Ripple: Functional Programs as Linked Data

Joshua Shinavier
josh@fortytwo.net

Soph-Ware Associates, Inc.,
624 W. Hastings Rd, Spokane, WA 99218 USA
http://www.soph-ware.com

Abstract. Ripple is dedicated scripting language for linked data whose programs both operate upon and reside in RDF graphs. Ripple is a variation on the *concatenative* theme of functional, stack-oriented languages such as Joy and Factor, and takes a multivalued, pipeline approach to query composition. The Java implementation includes a query API, an extensible library of primitive functions, and an interactive command-line interpreter. A demo application can be found at: http://fortytwo.net/ripple.

1 Introduction

Most of the data which populates today's Semantic Web is purely *descriptive* in nature, while the complex procedural machinery for querying, crawling, transforming and reasoning about that data is buried within applications written in high-level languages such as Java or Python, and is neither machine-accessible nor reusable in the Semantic Web sense. This project explores the notion of *linked* or distributed programs as RDF graphs, and presents a functional, *concatenative* interpreted language, closely related to Manfred von Thun's Joy[1], as a proof of concept.

For a Turing-complete RDF query language, Ripple is an exercise in minimalism, both in terms of syntax and semantics. A Ripple *program* is a nested list structure described with the RDF collections vocabulary, and is thereby a first-class citizen of the Semantic Web. In the Java implementation, conversion between the RDF graph representation of a program and its more efficient linked list counterpart is transparent to the user application.

When we assign a program a URI, we're pushing its definition to the same RDF model from which our query results are drawn:

```
@define hello:
    "Hello world!".
```

Given an appropriate base URI and web-visible triple store, the program itself becomes a part of the global graph of linked data[2], enabling a remote application to read it in, execute it, and build upon it without restriction.

[1] http://www.latrobe.edu.au/philosophy/phimvt/joy/j01tut.html
[2] http://www.w3.org/DesignIssues/LinkedData.html

2 Examples

Ripple's query model combines a computational scheme based on *stack manipulation* with a functional "pipes and filters" pattern. The stack paradigm makes for minimal, *point-free* syntax at the RDF level, while the *pipeline* mechanism accommodates RDF's multivalued properties by distributing operations over arbitrary numbers of intermediate results. Programs resemble path expressions in which RDF lists, RDF properties, and "black box" primitive functions are interchangeable. For instance, in the following program, `toString`, and `sha1` are primitive functions, whereas `foaf:mbox` is a property:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/>.

# p => calculated mbox sha1 sum of p
@define mboxSha1: /foaf:mbox/toString/sha1.
```

The following example contains a more sophisticated query.

```
@prefix owl:  <http://www.w3.org/2002/07/owl#>.

# foaf1 => foaf1, foaf2, foaf3, ...
@define foafStep:   # iterator for a FOAF crawler
    (   id          # include foaf1 itself
        owl:sameAs  # include nodes identified with foaf1
        foaf:knows  # include those foaf:known by foaf1
    )/each/i        # apply all three patterns at once
    /unique.        # eliminate duplicate results

# => names of Tim Berners-Lee and friends
<http://www.w3.org/People/Berners-Lee/card#i>
    :foafStep    # the iterator function
    1/times      # (change this value to extend the crawl)
    /foaf:name.  # grab all matching individuals' names
```

3 Conclusion and Future Work

Ripple is an exploratory project, which is to say that further development will be driven by discoveries made along the way. Thus far, Ripple has been most useful for discovery, navigation and mapping of linked data. If the hypertext web is any indication of the future of the Semantic Web, it be vast, complex, and overall, loosely structured. As it grows, we will need an equally sophisticated *web of programs* to keep pace with it.

References

1. Tim Berners-Lee et al. Tabulator: Exploring and Analyzing linked data on the Semantic Web. In Proceedings of the 3rd International Semantic Web User Interaction Workshop, 2006.