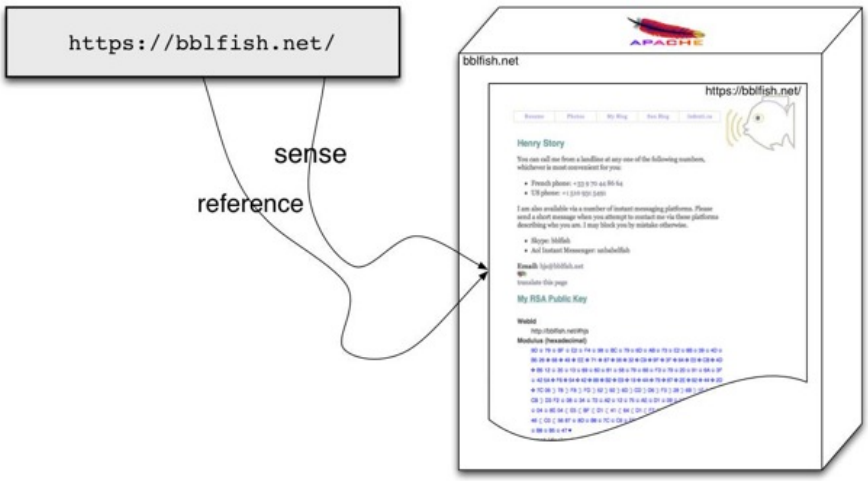
From: Henry Story <henry.story@bblfish.net>
Subject: Re: Certificate Triplify Challenge
Date: 12 January 2012 19:21:53 CET
To: Kingsley Idehen <kidehen@openlinksw.com>
Cc: public-xg-webid@w3.org

2 Attachments, 79 KB

Let me now jump to the more positive parts of Kingsley's reply.

On 12 Jan 2012, at 03:54, Kingsley Idehen wrote:

> On 1/11/12 8:06 PM, Henry Story wrote:
>> In the grey box below we have put the URI which is the name for a document served by an apache server. The name is one thing (it has 21 characters) the document is another thing (it contains a public key and many more characters, and can change over time)
>>
>> 
>>
>>
>> When, as in this case, the URI _refers_ to a document, the sense and the reference of the URI coincide: they are both the document, or information resource referred to.
>>
>> But with "https://bblfish.net/#hjs" things are different. That URI refers to <https://bblfish.net/#hjs> ie to me. The sense of the URI can be found at <https://bblfish.net/> which is a document, the same document we were discussing in the previous diagram.
>
> In computer science terms e.g., 'C':
>
> 1. https://bblfish.net/#hjs -- "*" (de-reference / indirection unary operator)
> 2. https://bblfish.net/ -- "&" (address-of unary operator).

yes, that is interesting. This is very likely where we got the idea of dereferencing URLs from. So this is a good time to see if this analogy is a good one. Ie, we have to see where it holds and where it does not. I don't think I ever looked at that carefully, and since we use these terms in the spec quite a lot, it is a good time to consider this.

So in order to answer this I first had to remind myself of C addressing since it has been some time that I have not used it. I found a good tutorial here http://augustcouncil.com/~tgibson/tutorial/ptr.html

[...reading...]

In C the address-of operator '&' is an operator that takes a variable and gives you an unsigned int.

```
4:   float fl=3.14;
5:   unsigned int addr=(unsigned int) &fl;
```

So if we apply this to URIs then we would need a function from URIs to their addresses, which could be expressed

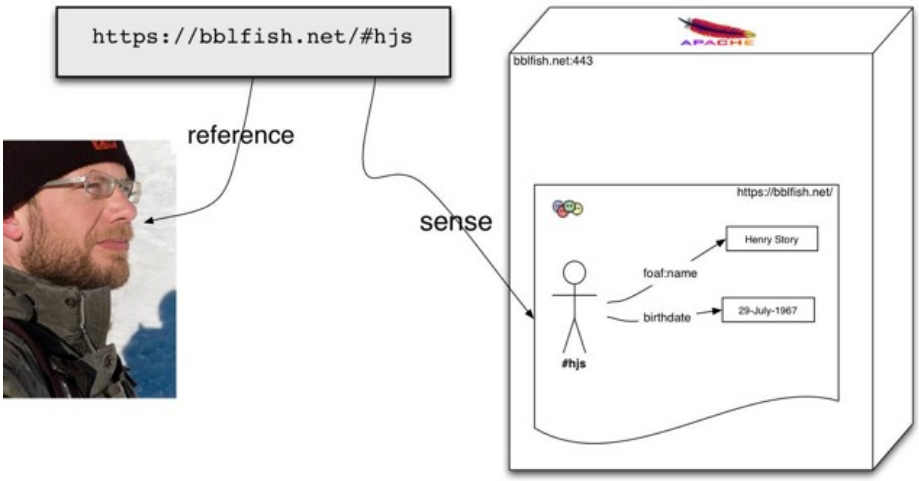
as an owl:FunctionalProperty. Let us call it :addr

"https://bblfish.net/#hjs"^^xsd:anyURI :addr ?addr1 .
"https://bblfish.net/"^^xsd:anyURI :addr ?addr2 .

It turns out that the value of the :addr relation is not determinable a-priori, since one needs in http for example to follow the possible redirects to be able to determine what the value of it is. So for example it is quite possible that following statement satisfy the above query:

"https://bblfish.net/"^^xsd:anyURI :addr <http://bblfish.net/index.html> .

Now what is the equivalent of the C de-reference * operator for us? This is a function that when given an address, returns the value of that address. There is a relation in existence that does this `log:content` (see http://www.w3.org/2000/10/swap/doc/Reach). I am not sure that relation is exactly what we are looking for, as it says "for documents", and we are looking for something that gives us the content of any uri. This is what in my diagrams I call the :sense relation. In all we were dealing with was the document web then log:content would be enough. But since we also dealing with referents in the real world, as shown in this diagram, then we need the :sense relation here (perhaps better called senseDoc) which relates any URI to the document that gives its canonical sense.



"https://bblfish.net/#hjs"^^xsd:anyURI :sense """ ... <#hjs> foaf:name "Henry Story" ... """ .

Interestingly it looks like we can now combine :addr with log:content to get :sense .

{ ?uri :addr ?addr .
?addr log:content ?content } => { ?uri :sense ?content } .

So perhaps we are not that far off from combining & and *

```
1: #include <stdio.h>
2: int main()
3: {
4:   float fl=3.14;
5:   unsigned int addr=(unsigned int) &fl;
6:   printf("fl's address=%u\n", addr);
7:   printf("addr's contents=%.2f\n", * (float*) addr);
8:   return 0;
9: }
```

Ok. So that is interesting.

Good so with this mapping clarified, I will next need to see what the issue is that we were trying to resolve originally.

Henry