Hey! You Got Imperative in My Declarative! *or* A Mashup Made in Heaven: Making OWL friendlier with Javacript.

Alan Ruttenberg[1]

Science Commons, Cambridge, MA 02127, USA

Abstract. We propose that OWL 1.1 incorporate the use of Javascript. By choosing to make use of Javascript within OWL, adoption and utility of OWL might be improved in two different ways. First, it could be used in a facility that lets users specify Javascript code to translate domain specific languages into OWL.

Second, Javascript is well known to a large number of web developers. By enabling OWL documents to have property values that are computed by Javascript functions, we extend the language in a useful way, and encourage the use of OWL in different applications than it might otherwise be used.

OWL needs a method of extending its syntax to enable concise expression of domain statements without compromising its expressiveness. For example, a current debate over the whether to use OWL as the "native" format for the OBO ontologies, is driven (away from using OWL) by OWL's unappealing syntax and the relative ease of understanding OBO's current format. We describe a model for how to use Javascript, included as part of an OWL ontology, to parse domain specific languages into native OWL. Two levels of operation are distinguished, the first a lexical translation into a language which extends the functional syntax, and a second macro expansion which translates the extended functional syntax to expressions that only use the defined OWL vocabulary.

For an implementation of computed property values, we propose that they have a status similar to annotation properties, in that they are not reasoned over. Instead they can be computed in terms of non-annotation property values. In this way, we allow for useful expressivity gains without complicating the OWL reasoning or imposing a complicated evaluation model. We discuss what access to the environment, such as the ability to query against the ontology, could profitably (and safely) be had by these scripts. Use cases motivated from experience in validating BioPAX and from other applications are provided.

Javascript interpreters are available for use from within the programming languages that the major reasoners are implemented in, and should therefore not pose an excessive burden on reasoner developers.