

Figure 1: A CIA Factbook page

We will illustrate Ariadne by considering an example application that involves answering queries about the world's countries. An excellent source of data is the CIA World Factbook, which has an HTML page for each country describing that country's geography, economy, government, etc. The top of the factbook page for the Netherlands is shown in Figure 1.² Some

²All the web sources in our examples are based on real sources that Ariadne handles, but we have simplified some of them here for expository purposes.

Figure 2: NATO members page

In the following section we describe our basic approach to query planning, where a unifying domain model is used to tie together multiple information sources. We then describe the details of our modeling approach: how we represent and query individual web pages, how we represent the relationships among multiple pages in a single site, and how we integrate data that spans multiple sites. In each section, we also describe the AI methods that are used in modeling and query processing, and how the uniform representational scheme supports these methods.

Approach to Information Integration

Ariadne’s approach to information integration is based heavily on the SIMS mediator architecture (Arens *et al.* 1996; Knoblock 1995). SIMS enables users to obtain information from multiple heterogeneous information sources. The framework consists of two parts: 1) a query planner/executor that determines how to efficiently process a query given the set of available information sources and 2) wrappers that provide uniform access to the information sources so that they can be queried as if they were SQL databases. The SIMS framework was designed with specific types of information sources in mind, primarily databases and knowledge bases (and to some extent programs), but as we will explain, the approach can be extended to handle web sources.

One of the most important ideas underlying SIMS

is that for each application there is a unifying *domain model* that provides a single ontology for the application. The domain model is represented using the Loom knowledge representation system (MacGregor 1988) and is used to describe the contents of each information source. Given a query in terms of the domain model, the system dynamically selects an appropriate set of sources and then generates a plan to efficiently produce the requested data.

To illustrate this, let us first suppose that the information in the three web sites described earlier, the CIA World Factbook, the World Governments site, and the NATO members page, are each available in three separate databases, along with a fourth database containing a map for each country. To define a new information agent, one would first define a domain model that contains the set of terms that the user might want to query about. An example domain model is shown in Figure 3. The model contains four classes with some relations between them, e.g., ‘NATO Country’ is a subclass of ‘Country’, and ‘Country’ has a relation called ‘Head-of-State’ which points to a class with the same name. We then use the domain model to describe each of the individual information sources. This provides the glue for answering queries that span multiple sources. For example, the figure shows that the CIA factbook is a source for information about Countries, and the World Governments database is a source for Heads of State. Each class has a set of attributes (e.g., total area, latitude, population, etc.) which may be available from one or more sources.

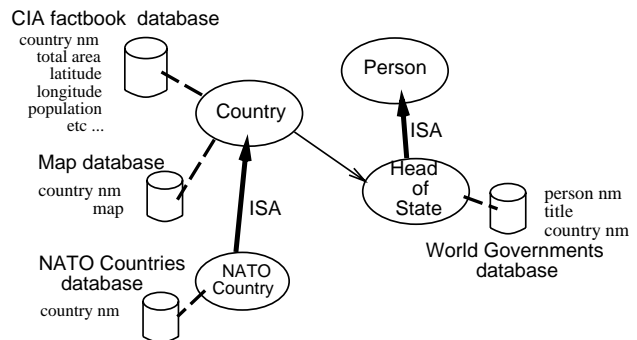


Figure 3: Domain Model with Database Sources

Query Processing

Queries are presented to the system in terms of the domain model. For example, a query might be “List the heads of state of all the countries whose population is less than ten million.”³ The system then decomposes the query into subqueries on the individual sources, such as the World Governments and Factbook

³In actuality, queries are phrased in the Loom KR language, the same language used to express the domain theory. We use English translations for clarity.

sources, producing a partially-ordered query plan consisting of a series of relational operators, i.e., joins, selects, projects, remote subqueries, etc.

The SIMS query planner (Knoblock 1995) was designed primarily for database applications, but database applications typically involve only a small number of databases, while web applications can involve accessing many more sources. Since the SIMS planner did not scale well to large numbers of sources, we developed an approach capable of efficiently constructing very large query plans. We addressed this problem by combining preprocessing techniques with a local-search method for query planning.

In Ariadne, query processing is broken into a preprocessing phase and a query planning phase. In the first phase the system determines the possible ways of combining the available sources to answer a query. Since sources may be overlapping (i.e., an attribute may be available from several sources) or replicated, the system must determine an appropriate combination of sources that can answer the query. The Ariadne source selection algorithm (Ambite *et al.* 1998) preprocesses the domain model so that the system can efficiently and dynamically select sources based on the classes and attributes mentioned in the query.

In the second phase, Ariadne generates a plan using a method called Planning-by-Rewriting, developed by Ambite and Knoblock (Ambite and Knoblock 1997; 1998). This approach takes an initial, suboptimal plan and then attempts to improve it by applying rewriting rules. In the case of query planning, producing an initial, suboptimal plan is straightforward; we can generate an initial plan in $O(n)$ time, where n is the length of the query, based on a depth-first parse of the query. The rewriting process iteratively improves the query via a local search process that can change both the sources used to answer a query and the order of the operations on the data.

Consider the processing required to retrieve all NATO countries that have a population of less than 10 million. Using the domain model in Figure 3, the source selection step would determine that the NATO source is the only source for the class of NATO countries. This source provides only the names of the NATO countries and not their populations. However, the population information can be extracted from the Factbook source since it provides data for a superclass of NATO countries. The reasoning to combine sources in this way is done efficiently by precomputing the way sources can be combined before any queries are processed.

The next step in the example is to construct a plan for efficiently retrieving and combining the data. In this case, the system might first construct an initial plan that retrieves the data from the NATO source, separately retrieves the names and population for all countries from the Factbook source, and then combines the data locally, which is very costly since the Factbook source is quite large. This initial, suboptimal plan is then improved in a series of rewriting steps that would

order the retrieval of the NATO source before the factbook source so that only population data on the NATO countries would need to be retrieved. The optimized plan would then be executed, returning only Denmark, which has a population of just over 5 million.

Because Ariadne combines an efficient source selection algorithm with an efficient, anytime planning algorithm, the system can produce query plans for web environments in a robust, efficient manner. Ariadne's development was aided by the fact that the relational algebra is very simple and well understood, so that we could concentrate on the issues involved in searching for a plan, rather than on the underlying plan representation, which simply consists of a partially ordered set of relational operators. To move to the Web, we only needed one extension to the basic representation, which was the inclusion of "binding patterns" (Kwok and Weld 1996). That is, unlike database sources, web sources may have input/output constraints (e.g., a stock quote server requires a ticker symbol in order to retrieve a stock quote). This is a small extension that is naturally handled by the source selection algorithm and planning operators.

In the remainder of the paper we consider in more detail the modeling issues involved in creating a database-like view of the Web.

Modeling the Information on a Page

The previous section describes how the planner decomposes a complex query into simple queries on individual information sources. To treat a web page as an information source so that it can be queried, Ariadne needs a wrapper that can extract and return the requested information from that type of page. While we cannot currently create such wrappers for unrestricted natural language texts, many information sources on the Web are *semistructured*. A web page is semistructured if information on the page can be located using a concise formal grammar, such as a context-free grammar. Given such a grammar, the information can be extracted from the source without recourse to sophisticated natural language understanding techniques. For example, a wrapper for pages in the CIA factbook would be able to extract fields such as the Total Area, Population, etc. based on a simple grammar describing the structure of factbook pages.

Our goal is to enable application developers to easily create their own wrappers for web-based information sources. To construct a wrapper, we need both a semantic model of the source that describes the fields available on that type of page and a syntactic model, or grammar, that describes the page format, so the fields can be extracted. Requiring developers to describe the syntactic structure of a web page by writing a grammar by hand is too demanding, since we want to make it easy for relatively unsophisticated users to develop applications. Instead, Ariadne has a "demonstration-oriented user interface" (DoUI) where users show the system what information to extract

the items themselves can be catalogs. As an example, consider a CIA factbook page (see Figure 1). The top level consists of an 8-tuple distinguished by section headings: Geography, People, etc. The Geography section is a tuple consisting of Map References, Area, Coastline, etc. These can be decomposed further if necessary; Coastline is a tuple consisting of a number and the string “km”.

Because web pages are intended to be human readable, special markers often play a role identifying the beginning or ending of an item in an embedded catalog, separating items in a homogeneous list, and so on. These distinguishing markers can be used as landmarks for locating information on a page. For instance, to find the longitude, simply skip down to the heading “Geography”, then to “Geographic Coordinates:”, and then skip past the first comma.

A *landmark grammar* describes the position of a field via a sequence of landmarks, where each landmark is itself described by a deterministic finite automaton. Our recent work (Muslea *et al.* 1998) shows that in practice, a subclass of landmark grammars (linear, augmented landmark grammars) can be learned rapidly for a variety of web pages using a greedy covering algorithm. There are several reasons for this. Firstly, because web pages are intended to be human readable, there is often a *single* landmark that distinguishes or separates each field from its neighbors. Therefore the number of landmarks for a field in an embedded catalog will generally be equal to its “depth” in the catalog. Since most catalogs are very shallow, this means that the length of the grammar rules to be learned will be very small, and learning will be easy in practice. Secondly, during the demonstration process, users traverse a page from top-to-bottom, picking out the positive examples of each field. Any position on the page that is not marked as a positive example is implicitly a negative example. Thus, for every positive example identified by the user, we obtain a huge number of negative examples that the covering algorithm can use to focus its search.

The modeling tool we have described enables unsophisticated users to turn web pages into relational information sources. But it has a second advantage as well. If the format of a web source changes in minor respects, the system could induce a new grammar by reusing examples from the original learning episode, without any human intervention (assuming the underlying content has not changed significantly). This is a capability we are currently exploring.

Modeling the Information in a Site: Connections between Pages

The previous section showed how Ariadne extracts information from a web page to answer a query. However, before extracting information from a page, Ariadne must first locate the page in question. Our approach, described in this section, is to model the information required to “navigate” through a web site, so

Figure 4: Creating a Wrapper by Demonstration

The second function achieved by the user’s demonstration is to provide examples so that the system can induce the syntactic structure of the page. Ideally, after the user has picked out a few examples for each field, the system will induce a grammar sufficient for extracting the required information for all pages of this type. Unfortunately, grammar induction methods may require many examples, depending on the class of grammars being learned. However, we have observed that web pages have common characteristics that we can take advantage of, so that a class of grammars sufficient for extraction purposes can be rapidly learned in practice.

More specifically, we can describe most semistructured web pages as *embedded catalogs*. A *catalog* is either a homogeneous list, such as a list of numbers, (1,3,5,7,8), or a heterogeneous tuple, such as a 3-tuple consisting of a number, a letter, and a string, (1,A,“test”). An *embedded catalog* is a catalog where

each wrapper (shown by the small directional arrows in Figure 6). The country page wrapper takes a country-URL, and acts as a source for “total area”, “population”, “latitude”, etc. The index wrapper takes a country name⁵ and acts as a source for “country-URL”. Given the domain model and the binding constraints, the system can now construct query plans. For instance, to obtain the population of a country given its name, the planner determines that the system must first use the country name to retrieve the country-URL from the index page wrapper, and then use the country-URL to retrieve the population data from the country page wrapper.

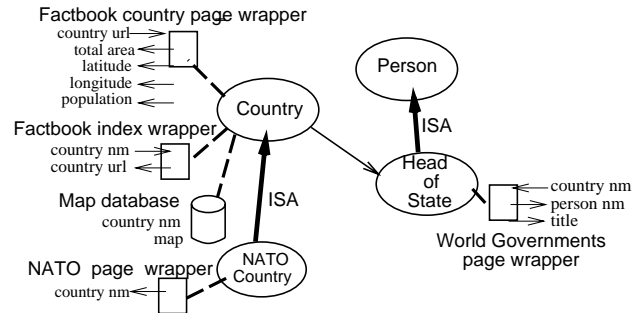


Figure 6: Domain Model with Web Sources

Figure 5: CIA Factbook Index

To create a wrapper for the index page, the developer uses the approach described in the last section, where we illustrated how a wrapper for the factbook’s country pages is created. There is only one difference: this wrapper only wraps a single page, the index page. The developer creates a semantic model indicating that the index page contains a list of countries, each with two attributes, country-nm and country-URL.⁴ The learning system induces a grammar for the entire page after the developer shows how the first few lines in the file should be parsed.

As the wrappers for each source are developed, they are integrated into the unifying domain model. Figure 6 shows the domain model for the completed geopolitical agent. (Notice that we have substituted web source wrappers for the hypothetical databases used previously.) To create the domain model, the developer specifies the relationship between the wrappers and the domain concepts. For instance, the developer specifies that the Factbook country wrapper and the Factbook index wrapper are both information sources for “country” information, and he identifies which attributes are keys (i.e., unique identifiers). In the example, “country-nm” and “country-URL” are both keys. Binding constraints specify the input and output of

⁴During the demonstration, a special copy command is used to obtain a URL from a hyperlink, as opposed to grabbing text.

Explicitly modeling ‘navigation’ pages, such as the factbook index, as information sources enables us to reuse the same modeling tools and planning methodology underlying the rest of the system. The approach works well in part because there are only two common types of navigation strategies used on the Web – direct indexing and form-based retrieval. We have already seen how index pages are handled; form-based navigation is also straightforward. A wrapper for an HTML form simply mimics the action of the form, taking as input a set of attributes, each associated with a form parameter name, and communicating with the server specified in the form’s HTML source.

When the resulting page is returned, the wrapper extracts the relevant attributes in the resulting page. Imagine, for instance, a form-based front end to the factbook, where the user types in a country name and the form returns the requested country page. To create a wrapper for this front end, the developer would first specify that the parameter associated with the type-in box would be filled by a “country-nm”. He would then specify how the system should extract information from the page returned by the form using the approach described in the last section.

The Factbook example described in this section illustrates our basic approach to modeling navigation pages. Many web sites are more complex than the factbook. The approach still works, but the models

⁵No URL is needed as input to the index page wrapper since the URL of the index page is a constant.

and “Republic of Korea”, rather than the other way around. Our approach can also be used to automatically update mapping tables when new sources are released. For instance, each year a new version of the CIA factbook is released, and sometimes countries have new names, or countries merge or split. These name confusions can often be resolved using geographical information (e.g., land area, latitude and longitude).

Applications

Below we list some Ariadne applications we are developing, illustrating the generality of our approach:

World-Wide Geographic Information Server:

We are collaborating with another group that is building a geographic information system that integrates a variety of map-based information sources. These sources include satellite images, detailed street maps, parcel data, historical aerial photographs, etc. We are using Ariadne to extract geographically referenced data from the Web and integrate it with map data. We have built a system using Ariadne that extracts restaurant data from the Zagats Restaurant Reviews site, feeds the restaurant address into a geocoder, and then places the restaurant on an aerial map. Other web sources that we plan to incorporate include census data, US Geological Survey data, and real estate data from the Multiple Listing Service.

Electronic Catalog Access: We are applying Ariadne to provide access to online electronic catalogs for the Defense Logistics Agency. One implemented application provides real-time access to pricing and availability data from the General Services Administration web pages. This application accesses only a single site, but retrieves pricing data for parts by extracting and integrating data from multiple pages in the site.

Financial Information Agent: We have done initial work on an agent that accesses stock quote servers, stock exchange sources, and the SEC’s EDGAR Archives (which contains copies of financial filings, such as annual reports, by publicly traded companies and mutual funds). By integrating these sources, the agent could answer queries such as “Find all airline companies whose stock has risen more than thirty percent in the last year” and “Find all people who serve as directors of two or more companies located in Los Angeles”. Using the modeling tools described earlier, users could also include their own personal financial data sources, tailoring the system to their needs.

Related Work

There is large body of relevant literature on information integration (Wiederhold 1996), but the most closely related work focuses specifically on the problems of information integration on the Web, such as Information Manifold (Levy *et al.* 1996), Occam (Kwok and Weld 1996), Infomaster (Genesereth *et al.* 1997),

and InfoSleuth (Bayardo Jr. *et al.* 1997). These systems focus on a variety of issues, including the problems of representing and selecting a relevant set of sources to answer a query, handling binding patterns, and resolving discrepancies among sources. All of this work is directly relevant to Ariadne, but the issue addressed in this paper that has not been addressed previously is how one represents the information within a single page, across pages at a site, and across sites to support web-based information integration.

Another closely related body of work is on the extraction of data from web sources (Hammer *et al.* 1997; Doorenbos *et al.* 1997; Kushmerick 1997). The focus of all of these systems are on building wrappers for semi-structured sources. The systems either take a template-based specification of a source, as in (Hammer *et al.* 1997), or learn the structure of the source by example and then compile a wrapper that provides access to the source, as in (Kushmerick 1997). Our work on inducing wrappers takes the latter approach. The induction method is not only very general, but is also integrated into the larger Ariadne development system so that the learned wrappers can be used directly by the query planner.

Discussion

There are many examples of impressive AI systems based on relatively simple representational schemes. In the realm of planning, recent examples include SAT-plan (Kautz and Selman 1996) and Graph-plan (Blum and Furst 1995); the former employs a propositional CSP approach, the latter, a graph-based search. In machine learning, propositional learning schemes (e.g., decision trees) have been dominant. Though it is often difficult to understand exactly what a simple representational scheme buys you computationally, one thing seems clear: systems with simple representations are often easier to design and understand.

We believe that Ariadne is successful, in terms of the broad applicability of the approach, because it combines a simple representation scheme with sophisticated modeling tools that map web information sources into this simple representation. Ariadne capitalizes on a representation scheme adopted from database systems, where the world consists of a set of relations (or tables) over objects, and simple relational operators (retrieve, join, etc.) are composed to answer queries. This representation makes it straightforward to integrate multiple databases using an AI planner. Ariadne’s planner can efficiently search for a sequence of joins, selections, etc. that will produce the desired result without needing to do any sophisticated reasoning about the information sources themselves.

The Web environment is much richer than the database world, of course. What makes Ariadne possible are the modeling tools that enable a user to create a database-like view of the Web. Where our approach becomes challenging (and could break down) is in situations where the “natural” way to represent a web

source is not possible due to limitations of the underlying representation.

One such limitation is that Ariadne cannot reason about recursive relations. (To do this properly would require query plans to contain loops.) This has many practical ramifications. For example, consider web pages that have a 'more' button at the bottom, such as Alta Vista's response pages. It would be natural to represent each 'more' button as a pointer to the next page in a list, but there is no way to do this without a recursive relation. Instead, we can build knowledge about 'more' buttons in our wrapper generation tools, so the process of following a 'more' buttons is done completely within a wrapper, hiding the complexity from the query planner.

Another ramification of the planner's inability to reason about recursive relations shows up with hierarchical indexes like Yahoo, where there is no fixed depth to the hierarchy. The natural way to model such pages is with a parent-child relation. Instead, the alternative is to build a more sophisticated wrapper that computes the transitive closure of the parent-child relationship, so that we can obtain all of a node's descendants in one step.

There is an obvious tension between the expressiveness of the representation and the burden we place on the modeling tools. Our approach has been to keep the representation and planning process simple, compensating for their weaknesses by relying on smarter modeling tools. As we have described, the advantage is that we can incrementally build a suite of modeling tools that use machine learning, statistical inference, and other AI techniques, producing a system that can handle a surprisingly wide range of tasks.

Acknowledgements

This work was supported in part by USC's Integrated Media Systems Center (IMSC) - an NSF Engineering Research Center, by the U.S. Air Force under contract number F49620-98-1-0046, by the Rome Laboratory of the Air Force Systems Command and the Defense Advanced Research Projects Agency (DARPA) under contract number F30602-97-2-0352, by the Defense Logistics Agency, DARPA, and Fort Huachuca under contract number DABT63-96-C-0066, and by a research grant from General Dynamics Information Systems. The views and conclusions contained in this paper are the authors' and should not be interpreted as representing the official opinion or policy of any of the above organizations or any person connected with them.

References

- Ambite, J.L. and Knoblock, C.A. 1997. Planning by rewriting: Efficiently generating high-quality plans. In *Proceedings of AAAI-97*.
- Ambite, J.L. and Knoblock, C.A. 1998. Flexible and scalable query planning in distributed and heterogeneous environments. In *Proceedings of AIPS-98*.
- Ambite, J.L.; Knoblock, C.A.; Muslea, I.; and Philpot, A. 1998. Compiling source descriptions for efficient and flexible information integration. Technical report, USC Information Sciences Inst.
- Arens, Y.; Knoblock, C.A.; and Shen, W.M. 1996. Query reformulation for dynamic information integration. *Journal of Intelligent Information Systems*, 6(2/3):99-130.
- Bayardo Jr., R.J.; Bohrer, W.; Brice, R.; Cichocki, A.; Fowler, J.; Helal, A.; Kashyap, V.; Ksiezzyk, T.; Martin, G.; Nodine, M.; Rashid, M.; Rusinkiewicz, M.; Shea, R.; Unnikrishnan, C.; Unruh, A.; and Woelk, D. 1997. Infosleuth: Agent-based semantic integration of information in open and dynamic environments. In *Proceedings of ACM SIGMOD-97*.
- Blum, A. and Furst, M. 1995. Fast planning through planning graph analysis. In *Proceedings of IJCAI-95*.
- Cohen, W.W. 1998. Integration of Heterogeneous Databases without Common Domains using Queries Based on Textual Similarity. In *Proceedings of ACM SIGMOD-98*.
- Doorenbos, R.B.; Etzioni, O.; and Weld, D.S. 1997. A scalable comparison-shopping agent for the worldwide web. In *Proceedings of the First International Conference on Autonomous Agents*.
- Genesereth, M.R.; Keller, A.M.; and Duschka, O.M. 1997. Infomaster: An information integration system. In *Proceedings of ACM SIGMOD-97*.
- Hammer, J.; Garcia-Molina, H.; Nestorov, S.; Yerneni, R.; Breunig, M.; and Vassalos, V. 1997. Template-based wrappers in the TSIMMIS system. In *Proceedings of ACM SIGMOD-97*.
- Huang, T. and Russell, S. 1997. Object Identification in a bayesian context. In *Proceedings of IJCAI-97*.
- Kautz, H. and Selman, B. 1996. Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proceedings of AAAI-96*.
- Knoblock, C.A. 1995. Planning, executing, sensing, and replanning for information gathering. In *Proceedings of IJCAI-95*.
- Kushmerick, N. 1997. *Wrapper Induction for Information Extraction*. PhD thesis, Computer Science Dept., University of Washington.
- Kwok, C.T. and Weld, D.S. 1996. Planning to gather information. In *Proceedings of AAAI-96*.
- Levy, A.Y.; Rajaraman, A.; and Ordille, J.J. 1996. Query-answering algorithms for information agents. In *Proceedings of AAAI-96*.
- MacGregor, R. 1988. A deductive pattern matcher. In *Proceedings of AAAI-88*.
- Muslea, I.; Minton, S.; and Knoblock, C.A. 1998. Wrapper induction for semistructured, web-based information sources. In *Proceedings of the CONALD-98 Workshop on Learning from Text and the Web*.
- Wiederhold, G. 1996. *Intelligent Integration of Information*. Kluwer.