Extending Semantic Interoperability To Legacy Systems and an Unpredictable Future

John F. Sowa

VivoMind Intelligence, Inc.

Abstract. This talk discusses issues of semantic interoperability, the role of ontology, and the requirements for supporting a graceful migration from legacy systems and for accommodating unexpected circumstances that may arise in the future. A central theme is the role of formal systems of logic, ontology, and knowledge representation. To provide some perspective on current systems and the potential for the future, it begins with a survey of some historical developments in computer science and artificial intelligence.

15 August 2006

Collaborative Expedition Workshop, National Science Foundation, Arlington, VA

Interoperability and Ontology

Computer systems interoperate by passing messages.

Every message has a meaning (semantics) and a purpose (pragmatics).

The role of ontology is to make the semantics and pragmatics explicit in terms of the people, places, things, events, and properties involved.

Questions:

- How can ontology facilitate interoperability?
- What if the systems have different ontologies?
- Or no explicit ontology?
- Or a dynamically changing ontology?
- How can we support a smooth migration path from legacy systems and to future systems?

To Predict the Future — Look at the Past

Questions:

- How have formal, logic-based ontologies been used?
- Are there any success stories for formal ontologies?
- Are there any success stories for formal methods of any kind?
- How do they compare with informal methods?
- Can formal and informal systems coexist?
- Is it possible to migrate from informal systems to formal ones?
- Is it possible to derive formal structures from informal resources?

Early History

1940s:

- Vannevar Bush: Proposed the WWW in 1945.
- Warren Weaver: Suggested machine translation as an important application for "Giant Brains".

1950s:

- Formal grammars for natural and artificial languages (by Noam Chomsky and John Backus).
- Semantic networks for machine translation.
- Knowledge representation languages for artificial intelligence.
- Hao Wang's theorem prover took a total of 7 minutes to prove the first 378 theorems of *Principia Mathematica* on an IBM 704, averaging about 1.1 seconds per theorem on an 83KHz vacuum-tube machine with a maximum of 144K bytes of main memory.

Vannevar Bush's article of 1945: http://www.theatlantic.com/doc/194507/bush

Warren Weaver's memo: http://ourworld.compuserve.com/homepages/WJHutchins/Weaver49.htm

Wang's paper on theorem proving: http://www.research.ibm.com/journal/rd/041/ibmrd0401B.pdf

A history of semantic networks: http://www.jfsowa.com/pubs/semnet.htm

The New Field of Computer Science

1960s:

- First compiler-compilers based on formal methods.
- First Object-Oriented Programming Language (Simula 67).
- Vienna Definition Language (VDL) as a basis for a formal methodology (VDM).
- Ted Codd proposed logic as a foundation for relational databases.
- Petri defined networks for specifying interacting events.
- Dijkstra designed a provably correct operating system.
  In 5 years of operation, the only bugs were minor coding errors.
- Mac Hack chess program beat the philosopher Hubert Dreyfus,
  who had claimed that a chess program could never beat an amateur.
- IBM implemented the Generalized Markup Language (GML),
  which later evolved into SGML, HTML, and XML.
- Research terminated on the Georgetown Automatic Translator (GAT),
  but under the name Systran it is very widely used today,
  and under the name Babelfish it is available on the WWW.

Dijkstra: http://www.cs.virginia.edu/~zaher/classes/CS656/p341-dijkstra.pdf

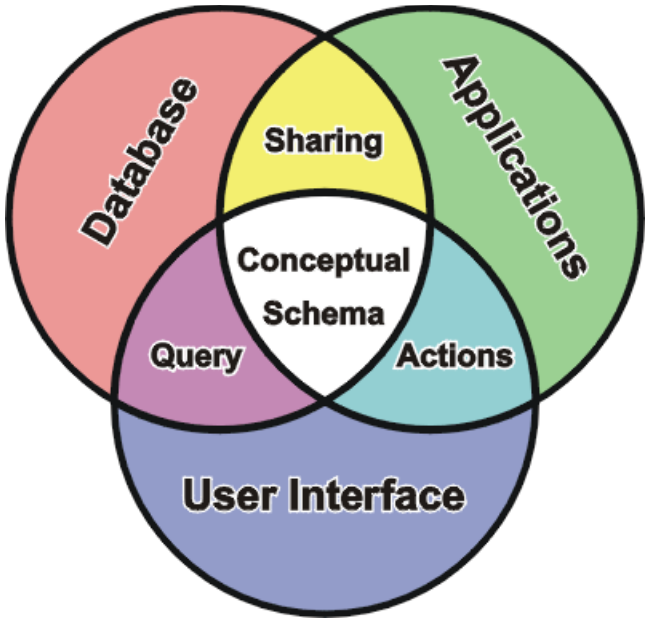
---

Computer Science Reaches Maturity

1970s:

- Xerox PARC invented the WIMPy user interface
  (Windows, icons, menus, pointing device).
- Algorithmic complexity theory and NP completeness.
- IBM designed SQL as a logic-based query language for relational DBs.
- A tiny company called Oracle implemented SQL for the CIA.
- Rule-based expert systems, such as MYCIN and OPS5.
- Prolog was the first logic-programming language.
  Ted Codd said that he wished he had invented Prolog.
- Natural language query systems supported full first-order logic.
  Users loved them, but defining the vocabulary was a major obstacle.
- ANSI/SPARC Conceptual Schema for facilitating DB interoperability.
- In 1976, first published paper on conceptual graphs proposed them as a schema language for databases and NL interfaces: http://www.research.ibm.com/journal/rd/204/ibmrd2004E.pdf

ANSI/SPARC Conceptual Schema



Three-schema architecture was proposed as the basis for interoperable databases in 1978.

Later revived as an ISO standards project. But in 1999, it ended as a technical report.

To order CD-ROMs with ACM publications on database systems dating back to the 1960s:
http://www.informatik.uni-trier.de/~ley/db/anthology.html

The PC Revolution

A new generation of PC users are unaware of anything that went before.

Many great new "killer apps" — e.g., spreadsheets.

Most of the mistakes and some of the lessons of the past are repeated.

IBM mainframes begin a long, slow decline.

Old timers continue to do research on formal methods.

But anything that cannot be done by WIMPy tools is ignored.

The Unified Modeling Language (UML) has prettier diagrams than VDM.

Those diagrams could be defined in logic, but they're not.

Without a formal definition in logic, the diagrams lack the precision and coherence of VDM.

World's Largest Ontology Project

Cyc project started in 1984 by Doug Lenat.

- Name comes from the stressed syllable of *encyclopedia*.
- Goal: implement the commonsense knowledge of an average human being.
- After $70 million and 700 person-years of work,

  600,000 categories
  defined by 2,000,000 axioms
  organized in 6,000 microtheories.

For more info, see http://www.cyc.com

Project Halo

Project for evaluating methods of knowledge representation.

Goal: Build an intelligent tutor.

Test case: Encode knowledge from a chemistry textbook in order to answer questions on a freshman chemistry exam.

Three companies participated: Cycorp, OntoPrise, SRI International.

Results:

- None of the three systems could read the English statements of the questions. For each of them, some knowledge engineer had to translate each question to the knowledge representation language used by that system.
- Average score: about 40% to 47% correct.
- Cost to encode knowledge: average about $10,000 per page from the textbook.
- Despite its large knowledge base, Cyc had the lowest score.
- Contrary to expectations, the large amount of general-purpose knowledge was not very helpful for enabling Cyc to acquire the special-purpose, domain-dependent knowledge.
- In some cases, general-purpose knowledge can be a distraction, and it is better to start from scratch with a representation that is tailored to the task.

Reference: http://www.aifb.uni-karlsruhe.de/~sst/Research/Publications/2004/ai-mag-preprint04.pdf

Some commentary on this and related issues with a somewhat less positive "spin":

The Challenge of Knowledge Soup: http://www.jfsowa.com/pubs/challenge.pdf

Lessons to be Learned

Why did UML succeed, but VDL or VDM was largely ignored?

- UML is much easier to learn.
- UML adds useful features as incremental, evolutionary additions to current technologies for software design and development.
- VDM has had some very successful applications, but it requires a considerable amount of training in logic and methods of formal definition.

Why hasn't Cyc been used in commercial applications?

- Cyc introduced a radically new paradigm that has very little support for anything that commercial systems currently do.
- The Cyc language (CycL) is just as hard to learn and use as VDM.
- Cyc has focused on "pure" research — that's not bad, but...

A book about VDM, which is also a good introduction to logic and formal methods:
http://www.csr.ncl.ac.uk/vdm/ssdvdm.pdf.zip

Recommendations

Take advantage of what developers already know:

- SQL is the most widely used logic-based notation on earth.
- UML diagrams haven't been formally defined, but they can be defined and used as precisely as VDM.
- Use controlled English as a supplement to UML.

Focus on bottom-up tasks, rather than top-down ontologies:

- Look at the kinds of messages generated for a given task.
- Two people (or computer systems) can agree on the semantics of a specific task without realigning their global ontologies.
- Even legacy systems can perform useful tasks while passing messages to a more sophisticated knowledge-based system.

Use ISO Common Logic as the underlying formalism for the database language, the UML diagrams, and controlled English.

Even more important, integrate Common Logic with popular development methodologies.

ISO Common Logic

CL is a framework for a family of logic-based languages:

- Three general-purpose dialects: CLIF, CGIF, and XCL.
- With a semantics that is a superset of the semantics of many other logic-based languages — including RDF, OWL, and SQL.
- With an abstract syntax that can be specialized to the concrete syntaxes of other logic-based languages.
- Designed to preserve the semantics when information is interchanged among heterogeneous systems.

Purpose: Guarantee that content exchanged between CL-conformant languages has the same semantics in each language.

CL web site: http://cl.tamu.edu

Representing Rules in Controlled English

Attempto Controlled English:

```
If a copy of a book is checked out to a borrower
   and a staff member returns the copy
then the copy is available.
```

```
If a staff member adds a copy of a book to the library
   and no catalog entry of the book exists
then the staff member creates a catalog entry
        that contains the author name of the book
           and the title of the book
           and the subject area of the book
   and the staff member enters the id of the copy
   and the copy is available.
```

These statements can be automatically translated to or from CL.

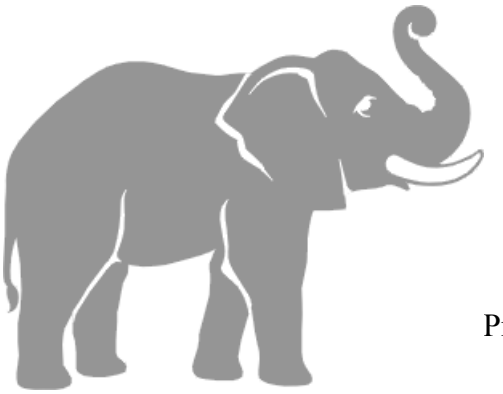
Web site for controlled natural languages:

http://www.ics.mq.edu.au/~rolfs/controlled-natural-languages/

Use of Controlled English

- Controlled English is a formal language that can be read by people who have never studied logic or programming languages.
- Controlled English is not true English, and it requires tools that ensure the authors stay within the limited subset.
- Can be implemented in integrated development tools.
- Can be generated automatically for help and diagnostics.
- Can serve as an agent communication language — which is readable by both humans and computers.
- But the real challenge is to deal with legacy systems.

Elephant 2000



I meant what I said, and I said what I meant.
An elephant's faithful, one hundred per cent.

Moreover,

An elephant never forgets.

Proposal by John McCarthy:

A language based on logic and speech acts.

McCarthy's original paper: http://www-formal.stanford.edu/jmc/elephant/elephant.html

A paper based on the ideas:

Architectures for Intelligent Systems, by John F. Sowa, http://www.jfsowa.com/pubs/arch.htm

Abstract. People communicate with each other in sentences that incorporate two kinds of information: propositions about some subject, and metalevel *speech acts* that specify how the propositional information is used — as an assertion, a command, a question, or a promise. By means of speech acts, a group of people who have different areas of expertise can cooperate and dynamically reconfigure their social interactions to perform tasks and solve problems that would be difficult or impossible for any single individual. This paper proposes a framework for intelligent systems that consist of a variety of specialized components together with logic-based languages that can express propositions and speech acts about those propositions. The result is a system with a dynamically changing architecture that can be reconfigured in various ways: by a human knowledge engineer who specifies a script of speech acts that determine how the components interact; by a planning component that generates the speech acts to redirect the other components; or by a committee of components, which might include human assistants, whose speech acts serve to redirect one another. The components communicate by sending messages to a Linda-like blackboard, in which components accept messages that are either directed to them or that they consider themselves competent to handle.

Challenge to AI for the Next Fifty Years

Position paper by Alan Bundy, a pioneer in automated problem solving:

- A few minutes studying any particular representation rapidly reveals deficiencies in expressivity or efficiency or both.
- The world is infinitely complex, so there is no end to the qualifications, ramifications and richness of detail that one could incorporate, and that you might need to incorporate for a particular application.
- For a narrow application, it is often sufficient to hand-craft a representation that hits the desired sweet spot.
- In general, the representation itself needs to be manipulated automatically.
- Such manipulation must be able to change the underlying syntax and semantics of the ontology.
- We believe that automatic representation development, evolution and repair must be a major goal of artificial intelligence research over the next 50 years.

Source: http://www.inf.ed.ac.uk/publications/online/0836.pdf

This view is consistent with the paper, "The Challenge of Knowledge Soup":

http://www.jfsowa.com/pubs/challenge.pdf

Conclusions

Legacy systems survive for many decades —
and their ontologies are inherited by their successors.

Communications among people ***and*** computers are always
based on task-oriented ontologies.

Those ontologies are bottom-up, highly specialized, and usually de facto.
Example: Amazon.com ontology for transactions, which suppliers are forced to adopt.

Formal definitions are important for both upper and lower ontologies.

Upper-level ontologies are important as guidelines.

When conflicts occur, the lower level wins.

At every level, intentions, expressed in speech acts, are fundamental.

But logic-based languages must be seamlessly integrated with the popular methodologies
that people use for application design, development, ***and*** maintenance.

If you want people to be virtuous, make virtue the path of least resistance.