Conceptual Modeling for a Web of Knowledge

David W. Embley[†][⋆], Stephen W. Liddle[‡], and Cui Tao[†][*]

[†]Department of Computer Science
[‡]Department of Information Systems
Brigham Young University, Provo, Utah 84602, U.S.A.

Abstract. The current web is a web of linked pages. Frustrated users search for facts by guessing which keywords or keyword phrases might lead them to pages where they can find facts. Can we make it possible for users to search directly for facts? Equivalently, can we turn the web into a web of facts (instead of a web of pages containing facts)? Ultimately, can the web be a knowledgebase—a web of knowledge—that can provide direct answers to factual questions and also provide the confidence necessary to make those answers believable? We answer these questions by showing how to superimpose a web of data over the web of pages, resulting in a web of knowledge. Our solution, which is based on conceptual modeling, calls for turning raw symbols contained in web pages into knowledge and making this knowledge accessible via the web. The particulars of our solution show ways to overcome three impeding challenges: (1) automatic (or near automatic) creation of ontologies, (2) automatic (or near automatic) annotation of web pages with respect to these ontologies, and (3) simple but accurate query specification, usable without specialized training. Meeting these basic challenges can simplify knowledge-web content creation and access to the point that the vision of a web of knowledge can become a reality. Throughout, we show that conceptual modeling plays a key role in actualizing these ideas.

1 Introduction

To think about turning raw data into accessible knowledge in a *Web of Knowledge* (*WoK*), we first ask some fundamental questions: What is data? What are facts? What is knowledge? How does one reason and know? Philosophers have sought answers to these questions for millennia; and although we do not pretend to be able to contribute to Philosophy, we can use their ideas about *ontology*, *epistemology*, and *logic* to guide us in how to build a WoK.

- *Ontology* is the study of existence. It asks: "What exists?" In our quest to build a WoK, we must find computational solutions the question: "What concepts, relationships, and constraints exist?" We answer computationally, saying that we can declare a formal conceptual model for some domain of knowledge that captures the relevant concepts along with the relationships

[⋆] Supported in part by the National Science Foundation under Grant #0414644.

among these concepts and the constraints over these concepts and relationships.[1]
- *Epistemology* is the study of the nature of knowledge. It asks: "What is knowledge?" and "How is knowledge acquired?" To build a WoK, we provide computational answers to "What is digitally stored knowledge?" and "How does raw data become algorithmically accessible knowledge?" Our answer is to turn raw data into knowledge by populating conceptual models—by embedding facts in the concepts and relationships in accord with constraints.
- *Logic* comprises principles and criteria of valid inference. It asks: "What is known?" and "What can be inferred?" In the computational context of a WoK, it can answer the question: "What are the known facts (both given and implied)?" We ground our conceptual model in a description logic—a decidable fragment of first-order logic [BN03]. To make this logic practical for non-logicians, we must and do add a query generator whose input consists of ordinary free-form textual expressions or ordinary fill-in-the-blank query forms. As we explain later in this paper, both query modes fundamentally depend on conceptual models.

To actualize these ideas, we present a way to turn raw symbols contained in web pages (or other source documents) into computational knowledge and to make this knowledge accessible by average web users. The key research problems are: (1) How do we make ontology creation—conceptual-model creation—easy enough to be usable by typical human knowledge workers? (2) How do we make epistemology—content annotation with respect to an ontology—easy enough to require little, if any, training for human annotators? (3) How do we make logic—query specification—easy enough for web users to apply without specific training in writing queries in sophisticated query languages? Not only do these activities need to be easy enough, they also have to be good enough. Without a resolution to these problems, the barrier to WoK content creation and usage will remain too high, and the envisioned WoK will remain elusive.

Our paper addresses these challenges and contributes by showing how answers can enable a WoK. As a main thread, it highlights the role of conceptual modeling. In Section 2 we provide an overview of our vision of how to superimpose a web of knowledge over a web of pages. In Section 3 we give the details of two types of tools: (1) tools to create and populate ontologies for a WoK and (2) tools to query populated WoK ontologies. We explain why believe these tools provide the simplicity, as well as the flexibility, needed to make them practical for typical knowledge workers. Additionally, we explain how these tools point to future work that can expand the ability of knowledge workers to successfully generate, populate, and query ontologies. We conclude in Section 4 by reiterat-

[1] Purists argue that conceptual models are not ontologies [Gru93,Gua98,Smi03]. We agree that when conceptual models play their traditional role to aid in database schema design, they typically are not ontologies. But when they are used to answer "What exists?" and thus when they formally capture the concepts, relationships, and constraints that exist in a domain, they are ontologies.

ing the key role that conceptual modeling plays in enabling the superposition of a web of knowledge over the current web of pages.

2 From a Web of Pages to a Web of Knowledge

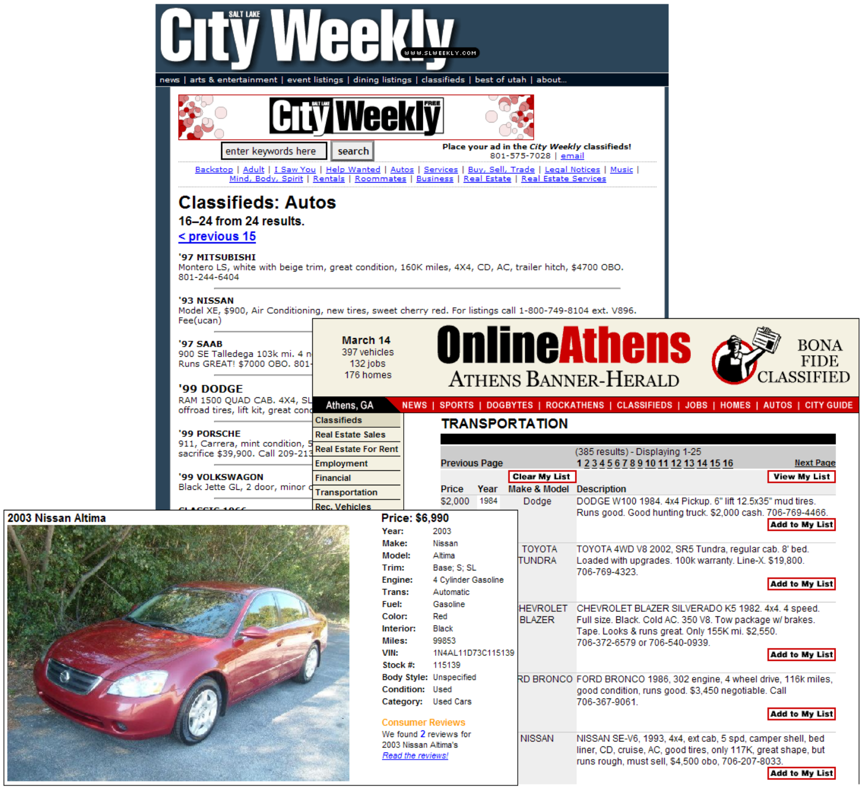
We use two examples to show how to turn a web page into a page of queriable data. Figure 1 shows part of three ordinary, human-readable web pages about cars for sale. The facts in these pages are obvious: e.g., a '93 NISSAN is for sale; it is sweet cherry red, has air conditioning, and sells for $900. Figure 2, which is about genes and molecular biology, shows a second example. Facts on this page are much less obvious to ordinary readers, but a specialist can see a myriad of facts: Gene cdk-4 has genetic position X:12.68 +/- 0.009 cM and genomic position X:13518824.13515774bp. Users would like to be able to query the facts on these pages directly: "Find me a red Nissan for under $5000; a 1990 or newer with less than 120K miles on it." Or, "Tell me the genomic location of cdk-4." We cannot, however, directly access these facts with the current structure of the web. Our approach makes these facts visible from outside the page and directly accessible to query engines (as opposed to search engines).

2.1 From Symbols to Knowledge—Ontological and Epistemological Tools

To make facts available to query engines, we first map out a guiding pathway to turn raw symbols into knowledge. *Symbols* are characters and character-string instances (e.g., $, mileage, red, Gene, Protein, WP:CE18608). *Data* builds on *symbols* by adding conceptual meta-tags (e.g., Price: $900, Color: red, Protein: WP:CE18608). *Conceptualized data* groups data tagged with conceptual identifiers into the framework of a conceptual model (e.g., an ordinary, extended ER model, although in our work the conceptual models we use are fact-oriented, like ORM [Hal95]). We have *knowledge* when we populate a conceptual model with correct[2] conceptualized data.

To specify ontological descriptions, we need a conceptual-modeling language. We use OSM [EKW92], which lets us classify and describe things that exist as object sets of things, relationships among these things as relationship sets, generalization/specialization hierarchies for is-a-related object sets, and constraints over object and relationship sets. In principle, it does not matter which conceptual-modeling language we use, but for a WoK, as we envision it, the language should have certain characteristics:

[2] *Correct* is interesting. How do we know whether conceptualized data is correct? Humans struggle to know; machines may never know. For the WoK we are contemplating, we rely on evidence and provenance by always linking conceptualized data back to its original source—the human-readable web page from which it was extracted.


SALT LAKE City Weekly WWW.SLWEEKLY.COM
news | arts & entertainment | event listings | dining listings | classifieds | best of utah | about...
SALT LAKE City Weekly FREE
enter keywords here
search

Place your ad in the *City Weekly* classifieds!
801-575-7028 | email

Backstop | Adult | I Saw You | Help Wanted | Autos | Services | Buy, Sell, Trade | Legal Notices | Music | Mind, Body, Spirit | Rentals | Roommates | Business | Real Estate | Real Estate Services

Classifieds: Autos

16–24 from 24 results.

< previous 15

'97 MITSUBISHI
Montero LS, white with beige trim, great condition, 160K miles, 4X4, CD, AC, trailer hitch, $4700 OBO. 801-244-6404

'93 NISSAN
Model XE, $900, Air Conditioning, new tires, sweet cherry red. For listings call 1-800-749-8104 ext. V896. Fee(ucan)

'97 SAAB
900 SE Talledega 103k mi. 4 n
Runs GREAT! $7000 OBO. 801-

'99 DODGE
RAM 1500 QUAD CAB. 4X4, SL
offroad tires, lift kit, great cond

'99 PORSCHE
911, Carrera, mint condition, 5
sacrifice $39,900. Call 209-213

'99 VOLKSWAGON
Black Jette GL, 2 door, minor c

TRANSPORTATION

(385 results) - Displaying 1-25

Previous Page 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 Next Page
Clear My List
View My List

Price	Year	Make & Model	Description
$2,000	1984	Dodge	DODGE W100 1984. 4x4 Pickup. 6" lift 12.5x35" mud tires. Runs good. Good hunting truck. $2,000 cash. 706-769-4466. Add to My List
		TOYOTA TUNDRA	TOYOTA 4WD V8 2002, SR5 Tundra, regular cab. 8' bed. Loaded with upgrades. 100k warranty. Line-X. $19,800. 706-769-4323. Add to My List
		HEVROLET BLAZER	CHEVROLET BLAZER SILVERADO K5 1982. 4x4. 4 speed. Full size. Black. Cold AC. 350 V8. Tow package w/ brakes. Tape. Looks & runs great. Only 155K mi. $2,550. 706-372-6579 or 706-540-0939. Add to My List
		RD BRONCO	FORD BRONCO 1986, 302 engine, 4 wheel drive, 116k miles, good condition, runs good. $3,450 negotiable. Call 706-367-9061. Add to My List
		NISSAN	NISSAN SE-V6, 1993, 4x4, ext cab, 5 spd, camper shell, bed liner, CD, cruise, AC, good tires, only 117K, great shape, but runs rough, must sell, $4,500 obo, 706-207-8033. Add to My List

2003 Nissan Altima

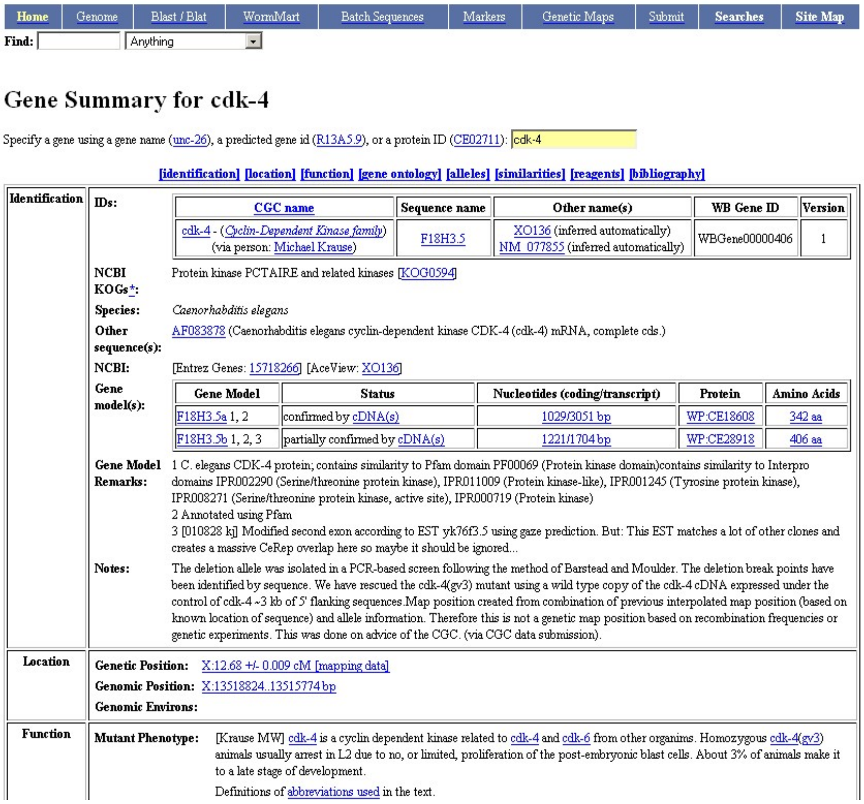
Price: $6,990

Year:	2003
Make:	Nissan
Model:	Altima
Trim:	Base; S; SL
Engine:	4 Cylinder Gasoline
Trans:	Automatic
Fuel:	Gasoline
Color:	Red
Interior:	Black
Miles:	99853
VIN:	1N4AL11D73C115139
Stock #:	115139
Body Style:	Unspecified
Condition:	Used
Category:	Used Cars

Consumer Reviews
We found 2 reviews for
2003 Nissan Altima's
Read the reviews!

Fig. 1. Sample Car Ads Web Pages.

- The conceptual-modeling language should have a direct correspondence to predicate calculus—the simpler and more direct the correspondence, the better. Fact oriented conceptual models (e.g., OSM [EKW92] and ORM [Hal95]), which model both entities and attributes as objects, have a more direct correspondence to predicate calculus. Each object set represents a one-place predicate, and each n-ary relationship set represents an n-place predicate. Populating object and relationship sets with values directly yields the facts for a first-order interpretation (which, incidentally, is also why these types of conceptual models are called "fact-oriented").
- Every constraint representable in the chosen conceptual-modeling language should be expressible as a well-formed formula of the first-order language induced by the representation of the object and relationship sets of the language. (Constraint translations for both OSM [EZ10] and ORM [Jar07] are examples.)
- To facilitate reasoning, the conceptual-modeling language should be merely an alternate representation for a first-order language, as are fact-oriented



Gene Summary for cdk-4

Specify a gene using a gene name (unc-26), a predicted gene id (R13A5.9), or a protein ID (CE02711): cdk-4

[identification] [location] [function] [gene ontology] [alleles] [similarities] [reagents] [bibliography]

Identification

IDs:

CGC name	Sequence name	Other name(s)	WB Gene ID	Version
cdk-4 - (*Cyclin-Dependent Kinase family*) (via person: Michael Krause)	F18H3.5	XO136 (inferred automatically) NM_077855 (inferred automatically)	WBGene00000406	1

NCBI KOGs*: Protein kinase PCTAIRE and related kinases [KOG0594]

Species: *Caenorhabditis elegans*

Other sequence(s): AF083878 (Caenorhabditis elegans cyclin-dependent kinase CDK-4 (cdk-4) mRNA, complete cds.)

NCBI: [Entrez Genes: 15718266] [AceView: XO136]

Gene model(s):

Gene Model	Status	Nucleotides (coding/transcript)	Protein	Amino Acids
F18H3.5a 1, 2	confirmed by cDNA(s)	1029/3051 bp	WP:CE18608	342 aa
F18H3.5b 1, 2, 3	partially confirmed by cDNA(s)	1221/1704 bp	WP:CE28918	406 aa

Gene Model Remarks: 1 C. elegans CDK-4 protein; contains similarity to Pfam domain PF00069 (Protein kinase domain)contains similarity to Interpro domains IPR002290 (Serine/threonine protein kinase), IPR011009 (Protein kinase-like), IPR001245 (Tyrosine protein kinase), IPR008271 (Serine/threonine protein kinase, active site), IPR000719 (Protein kinase)
2 Annotated using Pfam
3 [010828 kj] Modified second exon according to EST yk76f3.5 using gaze prediction. But: This EST matches a lot of other clones and creates a massive CeRep overlap here so maybe it should be ignored...

Notes: The deletion allele was isolated in a PCR-based screen following the method of Barstead and Moulder. The deletion break points have been identified by sequence. We have rescued the cdk-4(gv3) mutant using a wild type copy of the cdk-4 cDNA expressed under the control of cdk-4 ~3 kb of 5' flanking sequences.Map position created from combination of previous interpolated map position (based on known location of sequence) and allele information. Therefore this is not a genetic map position based on recombination frequencies or genetic experiments. This was done on advice of the CGC. (via CGC data submission).

Location

Genetic Position: X:12.68 +/- 0.009 cM [mapping data]

Genomic Position: X:13518824..13515774 bp

Genomic Environs:

Function

Mutant Phenotype: [Krause MW] cdk-4 is a cyclin dependent kinase related to cdk-4 and cdk-6 from other organims. Homozygous cdk-4(gv3) animals usually arrest in L2 due to no, or limited, proliferation of the post-embryonic blast cells. About 3% of animals make it to a late stage of development.

Definitions of abbreviations used in the text.

Fig. 2. Sample Molecular-Biology Web Page.

conceptual-modeling languages whose constraints are expressible as well-formed first-order expressions. To keep reasoning tractable as well as decidable, appropriate limitations are necessary. (Adjusting the conceptual-modeling language for our WoK vision as researchers come to better understand the decidability and tractability issues of description logics may be appropriate.)

An appropriately limited version of OSM satisfies these requirements better (in our opinion) than the more traditional conceptual-modeling languages such as extended ER or UML and is more easily tailored to the needs of WoK development than ORM, a more widely known fact-oriented language.

OSM, in general, has the power of predicate calculus, but the limited version we typically use is essentially an $\mathcal{ALCN}$ description logic [BN03]. Thus, OSM has the formal properties required (1) to ontologically represent concepts and their interrelationships, (2) to epistemologically represent knowledge as predicate-calculus theories in terms of formal interpretations, and (3) to logically establish inference rules and to query over base interpretations and inference rules. Further, in its limited version, OSM has reasonable processing efficiency.

A minimal necessary tool is an editor that allows users to construct and populate conceptual models. Building and populating ontologies by hand, however, becomes a bottleneck in the process of turning data into knowledge [BCL06]. Thus, we seek an answer to this question: Can we automatically construct and populate an ontology for a domain of knowledge from raw source domain information? If so, how? If not fully automatically, can we at least semi-automatically construct domain knowledge and do so with most of the burden of construction shifted to the machine?

Attempts to extract ontologies from natural-language text documents have been largely unsuccessful [BCL06]. The unbounded freedom in natural-language expression makes synthesizing conceptualizations extremely difficult. Attempts to extract ontologies from semi-structured documents, such as the ones in Figures 1 and 2, although challenging, appear promising [TEL$^+$05]. To the extent we can algorithmically discover meta-information and tie it to data values, we can synthesize conceptualizations. The degree of semi-structuredness is a rough measure of how amenable a document is to automatic conceptualization.

With algorithmic conceptualization in mind, we approach the problem of automatic and semi-automatic ontology construction by building tools that use the web pages we wish to turn into knowledge as sources to help us construct the ontology. This works by recognizing that the data is formatted in a particular way (e.g., the table in the OnlineAthens ads in Figure 1) and by using reverse-engineering techniques to construct a conceptual model (e.g., to discover that $Price$, $Year$, $Make$, and $Model$ in the table in Figure 1 are concepts for a car-sales conceptual model or to discover that *Genetic Position* and *Genomic Position* in Figure 2 are alternate ways to specify locations for a gene ontology). Since we cannot fully count on these automatic ontology-building tools, we also provide a way to build ontologies that leverages the idea of an ordinary form. People generally know how to design forms with single- and multiple-entry blanks. And we know how to algorithmically turn form-like nested structures into conceptual models [AK07].

Ontological descriptions, however, are not enough for our envisioned web of knowledge. We also need a way to link raw facts in web pages with ontological descriptions. We need epistemological tools as well as ontological tools. A way to link the actual facts with an ontology is to annotate a web page with respect to that ontology. We annotate a data value in a web page with respect to an ontology by mapping it to an object set in an OSM ontology. Likewise, we annotate related pairs of values (and, more generally, related n-tuples of values) in a web page by mapping them to a relationship set in an ontology.

Although it is possible to annotate a web page with respect to an ontology by hand, this is likely to be too tedious and time consuming to be practical for most applications.[3] We therefore augment OSM ontologies with instance

[3] Although tedious, we do foresee hand-annotation as a viable way to create annotated content. Moreover, we can also annotate images like some commercial enterprises do

recognizers (ontologies augmented with instance recognizers are called *extraction ontologies*). Instance recognizers contain regular expressions that recognize common textual items such as dates, times, prices, and telephone numbers. They can also contain lexicons that match with items such as car makes and models and protein names and functions. Much can and has been said about how to build and use these instance recognizers embedded within extraction ontologies [ECJ$^+$99,DEL06].

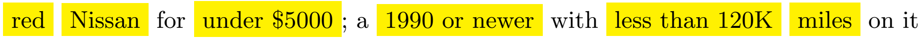
Although we can use extraction ontologies to automate the annotation of web pages, building instance recognizers for OSM extraction ontologies is laborious. We therefore seek for better ways to do automatic annotation without having to build many specialized recognizers by hand. We first observe that for many common data items such as ordinary numbers, dates, time, currency, and percentages, we can expend the effort to create good recognizers, and we can store them in a library and use them liberally in many extraction ontologies. Even so, however, many needed recognizers would not likely be among them. From our examples (Figures 1 and 2), we would not likely have recognizers for makes, models, and features of cars; nor for proteins, amino acids, and gene locations.

For these types of items, we seek alternative ways to build recognizers. We observe that many of these types of data items come from regular patterns in semi-structured pages. Columns in tables, like the makes and models in the table in Figure 1 or like the proteins and amino acids in the table in Figure 2, are common, as are lists of items, either in a single column or delimiter separated like the features in car ads. We should be able to classify and annotate values in semi-structured documents. Indeed, we observe that if we can extract ontological knowledge from semi-structured web pages, we should at the same time be able to also extract epistemological knowledge. For example, if we can recognize the table structures in Figure 2 well enough to derive the meta-knowledge necessary for an ontology, we should be able to link the data associated with the meta-knowledge (e.g., category labels in tables) to the ontological concepts to which they belong. This maps the data to an ontology and thus annotates the data in the web page. Furthermore, as we annotate data such as car makes and models and protein names and amino acids, we can keep the values we find in lexicons and thus automatically build and continuously improve instance recognizers for extraction ontologies.

2.2 Querying Knowledge—Logic Tools

After building tools to turn raw symbols in web pages into knowledge, we next need to provide appropriate query capabilities. Given that we have data on a web page annotated with respect to an ontology, we can immediately generate subject-predicate-object triples in RDF [W3Cb], the W3C standard for representing ontological data. We can then directly use the SPARQL query language

(e.g., [Foo07]), but with respect to ontologies so that they can be queried in the envisioned WoK. Likewise, sound bites and videos, and in general all multi-media objects, can be annotated with respect to ontologies.



red Nissan for under $5000; a 1990 or newer with less than 120K miles on it

Fig. 3. Free-Form Query.

[W3Cc], also a W3C standard, to write and execute queries over this RDF data. In addition, from OSM ontologies, we can also immediately generate OWL ontologies [W3Ca], another W3C standard. This further enhances our ability to reason with the data since we can write and execute inference rules over OWL ontologies (e.g., in Pellet [SPG$^+$07], an open source OWL description-logic reasoner).

If everyone could write SPARQL queries, we would basically have what we need to enable users to search the envisioned WoK. However, since we target users who are not trained in formal query specification, we should not expect these users to learn SPARQL or any other formal query language. Instead, we should provide a query system in which users can pose queries in their own terms. Figure 3 shows a free-form query for a tool we have built [Vic06,AM07]. The key to making these free-form queries work is not natural-language processing (at least not in the usual sense of natural-language processing), but rather is the application of extraction ontologies to the queries themselves. This lets us align user queries with ontologies and thus with facts in annotated web pages. In essence, these free-form queries are keyword queries over both the instances and the concepts in populated ontologies (as opposed to keyword queries over web pages). And, in addition, the system also uses keywords and instance recognizers to identify implied operators and operands for operations as well. As Figure 3 shows, the query engine highlights words, values, phrases, and operations it recognizes (e.g., the context keyword *miles*, the value *red*, and the operation *under* applied to the value *$5000*). The highlighting provides feedback to users, letting them know which words, values, phrases, and operations the search engine recognizes.

Anyone can readily pose free-form queries. To be successful, however, users do have to guess what keywords, values, and constraint expressions might be available in an extraction ontology for the domain of interest. This is similar to users having to guess keywords and values for current search-engine queries. Since arbitrary free-form queries may not always be successful, we also provide a form query language, based on the ontology, that allows a user to fill out a form and submit it as a query in much the same way users currently pose queries by filling in forms currently on the web. Interestingly, these query forms are automatically derivable from domain ontologies, and thus need not be specified by developers. Instead of reverse-engineering a form to create an ontological structure, we can forward-engineer (derive) forms from the ontology and use them as a natural-forms query language (e.g., [Emb89]).

Finally, we must make all of this scale globally. It is not enough to be able to return answers to user queries and to point them to a source page showing that the answer accurately reflects the facts as recorded on the pages. We also have to return answers in real time. Users are not patient enough to tolerate long delays in receiving answers to what they perceive as a simple question. To this end, we must define and build tools to process queries quickly. First, we cache all pages we annotate.[4] Second, we have defined semantic indexing [AMELT07], with which we expect to be able to quickly find applicable ontologies for user queries, which is likely the key bottleneck in the process. To pursue this further, we are inclined to follow the lead of modern search engines—to fully exploit massive parallelism and expansive data stores. In our research environment, however, we will not be able to build a full-scale system; we can, however, build prototypes that show the way and provide technical answers to make it work.

3 WoK Tools

Although we have not yet built the full set of tools we envision, we have developed some prototypes. In Section 3.1 we briefly describe our ontological/epistemological tools, and in Section 3.2 we briefly describe our query tools. In doing so, we emphasize again the central role of conceptual modeling.

3.1 Ontology/Epistemology Creation Tools

Automatic ontology/epistemology creation requires source documents that embody both the conceptual structure of the knowledge as well as the facts embedded in the structure. Automatic creation of a populated ontology then becomes a reverse-engineering process. Researchers in the conceptual-modeling community have worked on reverse-engineering structured data into conceptual models for many years (e.g., [Alh03,CBS94,LCWA07]). Web pages, however, normally include neither relational tables nor any other standard database structure. Can we extend these reverse-engineering techniques to data laid out in arbitrary, human-readable forms and tables, or to data laid out as human-readable semi-structured data, or even to unstructured human-readable data? The answer appears to be “yes”, but the task becomes increasingly more difficult as the structure of the input becomes increasingly less structured.

When data is structured in well-known ways, we know which symbols represent meta-data and which represent data, and we know how the meta-data relates to the data. When data is structured in less well-known ways, distinguishing data from meta-data and relating data and meta-data become more challenging. Even when data, meta-data, and their interrelationships are known, the reverse-engineering process is not necessarily straightforward. Reverse mappings are typically not unique, so selecting among the plausible solutions can also be an issue.

In our ontology/epistemology creation tools, we exploit two well-known and commonly-used information structures: forms and tables. In both cases we take a particular approach that limits the difficulty of the reverse-engineering process.

[4] Indeed we must, for otherwise, we cannot guarantee that our provenance links will be correct. This implies, by the way, that for sites whose pages change often, we must have a fully automatic way to re-annotate pages from the site.

For forms, we choose to stay within the bounds of well understood form fields in everyday use, and we establish in advance how we will reverse-engineer each type of form field as well as each combination of form fields. We call our form-based tool *FOCIH* (*Form-based Ontology Creation and Information Harvesting*). For tables, we consider only HTML tables, and, in particular, only HTML tables in sibling pages—machine-generated pages each laid out in the same way. We call our table-interpretation tool *TISP* (*Table Interpretation in Sibling Pages*), and we augment its name to *TISP*$^{++}$ when we refer to an extension of TISP that builds and populates ontologies from interpreted tables. We now describe FOCIH and TISP/TISP^{++}.

FOCIH

FOCIH is a tool that lets users specify ontologies without having to know any conceptual-modeling language or ontology language [Tao08]. We observe that forms are a natural way for humans to collect information. As an everyday activity, people create forms and ask others to fill in forms so that specified information can be gathered. Believing that users can specify and fill in forms, we let users create their own forms to describe information they wish to harvest. Once defined, users can fill in forms from web pages by copy and paste. From these user actions, FOCIH generates an ontology and annotates the web page with respect to the ontology. Further, if the web page is machine-generated and has sibling pages, FOCIH is able to harvest the specified information from all the sibling pages, usually without further user intervention.

FOCIH's form-creation mode provides users with a way to define different kinds of form features. FOCIH has five basic form-field elements from which users can choose: *single-label/single-value*, *single-label/multiple-value*, *multiple-label/multiple-value*, *mutually-exclusive choice*, and *non-exclusive choice*. Figure 4 shows an example of form creation in our prototype implementation.[5] To create a form, users click on form-field icons to say which type of form field they want and then click on the pencil icon and type in a name to provide a label for the form field. By clicking on the plus icon, users can extend the number of columns in a multiple-label/multiple-value form field and can extend choice elements to have has many choices as desired. In Figure 4 the only plus-icon appears with *Accessory*, the last choice for *Feature*. Clicking on the plus-icon would allow a user to add another feature category, for example, for safety features or anti-theft features. Finally, observe in Figure 4 that form-field icons appear inside every elementary form field; this lets users construct forms with form fields nested as deeply as desired.

As an example, suppose we wish to buy a car. We can let FOCIH harvest the information we wish to consider and then query the harvested information to find cars we may want to buy. We let *Car* be the title for the base form as Figure 4 shows. For each car we want to harvest the *Year*, *Make*, *Model*, and

[5] We note that our implementation is only a research prototype. A more intuitive interface can be built; our focus in the prototype is form-creation functionality leading to an ontology, plus automated information harvesting.

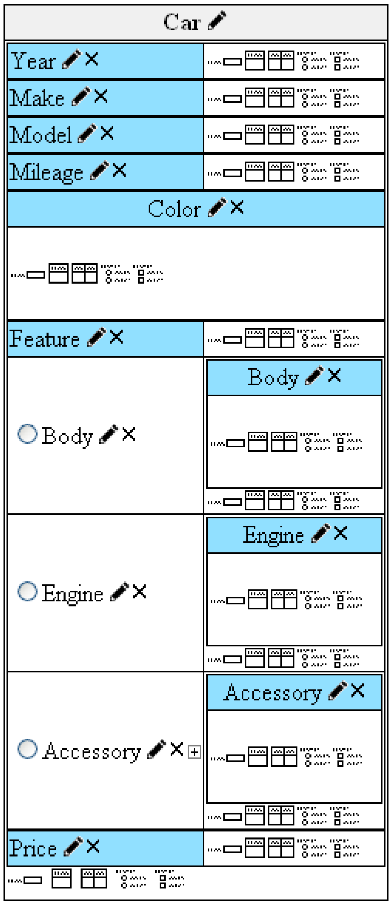
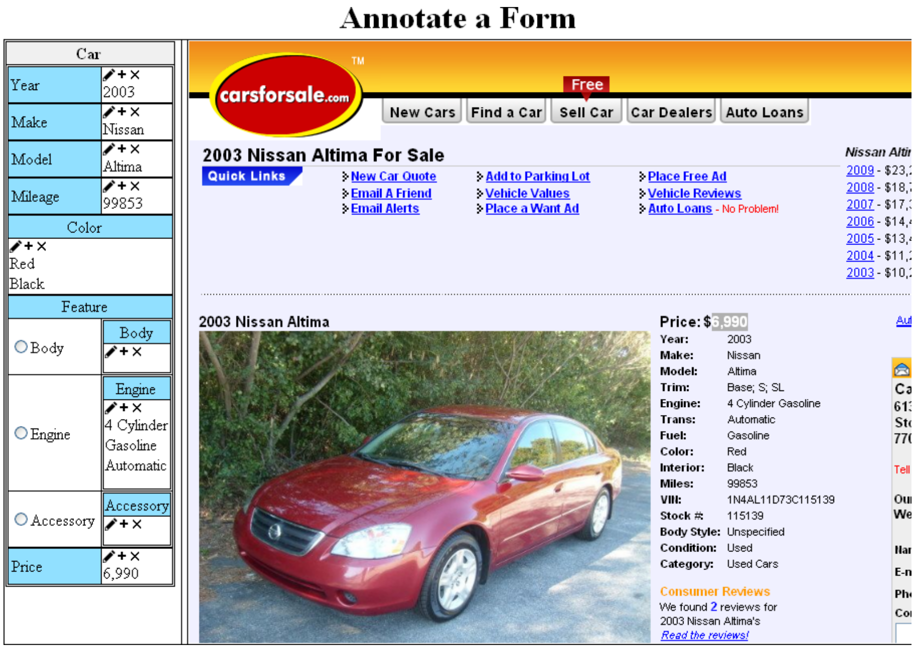


Fig. 4. A Sample Form.

Mileage. Since each car has only one year, make, model, and mileage, we choose single-value form fields for each as Figure 4 shows. We also want car colors; since a car may have more than one color, we let *Color* be a single-label/multiple-value form field. A car may have several features of interest, which we wish to think of as *Body* features, *Engine* features, and *Accessory* features. To accommodate these specializations of *Feature*, we add an exclusive choice field—"exclusive" because any particular feature can only be in one of the specializations. Further, because for each feature category we can have several features in the category, we nest a single-label/multiple-value form field in each. Of course, we also want to know how much the owner expects the buyer to pay for the car, so we add a *Price* single-value form field at the end. Figure 4 shows the resulting form.

FOCIH's form-fill-in mode lets users browse to a web page they wish to annotate and copy and paste values into form fields. A user highlights values in the web page and then clicks on the form field to fill in a value. Figure 5 shows



Annotate a Form

Car	
Year	2003
Make	Nissan
Model	Altima
Mileage	99853
Color	Red Black
Feature	
○ Body	Body
○ Engine	Engine 4 Cylinder Gasoline Automatic
○ Accessory	Accessory
Price	6,990

2003 Nissan Altima For Sale

2003 Nissan Altima

Price: $6,990

Year:	2003
Make:	Nissan
Model:	Altima
Trim:	Base; S; SL
Engine:	4 Cylinder Gasoline
Trans:	Automatic
Fuel:	Gasoline
Color:	Red
Interior:	Black
Miles:	99853
VIN:	1N4AL11D73C115139
Stock #:	115139
Body Style:	Unspecified
Condition:	Used
Category:	Used Cars

Consumer Reviews

We found 2 reviews for 2003 Nissan Altima's

Read the reviews!

Fig. 5. A Filled in Form with a Source Data Page.

the price $6,990 highlighted and entered in the *Price* form field. To add several values to a multiple-value field, a user adds each to the field one at a time. The values "4 Cylinder", "Gasoline", and "Automatic" for example are all *Engine* features. To concatenate values that may be separate such as "Altima SL" and "Black Interior" in Figure 5, a user adds subsequent components by clicking on the plus icon instead of the pencil icon.

From the filled-in form, FOCIH can generate both a conceptual model, eventually to be represented as an OWL ontology, and an annotation document, eventually to be represented as RDF triples. Further, FOCIH also records the annotation information: (1) paths to leaf nodes in the DOM tree of an HTML page containing each value and, for concatenated values, each value component; (2) for each value the most specific instance recognizer from the data-frame library (e.g., string, number, year, make, model, color); and (3) enough left, right, and delimiter context within each leaf node to identify the value or values within the DOM-tree node. This enables FOCIH to harvest the same information from all machine-generated sibling pages from the same web site.

Details about our implementation of FOCIH are elsewhere [Tao08]. Since we are focusing in this paper on the role of conceptual modeling in our WoK tool set, be briefly explain here how FOCIH generates a conceptual model from a form specification. Figure 6 graphically shows the result of converting the form in Figure 4 to a conceptual model. We limit our explanation here to the generation of the features in this conceptual model.

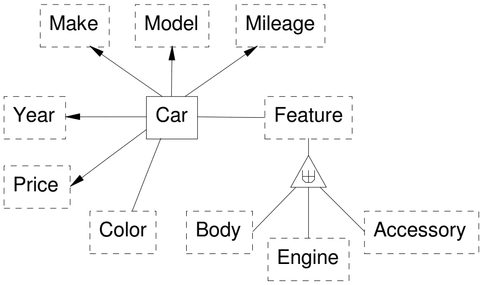


Fig. 6. Graphical View of the FOCIH-Generated Ontology.

A form title (e.g., *Car* in Figure 4) becomes a non-lexical object set (e.g., *Car* in a solid box Figure 6). Object identifiers represent values in non-lexical object sets (i.e., for each car, we generate an OID). A single-value field becomes a lexical object set functionally dependent on the object set of its enclosing form field. Thus, *Year*, *Make*, *Model*, *Mileage*, and *Price* are all lexical object sets (enclosed in dashed boxes, as opposed to solid boxes for non-lexical object sets), and each depends functionally on *Car*, the enclosing form field. As Figure 6 shows, we denote a functional relationship set with an arrow connecting a domain object set on the tail side to a range object set on the head side. A single-label/multiple-value field like *Color* in Figure 4 becomes a lexical object set linked non-functionally to the object set of its enclosing form field as Figure 6 shows. Because *Feature* is a choice form field, it is the generalization object set of a generalization/specialization. Its specializations are single-label/multiple-value form field, which implies the possibility of several features for each car and thus that the connecting relationship set between *Car* and the root generalization *Feature* should be many-many. Since no nested form field appears inside *Feature* itself (i.e., nested form fields only appear inside its specializations), all the values are in the specializations and therefore the union of the values in the specializations constitutes the values in the generalization. Hence, along with the choice being exclusive, this implies a partition constraint—the specialization object sets are mutually exclusive and their union constitutes all the values. The symbol ⊎ in the triangle in Figure 6 asserts that the specialization object sets *Body*, *Engine*, and *Accessory* partition the generalization object set *Feature*. Figure 6 faithfully represents all form-implied object sets, relationship sets, and implied constraints over the values in these object and relationship sets.

We note, however, that form specification, as provided in FOCIH, fails to provide some constraints. FOCIH provides no way to specify reverse cardinality constraints. We do not know from the form alone, for example, whether any single-value field or any combination of single-value fields constitutes a key for *Car*. FOCIH also fails to provide for mandatory/optional constraints. The mileage, for example, may not be listed in some car ads, and some car ads list no features. It is the form specification, not the underlying conceptual-modeling language that imposes these limitations. Potentially, we could augment the set of form features to allow a form creator to specify additional constraints—but at the expense of complicating form specification and likely losing the intuitive correspondence between FOCIH forms and forms in common usage. Thus, in the interest of maintaining simplicity, we do not add additional notation to FOCIH forms to capture these constraints or to capture other more advanced features of conceptual models. We do point out, however, that for some constraints, we can observe the data and adjust to what we see as we harvest information. For example, if we obtain enough evidence to conclude that a value in a form field uniquely identifies an object, we can infer that the form-field values are keys for the objects identified. We also point out that for the purpose of annotating web pages, neither reverse cardinality constraints nor optional/mandatory constraints matter very much, if at all.

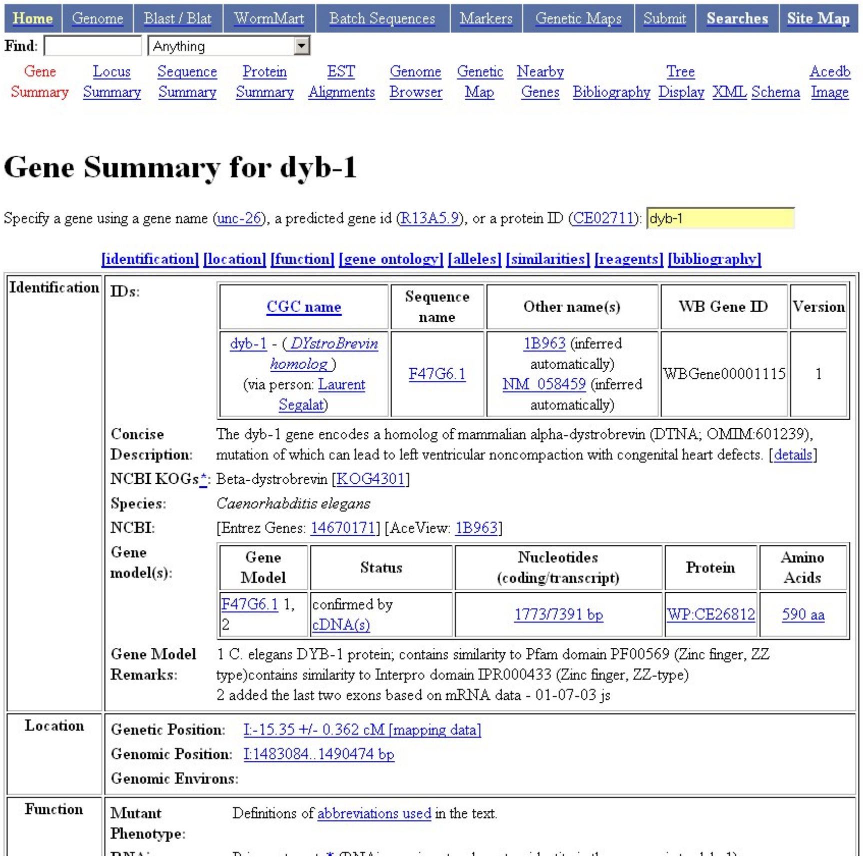
With the conceptual model generated and the data harvested from web pages with respect to the conceptual model, it is straightforward to generate an OWL ontology and RDF triples. In addition to the data, the RDF triples include annotation information: references to source pages and to each data item or data-item component within the source pages.

TISP/TISP^{++}

TISP is a tool that interprets tables in sibling pages [Tao08]. To interpret a table is to properly associate table category labels with table data values. Using Figure 2 as an example, we see that *Identification*, *Location*, and *Function* are labels for the large rectangular table. Inside the right cell of the first row is another table with headers *IDs*, *NCBI KOGs*, *Species*, etc. Nested inside of this cell are two tables, the first with labels *CGC name, Sequence name, Other name(s), WB Gene ID*, *Version* and the second with labels *Gene Model, Status, Nucleotides (coding/transcript)*, *Protein*, and *Amino Acids*. Most of the rest of the text in the large rectangular table comprises the data values. We associate labels with data values by observing the table structure. A cell in a table associates with its header label (or labels in the case of multi-dimensional tables). For nested tables, we trace the sequence of labels from data cell to the outermost label. Thus, for example, the associated label for the sequence-name value *F18H3.5* is the sequence of labels *Identification*, *IDs*, and *Sequence name*.

Although automatic table interpretation can be complex, if we have another page, such as the one in Figure 7, that has essentially the same structure, the system can usually obtain enough information about the structure to make automatic interpretation possible. We call pages that are from the same web site and have similar structures *sibling pages*.[6] The two pages in Figures 2 and 7 are sibling pages. They have the same basic structure, with the same top banners that appear in all the pages from this web site, with the same table title (*Gene Summary for* some particular gene), and a table that contains information about the gene. Corresponding tables in sibling pages are called *sibling tables*. If we

[6] Hidden-web pages are usually generated dynamically from a pre-defined templates in response to submitted queries; therefore they are usually sibling pages. Quite often hidden-web pages display their data in tables.



Gene Summary for dyb-1

Specify a gene using a gene name (unc-26), a predicted gene id (R13A5.9), or a protein ID (CE02711): dyb-1

[identification] [location] [function] [gene ontology] [alleles] [similarities] [reagents] [bibliography]

Identification

IDs:

CGC name	Sequence name	Other name(s)	WB Gene ID	Version
dyb-1 - (DYstroBrevin homolog) (via person: Laurent Segalat)	F47G6.1	1B963 (inferred automatically) NM_058459 (inferred automatically)	WBGene00001115	1

Concise Description: The dyb-1 gene encodes a homolog of mammalian alpha-dystrobrevin (DTNA; OMIM:601239), mutation of which can lead to left ventricular noncompaction with congenital heart defects. [details]

NCBI KOGs*: Beta-dystrobrevin [KOG4301]

Species: *Caenorhabditis elegans*

NCBI: [Entrez Genes: 14670171] [AceView: 1B963]

Gene model(s):

Gene Model	Status	Nucleotides (coding/transcript)	Protein	Amino Acids
F47G6.1 1, 2	confirmed by cDNA(s)	1773/7391 bp	WP:CE26812	590 aa

Gene Model Remarks: 1 C. elegans DYB-1 protein; contains similarity to Pfam domain PF00569 (Zinc finger, ZZ type)contains similarity to Interpro domain IPR000433 (Zinc finger, ZZ-type)
2 added the last two exons based on mRNA data - 01-07-03 js

Location

Genetic Position: I:-15.35 +/- 0.362 cM [mapping data]

Genomic Position: I:1483084..1490474 bp

Genomic Environs:

Function

Mutant Phenotype: Definitions of abbreviations used in the text.

Fig. 7. Sibling Page.

compare the two large tables in the main part of the sibling pages, we can see that the first columns of each table are exactly the same. If we look at the cells under the *Identification* label in the two tables, both contain another table with two columns. In both cases, the first column contains identical labels *IDs*, *NCBI KOGs*, ..., *Gene Model Remarks*. Further, the tables under *Identification.IDs* also have identical header rows. The data rows, however, vary considerably. Generally speaking, we can look for commonalities to find labels and look for variations to find data values.

Although we look for commonalities to find labels and look for variations to find data values, we must be careful about being too strict. Sometimes there are additional or missing label-value pairs. The two nested tables whose first column header is *Gene Model* in Figures 2 and 7 do not share exactly the same structure. The table in Figure 2 has five columns and three rows, while the table in Figure 7 has six columns and two rows. Although they have these differences, we can still identify the structure pattern by comparing them. The top rows in the two tables are very similar. Observe that the table in Figure 7 only has an additional *Swissprot* column inserted between the *Protein* and *Amino Acids* columns. It is still not difficult, however, to tell that the top rows are rows for labels.[7]

Given that we can interpret a table—find labels and values and properly associate them—our next task is to infer the general structure pattern of the table. Does the table have its labels across the top—as does the OnlineAthens table in Figure 1? Or, are the labels row headers—as are the labels in the table in the "2003 Nissan Altima" page in Figure 1? Or, does the table have both row and column headers? Or, is the table even more complex, having for example, tables nested inside one another such as the tables in Figures 2 and 7 or tables with labels that are implied or are in a tree structure? As implemented, TISP works only with tables that have labels as row headers, column headers, or both row and column headers where the row headers can be treated as values. Additionally, our TISP implementation works with tables having these structure patterns when they are nested inside one another.[8]

Observe that the structure patterns TISP can process are also structure patterns for FOCIH forms. We can therefore immediately generate an OSM ontology in the same way we generate an OSM ontology for FOCIH. Further, based on the TISP interpretation of the tables, we can also immediately populate the ontology and thus annotate the data. We call these additions to TISP, TISP^{++}. Figure 8 shows part of the ontology TISP^{++} generates for the sibling tables in Figures 2 and 7. Observe, for example, that nested under *Location* are three single-label/single-value location attributes—*Genetic Position*, *Genomic Position*, and *Genomic Environs*, the first two of which have values and the last of which has none. Thus, from the *Location* object set in Figure 8 emanate three functional relationship sets to these attributes, the last of which has optional participation. The non-functional relationship set between *Identification* and *Gene models* arises because a multiple-value form field—the nested table with labels *Gene Model*, ..., *Amino Acids*—appears as the only field nested in its outer structure, *Gene models.* This is similar to the nesting of the multiple-value form fields nested in *Feature* in Figure 4. The relationship between *Identification* and *IDs*, however, is functional because the (degenerate) table nested under *IDs* has only one data row in all known tables, and TISP^{++} therefore treats the degenerate table as a sequence of single-label/single-value form fields.[9]

[7] In our implemented TISP prototype, in addition to discovering the structure pattern for a web site, we also dynamically adjust the pattern if the system encounters a table that varies from the pattern. For example, if we had not seen the extra *Swissprot* column in our initial pair of sibling pages, TISP would add *Swissprot* as a possible label for the table when encountering it.

[8] Processing more complex structure patterns requires semantic enrichment procedures [Lyn08], which rely on semantic resources such as WordNet [Fel98].

[9] As TISP^{++} processes the sibling pages of a site, it may observe that in other sibling tables, the table nested under *IDs* is not degenerate. In this case, it would adjust the ontology, making the relationship from *Identification* to *IDs* non-functional.

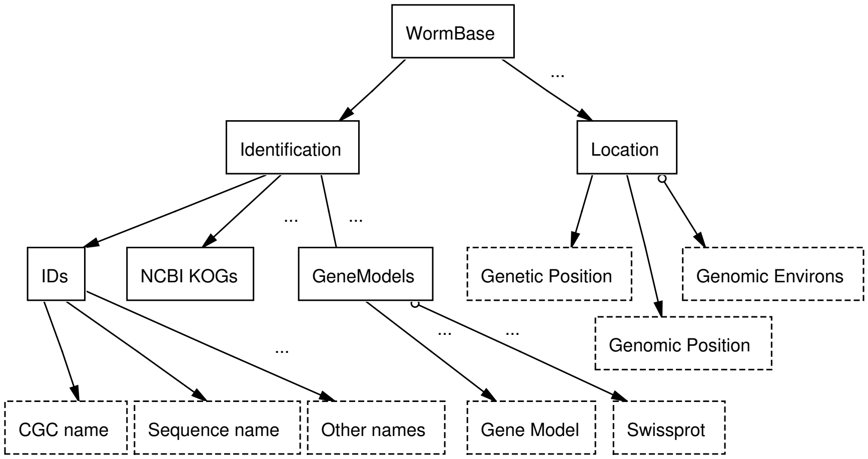


Fig. 8. Graphical View of the TISP^{++}-Generated Ontology.

Because of the isomorphic relationship between TISP tables and FOCIH forms, it is also possible to directly generate FOCIH forms. This leads to the possibility that users can enhance the TISP^{++}-generated ontology. Users may, for example, wish to rename the system-chosen name $WormBase$ with *Gene*—a more meaningful name for the root element of the ontology. Users may also wish to make the relationship set from *IDs* to *Other names* non-functional and show FOCIH, by example, how to recognize the list of other names, so that the names *XO136* and *NM 077855* in Figure 2 would be picked up out of the data cell and stored as individual names.

With or without any adjustments to the TISP^{++}-generated ontology, we are able to generate an OWL ontology and RDF triples. Again, as with FOCIH, we also have all the annotation information we need about the pages and the data items within the pages in a generated RDF file.

3.2 Query Tools

Given a generated file of RDF triples, we are immediately able to query the file using SPARQL. For typical, untrained users, however, we need a better way to query a WoK. Like current queries to web search engines, WoK queries will likely migrate to free-form text. Further, the free-form text is likely to be cryptic, keyword-based, and non-grammatical; an example is the query in Figure 3. How can we accept this kind query as input and produce a SPARQL query as output?

We base our approach on extraction ontologies. The essence of the idea is to (1) extract constants, keywords, and keyword phrases in a free-form query; (2) find the ontology that matches best; and (3) embed the query in the ontology yielding (3a) a join over the relationship-set paths connecting identified concepts, (3b) a selection over identified constants modified by identified operators, and

(3c) a projection on mentioned concepts. Both AskOntos [Vic06] and SerFR [AM07] implement this basic approach to free-form query processing.[10]

As a key feature of extraction ontologies, the concepts each have an associated data frame. A *data frame* describes information about a concept—its external and internal representations, its contextual keywords or phrases that may indicate the presence of an instance of the concept, operations that convert between internal and external representations, and other manipulation operations that can apply to instances of the concept along with contextual keywords or phrases that indicate the applicability of an operation. Figure 9 shows sample (partial) data frames for the concepts *Price* and *Make* for the ontology in Figure 6. As Figure 9 shows, we use regular expressions to capture external representations. The *Price* data frame, for example, captures instances such as "$4500" and "17,900". A data frame's context keywords are also regular expressions. The *Price* data frame in Figure 9, for example, includes context keywords such as "asking" and "negotiable". In the context of one of these keywords in a car ad, if a number appears, it is likely that this number is a price. The operations of a data frame can manipulate a concept's instances. For example, the *Price* data frame includes the operation *LessThan* that takes two instances of *Price* and returns a *Boolean.* The context keywords of an operation indicate an operation's applicability; context keywords such as "less than" and "<", for example, apply to the *LessThan* operation. Sometimes external representations are best described by lexicons. These lexicons are also regular expressions—simple lists of possible external representations—and can be used in place of or in combination with other regular expressions. In Figure 9, *CarMake.lexicon* is a lexicon of car makes, which would include, for example, "Toyota", "Honda", and "Nissan" and potentially also misspellings (e.g. "Volkswagon") and abbreviations (e.g. "Chev" and "Chevy").

We can apply an extraction ontology to obtain a structured representation of the unstructured information in a relevant document. For example, given that we have added data frames to the ontology in Figure 6, making it an extraction ontology, and given a car ad such as the first Nissan ad in the City Weekly page in Figure 1:

> **'93 NISSAN** Model XE, $900, Air Conditioning, new tires, sweet cherry red.
> For listings call 1-800-749-8104 ext. V896.

we can extract "**'93**" as the *Year*, "**NISSAN**" as the *Make*, "XE" as the *Model*, "$900" as the *Price*, "red" as the *Color*, and both "Air Conditioning" and "new tires" as *Accessory*s. As part of the extraction, the conversion routines in the data frames convert these extracted values to canonical internal representations, so that, for example, "**'93**" becomes the integer *1993* and "$900" becomes the integer *900.*

[10] SerFR builds on AskOntos by expanding its options to recognize more complex operations and to provide for advanced query specification and resolution. Further, SerFR can also link directly to web services.

```
Price
    internal representation: Integer
    external representation: \$?(\d+ | \d?\d?\d,\d\d\d)
    context keywords: price | asking | obo | neg(\.|otiable) | ...
    ...
    LessThan(p1: Price, p2: Price) returns (Boolean)
    context keywords: less than | < | or less | fewer | ...
    ...
end

Make
    external representation: CarMake.lexicon
    ...
end
```

Fig. 9. Sample Data Frames.

Now, consider the sample query in Figure 3 and assume that the ontology in Figure 6 is an extraction ontology—i.e. is augmented with data frames as illustrated in Figure 9. When we apply this extraction ontology to the query, the extraction ontology recognizes the highlighted strings in the query (see Figure 3). It recognizes "red" as a possible value for the *Color* object set, "Nissan" for *Model*, "$5000" for *Price*, "1990" for *Year*, and, with the aid of the recognized keyword "miles", "120K" as a *Mileage*. The conversion routines in the data frames normalize all these values, making the numbers integers, converting "red" to its RGB value, and standardizing car makes to have initial capital letters (no change to the string "Nissan" in this case). The extraction ontology also recognizes "under" as the less-than operator for *Price*, "or newer" as the greater-than operator for *Year*, and "less than" as the less-than operator for *Mileage*. From this extracted information along with the known ontological structure of the data provided by the generated OWL ontology and the data itself provided by the RDF triples, it is straightforward to generate a SPARQL query. In essence, the query searches for cars that satisfy the following constraints:

$$
\begin{aligned}
&\mathit{Year} \geq 1990\\
&\mathit{Make} = \text{`Nissan'}\\
&\mathit{Mileage} \leq 120000\\
&\mathit{ColorWithinRange}(255{,}0{,}0)\\
&\mathit{Price} \leq 5000
\end{aligned}
$$

Because we are processing queries under an open- rather than a closed-world assumption, we generate the SPARQL query with *OPTIONAL* clauses allowing it to return cars that, for example, have no color specified or no mileage specified so long as they meet the requirements for the fields where values do appear.

4 Conclusion

This work presents a grand vision of a "Web of Knowledge" (a "WoK")—a vision that others share [BL07]. A major barrier to realizing this vision is the overwhelming amount of human effort that appears to be required both for creating and querying WoK content. To surmount this barrier, we have described ontological and epistemological creation tools to significantly reduce or totally eliminate the barrier to creating WoK content, and we have described a query tool usable by anyone. FOCIH allows users with no training to specify simple ontologies and to annotate web pages with respect to these ontologies. FOCIH can also annotate and harvest specified information from all sibling pages of an initial hand-annotated web page. TISP/TISP^{++} uses sibling tables to interpret tables and from interpreted tables to generate ontologies and to annotate the information in these interpreted tables with respect to these generated ontologies. TISP/TISP^{++} is fully automatic, but is limited to information captured in sibling tables. AskOntos and SerFR provide for free-form query processing. And, although SerFR supports an advanced query interface for sophisticated users, less sophisticated users (most people) can only pose conjunctive queries and must limit their vocabulary to words, phrases, and symbols recognized by data frames associated with ontologies.

Conceptual modeling plays a key role in actualizing these ideas. An ontology is a conceptualization of a real-world domain in terms of object sets, relationship sets, generalizations, specializations, and constraints over these conceptualizations. Indeed an ontology can be thought of as a conceptual model grounded formally in a logic system. Automatic and semi-automatic ontology generation from data-rich, semi-structured web pages is akin to reverse engineering structured data into conceptual models—a task that has traditionally been associated with the conceptual-modeling community. Automatic and semi-automatic annotation of web pages can proceed bottom-up, occurring as a by-product of ontology generation via reverse engineering. Or annotation can proceed top-down, coming from extraction ontologies in which instance recognizers attached to conceptual object sets and relationship sets extract data on web pages with respect to conceptual models comprising these object and relationship sets. In either case, conceptual modeling plays the role of organizing this knowledge. For query processing, conceptual models grounded in description logics form a template to which free-form queries can be matched to yield formal queries to be processed by standard query engines.

Opportunities for future work abound. And many of these opportunities are best approached through conceptual modeling. Related to FOCIH, we see the following:

- Many OWL ontologies and ontologies in other structured forms already exist. We should be able to reverse-engineer them into FOCIH forms, which would immediately allow users to annotate web pages with respect to these existing ontologies. Users should also be able to alter these reverse-engineered forms and thus tailor them to suit their needs. Tailoring can also be a joint venture, which provides means for cooperative knowledge definition.

- Structured data repositories also already exist. We should also be able to reverse-engineer them into FOCIH forms. Further, we should be able to capture and annotate their data as well.
- Given an extraction ontology, in addition to being able to reverse-engineer it into a FOCIH form, we should also be able to use the extraction ontology to automatically do the initial form fill-in. A user could correct any annotation mistakes the augmented FOCIH system might make and complete the form fill-in for any data items it might miss. Information harvesting could then proceed with minimal user intervention—ideally, none at all.

Related to TISP/TISP^{++}, we see the following:

- TISP only interprets HTML tables. In principle, the ideas should apply to all tables, including Microsoft Word tables, Excel tables, and PDF tables, in which vast amounts of data appear.
- The idea of using sibling tables to identify category labels, data, and the relationships between category labels and data should extend beyond tables to semi-structured sibling pages in general. We should be able to identify and interpret lists and patterned layout as well as tables by sibling-page comparison.
- Because TISP works with HTML tables, which typically have simple label structures, it does very little to semantically enrich the tables it interprets. By considering semantic lexicons such as WordNet [Fel98] and other semantic resources such as a data-frame library, it is possible to considerably enrich interpreted tables by identifying generalizations, specializations, and aggregations within label structures and discovering constraints and interrelationships among data items not initially apparent in the data and meta-data of an interpreted table [Lyn08].

Related to free-form queries, we see the following:

- Often, even when users are unable to find the “right” vocabulary for making requests, a WoK system should be able to find an appropriate ontology. This ontology provides context for an interaction between system and user. We could exploit this context by exposing the vocabulary of the ontology and thereby allowing users to find the “right” words with which to ask their questions. It is also possible to generate, on the fly, a standard query form based on the ontology. Users should be able to fill in this form, as users do for typical HTML forms, to pose their queries. These forms would include drop-down selection lists for form entries with a short list of possibilities, range queries for ordered types, and facilities for more complex queries such as disjunctive queries and queries with negation.
- We can further explore advanced query specification. We could, for example, explore the use of natural-language processing techniques for generating logic statements [Rus04].
- It is possible to linguistically ground reasoning rules so that they too can play a role in free-form query processing. Further, then, in addition to providing

provenance for extracted facts by allowing users to click on results to see original sources, we can also provide provenance for inferred facts by exposing and explaining reasoning chains and showing how they eventually resolve into extracted facts.

Related to a data-frame library, we see the following:

- As FOCIH and TISP (or any other annotation technique) runs, we can enhance data frames. Data frames commonly have lexicons, and as new values are annotated, they can be added to these lexicons. For example, as new makes and models of cars become annotated, the envisioned WoK system can automatically add them to existing make and model lexicons.
- As FOCIH and TISP (or any other ontology-generation technique) creates ontologies, the envisioned WoK system could extract reusable knowledge components and store them in the data-frame library. For example, it is common to informally see car models as concatenations of what is technically the model and what is technically called the trim. (See Figure 1, which shows the model and trim in accord with these technical terms.) To accommodate both, a molecular-size knowledge component can describe *Model* as the aggregate concatenation of *Model* and *Trim*. In general, having a large collection of these molecular-size components in the data-frame library would aid in semantic enrichment and instance recognition.

We are implementing our WoK prototype with a MySQL backend database and programs written in Java. Currently, we have completed an initial implementation of FOCIH, TISP, and TISP^{++} as described here, and we have implemented them so that they run smoothly together with other tools such as OWL ontology generators, RDF instance generators, and a SPARQL query engine. We have also separately implemented prototypes for AskOntos and SerFR, and we have integrated a basic AskOntos query engine into our WoK prototype. In addition, we have established a data-frame library, but we have only begun to populate it with useful atomic and molecular-size data frames. Earlier versions of our extraction ontologies have been separately implemented. Currently, we are upgrading and integrating these prototypes into our WoK prototype. We have accomplished much, but, as always, there is much more to do.

References

[AK07] R. Al-Kamha. *Conceptual XML for Systems Analysis.* PhD dissertation, Brigham Young University, Department of Computer Science, June 2007.

[Alh03] R. Alhajj. Extracting the extended entity-relationship model from a legacy relational database. *Information Systems*, 28(6):597–618, 2003.

[AM07] M.J. Al-Muhammed. *Ontology Aware Software Service Agents: Meeting Ordinary User Needs on the Semantic Web.* PhD dissertation, Brigham Young University, Provo, Utah, August 2007.

[AMELT07] M.J. Al-Muhammed, D.W. Embley, S.W. Liddle, and Y. Tijerino. Bringing web principles to services: Ontology-based web services. In *Proceedings* *of the Fourth International Workshop on Semantic Web for Services and Processes (SWSP'07)*, pages 73–80, Salt Lake City, Utah, July 2007.

[BCL06] P. Buitelaar, P. Cimiano, and B. Loos. Preface. In *Proceedings of the 2nd Workshop on Ontology Learning and Population: Bridging the Gap Between Text and Knowledge, (COLING-ACL 2006)*, Sydney, Australia, July 2006.

[BL07] T. Berners-Lee. Future of the world wide web, March 2007. Testimony of Sir Timothy Berners-Lee Before the United States House of Representatives Committee on Energy and Commerce Subcommittee on Telecommunications and the Internet.

[BN03] F. Baader and W. Nutt. Basic description logics. In F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors, *The Description Logic Handbook*, chapter 2, pages 43–95. Cambridge University Press, Cambridge, UK, 2003.

[CBS94] R.H.L. Chiang, T.M. Barron, and V.C. Storey. Reverse engineering of relational databases: Extraction of an EER model from a relational database. *Data and Knolwedge Engineering*, 12(1):107–142, 1994.

[DEL06] Y. Ding, D.W. Embley, and S.W. Liddle. Automatic creation and simplified querying of semantic web content: An approach based on information-extraction ontologies. In *Proceedings of the First Asian Semantic Web Conference (ASWC'06)*, pages 400–414, Beijing, China, September 2006.

[ECJ$^+$99] D.W. Embley, D.M. Campbell, Y.S. Jiang, S.W. Liddle, D.W. Lonsdale, Y.-K. Ng, and R.D. Smith. Conceptual-model-based data extraction from multiple-record web pages. *Data & Knowledge Engineering*, 31(3):227–251, 1999.

[EKW92] D.W. Embley, B.D. Kurtz, and S.N. Woodfield. *Object-oriented Systems Analysis: A Model-Driven Approach.* Prentice Hall, Englewood Cliffs, New Jersey, 1992.

[Emb89] D.W. Embley. NFQL: The natural forms query language. *ACM Transactions on Database Systems*, 14(2):168–211, 1989.

[EZ10] D.W. Embley and A. Zitzelberger. Theoretical foundations for enabling a web of knowledge. In *Proceedings of the Sixth International Symposium on Foundations of Information and Knowledge Systems (FoIKS'10)*, pages 211–229, Sophia, Bulgaria, February 2010.

[Fel98] C. Fellbaum. *WordNet: An Electronic Lexical Database.* MIT Press, Cambridge, Massachussets, 1998.

[Foo07] Footnote.com. http://www.footnote.com, 2007.

[Gru93] T.R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[Gua98] N. Guarino. Formal ontologies and information systems. In N. Guarino, editor, *Proceedings of the First International Conference on Formal Ontology in Information Systems (FOIS'98)*, pages 3–15, Trento, Italy, June 1998.

[Hal95] T. Halpin. *Conceptual Schema & Relational Database Design.* Prentice Hall of Australia Pty. Ltd., Sydney, Australia, second edition, 1995.

[Jar07] M. Jarrar. Towards automated reasoning in ORM schemes: Mapping ORM into DLR_{idf} description logic. In *Proceedings of the 26th International Conference on Conceptual Modeling (ER'07)*, pages 181–197, Auckland, New Zealand, November 2007.

[LCWA07] N. Lammari, I. Comyn-Wattiau, and J. Akoka. Extracting generalization hierarchies from relational databases: A reverse engineering approach. *Data & Knowledge Engineering*, 63(2):568–589, 2007.

[Lyn08] S. Lynn. Automating mini-ontology generation from canonical tables. Master's thesis, Department of Computer Science, Brigham Young University, Provo, Utah, 2008.

[Rus04] V. Rus. A first evaluation of logic form identification systems. In R. Mihalcea and P. Edmonds, editors, *Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 37–40, Barcelona, Spain, March 2004.

[Smi03] B. Smith. Ontology. In L. Floridi, editor, *Blackwell Guide to the Philosophy of Computing and Information*, pages 155–166. Oxford: Blackwell, 2003.

[SPG$^+$07] E. Sirin, B. Parsia, B.C. Grau, A. Kalyanpur, and Y. Katz. A practical OWL-DL reasoner. *Journal of Web Semantics*, 5(2):51–53, 2007.

[Tao08] C. Tao. *Ontology Generation, Information Harvesting and Semantic Annotation for Machine-Generated Web Pages.* PhD dissertation, Brigham Young University, Department of Computer Science, December 2008.

[TEL$^+$05] Y.A. Tijerino, D.W. Embley, D.W. Lonsdale, Y. Ding, and G. Nagy. Toward ontology generation from tables. *World Wide Web: Internet and Web Information Systems*, 8(3):261–285, 2005.

[Vic06] M. Vickers. Ontology-based free-form query processing for the semantic web. Master's thesis, Brigham Young University, Provo, Utah, June 2006.

[W3Ca] OWL Web Ontology Language Reference Manual. www.w3.org/TR/owl-ref. W3C (World Wide Web Consortium).

[W3Cb] Resource Description Framework (RDF). www.w3.org/RDF. W3C (World Wide Web Consortium).

[W3Cc] SPARQL Query Language for RDF. www.w3.org/TR/rdf-sparql-query. W3C (World Wide Web Consortium).