Retrieving and Semantically Integrating Heterogeneous Data from the Web

Martin Michalowski, José Luis Ambite, Snehal Thakkar, and Rattapoom Tuchinda, *University of Southern California*

Craig A. Knoblock, *University of Southern California and Fetch Technologies*

Steve Minton, *Fetch Technologies*

Building Finder uses Semantic Web technologies to integrate different data types from various online data sources. The application's use of the RDF and RDF Data Query Language makes it usable by computer agents as well as human users.

The Semantic Web promises seamless integration of heterogeneous data from distributed sources, letting agents (human users or automated programs) perform sophisticated and detailed analyses of this data. An agent would send a query, expressed in terms of its preferred ontology (schema), to a system that would then find and integrate the relevant data from multiple sources and return it using the agent's ontology.

Before achieving this vision, however, we must address several challenges. We need technologies to integrate data described in different ontologies, for example, as well as different types of data, such as images or structured data. In addition, a Semantic Web-based system must recognize when different objects at different sites denote the same real-world entity. Other challenges include efficiently querying distributed information and converting legacy data in traditional databases and Web sites (HTML) into more semantic representations such as RDF.

Building Finder is a running application that showcases our approach to meeting these challenges. The application integrates satellite imagery, geospatial data, and structured and semistructured data from various online data sources using Semantic Web technologies. Users can query an integrated view of these sources and request Building Finder to accurately superimpose buildings and streets obtained from various sources on satellite imagery. The data sources integrated by Building Finder are heterogeneous not only in terms of the data, but also in terms of how the application accesses the sources.

Building Finder overview

Building Finder helps users obtain satellite imagery, street information, and building information about an area. Users can request that Building Finder superimpose the building information on the satellite imagery, or they can click on a particular building or house to obtain detailed information from property tax records or a white page directory. Similarly, Building Finder can superimpose street information or provide detailed information about a house. Building Finder accesses satellite imagery from Microsoft's Web service TerraService (http://terraserver-usa.com); streets from the US Census Bureau's Tigerline files, which are available as a database hosted at the University of Southern California; property tax information from the Los Angeles County Assessor's property tax information Web site; and residence information from the Yahoo White Pages Web site (http://people.yahoo.com).

What makes Building Finder even more attractive is that users can navigate through the Building Finder interface manually or have agents query the application using RDF Data Query Language (RDQL) queries and obtain results in RDF.

As Figure 1 shows, our GUI consists of an input form and an image. The input form lets users specify attributes the system will use to formulate RDQL queries and retrieve the URL as well as additional

1094-7167/04/$20.00 © 2004 IEEE
Published by the IEEE Computer Society

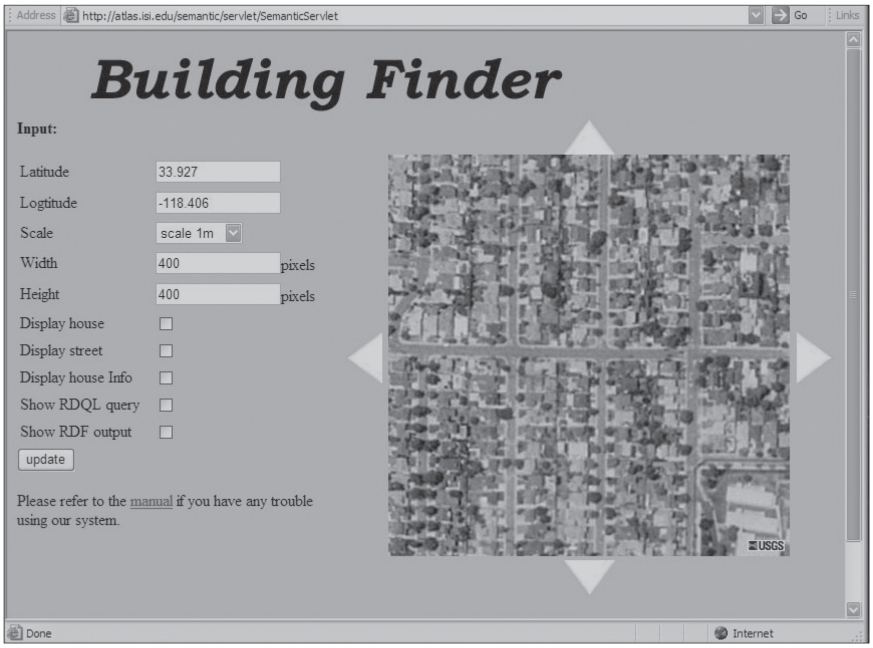
information, which the GUI displays. In the figure, the image on the right represents a satellite picture with middle point at coordinates (33.927, –118.406). The picture is 400 × 400 pixels, and each pixel equals 1 meter on the ground.

Unfortunately, Tigerline's street information is often inaccurate and rarely aligns with the streets on the TerraService satellite imagery. Aligning two geospatial data sets is a difficult problem often referred to as *conflation*.[1] For the Building Finder application, we obtain conflated street information from the Tigerline files for the city of El Segundo, Calif., using our localized image-processing approach for automatically aligning vector data with satellite imagery.[2] Although Building Finder's coverage is currently limited to El Segundo, it would be trivial to use this approach in the future to obtain conflated Tigerline files for all of the US.

There are three ways to use the application's GUI:

- The user retrieves a specific image by filling in its coordinates and clicking "update."
- The user navigates to nearby images by clicking one of four white arrows around the image.
- The user clicks on the box on the image (see Figure 2) that corresponds to a specific house the user wants information (owner and address) about.

The GUI in Figure 2 shows the results of running the user query. The boxes in the satellite image indicate building locations, and the lines represent streets.


Address http://atlas.isi.edu/semantic/servlet/SemanticServlet
Go
Links
Building Finder
Input:
update
Done
Internet
USGS

Latitude 33.927
Logtitude -118.406
Scale scale 1m
Width 400 pixels
Height 400 pixels
Display house
Display street
Display house Info
Show RDQL query
Show RDF output

Please refer to the manual if you have any trouble using our system.

Figure 1. The Building Finder GUI.

Technologies for efficient data interaction

To efficiently integrate semantically heterogeneous information from multiple data sources, Building Finder uses several technologies:

- Machine-learning techniques for converting traditional legacy Web sources and databases into Web services[3]
- A record linkage system for integrating data from multiple sources referring to a single entity
- A mediator system providing uniform access to data from various Web services
- An efficient execution system for information-gathering agents
- RDQL and RDF formalisms for representing queries and query results

Online source wrapping

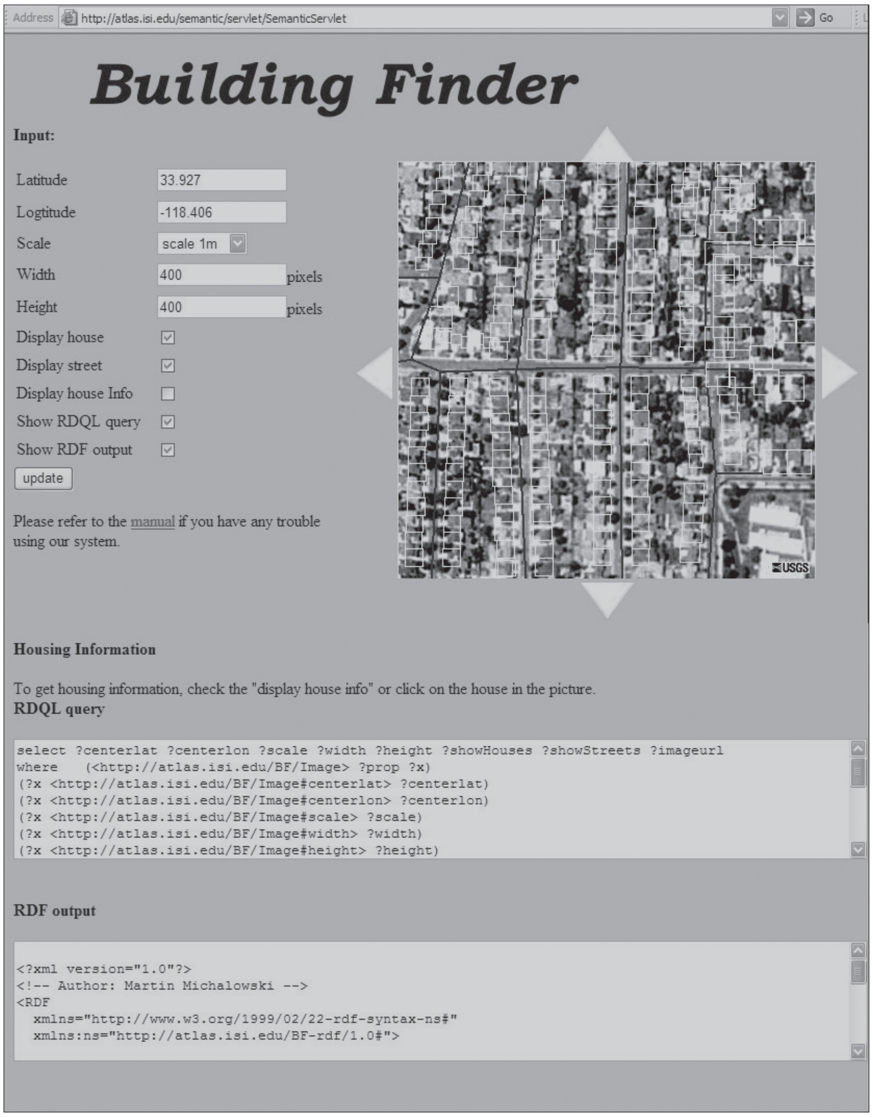
Much information on the Web is formatted for human readers, not machines. Software *wrappers* let programs or agents retrieve and translate data from Web sources into a format the software can easily manipulate. Building Finder relies on wrappers to provide the needed data on a per-query basis because the breadth and depth of queries prevents us from storing or caching all data locally.

Figure 3 shows how the wrapper navigates and extracts information from the Yahoo White Pages. Given a name, a city, and a state, the wrapper queries the Yahoo site by binding inputs to the wrapper to the corresponding variables in the search form on that site. Yahoo White Pages returns the results page, which lists names, addresses, and phone numbers, and the wrapper uses extraction rules to extract the data. If Yahoo returns more than one results page, the wrapper extracts the data from all pages. Building Finder filters the data, formats it into a table form, and sends it to the mediator, which chooses how to integrate the data to answer queries formulated according to user inputs.

Because most Web pages are written in HTML, extracting data is a complicated task. HTML is a language with which Web browsers specify a Web page's format; it's not intended to help locate specific data on a page. HTML tags have no semantics with respect to a page's content. Consequently, a wrapper must learn how to extract each data key on each Web page by searching for an HTML tag pattern leading to the data. For example, if we want to extract the person's name in Figure 4, we might notice that the tag always preceding the name data is `<html>Name:<b>` and the tag following the name data is `</b>`. Thus, we would create a rule for the wrapper to extract the keyword "A Smith" by starting extraction after the tag `<html>Name:<b>` and stopping at the tag `</b>`. This process, called *wrapper rule learning*, uses machine-learning techniques[4–6] for extracting data automatically from semistructured Web pages. We used the Fetch Agent Platform (www.fetch.com) to create the wrappers used to extract data used in Building Finder.

Record linkage

Integrating information from multiple Web sites is complicated because the same data objects can exist in inconsistent text formats across sites or because a search for a particular object can return multiple results. This makes identifying matching objects using an exact text match difficult. As Figure 5 shows, for a particular people query, the Yahoo White Pages site returns a record for Brandy Smith. When we use this person's address to query the property tax site, we receive more than 15 results. We must therefore use more than simple text-matching


Address http://atlas.isi.edu/semantic/servlet/SemanticServlet
Go
USGS

Building Finder

Input:

Latitude	33.927
Logtitude	-118.406
Scale	scale 1m
Width	400 pixels
Height	400 pixels
Display house	✓
Display street	✓
Display house Info	☐
Show RDQL query	✓
Show RDF output	✓

update

Please refer to the manual if you have any trouble using our system.

Housing Information

To get housing information, check the "display house info" or click on the house in the picture.

RDQL query

```
select ?centerlat ?centerlon ?scale ?width ?height ?showHouses ?showStreets ?imageurl
where   (<http://atlas.isi.edu/BF/Image> ?prop ?x)
(?x <http://atlas.isi.edu/BF/Image#centerlat> ?centerlat)
(?x <http://atlas.isi.edu/BF/Image#centerlon> ?centerlon)
(?x <http://atlas.isi.edu/BF/Image#scale> ?scale)
(?x <http://atlas.isi.edu/BF/Image#width> ?width)
(?x <http://atlas.isi.edu/BF/Image#height> ?height)
```

RDF output

```
<?xml version="1.0"?>
<!-- Author: Martin Michalowski -->
<RDF
  xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:ns="http://atlas.isi.edu/BF-rdf/1.0#">
```

Figure 2. RDF Data Query Language (RDQL) query and RDF output for the corresponding form input.

techniques to determine which record refers to the White Pages record of interest.

Active Atlas,[7] our record-linkage system, compares objects' shared attributes to identify matching objects. Some attributes are more important in determining whether a mapping should exist between two objects. Previous object-identification methods required manually constructing object identification or mapping rules for determining mappings between objects. This manual process is time-consuming and error-prone. Through limited user input, Active Atlas learns to tailor mapping rules to a specific application domain. In a single site, entities (people, places, countries, companies, and so on) are usually named in a consistent fashion. Across sites, however, the same entities might have different names. Other researchers have also developed approaches for solving the record-linkage problem.[8]

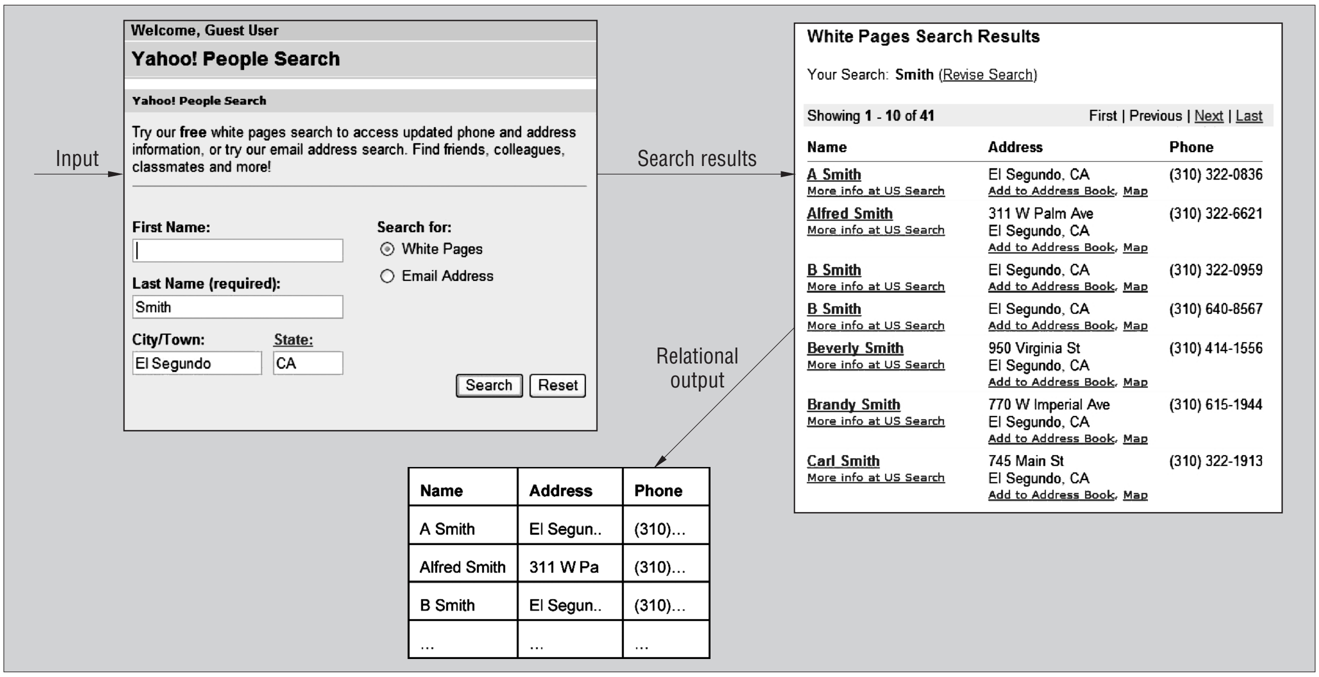
We begin by selecting a primary source for an entity's name and then provide a mapping from that source to every other source that uses a different naming scheme. One way to do this is to create a mapping table that specifies, for each entry in a data source, the name of the equivalent entity in another data source. Alternatively, if the mapping is computable, we can represent it using a mapping function—that is, a program that converts one form into the other form. We've developed a semi-automatic method for building mapping tables and functions by analyzing the underlying data in advance. The method attempts to pair each entity in one source with a corresponding entity (or entities) in another source. Essentially, it uses information-retrieval techniques (wrappers) to provide an initial mapping, and then applies machine-learning techniques to improve the mapping.

The initial mapping matches entities from two sources on the basis of their textual similarity. In the subsequent learning phase, the system learns two types of rules to help improve or verify the initial mapping. Transformation weights identify textual transformations such as acronyms, abbreviations, and phrase orderings common in the domain. For instance, the system can learn that "Rep" is commonly used as an abbreviation for "Republic," that one source typically uses acronyms, or that one source represents person names as "LastName, FirstName," whereas another uses "FirstName LastName." The system also learns mapping rules for comparing entities along multiple attributes.

We've used the underlying technologies developed for Active Atlas to augment Theseus,[9] our execution platform for information agents (discussed later), with the ability to consolidate information referring to the same entity from two different data sources. It performs this linkage in a streaming, dataflow-style execution that's both fast and efficient.

In Building Finder, the mediator must consolidate data from various sources—the property tax and Yahoo White Pages sites—referring to the same house or street. Using Theseus as the underlying execution engine gives the mediator access to the consolidation operator. The availability of such an operator lets the mediator replace all `join` instances in its generated plans with a consolidation step. This step performs a "smart join" between the two sites. In using multiple attributes present in each site, the consolidation operator links records referring to the same entity and thus integrates data more accurately.

The adaptive nature of the technologies used for consolidation makes Building Finder flexible and facilitates insertion of new data sources. To expand application support for areas beyond El Segundo, we would provide the mediator with access to Tigerline data on additional counties in California or the US. To integrate these new sources, the system would simply undergo a learning process to generate new mapping rules corresponding to each unique site. This is much easier than tailoring the entire application to



Welcome, Guest User

Yahoo! People Search

Yahoo! People Search

Try our **free** white pages search to access updated phone and address information, or try our email address search. Find friends, colleagues, classmates and more!

First Name:

Last Name (required): Smith

City/Town: El Segundo

State: CA

Search for: [✓ White Pages] Email Address

Search Reset

White Pages Search Results

Your Search: **Smith** (Revise Search)

Showing **1 - 10 of 41** First | Previous | Next | Last

Name	Address	Phone
A Smith More info at US Search	El Segundo, CA Add to Address Book, Map	(310) 322-0836
Alfred Smith More info at US Search	311 W Palm Ave El Segundo, CA Add to Address Book, Map	(310) 322-6621
B Smith More info at US Search	El Segundo, CA Add to Address Book, Map	(310) 322-0959
B Smith More info at US Search	El Segundo, CA Add to Address Book, Map	(310) 640-8567
Beverly Smith More info at US Search	950 Virginia St El Segundo, CA Add to Address Book, Map	(310) 414-1556
Brandy Smith More info at US Search	770 W Imperial Ave El Segundo, CA Add to Address Book, Map	(310) 615-1944
Carl Smith More info at US Search	745 Main St El Segundo, CA Add to Address Book, Map	(310) 322-1913

Name	Address	Phone
A Smith	El Segun..	(310)...
Alfred Smith	311 W Pa	(310)...
B Smith	El Segun..	(310)...
...	...	...

Figure 3. Extracting data from the Yahoo White Pages.

work with and understand the new sites.

We are also researching methods for improving record linkage using secondary sources.[10] In this research, we present an approach to accurately and automatically consolidate data from various data sources using a state-of-the-art object consolidation system in conjunction with a mediator system. The mediator system automatically determines which secondary sources to query if the object-consolidation system can't confidently determine whether two records refer to the same entity. In turn, the object-consolidation system uses this additional information to improve the accuracy of the data set consolidation. For example, the system could use a secondary source providing additional information on geographic coordinates for a given address (a geocoder) to improve the consolidation of data from the property tax and Yahoo White Pages sites. With the system developed by Martin Michalowski, Snehal Thakkar, and Craig A. Knoblock,[10] for example, we could better link people to properties based on addresses using geocoder information. Such an approach would reduce uncertainty when classifying two addresses as being the same.

```
<html>Name:<b>A Smith</b>Address:<b>El Segundo, CA</b>...
```

```
<html>Name:<b>Alfred Smith</b>Address: <b>311 W Palm Ave El Segundo, CA</b>...
```

Figure 4. A mapping pattern in HTML pages.

Mediator

The mediator system's goal is to provide unified access to various data sources. Building Finder uses the Prometheus mediator[11] to integrate satellite imagery from Microsoft TerraService, road vector data from US Census Tigerline files, and house information from white page directories. The user either sends a query through the user interface, as Figure 6 shows, or submits an RDQL query. The mediator passes the results back to the user interface.

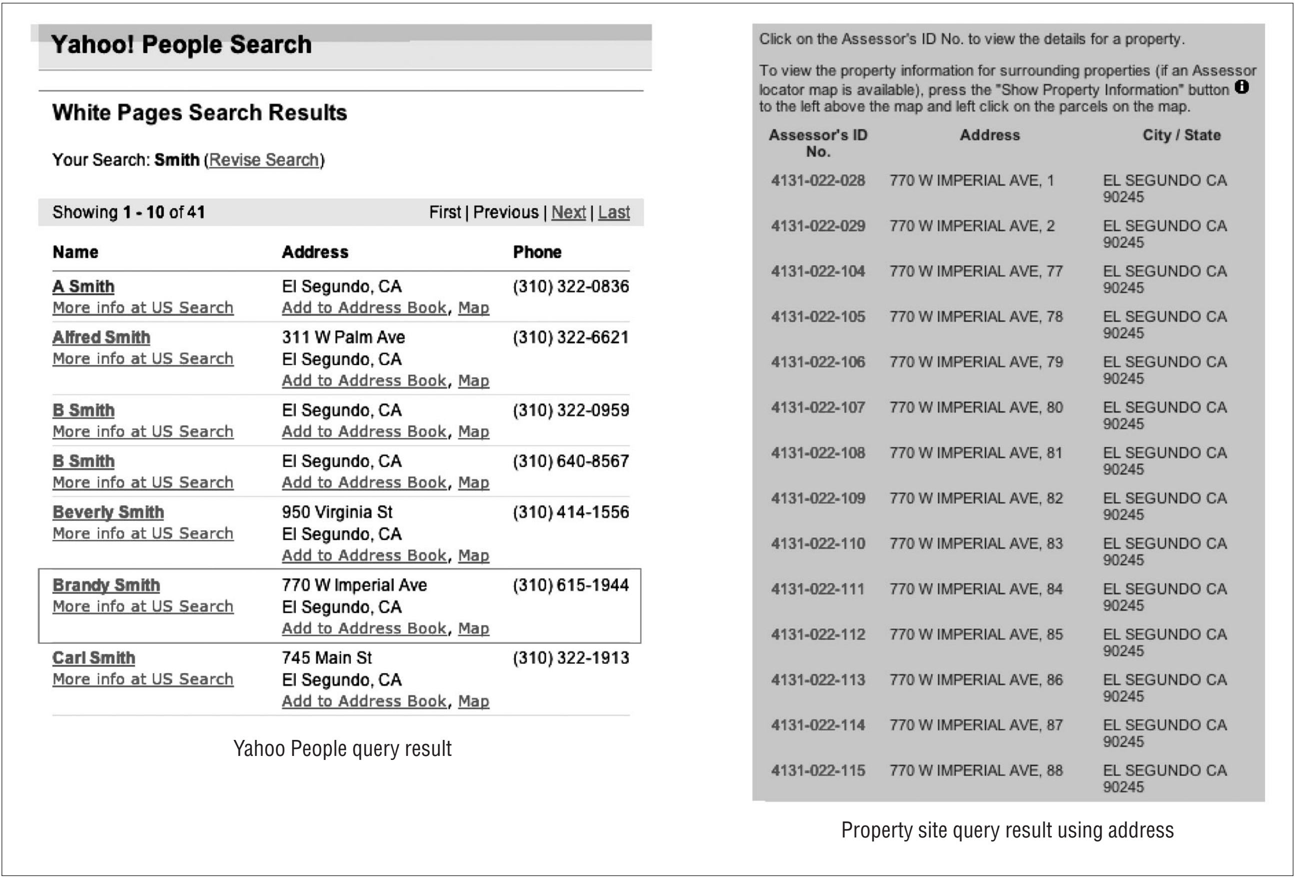
As Figure 7 shows, the Prometheus mediator has three parts:

- A data model
- A query reformulation component
- A query execution component using the Datalog to Theseus converter

The data model consists of information about various data sources as well as relationships between the data sources and domain predicates. Building Finder uses the domain predicates *street*, *image*, and *building*. Users can send queries in terms of these domain predicates.

Figure 8 shows the Building Finder application domain model. The domain predicate *building* is the consolidation of information from the Los Angeles County property tax data source and the white pages data source. The white pages data source provides the person's name, address, and phone number, while the property tax source provides the person's name and address and the property's lot size. The white pages data source gives information only for people listed in the directory. In the future, we'll add new data sources to the domain model to cover the entire continental US.

The mediator accepts queries on any combination of domain predicates. On receiving a query, the mediator merges it with the domain model to generate a datalog program that can answer the user query. The mediator then executes the generated program to find the results of the user query using the Theseus execution engine. For example, a



Yahoo! People Search

White Pages Search Results

Your Search: **Smith** (Revise Search)

Showing **1 - 10** of **41** First | Previous | Next | Last

Name	Address	Phone
A Smith More info at US Search	El Segundo, CA Add to Address Book, Map	(310) 322-0836
Alfred Smith More info at US Search	311 W Palm Ave El Segundo, CA Add to Address Book, Map	(310) 322-6621
B Smith More info at US Search	El Segundo, CA Add to Address Book, Map	(310) 322-0959
B Smith More info at US Search	El Segundo, CA Add to Address Book, Map	(310) 640-8567
Beverly Smith More info at US Search	950 Virginia St El Segundo, CA Add to Address Book, Map	(310) 414-1556
Brandy Smith More info at US Search	770 W Imperial Ave El Segundo, CA Add to Address Book, Map	(310) 615-1944
Carl Smith More info at US Search	745 Main St El Segundo, CA Add to Address Book, Map	(310) 322-1913

Yahoo People query result

Click on the Assessor's ID No. to view the details for a property.

To view the property information for surrounding properties (if an Assessor locator map is available), press the "Show Property Information" button to the left above the map and left click on the parcels on the map.

Assessor's ID No.	Address	City / State
4131-022-028	770 W IMPERIAL AVE, 1	EL SEGUNDO CA 90245
4131-022-029	770 W IMPERIAL AVE, 2	EL SEGUNDO CA 90245
4131-022-104	770 W IMPERIAL AVE, 77	EL SEGUNDO CA 90245
4131-022-105	770 W IMPERIAL AVE, 78	EL SEGUNDO CA 90245
4131-022-106	770 W IMPERIAL AVE, 79	EL SEGUNDO CA 90245
4131-022-107	770 W IMPERIAL AVE, 80	EL SEGUNDO CA 90245
4131-022-108	770 W IMPERIAL AVE, 81	EL SEGUNDO CA 90245
4131-022-109	770 W IMPERIAL AVE, 82	EL SEGUNDO CA 90245
4131-022-110	770 W IMPERIAL AVE, 83	EL SEGUNDO CA 90245
4131-022-111	770 W IMPERIAL AVE, 84	EL SEGUNDO CA 90245
4131-022-112	770 W IMPERIAL AVE, 85	EL SEGUNDO CA 90245
4131-022-113	770 W IMPERIAL AVE, 86	EL SEGUNDO CA 90245
4131-022-114	770 W IMPERIAL AVE, 87	EL SEGUNDO CA 90245
4131-022-115	770 W IMPERIAL AVE, 88	EL SEGUNDO CA 90245

Property site query result using address

Figure 5. Potential matches between the Yahoo White Pages and property tax sites.

user can send a request for information about houses in an area with coordinates (33.34, –118.54) to (33.37, –118.59). The datalog converter receives this request in RDQL format from the user interface and converts it to the following datalog query:

```
Q(name, address, phone, lat, lon) :-
        Houses(toplat, toplon, botlat, botlon,
                    name, address, phone, lat, lon) ^
        toplat = 33.37 ^ botlat = 33.34 ^
        toplon = -118.54 ^ botlon = -118.59.
```

When it receives this query from the datalog converter, the mediator uses its domain model to generate a Theseus plan to obtain the house information from the property tax and white pages Web sites. It then passes the generated plan to the Theseus execution engine.

Efficient query execution

A typical information-gathering plan might involve accessing and integrating data from many sources. However, unpredictable network latencies and varying remote source capabilities can significantly increase the total time required to execute such a plan. Moreover, many sources cannot be queried until a previous query has been answered. In most traditional approaches, integration plans consist of partial orderings of operators (that is, functions) and are executed based on the given order.

In Building Finder, Theseus[9] provides efficient execution of mediator-created information-gathering plans. Other methods of efficient query execution are also available.[12,13] We chose Theseus because it is a streaming dataflow execution system that uses both horizontal parallelism and vertical parallelism.

In horizontal parallelism, execution is decentralized, with independent operators executing at the same time. In vertical parallelism, operators "fire" when any part of an input data becomes available and output tuples to other operators in a pipeline fashion. Theseus has an array of operators geared toward information-gathering plans—for example, xwrapper executes a wrapper, xquery parses XML files using XML Query Language (XQuery), xmltorel converts an XML document into a relation, consolidate combines data using record linkage, and dbquery retrieves data from the database—in addition to relational algebra operations such as union, join, select, and project.

Figure 9 illustrates an information-gathering plan corresponding to the Building Finder query for information about houses (see the "Mediator" section). This plan retrieves data from the ptax and ywp wrappers, parses the XML output, translates the XML output into a relation, and consolidates and filters the data from the property tax Web site and Yahoo White Pages. The mediator sends Theseus an integration plan and all necessary inputs. Theseus uses the inputs to query the property tax and Yahoo White Pages Web sites in parallel. Because the area of interest can include more than one house, both Web sites might return several records. Theseus streams records from both data sources to the consolidate operator.

To put this in perspective, assume it takes

30 seconds to make a request to each Web site. After the initial request, obtaining one record from each data source takes five seconds. If both data sources return 10 records in a sequential implementation, obtaining records from both data sources would take 160 seconds; using Theseus, which queries sources in parallel, we can obtain the same number of records in 80 seconds. Furthermore, Theseus could return results for the first record from both sources in 40 seconds. Theseus further reduces total execution time because it streams resulting tuples from one operator in the plan to the next as soon as they are made available. Therefore, the Theseus execution engine greatly improves user query response time.

Semantic Web representation and query languages

RDQL is used for extracting information from RDF graphs. Our Building Finder application uses both formalisms. It presents queries to the application as RDQL queries, which the mediator subsequently processes and executes. The application uses an internal RDQL to datalog converter to interpret and process the query. On completing the query, the module converts the XML results constructed by the mediator to RDF and returns them to the user.

This type of approach makes Building Finder usable by both human users interacting with the GUI and computer agents using RDQL and RDF to interact with the application on behalf of human users. It also lets the application give meaning to the results and use semantics present in RDQL to interpret input queries. Building Finder can compose query results using personally developed ontologies describing buildings, images, and streets in conjunction with RDF. This lets computer agents further infer and reason about the results and use the additional semantic knowledge to process the results or pose more queries.

Advantages of our approach

Our mediator approach to data integration has several advantages over the data warehousing approach often used in Semantic Web applications (for example, RDF stores). Because the system accesses data only in response to user queries, the data is always fresh. Approaches that extract and store data locally are suitable for data that varies slowly but return stale values for sources that change faster than the warehouse update cycle.

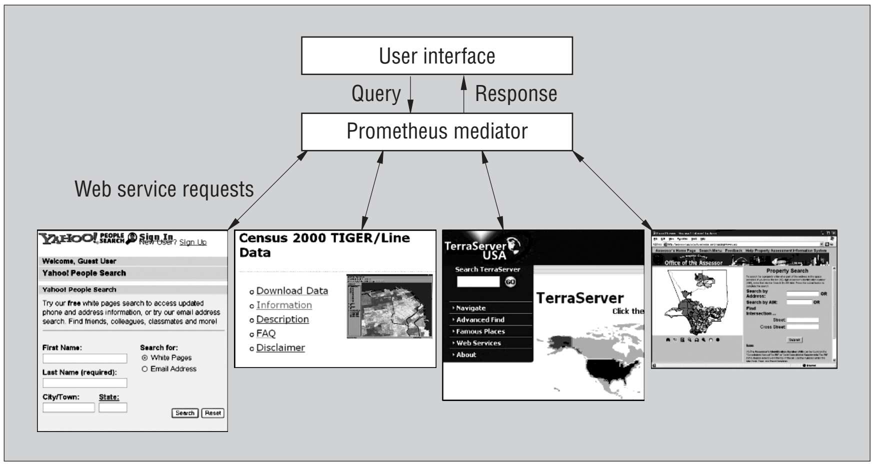


Figure 6. Mediator execution.

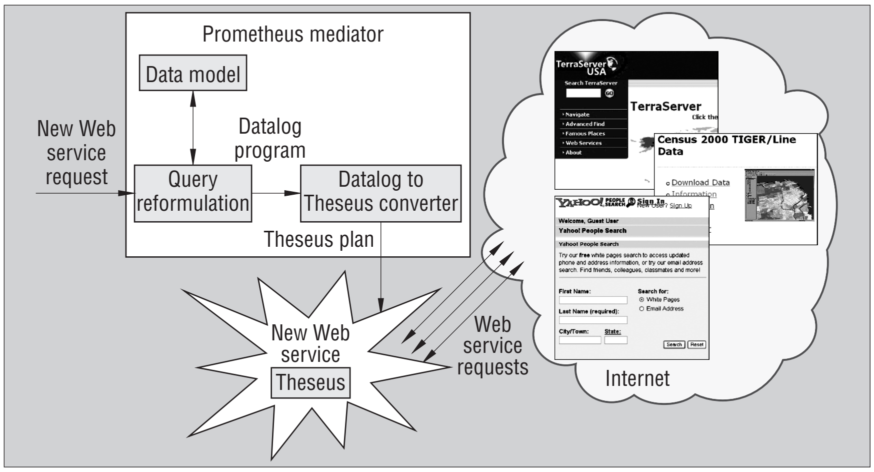


Figure 7. Mediator architecture.

```
Building(ctrlat, ctrlon, scale, width, height, name, address, city, state, phone, zip, lat, lon) :-
        findCorners(ctrlat, ctrlon, scale, width, height, botlat, toplat, botlon, toplon) ^
        whitepages(name, address, city, state, zip, phone) ^
        geocoder(address, city, state, zip, lat, lon) ^
        lat > botlat ^
        lat < toplat ^
        lon > botlon ^
        lon < toplon

Building(ctrlat, ctrlon, scale, width, height, "", address, city, state, "", zip, lat, lon) :-
        findCorners(ctrlat, ctrlon, scale, width, height, botlat, toplat, botlon, toplon) ^
        propertytax(address, city, state, zip, lotsize) ^
        geocoder(address, city, state, zip, lat, lon) ^
        lat > botlat ^
        lat < toplat ^
        lon > botlon ^
        lon < toplon

Street(ctrlat, ctrlon, scale, width, height, streetname, fraddl, toaddl, fraddr, toaddr, frlat, frlong, tolat, tolong) :-
        findCorners(ctrlat, ctrlon, scale, width, height, botlat, toplat, botlon, toplon) ^
        tigerlines(streetname, fraddl, toaddl, fraddr, toaddr, frlat, frlong, tolat, tolong) ^
        frlat > botlat ^
        frlat < toplat ^
        frlon > botlon ^
        frlon < toplon

Image(ctrlat, ctrlon, scale, width, height, showHouses, showStreets, imageurl) :-
        terraserverImage(ctrlat, ctrlon, scale, width, height, showHouses, showStreets, imageurl)
```

Figure 8. Building Finder domain model.

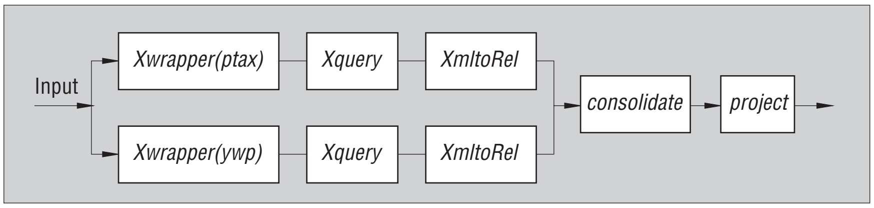


Figure 9. A plan to gather housing data used by the mediator.

Moreover, the data available on the Web will likely grow faster than a warehouse's ability to host it. The mediator only retrieves data relevant to user queries, making it more scalable than warehouse-based approaches.

Finally, the mediator approach is much more adaptable than a warehouse. Users will change their preferred worldview as their applications and data needs evolve and new sources become available. Changing the mediator's ontology and the source descriptions is significantly easier than rebuilding a warehouse.

The technologies showcased in the Building Finder have several important practical applications, such as disaster management and e-commerce. One of the most important tasks of a disaster-management system is finding information about the affected area, including street names, buildings, and residents. Building Finder can provide such information to Semantic Web-enabled disaster-management systems. Real estate applications could also use Building Finder to obtain information about properties and their surroundings.

We plan to explore several extensions to the core technologies of the Building Finder, such as richer models for the mediator. We currently integrate our sources using datalog rules, but we're interested in using object-oriented features such as reasoning about classes as is done in RDF or OWL. In addition, although we describe our integration model in a global-as-view style, we also want to use local-as-view models.[14] We also plan to integrate more RDF and RDQL sources as they become available.

Acknowledgments

This article is based on work supported by the US National Science Foundation under award number IIS-0324955; Darpa, through the Department of the Interior; NBC, Acquisition Services Division, under contract number NBCHD030010; the US Air Force Office of Scientific Research under grant numbers F49620-01-1-0053 and FA9550-04-1-0105; the US Air Force under contract number F49620-02-C-0103; and a gift from the Microsoft Corporation.

The US government is authorized to reproduce and distribute reports for governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of any of the above organizations or any person connected with them.

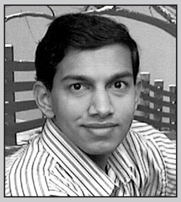
The Authors

Martin Michalowski is a PhD student in computer science at the University of Southern California. His research interests include information integration, record linkage, heuristic-based planning, and machine learning. He received his BS in computer engineering from the University of Minnesota and his MS in computer science from the University of Southern California. He can be reached at USC/Information Sciences Institute, 4676 Admiralty Way, Marina del Rey, CA 90292; martinm@isi.edu, www.isi.edu/~martinm.

José Luis Ambite is a senior research scientist at USC's Information Sciences Institute. His research interests include information integration, automated planning, databases, and knowledge representation, and his current focus is on automatic Web service composition. He received his PhD in computer science from USC. He is a member of the AAAI and ACM SIGMOD. He can be reached at USC/Information Sciences Institute, 4676 Admiralty Way, Marina del Rey, CA 90292; ambite@isi.edu, www.isi.edu/~ambite.

Craig A. Knoblock is a senior project leader at USC's Information Sciences Institute and a research associate professor in computer science. He is also the chief scientist for Fetch Technologies, a company commercializing some work developed at USC. His research interests include information agents, information integration, automated planning, machine learning, and constraint reasoning. He received his PhD in computer science from Carnegie Mellon University. He can be reached at USC/Information Sciences Institute, 4676 Admiralty Way, Marina del Rey, CA 90292; knoblock@isi.edu, www.isi.edu/~knoblock.

Steve Minton is the chief executive officer of Fetch Technologies. His research interests include machine learning, planning, and constraint satisfaction. He received his PhD in computer science from Carnegie Mellon University. Minton founded the *Journal of Artificial Intelligence Research* (JAIR) and served as its first executive editor. He has also served as an editor of the journal *Machine Learning*, and in 1998 he was elected to be a fellow of the American Association for Artificial Intelligence. He can be reached at Fetch Technologies, 2041 Rosecrans Ave. Suite 245, El Segundo, CA 90245; minton@fetch.com, www.fetch.com.


Snehal Thakkar is a PhD student in computer science at the University of Southern California. His research interests include information integration, automatic Web service composition, and geographical information systems. He received his MS in computer science from USC. He can be reached at USC/Information Sciences Institute, 4676 Admiralty Way, Marina del Rey, CA 90292; thakkar@isi.edu, www.isi.edu/~thakkar.

Rattapoom Tuchinda is a PhD student at USC and consultant for Hamlet, a company based on his research work on airfare prediction. His research interests include information integration, agent execution, mix-initiative planning, and machine learning. He received a BS in electrical engineering and computer science and an MS in engineering from the Massachusetts Institute of Technology. He can be reached at USC/Information Sciences Institute, 4676 Admiralty Way, Marina del Rey, CA 90292; pipet@isi.edu, www.isi.edu/~pipet.

References

1. A. Saalfeld, "Conflation: Automated Map Compilation," tech. report, Computer Vision Lab., Center for Automation Research, Univ. of Maryland, 1993.
2. C.-C. Chen et al., "Automatically Annotating and Integrating Spatial Datasets," *Proc. 8th Int'l Symp. Spatial and Temporal Databases*, LNCS 2750, Springer-Verlag, 2003, pp. 469–488.
3. I. Muslea, S. Minton, and C. Knoblock, "Hierarchical Wrapper Induction for Semistructured Information Sources," *J. Autonomous Agents and Multi-Agent Systems*, vol. 4, nos. 1–2, 2001, pp. 93–114.
4. C.-N. Hsu and M.-T. Dung, "Generating Finite-State Transducers for Semi-Structured Data Extraction from the Web," *Information Systems J.*, vol. 23, no. 8, 1998, pp. 521–538.
5. N. Kushmerick, "Wrapper Verification," *World Wide Web J.*, vol. 3, no. 2, 2000, pp. 79–94.
6. C.A. Knoblock et al., "Accurately and Reliably Extracting Data from the Web: A Machine Learning Approach," *Intelligent Exploration of the Web*, Springer-Verlag, 2003, pp. 275–287.
7. S. Tejada, C.A. Knoblock, and S. Minton, "Learning Domain-Independent String Transformation Weights for High Accuracy Object Identification," *Proc. 8th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining* (KDD-2002), ACM Press, 2002, pp. 350–359.
8. M. Bilenko and R.J. Mooney, "Adaptive Duplicate Detection Using Learnable String Similarity Measures," *Proc. 9th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining* (KDD 2003), ACM Press, 2003, pp. 39–48.
9. G. Barish and C.A. Knoblock, "An Expressive and Efficient Language for Information Gathering on the Web," *Proc. 6th Int'l Conf. AI Planning and Scheduling* (AIPS 2002) *Workshop: Is There Life Beyond Operator Sequencing?: Exploring Real-World Planning*, AAAI Press, 2002.
10. M. Michalowski, S. Thakkar, and C.A. Knoblock, "Exploiting Secondary Sources for Automatic Object Consolidation," *Proc. 2003 Knowledge Discovery and Data Mining, Workshop on Data Cleaning, Record Linkage, and Object Consolidation*, ACM Press, 2003, pp. 34–36.
11. S. Thakkar, J.L. Ambite, and C.A. Knoblock, "A View Integration Approach to Dynamic Composition of Web Services," *Proc. 2003 Int'l Conf. Automated Planning and Scheduling* (ICAPS), *Workshop on Planning for Web Services,* AAAI Press, 2003.
12. Z.G. Ives, A.Y. Halevy, and D.S. Weld, "An XML Query Engine for Network Bound Data," *VLDB J.*, vol. 11, no. 4, 2002, pp. 380–402.
13. J.F. Naughton et al., "The Niagara Internet Query System," *IEEE Data Eng. Bulletin*, vol. 24, no. 2, 2001, pp. 27–33.
14. A.Y. Levy, "Logic-Based Techniques in Data Integration," *Logic Based Artificial Intelligence,* J. Minker, ed., Kluwer Publishers, 2000, pp. 575–595.

For more information on this or any other computing topic, see our Digital Library at www.computer.org/publications/dlib.