Miscellaneous

John F. Sowa

16 October 2019
History of Interoperable Systems

1900 to 1959: Data representation and formatting.
   ● From punched cards to COBOL records and RPG reports.
   ● Fortran, LISP, COBOL, Algol, formal grammars.

1960s: Multiple concurrent programs that use the same data.
   ● Data structures, databases, data models, locking, virtual memory.
   ● PL/I, APL, SNOBOL, Simula 67, Algol 68, Pascal.
   ● Theorem provers, formal semantics, specification languages.

1970s: DB wars, conceptual schema, expert systems.

1980s: Knowledge bases, object-oriented systems. *

1990s: Shared reusable KBs, Need for ontologies, RDF.

2000s: Semantic Web, large ontologies, but slow adoption.

2010s: Some success stories, but no universal ontology.

* For more detail from 1980 to the present, see http://www.jfsowa.com/ikl
A single, universal ontology is more a hope than a reality.

No large hardware or software system can have a uniform ontology for every component.

Vagueness and ambiguity in natural languages result from the need to talk about anything from any possible perspective.

K. Janowicz and P. Hitzler: *

- “Meaning is not static, but dynamically reconstructed during language use.”
- “We should exploit the power of Semantic Web technologies and knowledge patterns to directly establish interoperability between purpose-driven ontologies without having to agree on a universal level before.”

* The Digital Earth as knowledge engine, Semantic Web (2012) vol. 3.
Three-schema architecture by ANSI SPARC in 1978:

- Conceptual schema defines the semantics of a database.
- Physical schema defines the storage and access methods.
- Application schema defines the APIs for programming languages.
The Conceptual Schema

Shared ontology to resolve the database wars of the 1970s.

- Early debates led to an ANSI technical report in 1979.
- Further discussions led to an ISO technical report in 1987.
- The Semantic Web became the next great hope.
DARPA Agent Markup Language

The diagram summarizes the requirements for the DAML project.

- From a presentation by Jim Hendler, the DARPA project manager. *
- The PI of the winning proposal was Tim Berners-Lee.

* See http://www.jfsowa.com/ikl/Hendler00.pdf
The diagrams show the evolution of the DAML project.

- The winning proposal in 2000 added detail to Hendler’s version.
- The diagram of 2001 moved logic to the side.
- In the final report, the “unifying logic” looks like an afterthought.

Hendler wrote that DAML must support heterogeneous systems.

- Tim B-L emphasized heterogeneous systems in his proposal.
- But the final report did not mention the word 'heterogeneous'.
The original diagram embodied many good ideas.

But building semantics on top of syntax was not one of them.

Result: Incompatible notations with unifying logic as a future hope.
Human Interfaces

Controlled English
Controlled Spanish
Controlled Chinese
FLIPP Diagrams
Topic Maps
Concept Maps
UML Diagrams

Common Logic

Machine Interfaces

XCL

SQL, Prolog, RDF(S), OWL

OCL, Datalog, RuleML

CGIF, CLIF
Approaches to Interoperability

A map of the heterogeneous methods that must be related. *

- Methods at the top address semantic issues.
- Methods in the middle address the external interfaces.
- Methods at the bottom address the low-level APIs.

Mapping Logics to Logics

Distributed Ontology, Model, and Specification Language. *

- An OMG standard for relating multiple logics.
- Common Logic is the most general.
- Open-source software is available to do the mappings.

* See the OMG standard for DOL: http://www.omg.org/spec/DOL/
Distributed OMS Language (DOL)

Mapping ontologies: *

- **OMS**: Ontology, Model, and Specification.
- **Goal**: Map an OMS expressed in one logic to equivalent versions in other logics.
- The diagram at right shows possible mappings.
- The target logic must have the same or greater expressivity.
- **Common Logic (CL)** is the most expressive logic shown.

Two related strategies:

- **Dialects**: Adopt a highly expressive logic such as CL or IKL as the base, and define all other logics as dialects of the base logic.
- **Mappings**: Use DOL to specify mappings (morphisms) among logics, but no logic is treated as a dialect of any other.

A Hierarchy of Mathematical Theories

See the Heterogeneous Tool Set, http://theo.cs.uni-magdeburg.de/Research/Hets.html
Three Kinds of Acyclic Graphs

A directed acyclic graph (DAG) has no cycles.
A tree is a DAG in which each node has exactly one parent.
Trees are used to show single inheritance paths, but other DAGs can show multiple inheritance paths.
A lattice is a DAG that shows all possible inheritance paths.
The word *hierarchy* is often used as a synonym for DAG.
Lattice of Theories

For any given logic, the set of all possible theories expressible in that logic forms a lattice.

The ordering is defined by specialization and generalization. Adding axioms to a theory creates a more specialized theory. Deleting axioms creates a more generalized theory.
For any logic L, the set of theories expressible in L forms a lattice.

- Every hierarchy of microtheories in L is a subset of that lattice.
- Top: The universal theory, which has no axioms, is always true.
- Bottom: The absurd theory, which has all axioms, is always false.
- Contraction: Delete axioms to form a more general theory.
- Expansion: Add axioms to form a more specialized theory.
- Revision: Contraction followed by expansion.
- Relabeling: Rename one or more predicates in a theory.
Formal Concept Analysis (FCA)

A theory with supporting algorithms and methodology:

- **Theory.** Define a minimal lattice that shows all inheritance paths among a set of concept types, each defined by a list of attributes.

- **Algorithms.** Efficient ways of computing the minimal lattice from a specification of concepts and attributes.

- **Methodology.** Techniques for describing concept types by attributes and using lattices for organizing ontologies and inference methods.

Applications:

- Ontology development and alignment; classification methods; machine learning; defining concepts used in other logics.

- FCA tools are commonly used to show that ontologies specified in OWL and other notations are consistent.

The FCA Homepage:  [http://www.upriss.org.uk/fca/fca.html](http://www.upriss.org.uk/fca/fca.html)

For deriving lattices from lexical resources:  [http://www.upriss.org.uk/papers/jucs04.pdf](http://www.upriss.org.uk/papers/jucs04.pdf)
Generating Lattices Automatically

FCA algorithms used the data in Roget’s Thesaurus to generate this lattice for the word 'happy' and its hypernyms (supertypes).

To generate this or similar lattices, enter 'happy' or any other word at the web site http://www.ketlab.org.uk/roget.html
Cyc Ontology

General-purpose upper level and a large hierarchy of microtheories.

See http://www.cyc.com and http://opencyc.org
A knowledge base with a very large formal ontology. *
- Over 600 reasoning modules, optimized for a variety of problems.
- SKSI for structured data and NLP for unstructured documents.

* See https://www.academia.edu/16911744/Common_Sense_Reasoning_From_Cyc_to_Intelligent_Assistant
Semantic Knowledge Source Integration

Goal: A logic, ontology, and system that can relate everything.

- Problem: Heterogeneous DBs and KBs may use different ontologies.
- But punched-card systems have been sharing data since 1900.
- And DBpedia relates independently developed web pages.
- An underspecified ontology may ignore some irrelevant details.

What reasoning methods can determine what is relevant?
Multiple paradigms and a growing number of modules. *
- For Jeopardy!, one API and about 100 reasoning methods.
- Now, a few dozen APIs and growing. **
- Versions of all the major paradigms for natural language processing, statistical, symbolic, semantic, pragmatic. ***

** Example: The Watson Tone Analyzer Service.
Generating a Response for Jeopardy!

Multiple steps that use a variety of algorithms:
1. Parse the question and analyze the relationships among key phrases.
2. Generate hypotheses, find evidence for them, and estimate their quality.
3. Combine the best hypotheses in possible answers.
4. Rank the answers by a confidence measure.
5. Select the best one and respond “Who is Edmund Hillary?”
6. Use feedback about success or failure for “dynamic learning.”
Finding Associations

The strength of association is fuzzy and context dependent.

Different algorithms for different kinds of associations:
- Indexes for finding co-occurrences in the data.
- Searching a network to find shortest paths.
- Deduction from definitions and axioms.

But Watson chose climate instead of religion as the context:
- Jeopardy! clue: “This kind of meat should not be shipped to Iraq.”
- Incorrect response: “What is reindeer?”
Tools for Processing Multiple Logics

Architecture of the heterogeneous tool set Hets

Tools for specific logics
- Text
- Parser
- Abstract syntax
- Static Analysis
  - (Signature, Sentences)
- Interfaces
  - XML, Aterm
- Theorem provers
- Rewriters
- Conservativity checkers
- Model finders

Logic graph
- Haskell
- Isabelle
- HasCASL
- SoftFOL
- CoCASL
- CASL
- ModalCASL
- CASL-DL
- OWL-DL

Grothendieck logic
(Flattened logic graph)

Tools for heterogeneous specifications
- Text
- Parser
- Abstract syntax
- Static Analysis
- Global Environment
- Interfaces
  - XML, Aterms
  - WWW, GUI
- Heterogeneous development graphs
  - Heterogeneous inference engine
  - Decomposition of proof obligations
  - Management of proofs
- Heterogeneous proof trees

http://www.informatik.uni-bremen.de/agbkb/forschung/formal_methods/CoFl/hets/index_e.htm
Describing Things in Different Ways

How can we describe what we see?
In ordinary language?
In some version of logic?
In a relational database?
In the Semantic Web?
In a programming language?

Even when people use the same language, they use different words and expressions.

How could humans or computers relate different descriptions to one another?
Structured and Unstructured Representations

A description in tables of a relational database:

<table>
<thead>
<tr>
<th>Entity</th>
<th>Shape</th>
<th>Color</th>
<th>Supporter</th>
<th>Supportee</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>pyramid</td>
<td>red</td>
<td>A</td>
<td>D</td>
</tr>
<tr>
<td>B</td>
<td>pyramid</td>
<td>green</td>
<td>B</td>
<td>D</td>
</tr>
<tr>
<td>C</td>
<td>pyramid</td>
<td>yellow</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>D</td>
<td>block</td>
<td>blue</td>
<td>D</td>
<td>E</td>
</tr>
<tr>
<td>E</td>
<td>pyramid</td>
<td>orange</td>
<td>D</td>
<td>E</td>
</tr>
<tr>
<td>F</td>
<td>block</td>
<td>blue</td>
<td>F</td>
<td>G</td>
</tr>
<tr>
<td>G</td>
<td>block</td>
<td>orange</td>
<td>F</td>
<td>G</td>
</tr>
<tr>
<td>H</td>
<td>block</td>
<td>blue</td>
<td>H</td>
<td>G</td>
</tr>
</tbody>
</table>

A description in English:

“A red pyramid A, a green pyramid B, and a yellow pyramid C support a blue block D, which supports an orange pyramid E.”

The database is called structured, and English is called unstructured. Yet English has even more structure, but of a very different kind.
“A red pyramid A, a green pyramid B, and a yellow pyramid C support a blue block D, which supports an orange pyramid E.”

The concepts (blue) are derived from English words, and the conceptual relations (yellow) from the case relations or thematic roles of linguistics.
Mapping Database Relations to Conceptual Relations

Each row of each table maps to one conceptual relation, which is linked to as many concepts as there are columns in the table.
Join concept nodes that refer to the same entities.

Closely related entities are described by connected graphs.
Mapping the Two Graphs to One Another

Very different ontologies: 12 concept nodes vs. 15 concept nodes, 11 relation nodes vs. 9 relation nodes, no similarity in type labels.

The only commonality is in the five names: A, B, C, D, E.

People can recognize the underlying similarities.

How is it possible for a computer to discover them?
Aligning Ontologies by Mapping Graphs

Repeated application of these two transformations completely map all nodes and arcs of each graph to the other.

This mapping, done by hand, is from an example by Sowa (2000), Ch 7. The VivoMind Analogy Engine (VAE) found the mapping automatically.
Observing, Learning, Reasoning, Acting

The reasoning methods identified by C. S. Peirce. Similar cycles occur in science and everyday life.
Boyd’s OODA Loop

John Boyd drew a four-step diagram for training fighter pilots to observe and respond rapidly.

The first two steps – Observe and Orient – involve the occipital, parietal, and temporal lobes.

The next two steps – Decide and Act – involve the frontal lobes for reasoning and motor control.

The four steps and the associated brain areas:

1. Observe: Visual input goes to the primary visual cortex (occipital lobes), but object recognition and naming involve the temporal lobes.
2. Orient: Parietal lobes relate vision, touch, and sound in “cognitive maps.”
3. Decide: Reasoning is under the control of the frontal lobes, but other areas store the “knowledge soup” and the “mental models.”
4. Act: “Action schemata” are patterns in the premotor cortex of the frontal lobes. Signals from the motor cortex go to the muscles.

In emergencies, each step must be traversed in milliseconds.

The fastest responses are controlled by the VR in the cerebellum.
Cycles are self-correcting: Any error in one cycle can be detected and corrected in later cycles.
Over the years, Boyd added more detail to the OODA Loop and applied it to decision-making processes of any kind. Both versions are compatible with Peirce’s cycles.

Diagram adapted from http://en.wikipedia.org/wiki/OODA_loop
The Hierarchical Cognition Affect architecture by Aaron Sloman includes a cycle similar to Peirce’s or Boyd’s. *

Albus Cognitive Architecture

A cycle that resembles those by Peirce, Boyd, and Sloman.
Implementing the Cycles

An open-ended variety of methods for learning and reasoning.
Context in Language

Hi & Lois

If you can tell me what kind of cookie I have in this box, I'll give it to you!

Chocolate chip!

Right! Here's the box!

Mom!

Four kinds of context: The text or discourse; the situation; common background knowledge; and the intentions of the participants.

Linguistics: Parse the sentences, resolve the referents of noun phrases, and determine the literal meaning of the text.

Pragmatics: Determine the implications by relating the meaning to the situation, the background knowledge, and the intentions.
Syntax is easy: Parse the question and the answer.

Semantics is harder: Use the context to
- Recognize the situation type and the roles of the two participants,
- Relate the word 'thing' to the car that is in a garage,
- Relate the verbs 'take' and 'move' to the situation,
- Apply the laws of physics to understand the answer.

Pragmatics is the hardest: Determine the intentions of the participants and their implications for the irony and humor.

* Source of cartoon: search for 'moving' at http://www.shoecomics.com/
Actual, Modal, and Intentional Contexts

Three kinds of contexts, according to the source of knowledge:

- **Actual**: Something factual about the world.
- **Modal**: Something possible, as determined by some hypothesis.
- **Intentional**: Something an agent believes, desires, or intends.
The three situations may be described as actual, modal, or intentional.

1. Actual: *Pierre is thinking of Marie, who is thinking of him.*

2. Modal: *Pierre is thinking of Marie, who might be thinking of him.*

3. Intentional: *Pierre thinks that Marie is thinking of him.*

In #1, both clauses are true, but Pierre may not know what Marie thinks.

In #2, the first clause is true, but the second may be true or false.

In #3, Pierre assumes or wishes that his thought is true, but it may be false.
A Tarski-style model evaluates axioms of a theory in terms of a world, which may be described by a set, a network, or a database of facts. For modal logic, the model may consist of a family of possible worlds. In computer applications, possible worlds are represented by sets of propositions that are true (facts) or necessarily true (laws).
In the situation $e$, John Perry is lecturing while Jon Barwise is standing on the right.

A language expression $\phi$ is a relation between a discourse situation $d$, a speaker connection function $c$, and a described situation $e$: $d, c \parallel \phi \parallel e$.

If $\phi$ is the expression "the number of sleeping students", its value is 3 at 3:01 pm, 5 at 3:15, 9 at 3:30, and 19 at 3:45.
Example of a Situation

This is a test picture used to diagnose patients with aphasia. A patient’s description of the situation can show the effects of lesions caused by wound, stroke, tumor, or infection.

The “cookie theft” picture was adapted from Goodglass & Kaplan (1972).
Situation: A woman, a girl, and a boy are in a kitchen of a house. The woman wipes a plate with a cloth. Water spills on the floor of the kitchen. The girl reaches for a cookie. The boy holds a cookie in his left hand. The boy grasps a cookie with his right hand. The boy stands on a stool. The stool tips over. The boy falls down.