Previewing Semantic Web Pipes

Previewing Semantic Web Pipes ⋆

Christian Morbidoni[2], Danh Le Phuoc[1], Axel Polleres[1], Matthias Samwald[1], and Giovanni Tummarello[1]

[1] DERI Galway, National University of Ireland, Galway
{firstname.lastname}@deri.org
[2] SeMedia Group, Universita' Politecnica delle Marche, Ancona, Italy
christian@deit.univpm.it

Abstract. In this demo we present a first implementation of Semantic Web Pipes, a powerful tool to build RDF-based mashups. Semantic Web pipes are defined in XML and when executed they fetch RDF graphs on the Web, operate on them, and produce an RDF output which is itself accessible via a stable URL. Humans can also use pipes directly thanks to HTML wrapping of the pipe parameters and outputs. The implementation we will demo includes an online AJAX pipe editor and execution engine. Pipes can be published and combined thus fostering collaborative editing and reuse of data mashups.

1 Introduction

Making effective use of RDF data published online (e.g. in sources as RDF DBLP, DBPEDIA etc) is, in practice, all but straightforward: data might be fragmented or incomplete so that multiple sources needs to be joined, different identifiers (URIs) are usually employed for the same entities, ontologies need alignment, certain information might be need to be "patched", etc. The only approach available to these problems so far has been custom programming such transformations for the specific task to be performed in a Semantic Web application. In this paper we present a paradigm for creating and reusing such transformation in a easy way: a Web based Software Pipeline for the Semantic Web.

A similar metaphor has been implemented in Yahoo Web Pipes[3], which allows to implement customized services and information streams by processing and combining Web sources (usually RSS feeds) using a cascade of simple operators. Since Web pipes are themselves HTTP retrievable data sources, they can be reused and combined to form other pipes. Also, Web pipes are "live": they are computed on demand at each HTTP invocation, thus reflect the current status of the original data sources.

Unfortunately Yahoo Web Pipes are engineered to operate using fundamentally the RSS paradigm (item list) which does not map well at all with the graph based data model of RDF. For this purpose Semantic Web Pipes have been written from the start to operate also on Semantic Web data, offering specialized operators to perform the most important data aggregation and transformation tasks.

When a pipe is invoked, simply fetching the pipe URL, the external sources are fetched dynamically and transformed transparently and thus the Semantic Web pipe will reflect the most up to date data available online.

⋆ This work has been supported by the European FP6 project inContext (IST-034718), by Science Foundation Ireland under the Lion project (SFI/02/CE1/I131), and by the European project DISCOVERY(ECP-2005-CULT-038206).

[3] http://pipes.yahoo.com/

2 Basic Operators

A *Semantic Web pipe* implements a predefined workflow that, given a set of RDF sources (resolvable URLs), processes them by means of special purpose operators. Unlike fully-fledged workflow models, our current pipes model is a simple construction kit that consists of linked *operators* for data processing. Each operator allow a set of unordered inputs in different yformats (to make them distinguishable) as well as a list of optional ordered inputs, and exactly one output.
Figure 1(b) shows a set of base operators which we implemented so far and which we will shortly explain below. *The ⊎-Operator: RDF Merge:* This operator takes

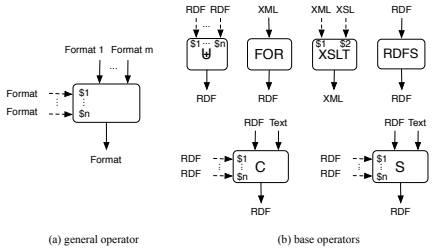

Fig. 1. Semantic Web pipe operators.

a list of RDF graphs as inputs, expressed in RDF/XML, N3 or Turtle format, and produces an RDF graph that is composed by the merge of its inputs. The standard implementation of the ⊎-operator simply standardizes blank nodes apart, according to RDF merge definition in [2], thus possibly generating non-lean graphs.
The C- and S-Operators: CONSTRUCT and SELECT: The C-operator outputs the result of a SPARQL [4] CONSTRUCT query given as textual input performed on the standard input RDF graphs. Similarly, the S-Operator performs a SELECT query and outputs the result in the SPARQL-Result XML format.
The RDFS-Operator: This operator basically performs materialization of the RDFS closure of the input graph by applying RDFS inference rules. We currently implement this using OWLIM.
The FOR-Operator: It works by taking a SPARQL XML result list (i.e. the output from the S operator) and binding each result with temporary variables which are then used as parameters in a subpipe which can be embedded inside it. The FOR operator is fundamental to enable many useful processing which involve discovering and using open data on the Semantic Web.
The XSLT-Operator: Finally, the XSLT-Adapter performs an XML transformation on a generic input XML document. This operator is particularly handy when custom XML output formats are needed or when an input source in a custom XML format shall be transformed to RDF/XML.
Examples for all operators can be found at `http://pipes.deri.org`.

2.1 A Semantic Web pipe example: about TBL

Pipes enable flexible aggregation of RDF data from various sources, here we present a simple example that show them in action. Data about Tim Berners-Lee

is available on various sources on the Semantic Web, e.g. his FOAF file, his RDF record of the DBLP scientific publication listing service and from DBPedia. This data cannot simply be merged directly as all three sources use different identifiers for Tim. Since we prefer using his self-chosen identifier from Tim's FOAF file, we will create a pipe as an aggregation of components that will convert the indentifiers used in DBLP and DBPedia. This is performed by using the C-operator with a SPARQL [4] query as shown below for DBLP:

```
CONSTRUCT {<http://www.w3.org/People/Berners-Lee/card#i> ?p ?o.
           ?s2 ?p2 <http://www.w3.org/People/Berners-Lee/card#i>}
   WHERE {{<http://dblp.l3s.de/d2r/.../Tim_Berners-Lee> ?p ?o}
   UNION {?s2 ?p2 <http://dblp.l3s.de/d2r/.../Tim_Berners-Lee>} }
```

A similar query is done to perform fix the identifier for Tim's DBPedia entry.[4] The whole use case is then easily adressed by the pipe shown in Figure 2: URIs are normalized via the C-operators and then joined with Tim's FOAF file.

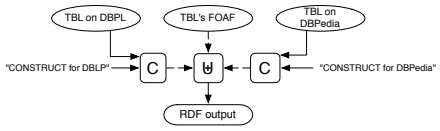

Fig. 2. A pipe that combines a Foaf file with DBLP and DBPedia entries.

For lack of space we do not discuss more compex examples here. However it would be simple to perform more interesting operations, such as fetching the 10 top hints from the Sindice[5] search engine (e.g. querying for TBL's URI or even for his email address as an IFP) and using a FOR block to merge them with the end results (possibly after a proper transformation or filtering).

3 Implementation

An open-source implementation is available online at `http://pipes.deri.org` and is composed by an execution engine and an AJAX based pipe editor. The engine supports the basic operators from Figure 1 plus more advanced ones which provide support for patching RDF graphs, or smushing URIs based on *owl:sameAs* relations. As the output of a pipe is an HTTP-retrievable RDF model or XML file, simple pipes can work as sources for more complex pipes. Additional functionalities are also available, such as "parametric pipes" which inject extra parameters via HTTP GET query string, allowing pipes to act within other pipes not only as sources but as full featured operators. Pipes are written in a simple XML language.[6] The following XML code show two pipes: a simple mix between two RDF sources (M-operator) and the one shown in Figure 2.

[4] `http://dbpedia.org/resource/Tim_Berners-Lee`

[5] `http://sindice.com`

[6] A grahical editor following the notation in Section 2 is currently under development

```
<mix>
  <source><fetch><location>
    http://www.w3.org/People/Berners-Lee/card
  </location></fetch></source>
  <source><fetch><location>
    http://g1o.net/foaf.rdf
  </location></fetch></source>
</mix>
```

```
<mix>
  <source><fetch><location>
    http://www.w3.org/People/Berners-Lee/card#i
  </location></fetch></source>
  <source><construct>
    <source><fetch><location>
        http://dblp.l3s.de/.../Tim_Berners-Lee
    </location></fetch></source>
    <query> <![CDATA[ CQ1 ]]> </query>
  </construct></source>
  <source><construct>
    <source><fetch><location>
        http://dbpedia.org/.../Tim_Berners-Lee
    </location></fetch></source>
    <query> <![CDATA[ CQ2 ]]> </query>
  </construct></source>
</mix>
```

Here, `CQ1` and `CQ2` stand for CONSTRUCT queries such as the ones previously shown. While it would be possible to implement pipe descriptions themselves in RDF, our current ad hoc XML language is more terse and legible. If an RDF representation will be later needed, it will be possible to obtain it via GRDDL.
HTTP-compliant caching is performed to avoid to recompute a pipe output if the sources have not changed. Whenever content is fetched it is hashed to detect changes. When no changes are detected the cached result is returned.
Circular invocations of the same pipe, which could create denial of services, can be easily detected within the same pipe engine, but not when different engines are involved. In this cases our solution relies on extra HTTP headers: whenever a model is fetched coming from an another pipe engine, an HTTP GET is performed putting an extra *PipeTTL* (Time To Live) header. The TTL number is decremented at each subsequent invocation. A pipe engine refuses to fetch more sources if the PipeTTL header is ≤ 1.
The AJAX pipe editor provides inline operator documentation when inserting a component. It presents a list of available pipes, fostering pipe reuse and composition. While normal runtime behavior is very accommodating to network errors (using copies of previous files on network timeouts or treating malformed input as empty sources), a debug mode is available, which highlights execution errors.
Finally, thanks to HTTP content negotiation, humans can use Semantic Web Pipes directly. Pipes parameters can be inputted directly in HTML boxes and the results will be shown by the use of the Simile Exhibit data browser[7].

4 Related Works

Semantic Web pipes as described in this paper are similar, in sense, to UNIX pipes[8] , but they allow to connect outputs to multiple inputs of other operators so that there can be multiple branches executed at the same time.

[7] `http://simile.mit.edu/exhibit/`

[8] `http://www.linfo.org/pipe.html`

Cascaded XML transformations are sometimes referred to as XML pipelines and have been successfully employed in projects like Apache Cocoon.[9]
The Yahoo Web Pipes framework was greatly inspiring our work, but lacks in functionality to address our desired use cases. Yahoo pipes provides an easy to use and powerful Web based graphic composer for pipes.
Concerning the Semantic Web world, the need for a cascade of operators to process RDF repositories is also addressed in the SIMILE Banach project[10], that enhance the Sesame triplestore by implementing pipelined stack of operators (implemented as SAILS). These can both process data and rewrite queries.

5 Conclusions and Future Works

Semantic Web pipes were also shown to be a paradigm that can do more than data harmonization alone: they implement workflows which can be used to model data flow scenarios that also include collaborative aspects. Most importantly, Semantic Web pipes are based on the union of functional operators specific to Semantic Web with the HTTP REST paradigm. Such combination fosters clean implementations, and promotes reuse of data sources as well as pipes themselves.
As we mentioned, a number of additional operators can then be imagined to aid ontology and data alignment when SPARQL CONSTRUCT queries are inconvenient or do not have the required features. Also, it will be interesting to consider how to achieve interaction between advanced RSS feed processing tools like Yahoo Pipes and Semantic Web operators. The SPARQL SELECT operator, producing XML, together with an XSLT transforms could provide a base for this. Many technical solutions can also be put in place to achieve scalability. These range from smart pipe execution strategies, advancing those explained in the previous sections, to others such as, for example, differential updates of the local copy of large remote RDF graphs [5].
Finally, while Semantic Web pipes (like Web pipes and Unix pipes) are certainly a tool for expert users, it is undeniable that the overall engine will be much more useful once a visual pipe editor is availale. A grahical editor for our XML format following the notation in Section 2 is currently under development.

References

1. D. Brickley and L. Miller. FOAF Vocabulary Spec., July 2005.
2. P. Hayes. RDF semantics, Feb.2004. W3C Rec.
3. C. Morbidoni, A. Polleres, G. Tummarello, and D. Le Phuoc. Semantic Web Pipes. Technical Report, see `http://pipes.deri.org/`. Nov. 2007.
4. E. Prud'hommeaux, A. Seaborne(eds.). SPARQL Query Language for RDF, June 2007. W3C Cand.Rec.
5. G. Tummarello, C. Morbidoni, R. Bachmann-Gmur, O. Erling. RDFSync: efficient remote synchronization of RDF models. *6th Int.l Semantic Web Conf. (ISWC'07)*, 2007.

[9] `http://cocoon.apache.org/`
[10] `http://simile.mit.edu/wiki/Banach`