Ontology-Based Integration of Knowledge from Semi-Structured Web Pages

James Masters
Cycorp
3721 Executive Center Dr. Ste. 100
Austin, TX 78731
01.512.514.2985
chip@cyc.com

Cynthia Matuszek
Cycorp
3721 Executive Center Dr. Ste. 100
Austin, TX 78731
01.512.342.4025
cynthia@cyc.com

Michael Witbrock
Cycorp
3721 Executive Center Dr. Ste. 100
Austin, TX 78731
01.512.342.4003
witbrock@cyc.com

ABSTRACT

In this paper, we describe an effort to use a very large knowledge base to provide the declarative underpinnings for a semantically rich, inferentially powerful approach to integrating web pages and other sources of semi-structured knowledge. We combine declarative wrappers around web page content with the Cyc Knowledge Base, which provides semantic information about the knowledge sources being mapped. While this makes the mapping process more complex, it provides richer inferential and querying power over the resulting combined information sources. We discuss the process of developing definitions of mapping in Cyc, and describe the long-term intent of the system, which we call SKSI.

Categories and Subject Descriptors

I.2.4: [**Knowledge Representation Formalisms and Methods**]

General Terms

Design, Experimentation

Keywords

Web integration, knowledge management, ontologies

1. INTRODUCTION

While the World Wide Web was originally conceived to meet the needs of a human audience, the explosion of information available on it has led to many research efforts to enable software agents to mine this huge source of structured, semi-structured, and unstructured data. Some effort has gone into mining the structure and arrangement of web documents [8], but—drawing from work on integrating multiple database knowledge sources via a *semantic mapping* layer [1]—we describe a mechanism for fully recovering the relational structure inherent in the knowledge underlying a web page from the page itself.

Semantic Knowledge Source Integration or SKSI [10], is a global-as-view [7] data integration technology that uses the Cyc[1] ontology as the global schema and enables mapping the local schemas of a general class of structured knowledge sources, including databases and web sites, into the representation of the Cyc knowledge base.[2] Once mapped to Cyc, the sources can be uniformly queried or modified using expressions in CycL, Cyc's knowledge representation language. [11] Meta-knowledge about a source's structure, and semantics that facilitate translating CycL into the source's native language, are also stored in Cyc, and comprise a conceptual model of the semantics of the source's data structures, syntactic organization, and its server's reasoning capabilities (if any).

While the technology has, so far been applied principally to integrating relational databases, SKSI is well suited for integrating the large and growing number of web sites that are front ends to databases and expose some limited form of querying capabilities, or sites which embed tabular information into the markup. Such sources have a relational structure to the data embedded in HTML document, and it makes sense to identify this structure and use it to model and extract the relevant data from the document. This sort of information is best described as *semi-structured*, and the current work owes much to others' work on capturing and mapping semi-structured knowledge. [12]

For example, the United States National Weather Service (among many other weather services) hosts a web site that provides current and forecast weather information for cities in the United States[3]. Given a reference to a US city in the form "City, ST" (where ST is the two letter abbreviation of a US State) the web site generates an HTML page with the weather conditions for that city. If the query processor cannot identify a known city from the input, an error page is generated. If the query processor cannot identify a unique city from the input, a disambiguation page is generated, permitting the user to select from the range of possible matches. For our initial work in integrating the web site, we limited our interest to extracting just the current weather conditions, which are grouped together inside of a single table of the result document. We extract the following information from the table: the input city, time of last update, weather description (cloudy, rainy, etc.), temperature, dew point, heat index (if any), wind chill (if any), barometer, humidity, visibility, wind speed, and wind direction.

The distinction between this work and other similar web mapping approaches, such as Ariadne [5], is that we use the Cyc Knowledge Base to provide *semantic* information about the knowledge sources being mapped. While this makes the mapping process more complex, it provides richer inferential and querying

[1] Cyc is a large-scale effort to create a Knowledge Base, or KB, of fundamental common-sense concepts and rules. [4] For more information, see http://cyc.com. Cyc available for research purposes at http://research.cyc.com.

[2] Here, a *structured knowledge source* is a data repository that is organized using an identifiable and consistent schema that is amenable to interpretation using a relational, object-oriented, or hierarchical conceptual model.

[3] http://nws.noaa.gov

power over the resulting combined information sources. [9] As an example, Cyc already contains a great many formally represented facts and rules about geography, politics, and geospatial reasoning. Once the NWS web page is wrapped with a wrapper that can communicate with Cyc, it is possible to query that combined interface about "the temperature of cities in Southern U.S. states." Even though no database explicitly containing that information has been mapped, the common-sense background knowledge in Cyc allows a fusion of the up-to-date knowledge available on the web and the political and geographical reasoning available in the Cyc KB. This allows the combined system to take advantage of pre-existing reasoning and querying capabilities of the Cyc inference engine.[4]

Once a source is integrated with the Cyc system, users can pose queries to Cyc that combine results from the web site with other integrated web sites, databases, or with the Cyc knowledge base. As another example, we have integrated the United States Geological Survey's geographic names database, which contains data about 1.9 millions places in the United States. Using both this source and the NWS site, Cyc can comprehensively answer questions that cannot be answered by either source individually, such as "Is it raining somewhere in New England?" The NWS web site does not know what cities are in New England, and the Cyc KB's knowledge is incomplete in this regard. However, Cyc does know that New England is comprised of Connecticut, Maine, Massachusetts, New Hampshire, Rhode Island, and Vermont, and the USGS database has a record for every city in every US state. The CycL version of this query is stated below:

```
(thereExists ?CITY
  (and
     (isa ?CITY USCity)
     (geographicallySubsumes
        NewEngland-USRegion ?CITY)
     (weather ?CITY RainyArea)))
```

The remainder of this article is organized as follows: we describe how the conceptual model of a web site is developed in Cyc, using the National Weather Service web site as the motivating example. We then describe how Cyc accesses the web site, requests URLs, and extracts relevant data from the result page. Finally, we describe plans for future work that will replace the brittle methods used to process web queries described in the preceding section with a robust and novel approach that uses Cyc's rule-based reasoning and machine learning capabilities to extract that data from documents. This will minimize or remove the need for human intervention, and can adapt to changes to the HTML structure over time.

2. SKSI

The SKSI technology uses a subset of the CycL knowledge representation language to build a conceptual model of a structured knowledge source. Once the conceptual model of a source's data content has been sufficiently completed, it is used to generate special code modules which can translate appropriate fragments of CycL queries into SQL, execute the queries on remote servers, retrieve the results, and translate them back into CycL. When the source is a database, the modules execute the SQL on the database's server. However, if the source is a web site, the modules execute the SQL query using a custom SQL query engine. Wrappers are used to extract the data from the result web page and pass it to the SQL query engine as a database tuple. The National Weather Service example will be used to illustrate development of a conceptual model of a source.

The model is comprised of three interrelated layers of knowledge about the source's structure and data content:

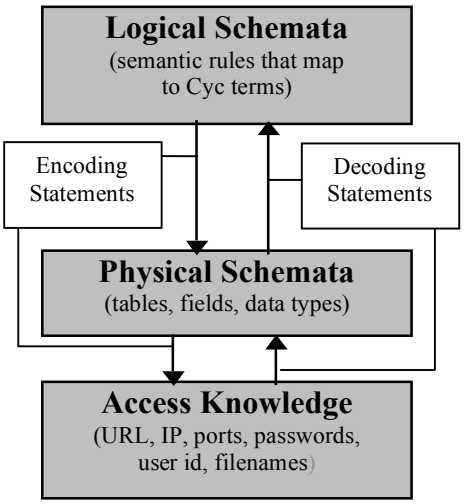

Figure 1. The three conceptual layers of the SKSI architecture.

2.1 The Access Knowledge Layer

The *access knowledge layer* contains the information needed to identify the source on the network, connect to it, and submit requests to it. For example, the primary pieces of information needed to submit a request to the NWS web site are the request URL and the request method (GET or POST). Other types of knowledge sources, such as databases or secure web sites that require log in, require more access knowledge to capture information such as user names, passwords, and flavors of SQL.

2.2 The Physical Schema Layer

The *physical schema layer* contains the information that describes the prima facie structure of how data is organized in the source. It is comprised of a set of *physical fields,* one field for each datum to be extracted from the source. A physical schema and its fields are close analogues to a relational table and its attributes or columns. Indeed, for database sources there is a one to one correspondence between database tables and their attributes, and physical schemas and their fields. All of the conceptual properties of a table and its attributes are reflected in its physical schema and fields, such as datatypes, primary key constraints, foreign key constraints, and other constraints.

To construct the physical schema of a web site, we model the data we wish to extract from the site as if it were a table or set of tables in a database. We then identify the datatypes of the data elements that appear in the web page and identify the constraints between fields that are enforced or implied by the web site's query interface. For example, Table 1 lists the physical fields that correspond to the National Weather Service web site with their datatypes and constraints.

[4] A difficulty with this approach is that the Cyc inference engine is a very large, modularized blackboard system, and as such, reasoning over combined queries of this sort is not readily described by a simple a

Table 1. Physical Fields, Types, and Constraints for the NWS web site.

Field Name	Data Type	Constraint
"zipcity"	Varchar	Primary key
"lastupdate"	Timestamp with timezone	Not null
"weather"	Varchar	Not null
"temperature"	Smallint	Not null
"dewpoint"	Smallint	Not null
"heatindex"	Smallint	
"windchill"	Smallint	
"barometer"	Float	Not null
"humidity"	Smallint	Not null
"visibility"	Float	Not null
"windspeed"	Smallint	Not null
"winddir"	Varchar	

An important constraint is enforced by the web site's query interface that is not stated in the table, and is typically not encountered when dealing with databases directly. That is, in order to retrieve a weather conditions page from the web site, one must provide the name of a city. That is, the value of "zipcity" must be specified. It is not possible to use the web site to, for example, request all cities where a certain weather condition is currently true. So we state in Cyc that the "zipcity" field is a *required field*.

2.3 The Logical Schema Layer

The *logical schema layer* interprets the meaning or semantics of the source's data in terms of the Cyc ontology. As with physical schemas, a logical schema is comprised of a set of *logical fields*. Whereas the physical schemas and fields model the explicit, literal structure of the source data, the logical schemas and fields interpret the meaning of the data, expressed in terms of the Cyc ontology. Each logical field is associated with a *collection* or *type* in the ontology, and logical fields are combined together with predicates in the ontology to form templates called *meaning sentences*. It is the meaning sentences that most fully express the formal interpretation of the source's data content in the Cyc system. Table 2 shows the logical field types and meaning sentence predicates for each of the data elements associated with the National Weather Service web site.

Table 2. Logical Field Types and Meaning Sentence Predicates for the NWS web site

Field Name	CycL Type	Cyc Predicate
"zipcity"	USCity	
"lastupdate"	Date	
"weather"	OutdoorLocation TypeByWeather	weather
"temperature"	Temperature	ambientTemperature
"dewpoint"	Temperature	ambientDewpoint
"heatindex"	Temperature	ambientHeatIndex
"windchill"	Temperature	ambientWindChill
"barometer"	Pressure	ambientPressure
"humidity"	RelativeHumidity	ambientRelativeHumidity
"visibility"	Distance	ambientVisibilityDistance
"windspeed"	Speed	ambientWindSpeed
"winddir"	Terrestrial-Direction	ambientWindDirection

2.4 Transformation Rules

The physical and logical schema layers are used during query processing to translate between the literal values that are sent to or retrieved from the external source and well formed terms in CycL. Often, interpreting a data literal requires some amount of manipulation or transformation. These *transformation rules* are expressed in terms of functional relationships between the physical and logical fields.

When a tuple is retrieved from the source, its members are associated with their corresponding physical fields. Then a *decoding* transformation is applied which converts the physical field values into logical field values. Finally, the meaning sentence templates are applied to the logical field values to arrive at CycL sentences.

The process is bi-directional. When a CycL sentence matches a meaning sentence template, its bound terms are associated with the appropriate logical field values, and *encoding* transformations are applied to convert them into physical field values of the proper data type, where they can then be used as query constraints (such as in SQL WHERE clauses) on the source's server side.

2.4.1 Example

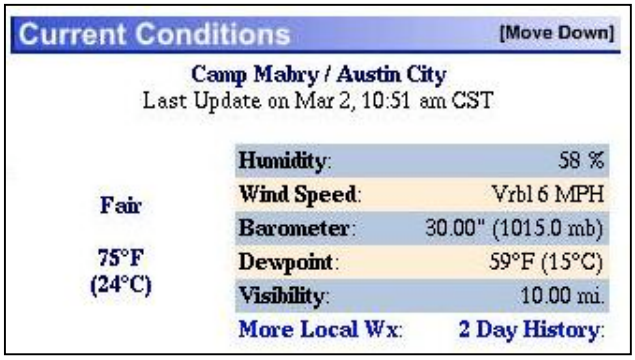

Figure 2. Current conditions in "Austin, TX," according to the NWS web site.

For example, consider the query: "What is the current temperature in Austin, Texas?" This query is expressed in CycL:

```
(ambientTemperature CityOfAustinTX ?TEMP)
```

Here ambientTemperature is a meaning sentence predicate that relates the city to its temperature, CityOfAustinTX is the CycL term for Austin, Texas, and *?TEMP* is a variable. The National Weather Service website is designed to take geographical reference strings of the form "City, ST" as the bound values to queries and return formatted web pages describing the current and forecasted weather conditions of the referenced city. Cyc uses the transformation rule below to encode CycL terms for US Cities, such as CityOfAustinTX, into geographical reference strings which are of the appropriate format to serve as input to web queries, such as "Austin, TX":

```
(fieldEncoding NWS-PS
   (ThePhysicalFieldValueFn NWS-PS "zipcity")
   NWS-LS (BestStringOfNLPhraseFn
              (TermParaphraseFn-
               CityWithStateOrProvinceAbbreviation
                   (TheLogicalFieldValueFn
                    NWS-LS USCity 1))))
```

To apply this rule, Cyc replaces the term

```
(TheLogicalFieldValueFn NWS-LS USCity 1)
```

with CityOfAustinTX, then evaluates the functional expression

```
(BestStringOfNLPhraseFn
   (TermParaphraseFn-
    CityWithStateOrProvinceAbbreviation
       CityOfAustinTX))
```

which evaluates to "Austin, TX"; that string is then used as the input to the web query.

When the result web page is retrieved, Cyc extracts the data value for the current temperature from the web page by applying an appropriate XPath query:

```
//TD[contains(text(), "Last Update
on")=true()]/../../TR/TD/TABLE/TR/TD/text()[2]
```

In the example shown, the extracted value is 75 (degrees Fahrenheit). Another transformation rule is now applied to decode the result of the XPath query into a CycL term:

```
(fieldDecoding
   NWS-LS
   (TheLogicalFieldValueFn NWS-LS
      Temperature 1)
   NWS-PS
   (DegreeFahrenheit
      (ThePhysicalFieldValueFn NWS-PS
       "temperature")))
```

Here, the term (ThePhysicalFieldValueFn NWS-PS "temperature") is replaced with the actual value of 75, resulting in (DegreeFahrenheit 75), which is then bound to the open variable in the query and returned as the answer:

```
?TEMP = (DegreeFahrenheit 75)
```

This example illustrates an additional important feature of the transformation rules: the fact that the temperature 75 is expressed in Fahrenheit was not stated in the physical data value; it was made explicit only in the decoding rule.

3. Query Processing

Next we turn to how CycL queries to external knowledge sources are evaluated by an external server and the relevant data extracted from the returned web pages. In general, CycL queries that are targeted at external sources are translated into another query language, typically SQL, and are passed off to external servers for resolution. When the target data source is a database, the database's native server processes the SQL query, however when the source is a web site, we use our own thin SQL query engine. In all cases, communication between Cyc and the external sources is mediated by a proxy server, which manages all database, web site, and Cyc connections between multiple Cyc clients and target servers.

The Semantic Database Connectivity (SDBC) proxy server communicates with external sources via JDBC. Currently, we write a separate JDBC driver for each web site integrated with Cyc. The driver is comprised of three components, the query engine that processes the SQL, the http request broker that submits requests and retrieves result pages, and a wrapper that extracts a tuple from the result page. For example, when the CycL query above is asked, Cyc converts it into the following SQL:

```
SELECT temperature FROM nws WHERE
zipcity='Austin, TX'
```

This SQL query is passed via SDBC to the driver for the NWS web site. The SQL query engine identifies the required input field in the WHERE clause, 'Austin, TX', and passes it to the http request broker. The broker submits the POST request, retrieves the result page, and passes it to the wrapper. The wrapper parses the HTML and extracts values for all of the physical fields in the source's physical schema, not just the "temperature" field, and returns the tuple of values to the SQL engine. The SQL engine then filters the tuple using the SELECT clause and returns the result set to Cyc. The SQL engine can process more complex SQL queries, with additional WHERE and SELECT clauses; however, the WHERE clause must contain a value for the required input field, otherwise the http request will either fail or never get executed.

There are definite limitations to this approach. Most notably, all of the logic for extracting the data from the result HTML is hard coded into the wrapper. Since web pages are subject to sometimes-frequent format changes, the wrappers are brittle and require routine maintenance. A better approach would be to write a generic wrapper that takes its instructions for parsing the HTML from Cyc. A step in this direction has been taken: we associate an XQuery expression with each physical field in Cyc, and will eventually use these queries within the generic wrapper to extract and properly type cast the data values. Here is an expanded version of the example above; the following XQuery query that extracts the value for "temperature" and cast it to an integer:

```
"declare namespace x = \"http://www.w3.org/1999/xhtml\";
xs:integer(tokenize(doc(\"xhtml.xml\")//x:td[contains(text(),
\"Last Update on\")=
true()]/../../x:tr/x:td/x:table/x:tbody/x:tr/x:td/text()
[2]) \"&#176;F\")"
```

Figure 3. An XQuery to extract temperature from the NWS web site

This improvement will make maintaining access to the web sites much more manageable, however our future work is aimed at applying knowledge about the content of the web site to drive extracting the data using a combination of machine learning and rule based AI.

4. Future Work

The true value of this work will only be realized if it provides access to web pages as diverse as the representations available in the Cyc KB. This depends on two things: 1) mapping in a new source must be extremely inexpensive and straightforward (ideally, the mapping process would be fully automatic), and 2) these mappings must be robust in the face of changes in the format of the page or query field.

We intend to develop methods that will use both general knowledge and specific knowledge about how to interpret web pages, and, where necessary, natural language dialogues with users, to semi-automatically create mappings, thereby shielding the users completely from the underlying CycL representations. The difficulty with fully automating the wrapper generation process, as it is typically understood [2,3], is the need to tie the

resulting wrappers into the existing CycL semantic content in a consistent way.

For example, suppose we are attempting to map in a weather site using a future Semantic Data Schema Elicitor (SDSE). The user would give the SDSE a URL, and a topic "weather". The system would use existing NL generation to list weather related predicates, including "Ambient temperature", that Cyc expects might be instantiated by the site. The known constraints and probable fillers for the arguments of ambientTemperature, would be used by Cyc to ask whether the site has ambient temperatures for towns, cities or airports, and whether the temperatures are given in Fahrenheit or Celsius. On being told "towns" and "Fahrenheit", the system could ask the user to highlight the field in which a town should be entered, and where the temperature would be found. In many cases, parsing of evident field labels could allow this step to be completely automated.

The system would then generate a series of queries, for known towns, and satisfy itself that the returned temperatures were in the range of town ambient temperatures, before storing the newly learned accessors for the site.

This automatic testing, based on KB content, could also be used to detect changes in a data source. Periodically, the system could generate queries with known results (in the case of slowly changing data, such as a movie data site), or using current results from other providers of the same type of information. If too many of these queries did not match expectations, a search within the web page document structure for close matches for the original, working Xquery would be used to support repair.

Similarly, the mappings for an existing weather site, say, could be used to automatically generate plausible queries and expected results for a new weather site, enabling a fully automated search for new extraction patterns. The use of automatically mapped parallel data source would increase coverage and data availability.

In the above example, KB information about the WeatherAttributes-Weather-Topic was used to infer what predicates the site might supply data for. More generally, the characterization of data sources is, itself, a kind of common sense knowledge. There is a general class of "entertainment schedule" sites, including movie listings, concert listings, DVD release date lists, TV schedules, etc, which share a common structure. For example, their searches often require entry of a locality, and may support entry of a time. These commonalities may aid in more fully automating the task of identifying appropriate entry fields and result extraction patterns, and of testing those patterns for validity.

5. Conclusion

By tying web access and extraction techniques to a large, preexisting ontology, information can be successfully integrated across web sites, databases and background knowledge to answer integrative queries. Without this common basis, integration would be incomplete and ad hoc, but producing the required formal mappings is expensive, specialized work.

While the work done so far has allowed useful access to existing web sources—thus allowing querying and inference over the contents of those pages, it requires significant expert effort to creating the mapping schemas, and remains non-robust against even minor changes to those schemas. The real value of the work is in providing a proof of concept of integrating existing mapping and web extraction technology with the semantic information available in large ontologies. To be practical, this process must be automated. Some success has been reported with using numerous volunteers to construct the mapping [4], but that process is tedious and may not be applicable to important classes of specialty sites, such as those on company intranets, or inside classified facilities.

Ultimately, we believe that the use of formal representations of the mappings will permit the necessary automation, allowing the use of background knowledge to identify, and test, hypotheses about how queries should be addressed to a page, and where the answer should be expected. This should allow a gradual move away from the rather brittle hand-created wrappers that allow extraction of data from these pages.

6. ACKNOWLEDGMENTS

We acknowledge the support of ARDA/DTO in funding the development of the technology described herein. This material is based upon work funded in whole or in part by the U.S. Government and any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the U.S. Government.

7. REFERENCES

[1] Arens, Y., Chee, C., Hsu, C., Knoblock, C. Retrieving and Integrating Data from Multiple Information Sources. In the International Journal of Cooperative Information Systems, 2(2), p. 127-158. 1993.

[2] Ashish, N., and Knoblock, C. Semi-automatic wrapper generation for internet information sources. In Proc. of the Second IFCIS International Conference on Cooperative Information Systems (CoopIS), Charleston, SC, 1997.

[3] Chidlovskii, B., Ragetli, J., de Rijke, M. Automatic Wrapper Generation for Web Search Engines. In Web-Age Information Management, p. 399-410, 2000.

[4] Doan, A., McCann, R., Shen, W. Collaborative Development of Information Integration Systems. In Proc. of the AAAI Spring Symposium on Knowledge Collection from Volunteer Contributors, Technical Report SS-05-03, AAAI Press (2005) pp. 34-41.

[5] Knoblock, C., Minton, S., Ambite, J.L., Ashish, N., Modi, P.J., Muslea, I., Philpot, A.G., Tejada, S. Modeling Web Sources for Information Integration. Proc. of The 15th National Conference on Artificial Intelligence. 1998.

[6] Lenat, D. 1995. Cyc: A Large-Scale Investment in Knowledge Infrastructure. In *CACM,* 38, No. 11. pp. 33-38.

[7] Lenzerini, M. Data Integration: A Theoretical Perspective, Proceedings of the Twenty-first ACM Symposium on Principles of Database Systems, Madison, Wisconsin, June 2002.

[8] Liu, M., and Ling, T.W. A Conceptual Model and Rule-based Query Language for HTML. In Proc. of the 7th International Conference on Database Systems for Advanced Applications (DASFAA 2001), Hong Kong, China, April 2001.

[9] Masters, J., and Güngördü, Z. Structured Knowledge Source Integration: A Progress Report. In Integration of Knowledge Intensive Multiagent Systems, Cambridge, Massachusetts, USA, 2003.

[10] Masters, J. Structured Knowledge Source Integration and its applications to information fusion. In Proceedings of the Fifth International Conference on Information Fusion, Annapolis, MD, July 2002.

[11] Matuszek, C., Cabral, J., Witbrock, M., DeOliveira, J. On the Syntax and Content of Cyc. In Proc. of the AAAI Spring Symposium on Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering (forthcoming).

[12] Smith, D., and Lopez, M. Information Extraction for Semi-Structured Documents. On Proc. of the Workshop on Management of Semistructured Data, pp. 225-238, 1997.