)(I_{n.l,1}(n.c_1)...I_{n.l,m}(n.c_m)) \quad (2)$$

$$I_{n.l,i} \equiv \begin{cases} \iota \\ I \end{cases} \quad (3)$$

where by definition (3) we mean that the interpretation function $I_{n.l,i}$, can be either I (per previous line) or ι (a function which maps a node to its label, i.e. $\iota \equiv \lambda(n)\ n.l$), depending both on the label of the node $n.l$ and the index of the child i.

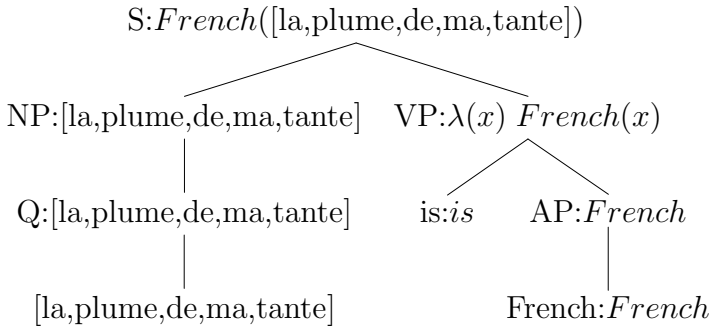
So for example if we add the following to our grammar:

$$\begin{aligned} \text{NP} &\rightarrow \text{Q} \\ \text{Q} &\rightarrow \text{[la,plume,de,ma,tante]} \\ \text{Q} &\rightarrow \text{[Wo,die,Zitronen,bluehn]} \\ \text{AP} &\rightarrow \text{French} \\ \text{AP} &\rightarrow \text{German} \end{aligned}$$

and the following definitions:

$$M(\text{Q}) \equiv lambda(x)x \tag{4}$$
$$I_{\text{Q},1} \equiv \iota \tag{5}$$
$$I_{l,i} \equiv I \text{ for all other labels } l \text{ and indices } \iota \tag{6}$$

then we get the following interpretation tree for : " 'La plume de ma tante' is French"



The crucial step is the computation of I applied to the 'Q' node (call it Q):

$$\begin{aligned} I(Q) &\equiv M(Q.l)(I_{Q.l,1}(Q.c_1)) \\ &\equiv M(\text{Q})(I_{\text{Q},1}(Q.c_1)) \\ &\equiv M(\text{Q})(\iota(Q.c_1)) \\ &\equiv (lambda(x)x)((lambda(n)n.l)(Q.c_1)) \\ &\equiv (lambda(x)x)(\text{[la,plume,de,ma,tante]}) \\ &\equiv \text{[la,plume,de,ma,tante]} \end{aligned}$$

4 The compositional 'interpretation' of XML documents

Many XML languages are defined, or at least can be understood, in terms of a (more-or-less abstract) data model and rules for how to construct instances of that model given a document. So W3C XML Schema has schema components and mapping rules, (X)HTML has a box model and RDF has a graph. Does our compositionality story work here? Broadly speaking, yes.

For illustration we will define interpretation over only a subset of the XML information set, considered as just a tree of element information items (EIIs) and strings, with EIIs having only three properties: name, attributes and children, notated, for an element e, as $e.n$, $e.a$ and $e.c$ respectively. We'll write names using the familiar QName notation, consider attributes as (partial) functions from names to strings and children as a sequence of EIIs and/or strings. Square brackets ([...]) are used to delimit sequences, and angle brackets (< ... >) to delimit tuples.

Given this, we can define the interpretation rule for an EII e as follows:

$$I(e) \equiv M(e.n)(e.a, [I(e.c_1), ..., I(e.c_m)]) \tag{7}$$

For example, if we write an RDF graph edge labeled p from s to o as $< s, p, o >$, the following meaning definitions are sufficient to add the right three edges to a graph given an infoset for Example 4 from [3]:

$$\begin{aligned} M(\texttt{rdf:Description}) \equiv\ & \lambda(a, J) \\ & \forall\, j \in J, \textbf{add\_to\_graph}(j(U(a(\texttt{rdf:about})))) \end{aligned} \tag{8}$$

$$M(op) \equiv \lambda(a)\ \lambda(s)\ < s, R(op.n), U(a(\texttt{rdf:resource})) > \tag{9}$$

$$M(dp) \equiv \lambda(i)\ \lambda(s)\ < s, R(op.n), i > \tag{10}$$

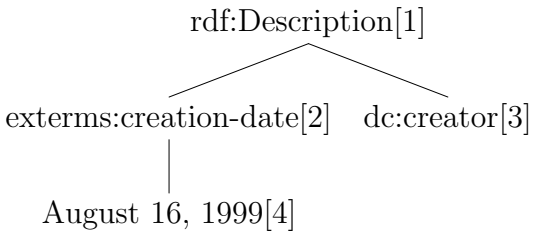
$$M(str) \equiv str \tag{11}$$

where *op* is the name of any element corresponding to an object property, *dp* is the name of any element corresponding to a datatype property, *str* is any string child, R is a function which converts QNames to URIs and U is a function which coerces strings to URIs.

Looking in detail at how this works for `rdf:Description`, combining its meaning (8) with the EII interpretation rule (7) we get that the interpretation of an `rdf:Description` EII adds one triple to the graph for each child, based on the EIIs `rdf:about` attribute and the interpretation of the child. Specifically, the triple is constructed by applying the interpretation of the child to the value of the `rdf:about` attribute coerced to a URI. It follows that the interpretation of each child must be an open proposition, which gets its subject filled in with the value of `rdf:about`.

To make this work, the interpretations of elements encoding properties have to be functions on one argument, and the meanings of those elements ($M(op)$ (9) and $M(dp)$ (10)) are designed to produce exactly that.

How all this works together can be seen in the following worked example. Given a two-child version of the Example 4 document (attributes not shown):



we would get the following interpretation computation:

[1] **add_to_graph**(< `http://www.example.org/index.html`,
`http://www.example.org/terms/creation-date`,
`"August 16, 1999"` >)
add_to_graph(< `http://www.example.org/index.html`,
`http://purl.org/dc/elements/1.1/creator`,
`http://www.example.org/staffid/85740` >)

[2] $\lambda(s)$ < s,
`http://www.example.org/terms/creation-date`,
`"August 16, 1999"` >

[3] $\lambda(s)$ < s,
`http://purl.org/dc/elements/1.1/creator`,
`http://www.example.org/staffid/85740` >

[4] "August 16, 1999"

Certain invariants have to be satisfied by the XML language in question for this kind of compositional story to succeed without complex conditional meaning functions. Either the interpretation of every kind of node has to be essentially of the same type (as is the case for (a large part of) XHTML, where every node correspondes to a box or a slug), so that almost any node's *meaning* function can be written independently of the kind of nodes its children are, or each node's meaning function has to know in advance the types of the interpretations of its children (as for RDF and XML Schema). Exactly what kinds of language schemas provide these invariants is an interesting question for further research.[1]

4.1 Quotation for XML

Some XML languages require quotation, that is, take the interpretation of infoset items in some context to just be those infoitems. Accordingly we need to adapt definition (7) in much the same way we adapted definition (1) to produce definition (2):

$$I(e) \equiv M(e.l)(e.a, [I_{e.l,1}(e.c_1), ..., I_{e.l,m}(e.c_m)]) \quad (12)$$

$$I_{e.l,i} \equiv \begin{cases} \iota \\ I \end{cases} \quad (13)$$

$$\iota \equiv \lambda(e)\ e \quad (14)$$

Now the recursive application of the interpretation function can be avoided on a case-by-case basis.

5 Elaborating infosets

I've proposed elsewhere [1] an approach to defining a notion of *elaborated* infoset, to address the need to identify a small set of 'standard' processes which will often/typically/normally be applied to the result of parsing an XML document before application-specific processing. At a more abstract level, the elaborated infoset of a document can be seen as answering the question "what information does the author of this document take responsibility for?".

[1] gives a recursive definition of elaboration for an entire document or subtree, but recognises that this is not adequate in some situations, particularly those in which multiple XML

[1] It's worth noting here how easy a 'must-ignore' rule as defined by (X)HTML is to implement in this approach—we just say that the meaning of unknown elements is the identity funcion, i.e. $M(unknown) \equiv \lambda(i)\ i$!

languages with different quoting requirements are combined in a single document. To address this, we can extend the approach developed here by first defining a non-recursive version of elaboration, and then integrating it into our compositional semantics equation.

Elaboration is essentially a process whereby an EII is replaced in an infoset with zero or more new EIIs, as determined by the name and/or attributes of the original EII. So for example an `xi:include` EII will be replaced (in the simplest case) by the EIIs resulting from parsing the XML document retrieved from the URI in its `href` attribute, an `xenc:EncryptedData` EII will be replaced by the result of decrypting its contents and an EII with an `xsl:version` attribute may be replaced by the result of treating the EII as an XSLT Literal Result Element. We call the `xi:include` and `xenc:EncryptedData` element names and the `xsl:version` attribute *elaboration signals*.

It is necessary to provide for both externally and internally specified quotation. That is, some applications may want to ignore some elaboration signals, and some documents may with to explicitly disable elaboration for some or all of their contents. A formal definition of elaboration (E) as a function from an EII (e) and a means of specifying signals to be ignored (X) to sequences of EIIs can be specified as:

$$E(e, X) \equiv \begin{cases} X(e.l, e.a) \rightarrow e \\ \texttt{q:q} \in e.a \rightarrow EII(e.l, e.a - \texttt{q:q}, e.c) \\ < e.l, e.a > \in \text{domain}(EE) \rightarrow \\ \qquad [\forall f \ \in E(EE(e.l, e.a)(e), X)\ E(f, X)] \\ (\text{otherwise}) \rightarrow e \end{cases} \tag{15}$$

where X is a (partial) function from EII names and attributes to $\{true, false\}$, which should be *true* for those elaboration signals which should be ignored, `q:q` is an attribute authors can use to signal that a given EII is not to be elaborated, but which *is* itself removed from the result[2] (EII is a constructor for EIIs) and EE is where the details of known elaborations are captured – it must be a partial function from EII names and attributes to (a function from EIIs to (sequences of) EIIs). In the event an unquoted elaboration signal is encountered, the result of the consequent elaboration will itself then be considered for elaboration—that is the impact of the application of E to the value of the application of EE's value to e in the third clause of the above definition. The resulting recursion will continue until a fixed point is reached.

The detailed definition of EE consists of the sum of all the relevant definitions from all the XML languages which are determined to belong to the set of elaborating languages. Just what those languages *are* is outside the scope of this document—see [1] for some discussion on this point.

6 Combining elaboration with interpretation

The final step is to integrate elaboration into our compositional definition of interpretation for EIIs (12). This gets a bit messy, in particular as we have *two* kinds of quotation to provide for: one with respect to elaboration, and the other with respect to interpretation.

[2]Two problems here: not sure removal is the right thing, and not clear how this will stop elaboration of the children, now that the recursion is going to be handled elsewhere. . .

A straightforward combination of definitions (12) and (15) gives the following:

$$I(e,X) \equiv \left[\begin{array}{l} \forall f \in E(e,X) \\ \quad M(f.l)(f.a, [I_{f.l,1}(f.c_1, X) + ... + I_{f.l,m}(f.c_m, X)]) \end{array} \right] \quad (16)$$

$$I_{f.l,i} \equiv \left\{ \begin{array}{l} \iota \\ I \end{array} \right. \quad (17)$$

$$\iota \equiv \lambda(e,X)\ [e] \quad (18)$$

where we now combine the results of recursive interpretation with + (for concatenation), since the value of our new I is always a sequence, and our quotation function ι is adjusted similarly to produce a single-item sequence.

There is at least one flaw with this approach: it suggests that X, the means by which elaboration may be restricted, is constant throughout the recursive descent down the infoset tree. But this is unlikely to be correct for documents which combine elements from more than one XML language, and may even not be correct for some single-language documents. To allow for context-appropriate changes to restriction, we can parameterise X by the relevant parent label and child position, in a move parallel to the way we handled infoset quotation:

$$I(e,X) \equiv \left[\begin{array}{l} \forall f \in E(e,X) \\ \quad M(f.l)(f.a, [I_{f.l,1}(f.c_1, X_{f.l,1}) + ... + I_{f.l,m}(f.c_m, X_{f.l,m})]) \end{array} \right] \quad (19)$$

$$I_{f.l,i} \equiv \left\{ \begin{array}{l} \iota \\ I \end{array} \right. \quad (20)$$

$$\iota \equiv \lambda(e,X)\ [e] \quad (21)$$

This amounts to enbodying in the definitions the fact that just as the original X is up to the application to determine, so the version of X to use in the recursion step is up to the application as well.

7 Conclusion

The sequence of definitions presented here establishes that it is possible to give a fully explicit account of how a compositional semantics for XML languages can be specified, taking advantage of independently motivated and defined notions of XML-generic elaboration and XML-language-specific compositional semantics. Although the ultimate result (19) is complex, it actually represents a simplification for specification writers and language designers, because it allows them to approach four tasks independently, knowing that the results can be combined:

- The compositional semantics of an XML language;
- The contextual determinants of quotation for that language;
- The contextual determinants of the amount of elaboration appropriate for applications processing that language;
- The overall inventory of elaborating languages.

Whether the notational and conceptual complexity of this story will stand in the way of the benefits of this simplification remains to be seen.

References

[1] Thompson, Henry S., *The elaborated infoset: a proposal*, W3C, 2007. Available online at `http://www.w3.org/2001/tag/doc/elabInfoset/`.

[2] Berners-Lee, Tim, *The Interpretation of XML documents*, W3C, 2007. Available online at `http://www.w3.org/DesignIssues/XML`.

[3] Manola, Frank and Eric Miller, eds. *RDF Primer*, W3C, 2004. Available online at `http://www.w3.org/TR/REC-rdf-syntax/#example4`.