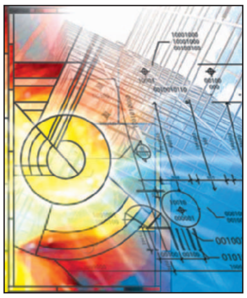
Talk to Your Semantic Web



Craig W. Thompson • *University of Arkansas*
Paul Pazandak • *Object Services and Consulting*
Harry R. Tennant • *Harry Tennant and Associates*

Natural language seems like a great idea for communicating with computers, yet that capability remains the grand challenge problem it was 30 years ago in the hey-day of question-answering systems. Back then, we knew how to parse sentences using augmented finite state machines and context-free and transformational grammars, and could translate results into SQL, predicate calculus, or equivalent semantic networks. We knew we'd need more powerful representations, so we targeted or invented temporal logics, modal logics, and nonmonotonic logics. But by the 1980s, we realized that we had only scratched the surface — just beneath was a complex of common-sense meaning representations, then referred to as frames, scripts, and knowledge representations. At present, we've made real progress in natural language technologies: keyword-based information retrieval is in everyday use; spelling and grammar checkers are commonplace; speech-to-text and text-to-speech utilities are widely available if not yet an integral part of how most people communicate with computers; and limited but useful translation services on the Web convert documents from one language to another. But we still cannot successfully use natural language to query and command computers.

To understand the difficulty, try typing or speaking to a system that has a natural language interface (NLI) — it won't understand many of your questions and commands because they overshoot the capabilities of the NLI system or the underlying application to which it interfaces. Not only that, you can't really know what you can ask about — does the system handle map or statistical queries? — so your questions and commands also undershoot the system's capabilities. This mismatch between user expectations and NLI system capabilities is called the *habitability* problem.

We've developed LingoLogic, an interface technology that uses menus to specify natural language queries and commands in order to combat the habitability problem.[1] It doesn't claim to provide deep understanding of natural language. Rather, we see it as a potentially widely useful human interface technology that can extend the Semantic Web.

Menu-Based Natural Language Interfaces

LingoLogic technology uses standard NLI technology in a directed completion-based way to restrict the user to performing commands and queries that the underlying system can understand. The technology targets an unfilled niche in user interface design that lets untrained users make complex queries and commands. LingoLogic interface development is much easier than traditional NLI development, which must cover every possible way of expressing a question or command.

The user composes a sentence either by typing it in or selecting items from a cascade of menus driven by a grammar and predictive parser. Domain experts provide domain-specific pop-up menu support for specifying values. When the user selects an expert, a menu supports the user in specifying a value (Figure 1).

The user can also choose from the following options:

- *Restart* erases the entire sentence and reinitializes the parser to begin again.
- *Rubout* erases the last phrase selected.
- *Translate* translates the sentence to a target language, such as SQL.
- *Execute* translates the sentence, sends a message to the target application, and displays the result.

In the case of relational databases, default standard grammars and translations are provided and can be automatically combined with DBMS

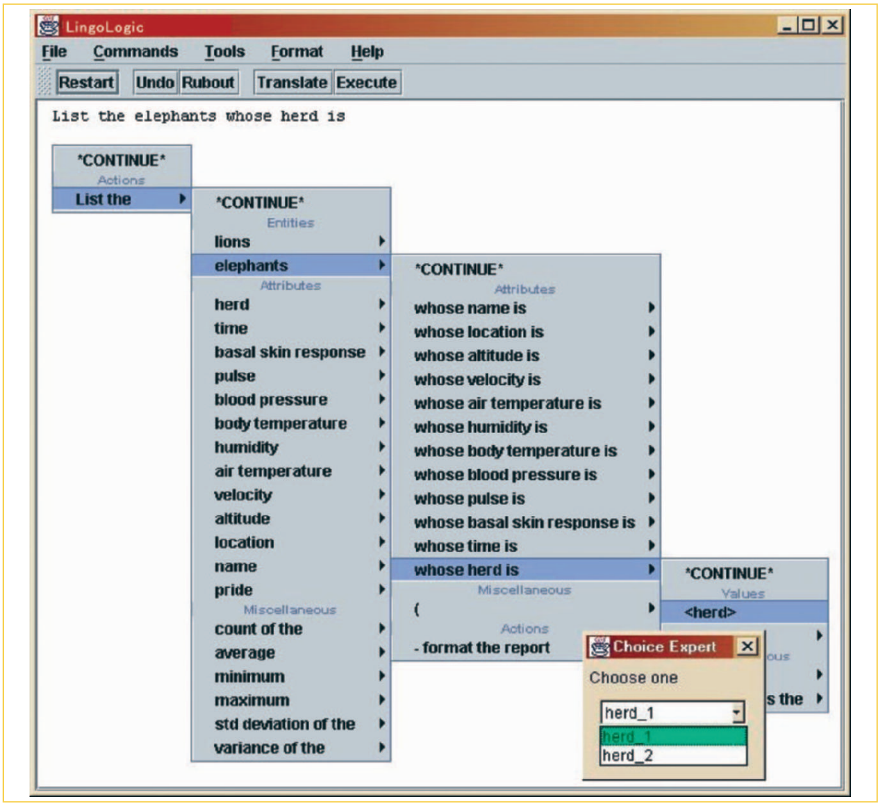

List the elephants whose herd is

Figure 1. Specifying a natural language query using a LingoLogic interface. A user selects words and phrases from menus to create a query. Using menus avoids the habitability problem and ensures that users cannot overshoot or undershoot the interface's capabilities.

schemas at runtime to automate the generation of LingoLogic interfaces to DBMS systems that use SQL. After phrasing a query, the user can see the translation or execute the query.

The LingoLogic interface developer — either an end user or someone who builds interfaces for end users — employs an interface generator to produce new interfaces for relational databases. The generator attaches to any open database connectivity (ODBC) database and extracts schema and other information. These parameterize a grammar and lexicon so that a new LingoLogic interface is ready for use at the push of a button, as Figure 2 shows.

LingoLogic System Architecture

In its simplest form, a LingoLogic system is a kind of translator. It acts similarly to how a compiler translates from a higher-level language source to a target lower-level language, except that it is left-to-right completion driven. The architecture, shown in Figure 3, consists of

- a user interface that displays a collection of menu selections. The user selects one of these and this choice is returned to the parser;
- a grammar and lexicon that defines the language the LingoLogic system recognizes, as well as defines how to translate to a target system's interface language;
- a parser that uses the grammar and lexicon, parses one word or phrase at a time, and computes the completion of the next legal words or phrases, which it then sends to the user interface.

After the user completes a question or command, the system translates it into a command that is sent to the back-end system for execution. For instance, if the translation results in a SQL query, the query is sent to a DBMS and an answer is returned.

Extending the Semantic Web with LingoLogic

In its earliest, simplest form, the World Wide Web consisted of client-side browsers that could display HTML documents to users — that is, it was a document server with a nice user interface. Now, though, the Web is much more sophisticated, and many of its pages are created on the fly. In the past several years, a more semantically rich Web has been the focus of much research,[2] with a goal of extending the Web so that programs can communicate with each other. Toward this end, Web page developers would add metadata to Web pages to include classification information and rules among other data. This extra information can be discovered and added to registries, and programs can then find the metadata and use it to better process other remote content. So far, the most successful form of metadata is Web service interfaces. These can be written in Web Service Description Language and registered in UDDI interface broker repositories on the Web; other programs can then connect to these remote Web services using SOAP, a lightweight XML-based messaging protocol.

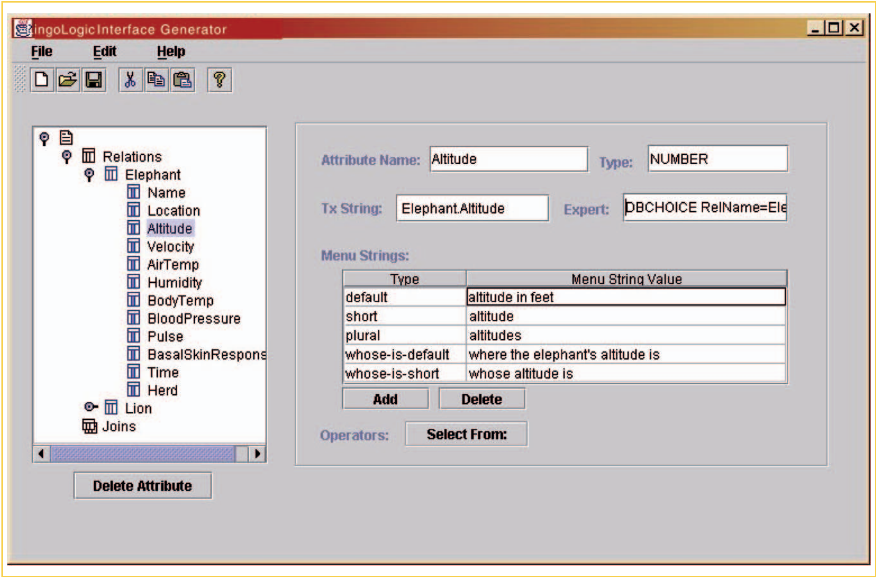
So far, there's no simple way to extend the Semantic Web with NLIs, but we've developed a recipe to accomplish this. To the Semantic Web metadata on a Web page, add the idea of a LingoLogic Interface Descriptor (LL-ID) as additional metadata that can be stored on any Web page. There might be several kinds of LL-IDs, such as a LingoLogic grammar and lexicon. The corresponding translations might translate to WSDL; that is, for a given WSDL command, there might be a LingoLogic phrase or sentence. Or the translations could map to other interface description languages or to SQL. In the latter case, the LL-ID could specify the DBMS from which to extract the schema (and login and password) and a template grammar and lexicon, and a LingoLogic grammar could be created on the fly.

Scaling LingoLogic to the Web requires changes to the simple LingoLogic architecture, as Figure 4 shows. Many users might be accessing the same or different Web pages, asking questions, so a parser farm would be necessary to have, with millions of parsers processing LingoLogic sentences that would translate to millions of target systems.

This extension would permit some users to browse and query stock Web pages, ask about price-to-earning ratios, and receive custom answers, while other users posed inquiries about recipes involving chocolate milk or price comparisons of digital cameras. Today's Web doesn't allow such precise questions, but the LingoLogic-enabled Semantic Web would.

Given that LingoLogic languages are small and domain-constrained, with a limited choice of words, speaker-independent speech recognition is probably feasible, and speaker-dependent speech recognition is clearly feasible. That means users could talk to their Web page content, a step beyond just speaking drop-down menu commands as found in some of today's dictation systems.

Of course, this scheme presumes the existence of a large number of LL-IDs. Because menus constrain the domain-specific language, the interface developer does not fall into the habitability trap of traditional natural language interfaces. The interfaces can be small and domain-specific. A cottage industry of LingoLogic interface developers could develop these interfaces for the rest of us.



Relations
- Elephant
  - Name
  - Location
  - Altitude
  - Velocity
  - AirTemp
  - Humidity
  - BodyTemp
  - BloodPressure
  - Pulse
  - BasalSkinRespons
  - Time
  - Herd
- Lion
- Joins

Attribute Name: Altitude Type: NUMBER

Tx String: Elephant.Altitude Expert: DBCHOICE RelName=Ele

Menu Strings:

Type	Menu String Value
default	altitude in feet
short	altitude
plural	altitudes
whose-is-default	where the elephant's altitude is
whose-is-short	whose altitude is

Add Delete

Operators: Select From:

Delete Attribute

Figure 2. The LingoLogic interface generator. The generator reduces the cost of developing new LingoLogic interfaces to DBMS systems. Rather than requiring a custom grammar for every DBMS, the interface generator extracts a schema from the DBMS; an interface designer then augments the specification with words and phrases mapping to relations, attributes, and joins. When this specification is automatically combined at runtime with a LingoLogic grammar targeted at SQL, the result is a new LingoLogic interface targeted at that DBMS.

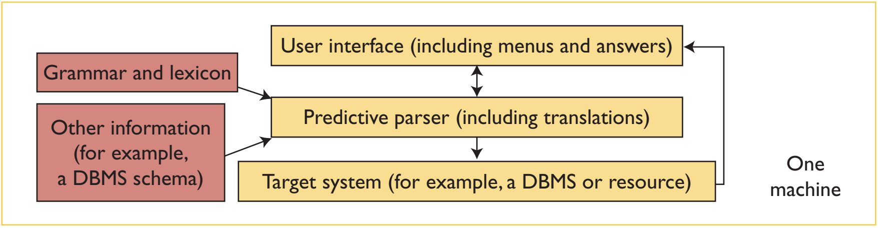

User interface (including menus and answers)
Grammar and lexicon
Predictive parser (including translations)
Other information (for example, a DBMS schema)
Target system (for example, a DBMS or resource)
One machine

Figure 3. LingoLogic system architecture. The architecture consists of an interface (cascading menus); a grammar for a particular domain; a predictive parser for recognizing strings, calculating next legal choices, and translating commands to a target system; and a target system that executes the commands and returns results.

Talking to the Internet of Things

An earlier Architectural Perspectives column described a coming Internet of Things where everything is alive, "a world in which common objects (including those that are inanimate and abstract) can have individual identities, memory, processing capabilities, and the ability to communicate and sense, monitor, and control their own behaviors."[3] A later column described "soft controllers," a kind of universal remote (actually a PDA with a pointer, GPS, and wireless) used to communicate with devices[4] in a move to extend the Internet and Web to the world of sensors, actuators, and devices. We can start by assuming that such devices (and even passive objects such as items and containers) have explicit identities. RFID technology provides one implementation. In addition, in this not-so-future world, such devices, sensors, or actuators will have APIs through which we can control them remotely. For instance, Open Geospatial Consortium's SensorML (www.opengeospatial.org/functional/?page=swe) and IEEE 1451 (the standard for transducer electronic data

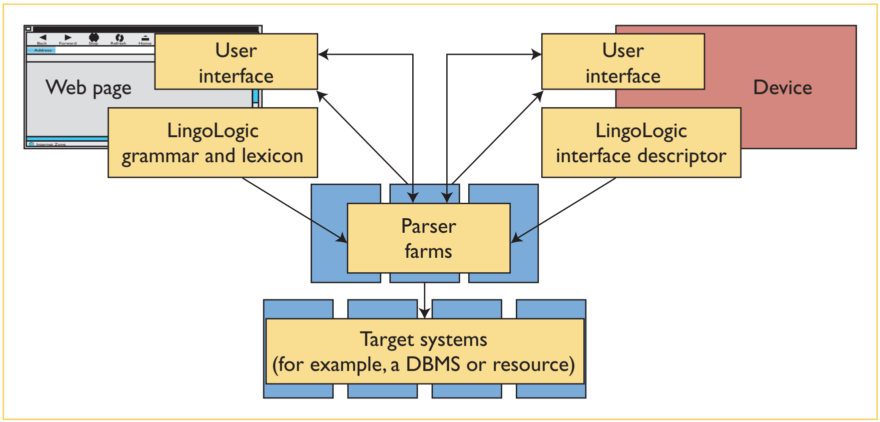

User interface
Web page
LingoLogic grammar and lexicon
User interface
Device
LingoLogic interface descriptor
Parser farms
Target systems (for example, a DBMS or resource)

Figure 4. Scaling the LingoLogic architecture. The LingoLogic architecture could be scaled to the Semantic Web by separating the interface from the parser and supporting a plethora of grammars and a parser farm. Any user at any machine could then start a query on any grammar, which will be parsed by some parser in the farm and executed by some back-end application.

sheets) are developing smart sensor technology.[5] Other metadata can be associated with device identities and APIs using normal Web mechanisms. For instance, we could associate a history wrapper with a light that recorded each time the light was turned on or off, as well as a scheduler wrapper that controlled the object according to a given schedule, or a security wrapper that allowed only certain people or programs to control the API.

For example, assume a room contains a light, thermostat, clock, and picture. If you pointed your universal remote at any of these, its identity, API, and associated metadata becomes available. If the "device" had a LingoLogic-ID (was LingoLogic-enabled), its interface descriptor would become available and you could "talk" to it using LingoLogic:

- "Light, turn on." ... "How many times have you been turned on in the past week?"
- "Tivo, record *Alias* and *24* reruns from season one."
- "Picture, who painted you?"
- "Kitty, where are you hiding?"

None of these interactions use the full spectrum of natural language. Instead, things could understand just small domain-restricted sublanguages related to their APIs (the things they can understand).

Beyond building and deploying a world-wide LingoLogic infrastructure complete with speech (pretty straightforward), there are some interesting technical challenges. We will want to talk not just to individual things but to collections of things. Sometimes this involves plurals like "Bedroom lights, turn off," and sometimes it involves rules referencing multiple objects such as, "If the tree needs water and the Internet weather service says rain is not in the forecast for the next two days, then turn on the sprinklers." Because we have individual grammars for individual kinds of things, we will need a way to combine grammars. Other operations on grammars can turn rules on and off on the basis of the user's access privileges, which can be reflected in the LingoLogic interface. This could restrict who could watch what stations, control the thermostat, or activate a smart car.

If we build LingoLogic and enough grammars to become interesting, and extend it to the Internet of Things, that will extend the Semantic Web in a straightforward way. Will it be enough to add natural language to our everyday interactions with computers? We predict so — we'll know in five to 10 years.

References

1. H. Tennant et al., "Menu-Based Natural Language Understanding," *21st Meeting of the Assoc. for Computational Linguistics*, Assoc. for Computational Linguistics, MIT, 1983, pp. 151–158.
2. T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Scientific Am.*, May 2001, pp. 34–43.
3. C.W. Thompson, "Everything is Alive," *IEEE Internet Computing*, vol. 8, no. 1, 2004, pp. 83–86.
4. C.W. Thompson, "Smart Devices and Soft Controllers," *IEEE Internet Computing*, vol. 9, no. 1, 2005, pp. 82–85.
5. *Draft Standard 1451 for A Smart Transducer Interface for Sensors and Actuators – Common Functions, Communications Protocols and Transducer Electronic Data Sheets (TEDS) Formats*, IEEE P1451.0 Working Group, Mar. 2005, http://grouper.ieee.org/groups/1451/0/.

Craig W. Thompson is a professor and Charles Morgan Chair in Database at the University of Arkansas and president of Object Services and Consulting, a middleware research company. His research interests include data engineering, software architectures, middleware, and agent technology. Thompson has a PhD in computer science from the University of Texas at Austin. He is a senior member of the IEEE. Contact him at cwt@engr.uark.edu.

Paul Pazandak is a senior computer scientist at Object Services and Consulting. His research spans distributed and agent-based systems, middleware, net-centric warfare, unmanned aerial vehicles, multimedia languages, and natural language interfaces. Contact him at pazandak@objs.com.

Harry R. Tennant is president of Harry Tennant & Associates, an Internet consulting firm focused on applications for K–12 education. While at Texas Instruments, he conceived of the notion of menu-based natural language interfaces. Tennant has a PhD in Computer Science from the University of Illinois. Contact him at harry@htennant.com.