Graph Fabric – Being Native on the Web, Not Migrant

Thomas Lörtsch

tl@rat.io

Abstract: While the web is ubiquitious today it is not as accessible and malleable to the user as it should be. Neither is it possible to annotate resources like one can with the printed papers of old, nor can resources on the web be integrated with personal data, nor are there natural ways to share data with friends, colleagues, the whole web or just across different personal devices. These shortcomings may seem like few and far between annoyances, but tackled altogether they will paradigmatically improve our grasp on the web.
Enhancing the web browser with an XPath-driven editor together with a filtering proxy will facilitate editing of any page on the web. A semantic toolchain comprised of graphstore, reasoner and query engine, will integrate any source and perspective the user cares about. Well defined parts of the personal graph will be shared with peers, on the web or just for personal backup. Access and control to this functionality will be provided through a user interface layer that acts like an overlay to any page rendered in the browser.

Keywords: Semantic Web, Read Write Web, Ontology, Conceptual Graphs, Federated Social Networks

1 Introduction

The World Wide Web today is ubiquitous as an everyday means of information, communication and living. All information we need is readily accessible on the web. Practically all our private and professional communication flows through its channels. We even live and organize our social life on the web. We have computers at work, on the lap, by the couch and even our smartphones are more powerful than last decades workstations. But still: this life is scattered across machines and providers, hidden in centralized data silos and application-specific repositories. All these data basins and streams have little to none integration, shared schema or cross cutting access, not even on a personal level, not even as an option.There is no fabric through which the different realms can be connected, interlinked, synchronized. All we get are amorphous tagclouds, breathless actuality (activity streams) or brute force fulltext search. Furthermore the web bars the user from any interaction that hasn't been explicitly implemented and sanctioned by the publisher. Appropriation, creation or collaboration can only take place in strictly demarcated areas. Why can't we annotate or scribble every resource on the web like we could with printed paper? Sure we can download a page, save a copy locally, print it, but then we loose the connection to the web, we loose the essence of the web: the interlinkedness.

An integrated web has to overcome these barriers that seemed natural in the age of personal computing and closed world competition:

- It has to become possible to annotate and modify any resource on the web - without anyone else noticing if we don't want them to.
- Sharing, collaborating and publishing to friends, colleagues, special interest groups or the world wide public has to be be a matter of a mouseclick or a predefined rule.
- Personal data must be allowed to live in multiple places at once - on a mobile phone, in the cloud, on a laptop - in a federated environment, allowing work and communication from everywhere, anytime and with everyone we chose to. There can not be any central services anymore that lock our data in.

- A unifying layer of structural meta information needs to be established that encompasses all aspects of data we use and generate. This can not be based on one central hierarchy or only one guiding principle like time or location: it must be a flexible, multidimensional structure that allows to map, to integrate and to cross-reference between the data realms we care about, and which does as much as possible on its own, semi-automatically as well as guided through user-generated rules and directives.
- With functionality comes complexity and with possibilities come dangers: it is essential that the workings of this machinery are logically structured, intuitively clear and concisely represented in an clear-cut user interface. Otherwise integration will not happen and mistakes might have bad consequences (eg you always want to be very sure what exactly you publish to whom exactly, especially when you automate that procedure).

Semantic web technologies, combined with some special tooling, can facilitate that graph fabric, flexibly integrating the multitude of tasks, domains and perspectives that typically constitute our digital life. The browser will be enhanced by a suite of semantic technologies, from graphstore to reasoner to semantic search engine. Other tools will provide functionalities and access: a local proxy assures us our personal view_on and part_of the web. Tools like XPath-engine and PDF-parser provide access far beyond the document level. OS-level services integrate filesystem and web, local and remote data, the personal and the public. Control and management is provided through an always reachable overlay of dynamic user interfaces in the browser, backed by rules, heuristics and flexible customizability. Distributed messaging middleware establishes smart channels between our machines, to selected peers and friends and circles, or to the world wide public.

2 The Web Is My Oyster

Integration of all the different data sources and streams that together form our data universe faces two obstacles: how to make them all reachable under one roof and how to unify their different structures into one schema.

Reachability in the browser is a solvable problem: in the case of web data it's trivial anyhow, in fact everything with a URI is perfectly acceptable. Bookmarks pose more problems since they tend to be hidden in a repository local to the browser application. Mail can be replicated into a webmail interface, which solves the problem good enough. New messaging services like Twitter or Google Wave most of the time get replicated on the web right from the start. Chat is more difficult since web accessible chatlogs tend to be an unpleasant read. Also problematic are "Social Web 2.0" services which tend to lock their users in by encapsulating their data. These practices can be circumvented by hackery but in the long run the success of open federated systems will have to make them politically and economically obsolete. The hardest problem is integration of local data like the contents of the users home directory. The HTML5 File API might become very helpful here, meanwhile a local WebDAV installation serving the home directory or any other location as localhost can do the job. That solves unified reachability of our dataspace in the browser, but now how to manage the mess?

One hierarchical ontology will most probably not be sufficient to do justice to data from all our activities, projects and interests, over time, not to mention the inconsistencies that inevitably arise when we start to integrate data sources from friends, colleagues or the open web. We have to develop means to connect different ontologies and different perspectives elastically. Ontologies can include sub-ontologies and provide mappings for equivalent classes, but sometimes we will have to be content with defining not more than crossings between dataspaces and accept that the resulting fabric develops into a multidimensional space of structured patches. Still that fabric can be enormously useful if we succeed in describing which premises lead to diverging structures while there sure are enough other aspects that still work across warped levels. There are for example always the rather universal categories of time, space and individuals, that are valid even in context of the most contradicting world views. Different perspectives on one and the same issue will most of the time diverge on one aspect but agree on others. A structural framework that makes it possible to express these perspectives can enable a great deal of operational advantages even in a cluttered, fractured data space.

While a multi-hierarchical ontological structure is one means to order and access the data scape, fulltext search is another and very popular option, and querying for certain properties is a promising approach when the data objects are already structured. Combinations of these strategies, like e.g. full text searches in certain parts of the ontology, filtered by results that meet certain criteria on properties, can be very effective. The integration of local and remote datasources also offers new opportunities

like searches that also encompass bookmarked websites. An automatic clustering engine can be trained with the structure it encounters in file system, mail folders and bookmarks and propose new ontological perspectives, evolving them over time as we write new texts, have new mail conversations, add new bookmarks etc.

As soon as all our data is reachable under the roof of one integrated ontological structure it's easy to make it accessible to others. Some parts we will want to publish to the whole world, other we might want to share with friends or colleagues. One of the dimensions of our meta structure will have to be access rules that govern precisely who is allowed to read (and eventually write) which resources. After publishing is in place a natural next step is consumption/ingestion. We might work together with colleagues on some project and subscribe to their data stream - the part of their data that they decided to publish on this topic to their colleagues. This might be their bookmarks, notes, a folder in their home directory or even a part of their ontological structure. With the subscription we also define rules where and how these resources should be integrated into our datascape. We might want them to blend in seamlessly or maybe we need to check them in seperately - that's all within our control. In the same way we could register for some public collaborative data stream like a video feed from YouTube or a subtree of the dmoz open directory and clip them into our ontology.

3 Edit-Able

One of the most annoying things about the web is that webpages can't be annotated by the reader. This is the most natural thing to do when reading a paper, a journal or a book: note once thoughts at the margin, mark something important, put a bookmark on a certain page. This is so natural and usefull when working with texts that it really hurts not to have these possibilities on the web. Printing out interesting articles is no alternative since one looses the immediate context of the page, the links and the ability to copy and paste parts of it. Of course it also not possible to grant the world wide public write access to every resource on the web: even if everybody was very responsible it would still make the web unreadable, buried under other peoples' remarks.

Instead the web browser can be enhanced by an XPath-capable mini-editor, a filtering proxy and some local storage to the effect that when the user wishes to annotate a page on the web, the editing tool via its XPath implementation can add the annotation to a specific part of the page without actually touching it. It just records the HTML element that the user clicked on and saves that location together with the annotation body and of course the URI of the annotated page. Every web access of the browser has to be routed through a local proxy and checked if the local database contains any metainformation regarding that page like eg. an annotation. The next time the user visits the annotated page the proxy checks its notes, sees the annotation and, via the XPath command, the exact element to which it belongs, and adds them to the document DOM so the browser can render them as an overlay into the web page. Annotations can be notes rendered on the margin of a page but they can also result in cutting out all parts of the page the user finds distracting or in inserting links the user wants to remember in the context of that web page.

A useful addition to the annotation facility would be an archival option that automatically archives any annotated page locally, ensuring that even if the page changes profoundly the annotation can still be meaningfully assigned - in this case the user would receive a warning or a prompt to choose if he'd prefer to see archived version or the new one. If the page didn't change too much the XPath engine might still be able to identify the correct element and proceed as usual.

4 Foundations

This proposal relies heavily on a notion of context. Integration of activities, tasks, realms, view points etc doesn't just happen by mixing everything together in a big pot. Integration most often means connecting while respecting the original situation, or context. Context is a tricky concept that fills conferences on it's own and is only half bakedly established on the Semantic Web. It is not part of RDF so far, but implemented in some way in most RDF stores and codified in the SPARQL standard. This usage will be standardized in the upcoming updated RDF 1.1 although what exactly the semantics will be is not clear so far.

The famous semantic web layer cake names a "unifying logic" encompassing OWL, RIF and SPARQL, but not much is clear about what this unifying logic will be. After years of concentration on Description Logics as a decidable subset of First Order Logic it becomes clear that decidability isn't a

sufficient goal since complexities can still be practically impossible to master. That means that even in the decidable subset of logic that is the foundation of OWL practical implementations have to rely on heuristics, and concepts like "locally closed worlds" are developed in an attempt to reduce complexities in real world usecases,

This project therefor tries to tackle the problem from another side, based on Nested Conceptual Graphs. This formalism doesn't emphasize decidability but comprehensibility. Logic is hard enough to grasp and use for the untrained user, it doesn't make much sense to make it even harder for the sake of a theoretic decidability which doesn't work in practice anyway. Better strive for a formalism that's as easy to use as possible and apply heuristics on the fly, eventually guiding the user in reformulating his tasks or adjusting his expectations towards a feasable goal.

In a context-driven environment we also have good means to reduce the sheer amount of data to crunch since we won't usually reason about the whole web but about a very limited subset like the data local to the user, eventually also the data his peers made available to him and some subset of the web referred to in his bookmarks. Still this is a much more practical datasat than say the whole LOD cloud. This too will help to achieve good performance even on the ground of a formalism that's not optimized for decidability.

Optionally logical subsets can be specified that are proven to be decidable. This can for example be very important when rules have to ensure that no data is published that was meant to remain private. This of course is an example where both theoretical decidability as well as practical comprehensibility and clear guidance to the user in formulating those rules are absolutely essential.

5 Technicalities

The browser environment is the central jacking point of this effort and it is only natural to base implementation on the programming language of the browser: JavaScript, a language with a growing reputation for its powerful declarative core, also nicely supported on the server side through frameworks like Node.js and scaling quite well on parallelized hardware. Java on the other hand is not as readily available in the browser but very well supported on the server, has very strong development support and a huge offering of libraries and softwares. A mixed environment is also thinkable where the interface and basic functionality are implemented in JavaScript and on demand available in the browser without the need for explicit download and installation, whereas a full blown suite providing all the more involved functionalities and implemented in Java runs locally and makes its services available as localhost.

Whereas a JavaScript-based solution is an attractive aim it will need a lot of work. A Java-based open source solution could be build on already available softwares: a semantic suite like Sesame, a quad store like Mulgara, a publishing framework like Play!, a WebDAV server like Jackrabbit, a search engine like Lucene, an autocategorizer like Carrot2, a webcrawler like Heritrix, a collaborative editing environment like Apache Wave etc.

6 Conclusion

This project can seem puzzling because it consists of so many parts, most of them rather smallish and none of them in itself really spectacular. But these little parts all fill little dents in the wheel of web - taken together they make the difference between a bumpy ride that merely gets us from A to B and an exciting journey which enriches our life. Being able to connect all the stuff we see and read and work on in one permeating graph, being able to annotate and rework everything at will, being able to specifically share certain parts or aspects with certain peers or groups, getting search results aligned with what we already work on - altogether it means that finally the user is in the center of the web (again, like in 1994, although with much more abilities), not the applications, not the offerings, not the providers. With this set of tools we can construct the fabric that allows us to reside on the web, become native where we now are migrants.