

Extracción de Datos de Sitios de la Web Profunda Anotados Semánticamente

Eduardo Martín Rojo, Vicente Luque Centeno

Universidad Carlos III de Madrid
Av. Universidad 30, 28911
Leganés (Madrid), España
{emartin,vlc}@it.uc3m.es

Resumen La navegación y recolección de información de forma automática de páginas de la Web Profunda requiere de la utilización de Wrappers Web que permitan simular la interacción realizada por usuarios humanos; sin embargo, este tipo de aplicaciones poseen algunos problemas debido a que su implementación posee una fuerte dependencia con la estructura del sitio web accedido y a que su código fuente requiere de un mantenimiento constante.

En este trabajo proponemos un modelo de anotación para sitios de la Web Profunda que pueda ser utilizado para la extracción de información. Las anotaciones realizadas con nuestro modelo se expresan desde el punto de vista del cliente permitiendo a usuarios finales y a terceras partes el desarrollo de sus propias anotaciones para cualquier sitio web. Esta anotaciones permitirán la creación de Wrappers Web más adaptables a los posibles cambios estructurales del sitio web accedido.

1. Introducción

Actualmente los sitios Web más populares proporcionan a sus usuarios de herramientas de desarrollo que permiten la creación de aplicaciones capaces de acceder a algunas de sus funcionalidades (por ejemplo, *eBay Developers Program*¹); y frecuentemente dichas funcionalidades son ofrecidas por medio de servicios Web que permiten ser accedidos con aplicaciones para *mashups* como *Google Mashups*², *Yahoo Pipes*³ or *Microsoft Popfly*⁴. Sin embargo, la inmensa mayoría de sitios Web están enfocados a ser manejados únicamente por usuarios humanos, con lo que no proveen de este tipo de herramientas. El acceso a este tipo de sitios se realiza por medio de los llamados *Wrappers Web*[5].

Algunos de los principales problemas de los Wrappers Web son: en primer lugar, requieren que los desarrolladores posean un alto nivel de conocimiento de la

¹ <http://developer.ebay.com/>

² <http://www.googlemashups.com/>

³ <http://pipes.yahoo.com>

⁴ <http://www.popfly.com>

estructura del sitio Web accedido dado que los Wrappers son soluciones específicas para cada sitio Web concreto; y en segundo lugar, los Wrappers Web requieren de un contante mantenimiento que les permita ser capaces de adaptarse a los cambios futuros. La evolución de las herramientas de desarrollo para Wrappers Web se ha dirigido a intentar resolver estas debilidades y al mismo tiempo hacer posible una integración de datos Web de fuentes heterogéneas de una manera más fácil. Algunos ejemplos de herramientas de desarrollo de Wrappers Web son las herramientas para creación de soluciones específicas como *GreaseMonkey*⁵ y *Ubiquity*⁶, las herramientas de usuario final que permiten la utilización de ciertos elementos de lenguaje natural como *Chickenfoot*⁷, o también herramientas gráficas como *OpenKapow RoboMaker*⁸.

La principal característica común a todas las soluciones de desarrollo de los Wrappers Web en la actualidad es que, aunque proveen de un manejo sencillo, siempre implican un tipo de desarrollo con fuerte dependencia respecto de la estructura del sitio Web.

En el trabajo que presentamos a continuación hemos definido una formalización que permite anotar semánticamente la estructura de un sitio Web para representar la navegación a través del sitio. Estas anotaciones poseen a su vez referencias que permiten localizar los fragmentos de información más importantes del sitio Web durante la navegación; utilizando las actuales herramientas de desarrollo de Wrappers Web nuestras anotaciones permitirán la implementación de Wrappers basados en el modelo del sitio web que eviten el solapado con la estructura del sitio. Además, el mantenimiento requerido por los cambios producidos en el sitio Web quedarán aislados dentro de la capa del modelo.

En nuestro trabajo hemos elegido utilizar el punto de vista del cliente para la elaboración de los modelos de navegación de los sitios Web debido a que:

- El punto de vista del cliente permite a terceras partes y usuarios finales participar y colaborar en la creación de anotaciones sobre cualquier sitio Web, ya que no requieren acceder al servidor.
- No es intrusivo; no requiere cambiar el sitio Web que está siendo anotado como lo requerirían otros métodos de anotación como RDFa[9] o Microformats[8].

2. Trabajos Relacionados y Contribución

La generación del modelo de navegación de Web Profunda es tratado por [10], donde los autores generan un modelo de navegación donde utilizan búsqueda de palabras clave para identificar las diferentes páginas de la Web Profunda. Otro trabajo, más enfocado a la interacción con sitios de la Web Profunda es *Trascendence*[1], un sistema que permite a los usuarios generalizar sus consultas sobre un formulario de búsqueda para poder expandir su espacio de búsqueda.

⁵ <http://www.greasespot.net/>

⁶ <http://ubiquity.mozilla.com/>

⁷ <http://groups.csail.mit.edu/uid/chickenfoot/>

⁸ <http://openkapow.com/>

Trascendence permite además definir patrones para extracción de datos, lo que permite combinar los datos obtenidos con otras fuentes de datos.

Uno de los principales problemas de la Web Profunda reside en que una página no siempre puede ser representada de forma unívoca por medio de una URL⁹. El problema de referenciar páginas de la Web Profunda es mencionado en [6], donde los autores presentan una solución para crear *bookmarks* a páginas de la Web Profunda utilizando una secuencia de pasos especificada por medio de scripts *Chickenfoot* que simulan la interacción requerida por parte del usuario para poder alcanzar la página marcada.

Extraer datos semánticos de sitios Web es un problema que ha sido afrontado con anterioridad en *Marmite*[13], un sistema que proporciona una interfaz al usuario final que le permite diseñar el flujo de procesos necesario para extraer datos; o también en *PiggyBank*[7], un sistema donde el usuario puede hacer uso de scripts llamados *Screen Scrapers* para convertir HTML en datos semánticos RDF.

En nuestro sistema demo hacemos uso de *Chickenfoot* como herramienta de desarrollo de Wrappers. Las principales características de esta herramienta son presentadas en [2] y [3]. *Chickenfoot* permite la especificación de interacciones Web del lado del cliente por medio de una extensión del lenguaje Javascript.

3. Modelo de Anotación de la Web Profunda

El objetivo de un cliente Web es alcanzar un determinado estado por medio de interactuar con la Web. Nuestro modelo representa los estados y transiciones que componen el grafo de navegación que refleja todas las posibles situaciones que pueden ocurrir en un sitio Web desde el punto de vista del cliente. En el grafo, los vértices representan todos los posibles estados lógicos en que se puede encontrar la aplicación durante la interacción producida por el usuario, mientras que las aristas del grafo representan las acciones producidas que permiten las transiciones entre estados.

Cada estado puede ser dividido en elementos llamados fragmentos, y estos fragmentos a su vez pueden ser divididos en nuevos fragmentos. Un fragmento se identifica por medio de una expresión XPath dentro de un estado, con lo que cada fragmento representa un tipo de contenido semántico que puede ser extraído al utilizar dicha expresión sobre el estado al que pertenece. La figura 1 muestra un ejemplo de división de estados en diferentes fragmentos. Los fragmentos permitirán extraer información semántica a partir de páginas de la Web Profunda si conocemos la localización del estado dentro del grafo de navegación del sitio Web.

Una transición representa las acciones disponibles para el usuario a partir de un estado origen. Estas acciones permitirán cambiar entre diferentes estados en el sitio Web. Las transiciones están compuestas de, en primer lugar, una secuencia ordenada de interacciones que originan el cambio de estado; y en segundo lugar, una lista de todos los posibles destinos que pueden ser alcanzables

⁹ Uniform Resource Locator, defined on <http://tools.ietf.org/html/rfc1738>

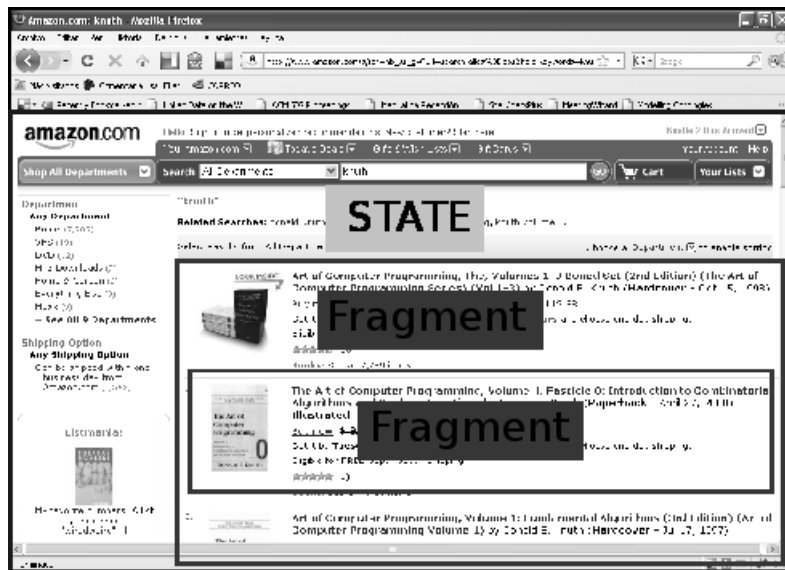


Figura 1. Estados y Fragmentos

por la utilización de estas transiciones. Nuestro grafo de estados y transiciones es **no determinista**, es decir, dado un origen determinado y una transición, pueden existir varios posibles destinos debido a que un sitio Web podría actuar de diferentes formas no predecibles para el cliente. Esto puede producirse como consecuencia de factores dinámicos como el estado de la plataforma que ofrece el servicio, el histórico de interacciones, la fecha y hora, etc.

Nuestro modelo de estados y transiciones para el grafo de navegación está representado por los siguientes tipos de elementos:

1. **PageState**: un estado unívoco identificable que representa una página de la Web Profunda. Un estado contiene fragmentos.
2. **Fragment**: representa un elemento dentro de un **PageState** o de otro **Fragment**. Esta clase se define como dominio de las siguientes propiedades:
 - a) **fragmentOf**: Define este fragmento como parte de otro fragmento o parte de un **PageState** dentro de una jerarquía.
 - b) **locatedBy**: una expresión XPath que identifica la posición del fragmento dentro de la representación XHTML del **PageState** o **Fragment** con que se enlaza por medio de la propiedad **fragmentOf**.
 - c) **semantic**: conjunto de triplas RDF que representa el conocimiento referenciado por este **Fragment**. Esta propiedad está definida también para las clases **Input** y **Action**.
3. **Transition**: representa un conjunto de acciones que permiten el cambio de estado entre un conjunto de estados de origen y un conjunto de posibles estados de destino. Tiene las siguientes propiedades:

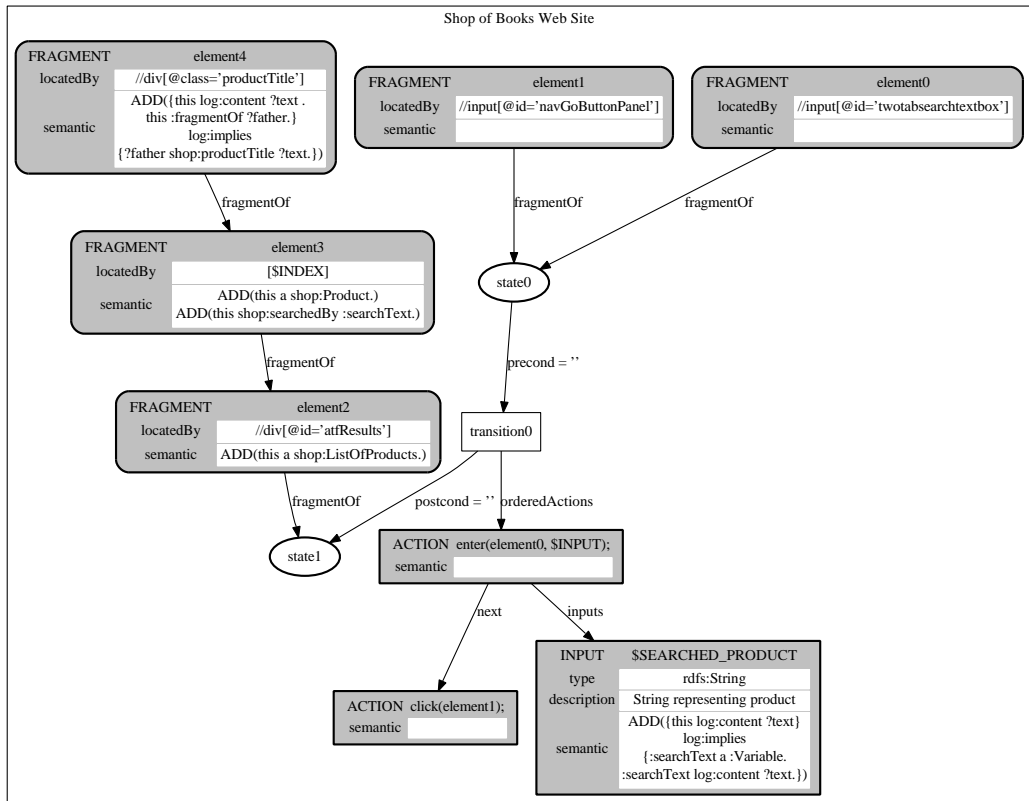


Figura 2. Modelo de navegación anotado para la realización de búsquedas en una supuesta tienda de libros

- a) **actions:** una transición es originada por una secuencia ordenada de acciones (instancias de la clase *Action*). Esta propiedad contiene dicha lista de acciones.
 - b) **sources y destinations:** todos los posibles estados que pueden ser fuente o destino de esta transición respectivamente.
4. **Action:** una interacción que puede ser llevada a cabo sobre un fragmento. Contiene las siguientes propiedades:
- a) **hasType:** tipo de interacción (clic sobre un elemento, selección de elemento, introducción de datos textuales...). Se representa por medio de un comando *Chickenfoot* en las anotaciones de nuestra demo.
 - b) **inputs:** todos los valores de formulario necesarios para llevar a cabo la interacción son referidos por esta propiedad como una lista que contiene elementos de tipo *Input*. Para cada entrada *Input* debe proporcionarse un nombre, un tipo de dato de XML Schema, y una descripción.

Hemos representado esta formalización por medio de una ontología en OWL¹⁰ que puede ser accedida desde [12].

En la figura 2 hemos representado un ejemplo de anotación para una supuesta Web que permite buscar y comprar libros. Desde la página inicial de este sitio representada por el nodo `state0`, existen dos fragmentos: `element0` (una caja de texto para búsquedas) y `element1` (un botón que inicia la tarea de búsqueda). Desde `state0`, es posible ir a `state1` a través de la transición `transition0`. Esta transición requiere desarrollar dos acciones (representadas por la lista de acciones `orderedActions`): la primera acción es llevada a cabo al insertar una cadena de texto dentro de la caja de texto `element0` mientras que la segunda acción se lleva a cabo por medio de la realización de un clic de ratón sobre el botón `element1`. En el estado `state1`, existen tres fragmentos que pueden ser accedidos: `element2`, `element3` y `element4`. Ya que cada uno de ellos es una parte de otro fragmento, la expresión XPath expresada en la propiedad `locatedBy` es relativa al fragmento o estado padre. Por ejemplo, `element4` es construido con las expresiones XPath de `element3` y `element2`, y por eso la localización XPath resultante es `//div[@id='atfResults'][$INDEX]//div[@class='productPrice']`.

En esta figura hemos utilizado una ontología hipotética para anotar conceptos relacionados con el escenario del ejemplo, pero el modelo de anotación presentado puede ser combinado con cualquier ontología para definir el contenido semántico dentro de los `Fragments`, `Actions` o `Inputs` del modelo. Este conocimiento puede ser proporcionado en formato RDF dentro de la propiedad `semantic` de estos elementos.

Todo `PageState` se compone de `Fragments` y puede ser alcanzado por medio de la realización de acciones (representadas por `Action`) que pueden utilizar entradas (`Inputs`). Todos estos elementos poseen un contenido semántico que definen el propio contenido semántico del estado `PageState`. No obstante, este contenido también se ve influido por la semántica de todas las transiciones que se han seguido y de todos los estados por los que se ha pasado para alcanzar el estado actual. Este conocimiento puede ser transformado durante las transiciones, y nuevo conocimiento puede ser añadido o eliminado del conjunto de aserciones almacenadas en la memoria de trabajo del cliente, dependiendo de las interacciones del cliente con el sitio de la Web Profunda. Es, por tanto, un sistema similar a los denominados Basados en Reglas o Sistemas de Producción en que el conocimiento se va modificando según las acciones anteriores.

Como se puede ver en la figura 2, `Fragment`, `Input` y `Action` han definido la propiedad `semantic` que indica qué conocimiento (representado como triplas RDF o reglas de inferencia) es añadido (`ADD`) o borrado (`DEL`) de la memoria de trabajo. Para añadir una tripla RDF sólo es necesario añadirla directamente a la memoria de trabajo, mientras que añadir una regla lógica en RDF requiere de la ejecución de dicha regla sobre todas las triplas de la memoria de trabajo cada vez que nuevas triplas RDF son añadidas. El hecho de borrar una tripla o una regla simplemente la elimina de la memoria de trabajo.

¹⁰ <http://www.w3.org/TR/owl-features/>

El efecto de añadir o borrar datos RDF de la memoria de trabajo del cliente sigue las siguientes reglas:

1. Si el cliente se encuentra sobre un estado `PageState` que contiene fragmentos, la propiedad `semantic` de los fragmentos es procesada dentro de la memoria de trabajo siguiendo la propiedad `fragmentOf` que los enlaza entre sí por medio de una jerarquía. Los fragmentos que se encuentran en el mismo nivel de jerarquía pueden ser procesados en cualquier orden (por ejemplo, todos los fragmentos que se encuentran directamente enlazados con un `PageState` se pueden procesar en cualquier orden).
2. Si el cliente ha viajado a través de una transición `Transition` que contiene una lista ordenada de acciones, la propiedad `semantic` de las acciones es procesada siguiendo el orden en que se definió la secuencia de acciones.
3. Si el cliente utiliza una entrada definida en un fragmento de un estado `PageState` o en una acción de una transición `Transition`, la propiedad semántica de la entrada es procesada después de todas las otras propiedades semánticas del resto de `PageStates` y `Transitions`.

La información semántica se encuentra asociada a cada instancia de las clases `Fragment`, `Action` e `Input` por medio de la utilización de la propiedad `semantic`. El usuario que está anotando un sitio Web indica qué tipo de información representa el fragmento, acción o entrada por medio de esta propiedad. Su contenido está expresado en RDF.

Para definir información semántica, en la figura 2 utilizamos las siguientes propiedades y conceptos basados en las funciones disponibles para el razonador CWM¹¹.

Con el objeto de facilitar la compartición y reutilización de anotaciones, es necesario el uso de ontologías para el modelado de este contenido semántico.

4. Extracción de Datos utilizando Anotaciones

El contenido de la propiedad `semantic` permite localizar una información particular dentro del mapa de navegación para llevar a cabo una consulta que especifica las condiciones en las cuales la información puede ser localizada dentro de la memoria de trabajo. Como nuestro modelo trata con conocimiento expresado por medio de triplas RDF, hemos seleccionado SPARQL¹² como lenguaje de consulta. SPARQL es un lenguaje que permite indicar el *Named Graph*[4] al que un conjunto de condiciones de la consulta se refiere. Hacemos uso de esta funcionalidad para relacionar anotaciones de diferentes sitios de la Web Profunda con el objeto de llevar a cabo una consulta distribuida entre sus grafos de navegación anotados. La figura 3 muestra un ejemplo de consulta que utiliza dos grafos diferentes. La consulta SPARQL requiere el precio de un libro que cumple las siguientes condiciones expresadas en la parte `WHERE` de la consulta:

¹¹ <http://www.w3.org/2000/10/swap/doc/CwmBuiltins.html>

¹² <http://www.w3.org/TR/rdf-sparql-query/>

1. Seleccionar desde un RSS cualquier elemento identificado como `Product` que tenga definido un título.
2. Seleccionar desde el sitio web de la tienda de libros cualquier elemento identificado como `Product` que haya sido buscado en el sitio web utilizando la cadena que representa el título del producto obtenida desde el RSS previamente accedido.

Un conjunto de condiciones expresadas por una consulta SPARQL puede ser satisfecho por una lista de transiciones. Las acciones asociadas a estas transiciones pueden requerir que el cliente proporcione determinadas entradas. Estas entradas a su vez pueden ser necesarias para acceder a fragmentos específicos dentro de un estado. En el ejemplo de la figura 3, el mapa RSS requiere una entrada llamada `INDEX` para acceder a un fragmento dentro de `rss.state0`, y el grafo de la tienda de libros requiere de la entrada `SEARCHED_PRODUCT` para llevar a cabo las acciones de `transition0`. Las `Inputs` son los puntos en los que la consulta SPARQL puede relacionar datos entre diferentes mapas de navegación. Debido a que las entradas requieren que se suministre cierta información, la consulta SPARQL puede enfrentarse a las siguientes posibles situaciones:

- Si las condiciones de la consulta especifican el valor de una entrada, entonces utilizar dicho valor.
- Si las condiciones indican que la información requerida por la entrada debe ser obtenida de otro grafo, entonces la consulta debe primero llevar a cabo sus acciones en el otro grafo. Esto ocurre en el ejemplo con el mapa de la tienda de libros en la condición `?product2 shop:searchedBy ?title`, ya que el título es referenciado por el mapa RSS.
- Si la información requerida no puede ser inferida, entonces la consulta SPARQL debe utilizar todas las posibles informaciones que podrían ser utilizadas para la entrada. En el ejemplo esto ocurre con la entrada `INDEX`, que no es proporcionada, con lo que la consulta debe iterar entre todos los posibles valores de entrada.

Hemos desarrollado un sistema demo que puede ser accedido a través de [11]. Nuestra demo genera scripts de Wrappers Web compuestos de comandos *Chickenfoot*¹³ que pueden ser ejecutados en un navegador *Firefox* que tenga instalado el plugin de *Chickenfoot*. *Chickenfoot* [2] es una extensión del lenguaje Javascript que proporciona un conjunto de funciones especiales para la realización de tareas Web tales como realizar clic en un enlace, introducir datos en un formulario HTML, etc.

Los scripts *Chickenfoot* son generados a partir de consultas SPARQL utilizando las anotaciones expresadas en la formalización que hemos presentado en este artículo. Esta formalización está accesible en OWL en la URL [12]. Aunque con SPARQL no podemos definir tareas demasiado complejas, pensamos que es un buen punto de partida para definir lenguajes más potentes que puedan ser utilizados en un desarrollo de Wrappers Web orientado a semántica.

¹³ <http://groups.csail.mit.edu/uid/chickenfoot/>

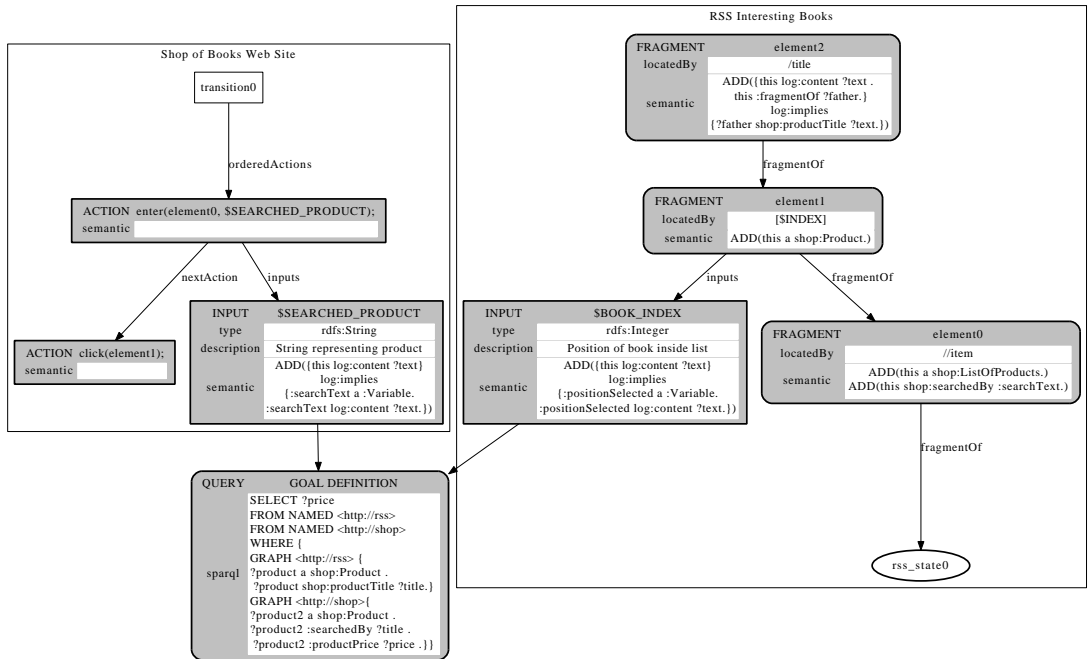


Figura 3. Ejemplo de realización de consultas entre mapas diferentes

5. Trabajos Futuros

Acceder e integrar datos procedentes de fuentes Web heterogéneas no estructuradas es un problema que actualmente se resuelve por medio de la realización de Wrappers Web a pesar de sus carencias. Los Wrappers Web son herramientas que pueden ser utilizadas para estructurar la información obtenida de fuentes no estructuradas y ponerla a disposición de la Web de Datos. Pensamos que el desarrollo de wrappers debe ser adaptado a este nuevo entorno de datos enlazados utilizando definiciones de Wrappers orientados a semántica, e implementaciones no enfocadas al sitio Web concreto. Para lograr esto, estamos interesados en la investigación de lenguajes más potentes y herramientas de desarrollo para esta nueva aproximación de desarrollo de Wrappers semánticos.

Estamos interesados en continuar trabajando en la integración de ontologías con las implementaciones de Wrappers semánticos con el objeto de aprovechar los conceptos comunes entre diferentes sitios web. La misma implementación de Wrapper semántico puede ser utilizada en cualquier sitio web que ha sido anotado utilizando la misma ontología.

6. Agradecimientos

Este trabajo ha sido parcialmente subvencionado por el Ministerio de Educación y Ciencia, proyecto ITACA No.TSI2007-65393-C02-01.

Referencias

1. Jeffrey P. Bigham, Anna C. Cavender, Ryan S. Kaminsky, Craig M. Prince, and Tyler S. Robison. Transcendence: enabling a personal view of the deep web. In *IUI '08: Proceedings of the 13th international conference on Intelligent user interfaces*, pages 169–178, New York, NY, USA, 2008. ACM.
2. Michael Bolin and Robert C. Miller. Naming page elements in end-user web automation. *SIGSOFT Softw. Eng. Notes*, 30(4):1–5, 2005.
3. Michael Bolin, Matthew Webber, Philip Rha, Tom Wilson, and Robert C. Miller. Automation and customization of rendered web pages. In *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 163–172, New York, NY, USA, 2005. ACM.
4. Jeremy J. Carroll, Christian Bizer, Pat Hayes, and Patrick Stickler. Named graphs, provenance and trust. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 613–622, New York, NY, USA, 2005. ACM.
5. Sudarshan Chawathe, Hector Garcia-molina, Joachim Hammer, Kelly Irel, Yannis Papakonstantinou, Jeffrey Ullman, and Jennifer Widom. The tsimmis project: Integration of heterogeneous information sources. In *In Proceedings of IPSJ Conference*, pages 7–18, 1994.
6. Darris Hupp and Robert C. Miller. Smart bookmarks: automatic retroactive macro recording on the web. In *UIST '07: Proceedings of the 20th annual ACM symposium on User interface software and technology*, pages 81–90, New York, NY, USA, 2007. ACM.
7. David Huynh, Stefano Mazzocchi, and David Karger. *Piggy Bank: Experience the Semantic Web inside your web browser*, volume 5, pages 16–27. Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, 2007.
8. Rohit Khare and Tantek Celik. Microformats: a pragmatic path to the semantic web. pages 865–866, 2006.
9. W3C. Rdfa primer. W3C Working Group Note 14 October 2008, October 2008.
10. Yang Wang and Thomas Hornung. Deep web navigation by example. In Tomasz Kaczmarek Marek Kowalkiewicz Tadhg Nagle Jonny Parkes Dominik Flejter, Slawomir Grzonkowski, editor, *BIS 2008 Workshop Proceedings, Innsbruck, Austria, 6-7 May 2008*, pages 131–140. Department of Information Systems, Pozna, University of Economics, 2008.
11. WebTlab. Site annotation demo. <http://corelli.gast.it.uc3m.es/siteannotation>.
12. WebTlab. Site annotation ontology. <http://corelli.gast.it.uc3m.es/siteannotation/ontology.owl>.
13. Jeffrey Wong and Jason I. Hong. Making mashups with marmite: towards end-user programming for the web. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1435–1444, New York, NY, USA, 2007. ACM.