Integrating Semantic Systems

Expressing, sharing, and using knowledge

John F. Sowa

10 July 2010

Outline of this Tutorial

1. What problems are semantic systems designed to solve?

2. What is a semantic system?

3. The Common Logic family.

4. Controlled natural languages.

5. Sharing and integrating ontologies.

6. Supporting new tools for the future.

This tutorial surveys technology that can be used to develop better tools and methodologies for the future.

1. What are the Problems?

Semantics should facilitate the interoperability and integration of all systems – legacy, current, and future.

But there is no such thing as a one-size-fits-all set of languages and tools for processing semantics.

People with different jobs and skills require different tools:

- **Most semantic tools and languages are too difficult for the average programmer, website developer, or subject matter expert (SME).**
- **They are also too limited for people who are capable of designing better tools for programmers, website developers, and SMEs.**
- **Different kinds of applications require different amounts of detail: lightweight, middleweight, or heavyweight semantics.**
- **Current technology is capable of supporting a better balance.**

Size of the Problems

Some estimates:

- **World-wide digital data in 2009: 800 million terabytes.**
- **Estimated digital data in 2010: 1.2 billion terabytes.**
- **Legacy software: Half a trillion lines of code.**
- **Percentage that is tagged with semantics: Slightly over 0%**

Some questions:

- **Can semantics help?**
- **How is semantics represented?**
- **What kind of training do the developers need?**
- **What can they do with currently available tools?**
- **What could be done with better tools?**
- **How and when can we get better tools?**

Two Views of Legacy Systems

By the philosopher Alfred North Whitehead:

> **"Systems, scientific and philosophic, come and go. Each method of limited understanding is at length exhausted. In its prime each system is a triumphant success: in its decay it is an obstructive nuisance."**

By the computer scientist Harlan Mills:

> **"OS/360 is like a cow." It's not the most beautiful or efficient, and many people think they can design a better one. But if you put hay and water in one end, you get fertilizer from the other end and milk from the middle. You can use it effectively if you recognize its limitations and remember which end is which.**

Both views are true:

- **Every system eventually becomes a legacy.**
- **But legacy systems can remain useful for a long time.**
- **They must interoperate with semantic systems.**

Cost of the Problems

A study of IT projects in 2006:

- **Of $364 billion spent on development, $160 billion was wasted.**
- **Only 35% of IT projects met the requirements,**
- **19% were outright failures,**
- **46% were "challenged."**

An IT failure can cost more than just the IT expense:

- **A large corporation lost 27% of their market share when an Enterprise Resource Planning (ERP) project failed.**

Better tools can help:

- **IT projects are late 72% of the time.**
- **That's a major improvement over the past.**
- **Can semantics enable us to do better?**

Source of statistics: William M. Ulrich & Philip Newcomb, *Information Systems Transformation: Architecture-Driven Modernization Case Studies*, MK/OMG Press, 2010.

How Could Semantics Help?

Typical programmer productivity with current tools:

- **10 to 15 lines of fully debugged code per person per day.**
- **Cost per line of code: $18 to $45.**
- **Most of the time is spent on analysis, design, and testing.**
- **Specification errors are the most costly and time consuming and the most likely to benefit from clearly defined semantics.**

Why semantic technology isn't used more widely:

- **Too much time, effort, and training to specify semantics.**
- **Delayed implementation without any obvious benefit.**
- **Difficulty of sharing semantics among different tools, especially tools designed for different methodologies.**

We need better tools and better integration among tools.

2. Semantic Systems

Computer systems that recognize, represent, and respond to the meaning of the data and the goals of the users.

Examples of semantic systems:

- Artificial intelligence, expert systems, knowledge-based systems,
- Deductive database systems,
- Natural language query and analysis systems,
- The Semantic Web and its applications,
- HAL 9000, but more cooperative.

Ultimate goal of semantic systems:

- Help people interact with computers in a more human way.
- Integrate all software around the semantics of the data.
- Enable legacy systems to participate in the integration.

What is Semantics?

In a broad sense, semantics is the study of meaning.

In a narrow sense, semantics is the second of three major subdivisions of the study of meaning:

1. **Syntax: The grammar of any language, linear or graphic.**
2. **Semantics: The content expressed by a language and its relationship to the subject matter.**
3. **Pragmatics: The goal or purpose for using the content.**

Since any practical application must have a purpose, pragmatics is essential to meaning and use.

But the same content can be used for multiple purposes.

What is Content?

The semantic content has three parts:

1. **Logic: The Boolean operators (and, or, not, if-then), the quantifiers (some, every, exactly one), and combinations.**
2. **Domain-independent ontology: The very general types, functions, and relations used by many different applications.**
3. **Domain-dependent ontology: The types, functions, and relations used in specific domains or applications.**

The domain-independent ontology includes numbers, sets, sequences, space, time, and other widely used types.

Many specification languages and modeling languages include a large amount of domain-independent ontology.

Various tools and methodologies use those languages to define the domain-dependent ontology.

Classical Artificial Intelligence

An approach to problem solving and question answering:

- **Some version of logic for knowledge representation,**
- **An inference engine for drawing conclusions,**
- **A database system for storing facts.**

This approach has been one of the mainstream paradigms of AI since the 1960s.

The largest and most sophisticated single system is Cyc, which has been in continuous development since 1984.

The Semantic Web is even larger, but it is a looser coalition of independently developed sites, and its knowledge representation is not as expressive as the CycL logic.

Rule-Based Expert Systems

A special case of classical AI that was popular in the 1980s and is still important for many kinds of applications.

Unfortunately, expert systems acquired a bad reputation:

- **They were overhyped as a panacea for almost everything.**
- **The tools and methodologies available in those days required complex skills.**
- **They were not well integrated with the mainstream technologies for software design, development, and deployment.**

Some of the companies started in the 1980s still provide rule-based systems with good tools and methodologies.

But for commercial applications, the term 'business rule' has become more acceptable than 'expert system'.

Cyc Project

Started in 1984 by Doug Lenat.

Name comes from the stressed syllable of 'encyclopedia'.

Goal: Implement a computable version of the background knowledge shared by most high-school graduates.

After the first 25 years:

- **100 million dollars and 1000 person-years of work,**
- **600,000 concepts,**
- **Defined by 5,000,000 axioms,**
- **Organized in 6,000 microtheories.**

The OpenCyc Foundation made the Cyc ontology and some applications available in open source.

To browse the ontology or to download OpenCyc, see http://opencyc.org/

Focus on Applications

The Cyc ontology is the world's largest body of knowledge represented in logic and suitable for detailed deduction.

The CycL language is a superset of many different versions of logic, including RDF, OWL, and rule-based systems.

Starting in 2004, Cycorp has put more emphasis on applications.

Cycorp earned more money from applications in the years 2008 to 2010 than in the previous 24 years.

Some of the fastest growing applications are to medical informatics.

At the Cleveland Clinic, about 1700 axioms from the general Cyc ontology are used to understand and respond to a typical query.

For white papers and research publications about Cyc, see http://cyc.com

Using Cyc as a Development Environment

Cyc is a good platform for defining semantics, and many applications require varying numbers of axioms.

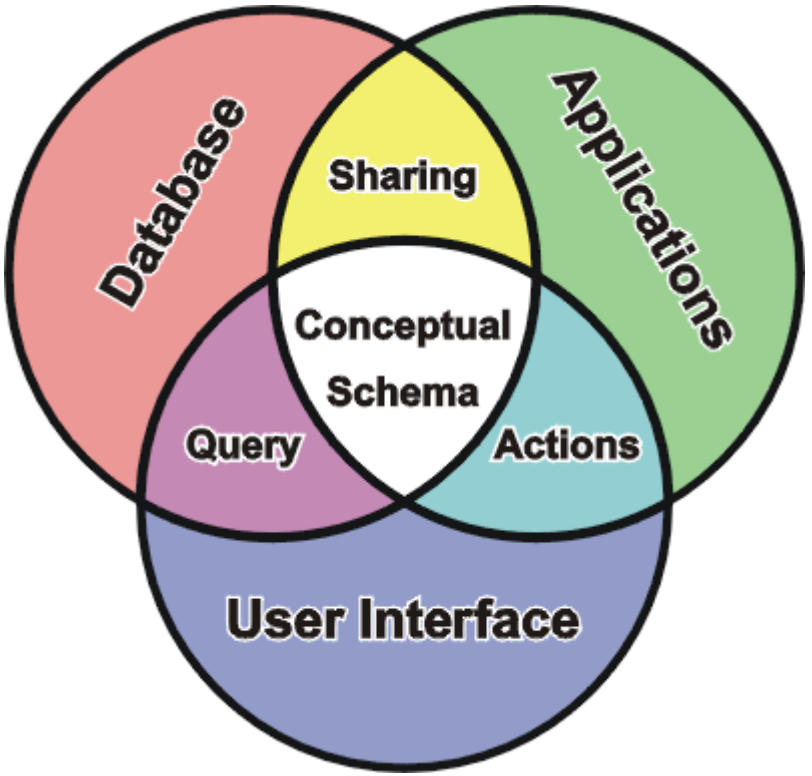
Suitable tools can extract axioms from Cyc, translate them to other notations, and integrate them with other software.

In fact, some Cyc users developed tools for extracting axioms and tailoring them to other platforms. *

More development work is needed to integrate Cyc with the tools and methodologies used in mainstream IT.

*** Peterson, Brian J., William A. Andersen, & Joshua Engel (1998) Knowledge bus: generating application-focused databases from large ontologies, http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-10/**

Conceptual Schema



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.**

Standardizing the Conceptual Schema

Many database experts agreed:

- Logic is neutral on the issue of data storage and access.
- Issues of storage and access are not relevant to semantics.
- Therefore, logic is the only logical choice for the conceptual schema.

But other DB experts disagreed:

- They claimed that programmers should specify the data formats.
- Some claimed that logic was hard for people to understand.
- Commercial DB vendors did not want to disrupt their implementations.

Over thirty years of R & D and ISO proposals:

- Better notations for logic than the SQL where-clause.
- Methodologies, tools, and technical reports, but no standards.

See The Orange Report ISO TR9007 (1982 – 1987): Grandparent of the Business Rules Approach and SBVR, History of the ISO TC97/SC5/WG3 Working Group, *Business Rules Journal,* by J. J. van Griethuysen, Vol. 10, No. 4 (April 2009), http://www.BRCommunity.com/a2009/b474.html

Database and Knowledge Base Design

Good methodologies are essential:

- Based on natural language and logic, not programming details.
- Respect for legacy systems, not a total rejection of operational systems.
- Supported by tools that subject-matter experts can use and understand.

Some tools have been designed, implemented, and used:

- Natural-language Information Analysis Methodology (NIAM).
- Object-Role Modeling (ORM).
- DOGMA workbench for modeling meaning in context.

Generic ontologies must be specialized for each context:

- Different applications may use shared data in very different ways.
- Enormous amounts of context-dependent details for each application.
- Tools and methodologies must support customization and negotiation.

See Why a data model does not an ontology make, by Robert Meersman,
http://www.starlab.vub.ac.be/website/files/MeersmanBuffaloAug2007.pdf

Semantics of Business Vocabulary and Business Rules

SBVR is a system for representing business rules in logic:

- **Ontology for business vocabulary.**
- **Mapping to and from SBVR Controlled English.**
- **Foundation based on Object-Role Modeling (ORM).**
- **Evolved from NIAM (Natural language Information Analysis Method).**
- **Graphic tools derived from NIAM and compatible with UML.**

SBVR Controlled English about the Nobel Prize:

- **Fact types are patterns that relate entity types and *roles*:**

  Work *was awarded in* Category *in* Year

  Laureate *was awarded in* Category *in* Year

- **Fact types combined with quantifiers are used to state constraints:**

  ***In* each Category *in* each Year *the prize is won by* at most two Works.**

  ***In* each Category *in* each Year *the prize is won by* at most three Laureates.**

Logic Programming (LP)

Logic can be used as a programming language:

- Specify a problem by axioms in some version of logic.
- Design a system that finds one or more models for those axioms.
- The result is the solution (or solutions) to the problem.
- It can also be the answer to some question.

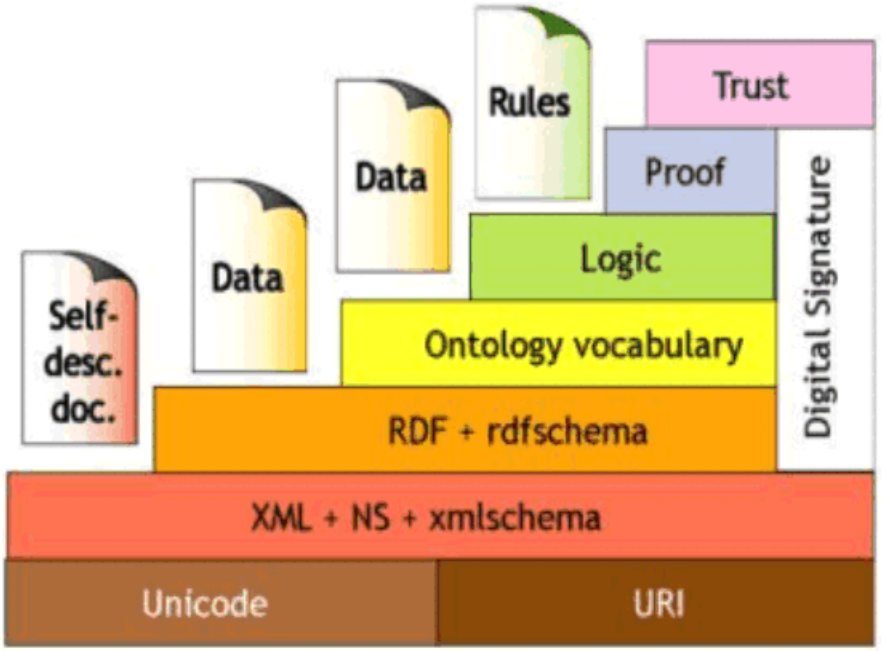
Prolog is the one of the most widely used LP systems.

- When Ted Codd saw Prolog, he said “I wish I had invented that.”

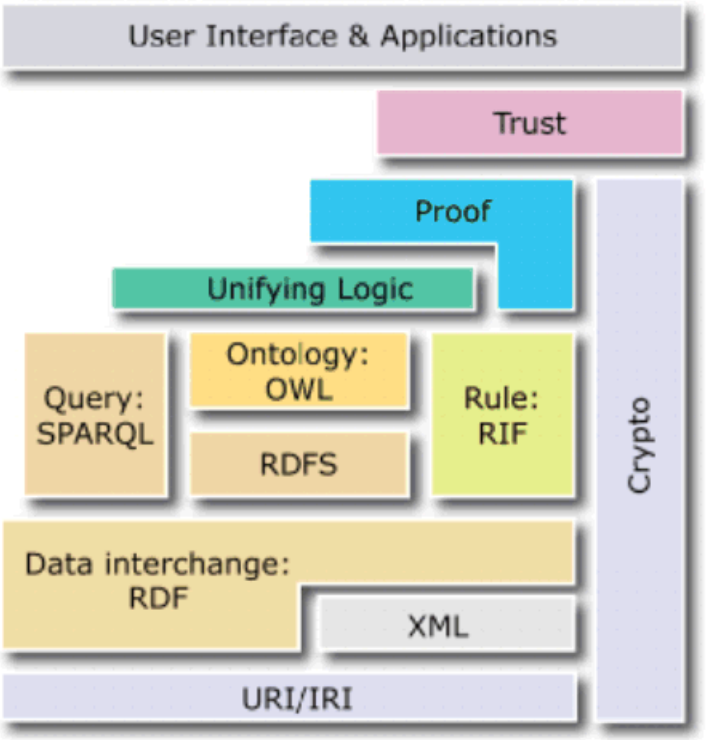
Example: The Experian credit bureau.

- They use rules written in Prolog to check everybody’s credit score.
- But their rules are proprietary and confidential.
- They are so heavily dependent on Prolog that they bought Prologia, the company founded by Alain Colmerauer, who invented Prolog.

Semantic Web



Original Layer Cake



New Layer Cake

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.

Missed Opportunity

When the Semantic Web appeared, most commercial web sites, large and small, were built around relational databases.

In fact, the acronym LAMP characterized small and medium sites: Linux, Apache, MySQL, and Perl, Python, or PHP.

The Semantic Web had a great opportunity to provide an upward compatible semantics for both relational and object-oriented DBs:

- **Type hierarchies defined by description logics.**
- **Query and constraint languages based on typed FOL.**
- **Deductive database tools based on a typed version of Datalog.**
- **Arbitrary n-tuples to support relational tables.**
- **A logic-based foundation for all semantic systems.**

Such an approach is still possible as an upward compatible extension of the current Semantic Web technology.

Integration, not Fragmentation

Systems from many different paradigms should be able to interoperate, if they focus on the meaning of the data.

Observation by an application developer in 1980:

> **"Any one of those tools, by itself, is a tremendous aid to productivity. But any two of them together will kill you." ***

Unfortunately, that complaint is just as true today.

The conceptual schema (slide 16) was a proposal to integrate all the tools with semantics at the center.

But the layer cakes of the Semantic Web shifted the focus to data structures with a unifying logic as an afterthought.

Integration requires the unifying logic at the center.

*** Terry Longstreth, at an IBM database conference in 1980.**

Knowledge Management

Karl-Erik Sveiby coined the term *Knowledge Management* for a human-oriented approach to organizational knowledge:

> **"The nurses in a Norwegian private hospital in Oslo wanted to solve a problem: how can we reduce the fear of patients going in to surgery? The idea came up: invite the old patients for coffee and cake together with the new patients and let them talk. The surgeons were against it, but the hospital decided to do a pilot test. It became a success! Both patient categories loved it, and both nurses and surgeons agreed afterwards: the patients' fear had been reduced."**

The hospital generalized that success story to a policy of promoting knowledge sharing among everybody – surgeons, nurses, patients, and support staff.

Semantic systems should focus on the human origin of the knowledge and its relationship to human needs and goals.

For the hospital example, see http://www.sveiby.com/articles/KMCaseHospital.pdf

For other articles by Sveiby, see http://www.sveiby.com/articles/

3. A Family of Logics

First-order logic is a subset or superset of most logic-based notations.

But people are constantly inventing new notations, and they don't want to abandon their favorite notation in favor anybody else's.

The ISO/IEC standard 24707 for Common Logic defines a very general semantic foundation for an open-ended family of dialects.

Three normative dialects are specified in the ISO standard:

- **CLIF — Common Logic Interchange Format**
- **CGIF — Conceptual Graph Interchange Format**
- **XCL — XML-based notation for Common Logic**

But any notation that uses the common semantics can join the family.

Syntax

The syntax is the most obvious part of any notation.

Syntax is also the easiest part to process by computer.

A good syntax is essential for both human factors and computer efficiency.

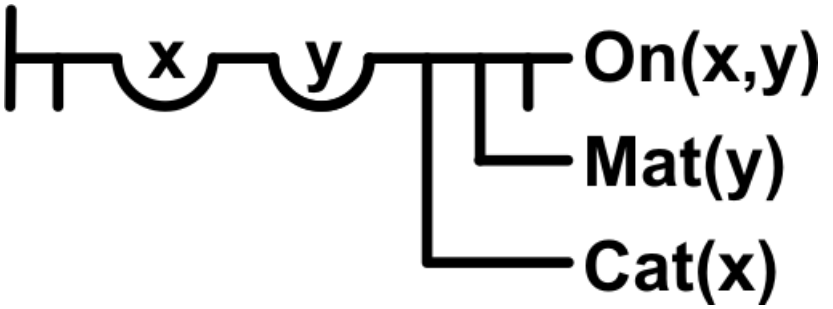
But no syntax can ever be ideal for all purposes.

Compiler technology for translating from one syntax to another became mature in the 1960s.

Therefore, the semantics can and must be specified in a way that can be translated to or from any desired syntax.

How to say “A cat is on a mat.”

Gottlob Frege (1879):



Charles Sanders Peirce (1885):

$$\Sigma_x \Sigma_y \text{Cat}_x \bullet \text{Mat}_y \bullet \text{On}_{x,y}$$

Giuseppe Peano (1895):

$$\exists x \exists y \text{Cat}(x) \wedge \text{Mat}(y) \wedge \text{On}(x,y)$$

Frege and Peirce developed their notations independently.

Peano adopted Peirce’s notation, but changed the symbols.

But all three notations have identical semantics.

Some Modern Notations

SQL query:

```
SELECT FIRST.ID, SECOND.ID
FROM   OBJECTS FIRST, OBJECTS SECOND, SUPPORTS
WHERE  FIRST.TYPE = "Cat"
AND    SECOND.TYPE = "Mat"
AND    SUPPORTS.SUPPORTER = SECOND.ID
AND    SUPPORTS.SUPPORTEE = FIRST.ID
```

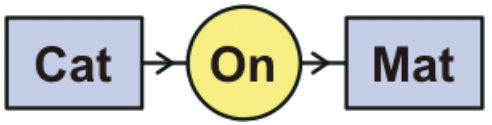
Common Logic Interchange Format (ISO 24707):

```
(exists ((x Cat) (y Mat)) (On x y))
```

Conceptual Graph Interchange Format (ISO 24707):

```
[Cat *x] [Mat *y] (On ?x ?y)
```

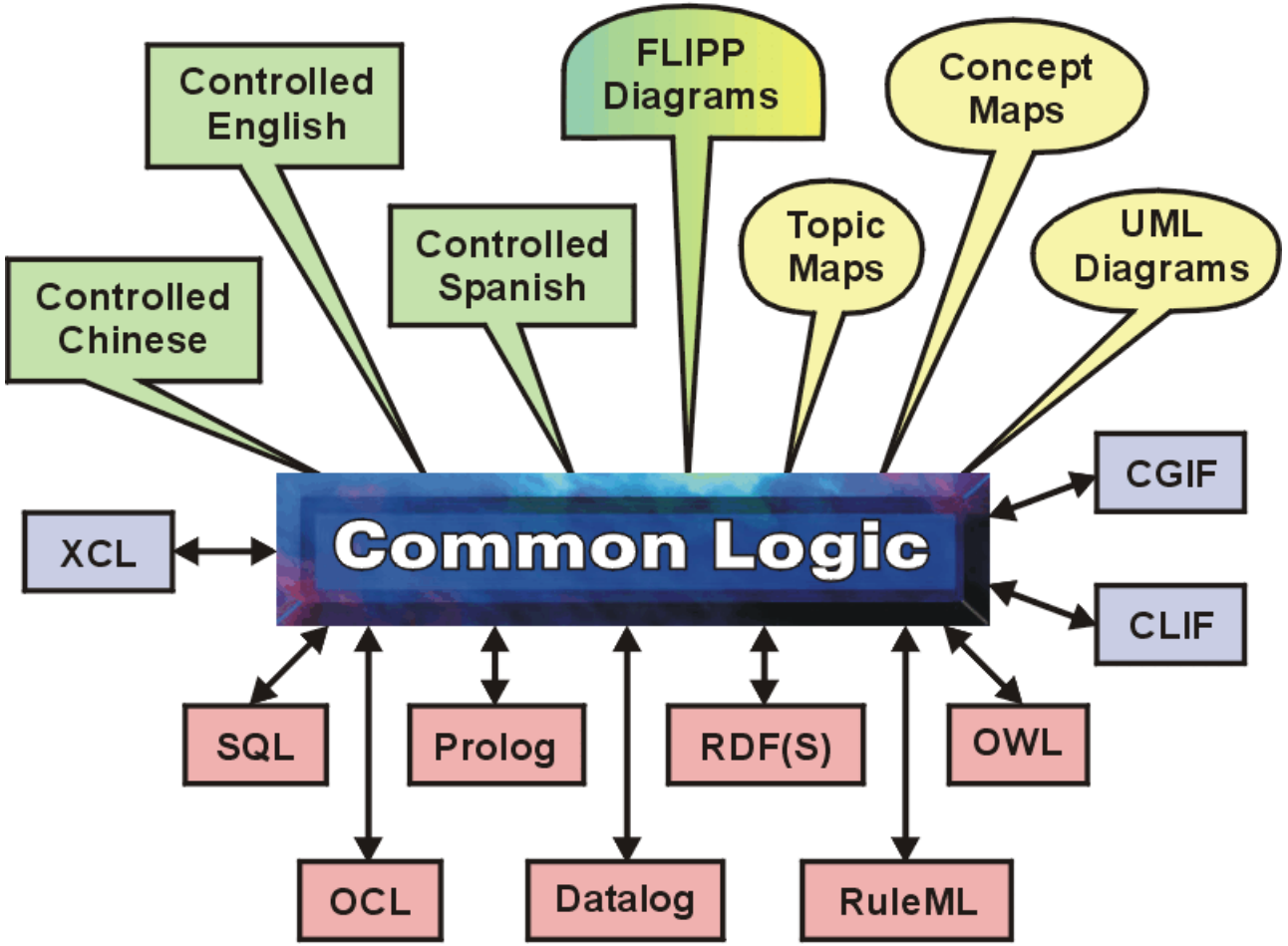
Conceptual Graph Display Form:



Controlled English:

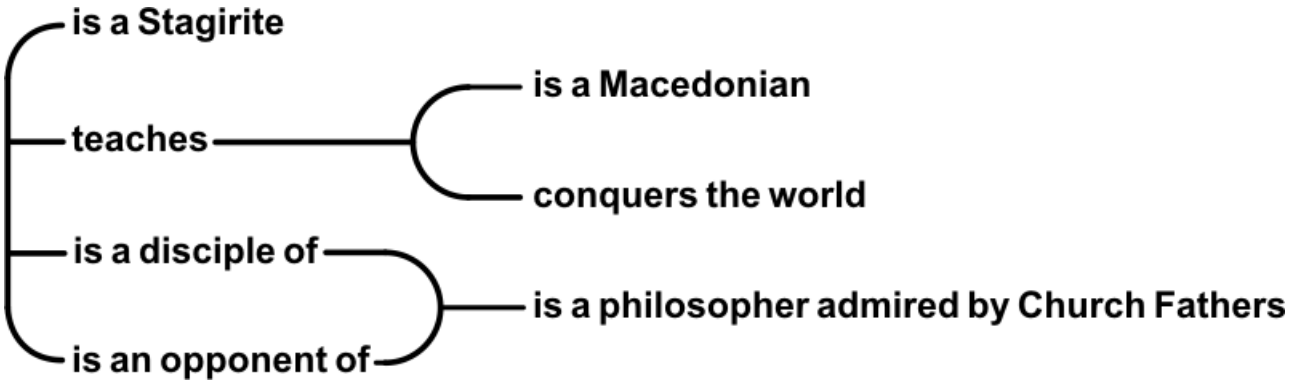
```
A cat is on a mat.
```

Human Interfaces



Machine Interfaces

Relational Graphs



A simple version of logic that can be expressed in RDF, Concept Maps, Topic Maps, and other widely used notations.

The above example by C. S. Peirce can be translated to the following formula in predicate calculus:

$$\exists x \exists y \exists z\ (\text{isaStagirite}(x) \wedge \text{teaches}(x,y) \wedge \text{isaMacedonian}(y) \wedge \text{conquersTheWorld}(y) \wedge \text{isaDiscipleOf}(x,z) \wedge \text{isanOpponentOf}(x,z) \wedge \text{isaPhilosopherAdmiredByChurchFathers}(z))$$

Note that the only logical operators expressed by relational graphs are conjunction $\wedge$ and the existential quantifier $\exists$.

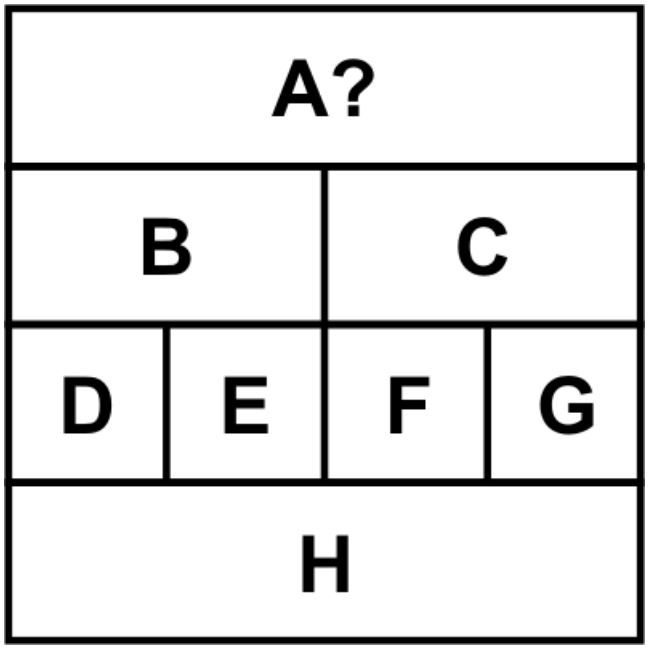
FLIPP Diagrams

Readable diagrams for expressing complex Boolean combinations.

Statements inside a box can be expressed in any notation for logic, including a controlled natural language.

A box that contains a question begins an if-then-else statement or a case statement.

A box on top of another box represents conjunction.



Two or more adjacent boxes represent disjunctions.

The diagram on the right represents the following formula:

$$(\text{if A then } (B \wedge (D \vee E)) \text{ else } (C \wedge (F \vee G))) \wedge H$$

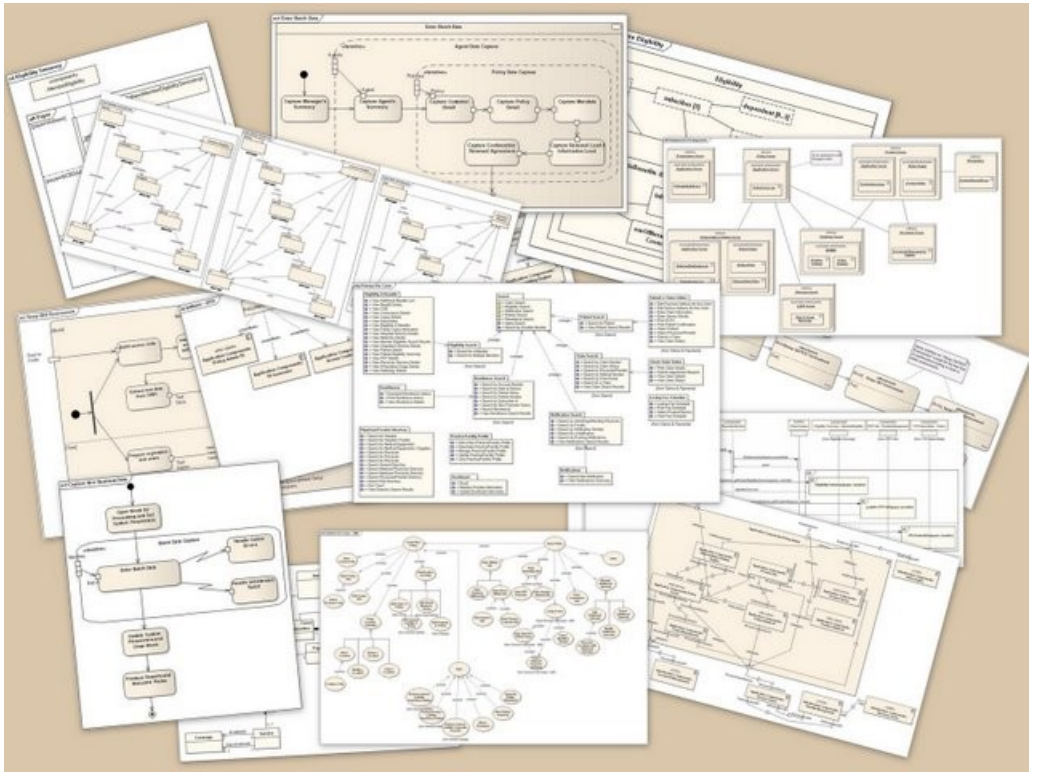
For examples, see http://www.flipp-explainers.org/casestudy1.htm

Unified Modeling Language (UML)

A family of readable diagrams that express various subsets of logic and ontology.

Adopted as a standard by the Object Management Group (OMG).

Originally specified as informal notations without a foundation in logic.



The current draft OMG standard specifies the base semantics in Common Logic.

See http://www.omg.org/spec/FUML/1.0/Beta2/PDF/

An Example of OWL

An example of OWL by Pat Hayes, slide 22, of http://is.gd/1ehQK

```
<owl:Class rdf:id="#ChildOfUSCitizenPost1955">
   <owl:intersectionOf rdf:parseType="Collection">
      <owl:Restriction>
         <owl:onProperty rdf:resource="#parentOf"/>
         <owl:allValuesFrom>
            <owl:Restriction>
               <owl:onProperty rdf:resource="#isCitizenOf"/>
               <owl:hasValue rdf:resource="#USA"/>
            </owl:Restriction>
         </owl:allValuesFrom>
      </owl:Restrrition>
      <owl:Restriction>
         <owl:onProperty rdf:resource="#dateOfBirth"/>
         <owl:allvaluesFrom rdf:resource="#YearsSince1955"/>
      </owl:Restriction>
   </owl:intersectionOf>
</owl:Class>
```

Translation to OWL-CL

A language semantically identical to OWL, but translated to Common Logic (with some supporting axioms written in CL).

Previous example translated to OWL-CL in the CLIF dialect:

```
(= ChildOfUSCitizenPost1955
    (And (AllVals parentOf (ValueIs isCitizenOf USA))
         (AllVals dateOfBirth YearsSince1955) )
```

This is a valid CLIF statement, which uses special terms 'And', 'AllVals', and 'ValueIs', which are defined by axioms in CLIF.

Any tool that generates, uses, or reasons with OWL could be adapted to generate, use, or reason with this notation.

More statements could be written in CLIF or any other dialect of CL to relate this statement to other CL statements.

Translation to Controlled English

The previous example of OWL or its translation to OWL-CL could also be written in controlled English:

Define "x is a ChildOfUSCitizenPost1955"
as "every parent of x is a citizen of USA,
and the date of birth of x is after 1955".

The noun 'ChildOfUSCitizenPost1955' is awkward, and a more natural statement would use simpler words:

If the date of birth of a person x is after 1955,
and every parent of x is a citizen of USA,
then x is a citizen of USA.

This statement can be automatically translated to many different notations for logic.

Translating OWL to ACE

ACE is a version of controlled English that can be translated to and from OWL 2.

The ACE "verbalization" of a hundred-line OWL 2 ontology:

> **Everything that is eaten by a goat is a leaf.**
> **Everything that eats nothing but leaves is a goat.**
> **Every animal is something that is a cat or that is a goat.**
> **John is a man.**
> **Everything is eaten by at most 1 thing.**
> **Everything that is eaten by something is a food that is not an automobile.**
> **Everything that is an apple or that is a leaf is a food.**
> **Every human is something that is John or that is Mary.**
> **Every man is a person.**
> **Everything eats at most 1 thing.**
> **Every human is a person that own an automobile.**
> **If X eats something that eats Y then X eats Y.**
> **Everything that eats something is an animal.**
> **If X eats Y then Y hate X.**
> **If X hate Y then Y eats X.**

See http://attempto.ifi.uzh.ch/site/docs/owl_to_ace.html

Using Common Logic as a Metalanguage

Common Logic is a superset of the logics used in many semantic systems, but some systems use even more expressive logics.

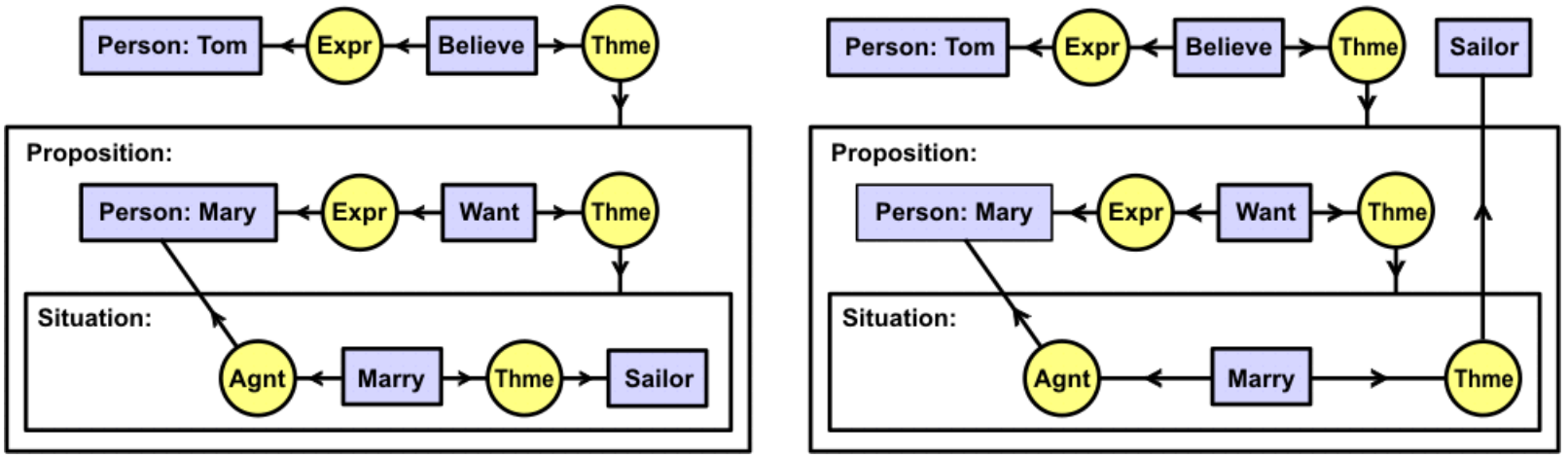
To support metalanguage, an extended version of CL called IKL marks an enclosed object-level statement with the keyword 'that'.

- **The enclosed statement denotes a proposition.**
- **The proposition may be a conjunction of many statements.**
- **It can be given a name, and other propositions can refer to it.**
- **In effect, IKL can be used as a metalanguage for talking about and relating packages of IKL statements nested to any depth.**

CL with the IKL extensions can represent a wide range of logics for modality, defaults, probability, uncertainty, and fuzziness.

For the IKL extensions, see http://www.ihmc.us/users/phayes/IKL/SPEC/SPEC.html and http://www.ihmc.us/users/phayes/ikl/guide/guide.html

CG Enclosures for Metalanguage



The two CGs above show two different interpretations of the sentence *Tom believes that Mary wants to marry a sailor:*

- **Tom believes a proposition that Mary wants a situation in which there exists a sailor whom she marries.**
- **There exists a sailor, and Tom believes a proposition that Mary wants a situation in which she marries the sailor.**

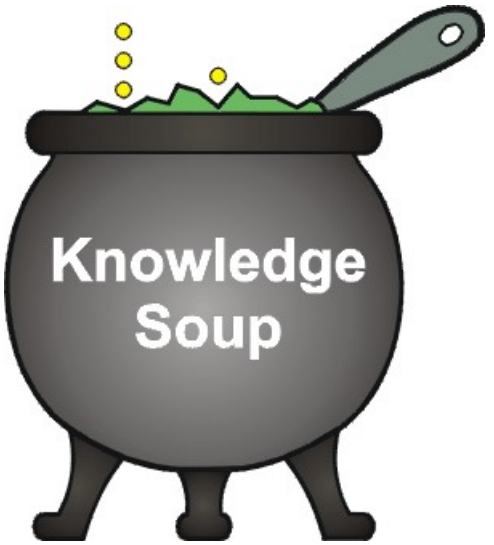
The IKL semantics permits the quantifier for "a sailor" to include the enclosed statements within its scope.

For further discussion of these issues, see http://www.jfsowa.com/pubs/cg4cs.pdf

Knowledge Soup

FOL can describe any digital computer and any program written for it.

But the world and the people in it create exceptions, uncertainty, vagueness, and fuzziness.



Many versions of logic have been invented to deal with those issues.

Cyc and other systems represent such logics by metalevel methods similar to the IKL extensions to CL.

For further discussion, see the "Challenge of Knowledge Soup,"
http://www.jfsowa.com/pubs/challenge.pdf

For the use of metalanguage to represent various logics,
http://www.jfsowa.com/pubs/laws.htm

Controlled Natural Languages

Natural languages evolved to express and support human ways of thinking, acting, and living.

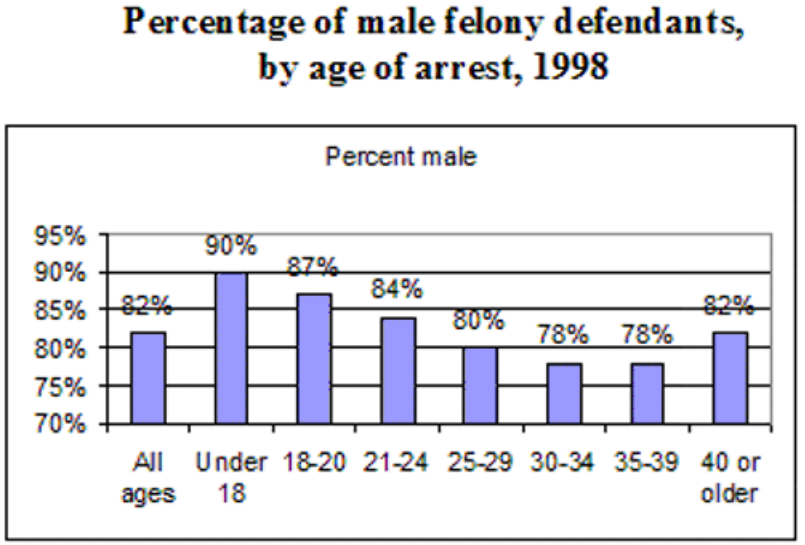
Computer languages enable IT professionals to think about the data and operations inside the computer system.

Forcing subject matter experts (SMEs) to think about their own subject in computer terms is often counterproductive:

- **Some of them become bad IT professionals.**
- **Some become good IT professionals, but compromise and distort their intuitions about their own subject.**
- **Very few become equally good at both.**

CNLs can form a bridge between the NL of the subject and the computational requirements for precision.

Same Semantics, Different Presentations



For the 75 largest counties, the highest percentage of male defendants is 90% for age Under 18, followed by 87% for age 18-20. The lowest percentage is 78% for age 30-34, with the remaining four age groups at or below the value of 84% for age 21-24. Averaging across all age groups, the percentage is 82%.

The chart and the controlled English text were generated from the same data.

A common semantic representation could produce either or both.

Source: http://www.cogentex.com/products/chartex/faq

Solving Problems Stated in English

Part of Project Halo, whose goal is to build a Digital Aristotle.

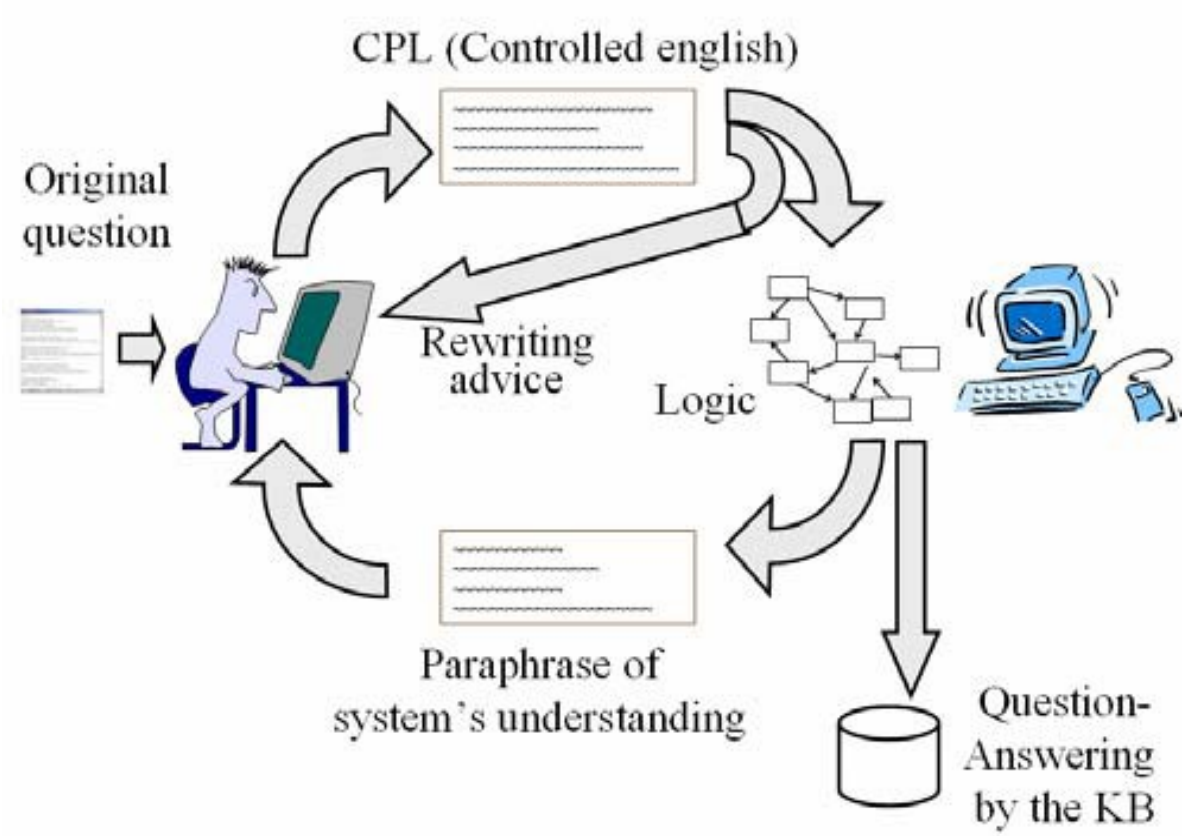
A question from the Advanced Placement Exam in physics:

> **A cyclist must stop her bike in 10 m. She is traveling at a velocity of 17 m/s. The combined mass of the cyclist and bicycle is 80 kg. What is the force required to stop the bike in this distance?**

Restated in a version of controlled English (CPL):

> **An object moves.**
> **The mass of the object is 80 kg.**
> **The initial velocity of the object is 17 m/s.**
> **The final velocity of the object is 0 m/s.**
> **The distance of the move is 10 m.**
> **What is the force on the object?**

Translating English to Controlled English



Source: P. Clark, S-Y Chaw, K. Barker, V. Chaudhri, P. Harrison, J. Fan, B. John, B. Porter, A. Spaulding, J. Thompson, P. Z. Yeh, Capturing and Answering Questions Posed to a Knowledge-Based System, http://www.ai.sri.com/pubs/files/1547.pdf

Finding the Right Combination

Controlled English is not magic.

It can't change the semantics of a user-hostile system.

A successful system requires a proper balance:

1. **Understanding the users' problems and working environment.**
2. **Developing a methodology that is natural for them.**
3. **Building a complete system that supports the methodology.**
4. **Designing a user interface that is a joy to work with.**

A good combination can be a spectacular success.

But it must be tailored to the users' needs and environment.

A Successful Combination

Tesco.com, a large Internet retailer, needed a flexible system that would allow employees to update business rules dynamically.

One vendor designed a system that would require Tesco employees to call an expert in RDF and OWL for every update.

Gerard Ellis, who had over ten years of R & D experience with conceptual graphs, designed and implemented a new system:

- **The internal knowledge representation was conceptual graphs.**
- **The interface for Tesco employees was controlled English.**
- **Tesco employees could extend or modify the rule base by typing the conditions and conclusions in controlled English.**
- **The system used the methodology of ripple-down rules to update the knowledge base, check for errors, and preserve consistency.**

See Qusai Sarraf and Gerard Ellis, "Business Rules in Retail: The Tesco.com Story," *Business Rules Journal*, Vol. 7, No. 6 (Jun. 2006), http://www.BRCommunity.com/a2006/n014.html

Typical Business Rules

Tesco employees typed information in controlled English, from which the system automatically generated the following rules:

- **If a television product description contains "28-inch screen", add a screen_size attribute_inches with a value of 28.**
- **a) If a recipe ingredient contains butter, suggest "Gold Butter" as an ingredient to add to the basket. b) If the customer prefers organic dairy products, suggest "Organic Butter" as an ingredient to add to the basket.**
- **If a customer buys 2 boxes of biscuits, the customer gets one free.**
- **If the basket value is over £100, delivery is free.**
- **If the customer is a family with children, suggest "Buy one family sized pizza and get one free".**

These rules were generated from a decision tree, as described on the next slide.

Ripple-Down Rules (RDR)

A methodology for subject matter experts to build and maintain a large, consistent rule base with little or no training:

- **Internally, the rules are organized as a decision tree.**
- **Each link of the tree is labeled with one condition.**
- **Each leaf (end point) is labeled with a conclusion (or a conjunction of two or more conclusions).**
- **Any update that would create an inconsistency is blocked.**
- **If the update is consistent, the tree is automatically reorganized.**
- **For maximum performance, the decision tree can be compiled to a nest of if-then-else statements in a programming language.**

For this application, the rules were represented in conceptual graphs, but they could be represented in any notation for logic.

See B. R. Gaines and P. Compton, Induction of Ripple-Down Rules Applied to Modeling Large Databases, http://pages.cpsc.ucalgary.ca/~gaines/reports/ML/JIIS95/index.html

Combination of CNL, RDR, and CGs

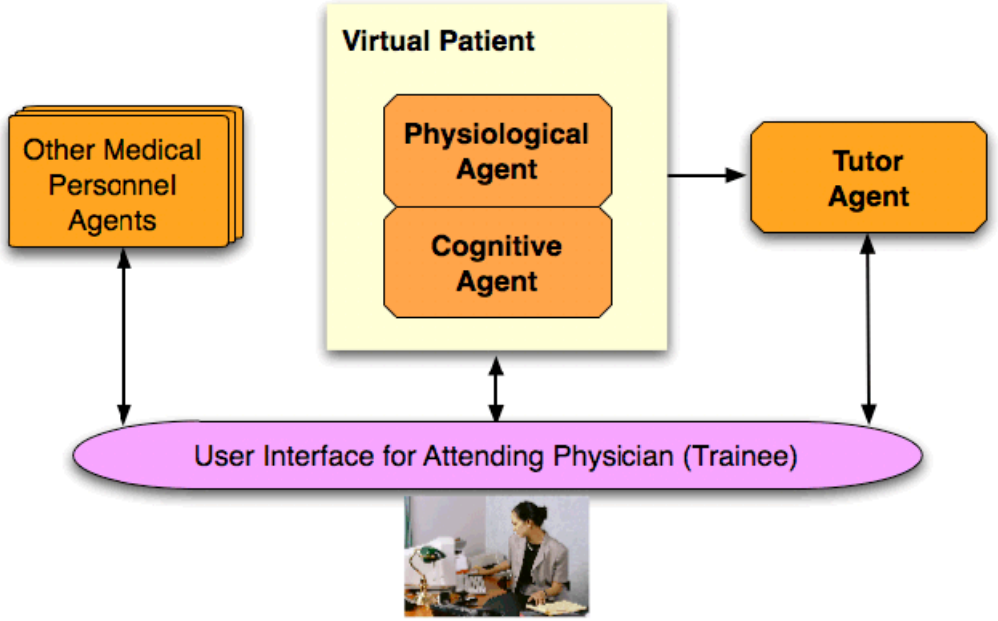
The three technologies have complementary strengths:

- Controlled English: Readable by anyone who can read English and easier to write than most computer notations.
- Ripple-down rules: Consistent knowledge bases with thousands of rules can be developed by subject matter experts with no training in programming or knowledge engineering.
- Conceptual graphs: A dialect of Common Logic, which can serve as an intermediate notation between CNLs and other formalisms.

Tesco applications that use this combination:

- Manage product information for the electrical and wine departments.
- Provide product information to business affiliates.
- Create dynamic rule-based linkages between recipes, ingredients, and products.

Maryland Virtual Patient



The MVP system simulates a patient who carries on a dialog with a medical student who tries to diagnose the patient's disease.

The student asks questions in unrestricted English, and MVP generates responses in a version of controlled English.

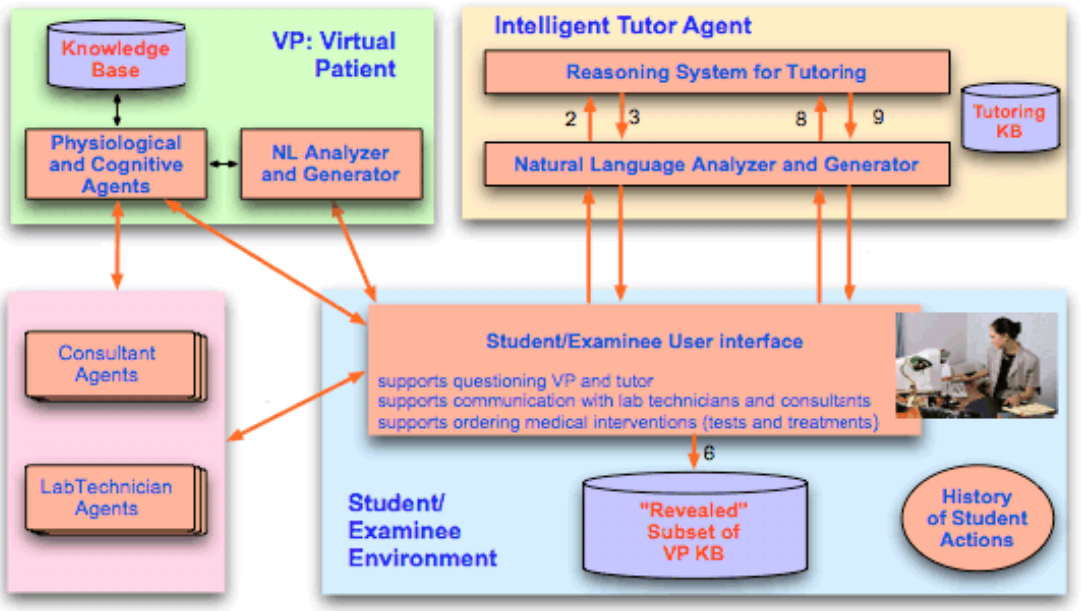
Source: Maryland virtual patient: a knowledge-based, language-enabled simulation and training system, by M. McShane, S. Nirenburg, B. Jarrell, S. Beale, & G. Fantry, http://bams.cm-uj.krakow.pl/bams3_pdf/bams%209.pdf

A Dialog with MVP

A medical student diagnoses an MVP "patient" named Mr. Wu:

Student:	**So you have difficulty swallowing?**
Mr. Wu:	**Yes.**
Student:	**Do you have difficulty swallowing solids?**
Mr. Wu:	**Yes.**
Student:	**Liquids?**
Mr. Wu:	**No.**
Student:	**Do you have chest pain?**
Mr. Wu:	**Yes, but it's mild.**
Student:	**Any heartburn?**
Mr. Wu:	**No.**
Student:	**Do you ever regurgitate your food?**
Mr. Wu:	**No.**
Student:	**How often do you have difficulty swallowing?**
Mr. Wu:	**Less than once a week.**
Student:	**It is too early to take any action. Please come back in 9 months.**
Mr. Wu:	**OK.**

Multiple Simulated Agents



Students talk with a patient, tutor, consultant, or lab technician.

All dialogs use the same ontology and knowledge base.

But each type of dialog is based on a different language game.

Source: Adaptivity in a multi-agent clinical simulation system, by S. Nirenburg, M. McShane, S. Beale, & B. Jarrell, http://www.cis.hut.fi/AKRR08/papers/nirenburg.pdf

MVP Syntax, Semantics, and Pragmatics

Two kinds of syntax:

- User inputs: A large English grammar that imposes very few restrictions on what the users can say.
- MVP responses: Controlled English, tailored to the subject matter.

Two kinds of semantics:

- Lexical semantics: Patterns of concept types and the expected relations among them. No detailed definitions or constraints.
- Subject matter: Detailed ontology, definitions, rules, constraints, and background knowledge about each disease and therapy.

Pragmatics tailored to each type of dialog:

- Different goals, speech acts, and language games.

A balanced combination of state-of-the-art technologies.

Common Logic Controlled English

A dialect of Common Logic that looks like English.

CLCE uses a subset of English syntax and vocabulary.

But the CLCE grammar avoids constructions that may cause ambiguities.

CLCE replaces pronouns with temporary names called *variables*.

Examples:

> **For every company C,**
> **exactly one manager in C is the CEO of C;**
> **every employee of C except the CEO reports to the CEO;**
> **the CEO of C does not report to any employee of C.**
>
> **If an integer N is 5, then (N^3 = 125).**

The scope of variables, such as C or N, extends to the period at the end.

Note: CLCE is not an ISO standard, but it uses the CL semantics.

CLCE Semantics

CLCE can express the full semantics of Common Logic.

A recursive definition of "reports" in terms of "directly reports":

> **Every employee who directly reports to a manager**
> **reports to that manager.**
>
> **If an employee of a company C directly reports to a manager M1 in C,**
> **and the manager M1 reports to a manager M2 in C,**
> **then the employee reports to the manager M2.**

Definitions link CLCE words and phrases to other versions of logic:

> **Define "x directly reports to y" as**
> **(DirectlyReports x y).**
>
> **Define "x directly reports to y" as**
> **SQL(select emp, mgr from employees).**

A Need for Good Tools

Anyone who can read English can read a CLCE statement.

But writing clear, precise, readable English is not easy.

And writing clear, precise, readable CLCE requires

- **The ability to write clear, precise, readable English,**
- **Some understanding of logical principles.**

Anyone who has the first requirement can learn the second.

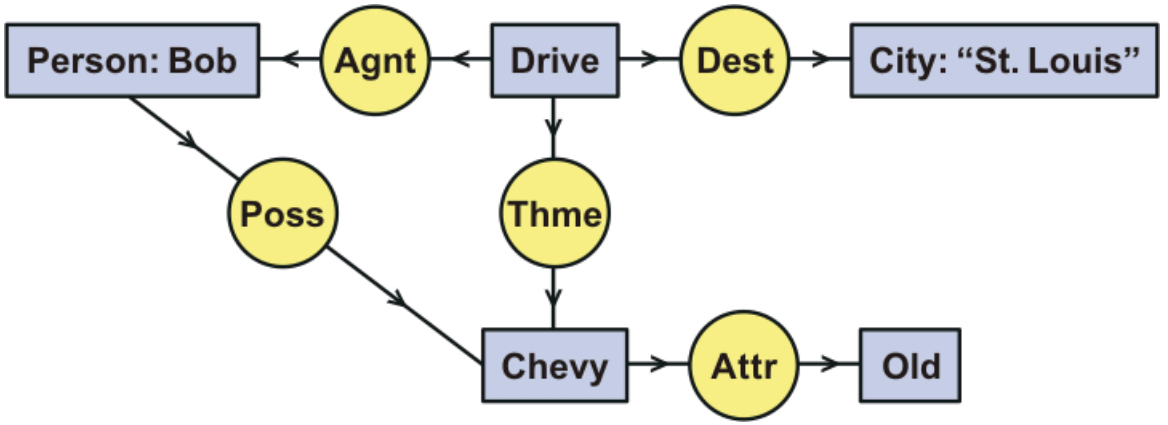
Tools for CLCE and other controlled languages have enabled subject-matter experts to update the knowledge base.

The amount of training for SMEs depends critically on the applications and the tools that support them.

- **Some tools provide extensive help and guidance.**
- **Others require a short course plus hands-on experience.**

CLCE: Bob drives his old Chevy to St. Louis.

Conceptual graph display form:



Conceptual Graph Interchange Format (CGIF):

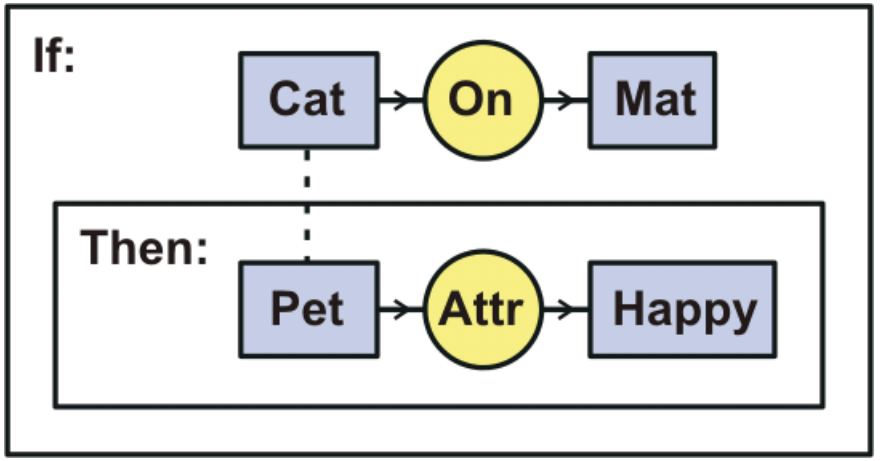
```
[Drive *x] [Person Bob] [City "St. Louis"] [Chevy *y] [Old *z]
(Agnt ?x Bob) (Dest ?x "St. Louis") (Thme ?x ?y) (Poss Bob ?y)
(Attr ?y ?z)
```

Common Logic Interchange Format (CLIF):

```
(exists ((x Drive) (y Chevy) (z Old))
  (and (Person Bob) (City "St. Louis") (Agnt x Bob)
    (Dest x "St. Louis") (Thme x y) (Poss Bob y) (Attr y z))
```

CLCE: If a cat is on a mat, then the cat is a happy pet.

Conceptual graph display form:



CGIF:

```
[If: [Cat: *x] [Mat: *y] (On ?x ?y)
      [Then: [Pet: ?x] [Happy: *z] (Attr ?x ?z) ]]
```

CLIF:

```
(not (exists (x y) (and (Cat x) (Mat y) (On x y)
     (not (exists (z) (and (Pet x) (Happy z) (Attr x z)))))))
```

A Logically Equivalent Variation

CLCE: For every cat x and every mat y,
if x is on y, then x is a happy pet.

CGIF:

```
[Cat: @every *x] [Mat: @every *y]
[If: (On ?x ?y)
   [Then: [Pet: ?x] [Happy: *z] (Attr ?x ?z) ]]
```

CLIF:

```
(forall ((x Cat) (y Mat))
   (if (On x y)
       (and (Pet x) (exists ((z Happy)) (Attr x z)))))
```

Most dialects of logic and natural languages permit many different ways of expressing semantically equivalent statements.

For common variations such as these, the proof of equivalence can be done in polynomial or even linear time.

Quantifying Over Relations

Although Common Logic has a first-order semantics, it does permit quantified variables to refer to functions and relations.

English: Bob and Sue are related.

CLCE: There is a familial relation between Bob and Sue.

CGIF:

```
[Relation: *r] (Familial ?r) (#?r Bob Sue)
```

CLIF:

```
(exists ((r Relation)) (and (Familial r) (r Bob Sue)))
```

Using CLCE to Express IKL

The operator 'that' of IKL can be used in CLCE:

Tom believes that Mary knows that (2 + 2 = 4).

And in CLIF notation:

(believes Tom (that (knows Mary (that (= (+ 2 2) 4)))))

The operator 'that' is a powerful metalevel extension.

It enables IKL to specify languages, define their semantics, and specify transformations from one language to another.

Anybody who has not spent years studying logic is unlikely to use CLCE correctly to state all the nuances.

But CLCE can express such nuances in a readable way that a wider audience, including logicians, can appreciate.

5. Sharing and Integrating Ontologies

The SIO project for sharing and integrating ontologies was started as a community project by the Ontolog Forum.

SIO is a companion to the Open Ontology Repository (OOR), which is also sponsored by the Ontolog Forum.

Various participants in the forum have discussed related theoretical issues over a period of several years.

Many of them have developed compatible methodologies and implemented open-source tools to support them.

Goal: Collaborate and coordinate the development of those tools, theories, and methodologies in support of OOR.

Supporting Interoperability

For over 50 years, computer systems have been interoperating and sharing data without using any formal semantics.

Every branch of science and engineering uses multiple, often inconsistent approximations for different kinds of problems.

There is no possibility of a single, unified, consistent, detailed ontology that can support multiple inconsistent applications.

Cyc, for example, has an underspecified upper-level ontology and 6000 detailed microtheories with complex interrelationships.

Questions:

- **How can applications with inconsistent semantics share data?**
- **What are the criteria for sharing data among inconsistent systems?**
- **Would semantic definitions with formal logic and ontology help?**
- **Or would they create more problems than they could solve?**
- **How can we detect, avoid, or work around the inconsistencies?**

Ontologies, Terminologies, and Lexical Resources

Ontology: The study of what exists and how it can be formally described and axiomatized.

Terminology: A classification of the technical terms used in any branch of science, engineering, business, and the arts.

Lexical resource: A dictionary, thesaurus, grammar, or other representation of the vocabulary, syntax, or semantics of some natural language.

These are three related, but different kinds of representations.

WordNet, for example, is a valuable lexical resource.

But aligning two ontologies to WordNet does not guarantee that corresponding categories have equivalent definitions.

Merging and relating ontologies requires logic-based tools.

Interfaces to Semantic Systems

All computer systems, including legacy systems, are becoming semantic systems that directly or indirectly access the WWW.

Different people require different interfaces:

- **Casual users – anybody who opens an unfamiliar application.**
- **Subject matter experts who develop the knowledge bases.**
- **IT professionals who develop the computer tools and systems.**

For any subject, terminology is the key to interoperability:

- **SMEs are the experts on the subject.**
- **Their terminology is the basis for all communications about the subject.**
- **Their preferred languages and diagrams must be translatable to and from computable representations.**
- **And the formal definitions of their terms must be consistent.**

The World as Interface

From a book by Otto Rössler with the above title:

Exophysics: The many, often incompatible theories about invisible entities such as atoms, molecules, fields, etc.

Endophysics: The many, often incompatible views of the world by all the people and beasts who live in it.

Interface: The phenomena that people and other animals see, feel, interpret, and manipulate.

Consistency with the phenomena at the interface is the ultimate criterion of accuracy for both exophysics and endophysics.

Ambiguities in ordinary language result from reusing the same words in multiple versions of exophysics and endophysics.

Any formal theory can only express one version at a time.

Multiple Incompatible Theories

Different versions of exophysics may use incompatible ontologies.

Example of a three-dimensional ontology:

- **A person is born at a time t1.**
- **A person dies at a time t2.**
- **For any time between t1 and t2, the person is alive at some location x and has attributes that may change from time to time.**

Example of a four-dimensional ontology:

- **A person corresponds to a 4D volume.**
- **The times t1 and t2 are the lower and upper bounds on the time coordinates of that volume.**
- **Some attributes of the person may be true or false at different points in that volume.**

Reconciling Ontologies

Any theory (exo- or endo-) that claims to be true about the world must be consistent with observations:

- **No observation can contradict a true theory.**
- **But many observations may be irrelevant to a theory that is true about some limited aspect of the world.**
- **Theories may also have internal variables that are not directly related to observations.**
- **The internal variables of one theory may be very different from or inconsistent with the internal variables of another theory.**
- **But all true theories must be consistent with observations.**

Conclusion:

- **Consistency of data shared among independently developed applications should be possible for observable phenomena.**
- **But the consistency of their internal data is unlikely, unless they are based on consistent theories.**

Methods of Reasoning

Theorem proving is only one of many ways of using logic.

The most common way of using logic in computer systems is to evaluate the truth of a statement in terms of a model:

1. **Database: The tables of a relational DB or the networks of an object-oriented DB are isomorphic to a Tarski-style model of the subject domain.**
2. **Query processing: An SQL query or a path-based query is evaluated to truth or falsity in terms of the given DB.**
3. **Constraint checking: For a DB update, each constraint must be evaluated to True before the update is permitted.**

For these purposes, first-order logic is highly efficient.

In the worst case, FOL takes polynomial time. With indexing, evaluating a statement can often be done in logarithmic time.

The world economy depends on this method of using FOL.

Consistency Test

Checking consistency in first-order logic can, in the worst case, be undecidable.

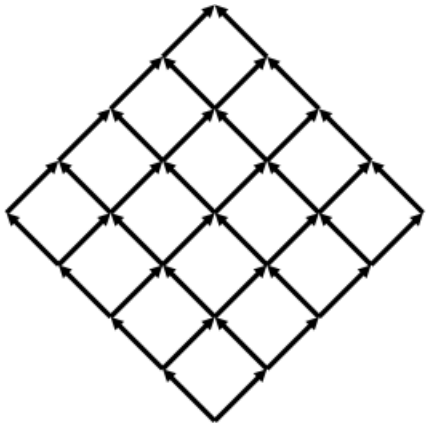
But any theory that has at least one model is guaranteed to be consistent.

Most ontology projects begin with a typical example, and the description of that example can serve as a model.

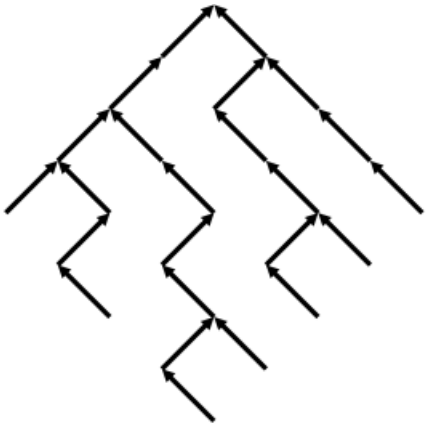
Simple and efficient consistency check:

1. **Represent the known example in a relational DB.**
2. **Translate each axiom of the theory to an SQL query.**
3. **Check whether all the queries are true of the given DB.**
4. **If so, the theory is both consistent and true of the example.**

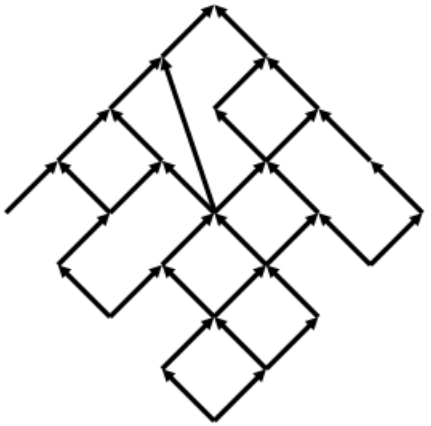
Three Kinds Acyclic Graphs



Lattice



Tree



Irregular

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*.

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.**

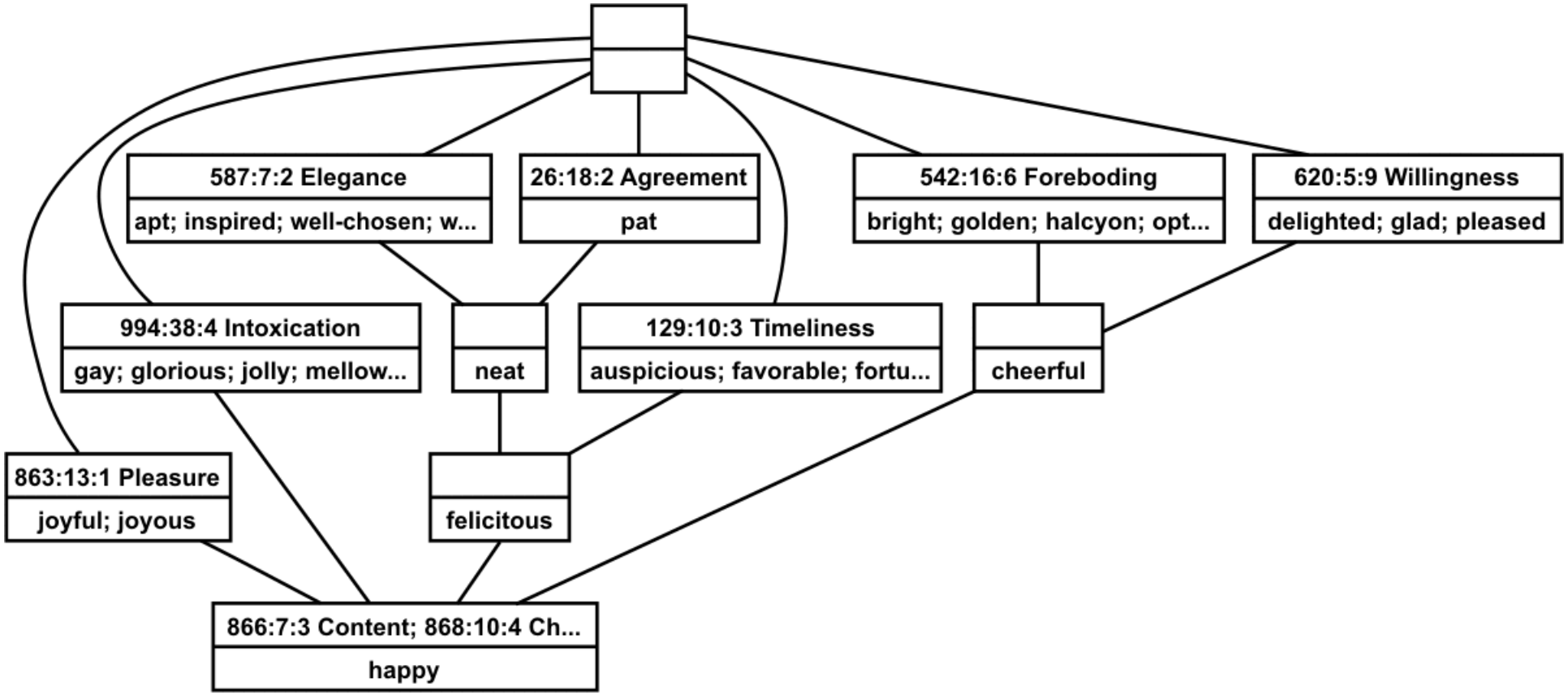
Applications:

- **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

For deriving lattices from lexical resources: 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

COLORE Project

COmmon Logic Ontology REpository (COLORE) addresses the question

> **How can the Open Ontology Repository (OOR) support the integration of upper ontologies and also support the design and reuse of ontologies for existing and emerging standards?**

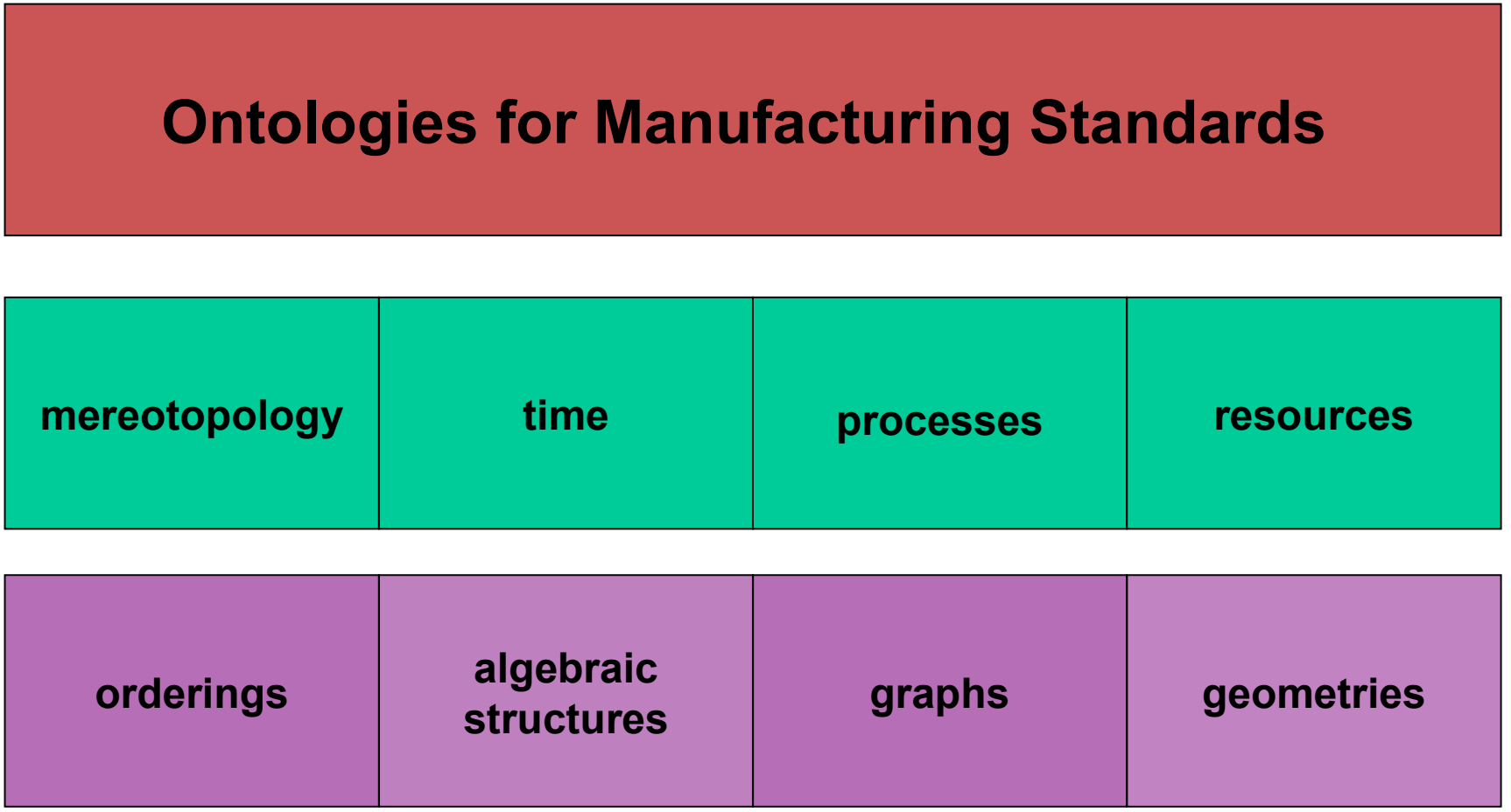
Goals:

- **Support ontologies expressed in Common Logic.**
- **Characterize logical relationships among ontologies.**
- **Develop logic-based tools to test those relationships.**
- **Use them to develop ontologies for manufacturing standards.**

Project led by Michael Grüninger, Department of Mechanical and Industrial Engineering, University of Toronto.

http://stl.mie.utoronto.ca/colore/index.html

COLORE Foundation Ontologies



Bremen Ontology Research Group

Ongoing activities:

- Collaborative Research Center on Spatial Cognition
- Within the EU FP7 Project OASIS (Open Architecture for Accessible Services Integration and Standardization)
- Developing logic-based tools and methodologies for ontology structuring, sharing, and integration.
- Plan to collaborate with OOR, SIO, and COLORE projects.

Project members:

John Bateman (project leader), Oliver Kutz,
Joana Hois, Till Mossakowski, Immanuel Normann,
Alexander Garcia Castro, Bernd Krieg-Brückner, Mehul Bhatt

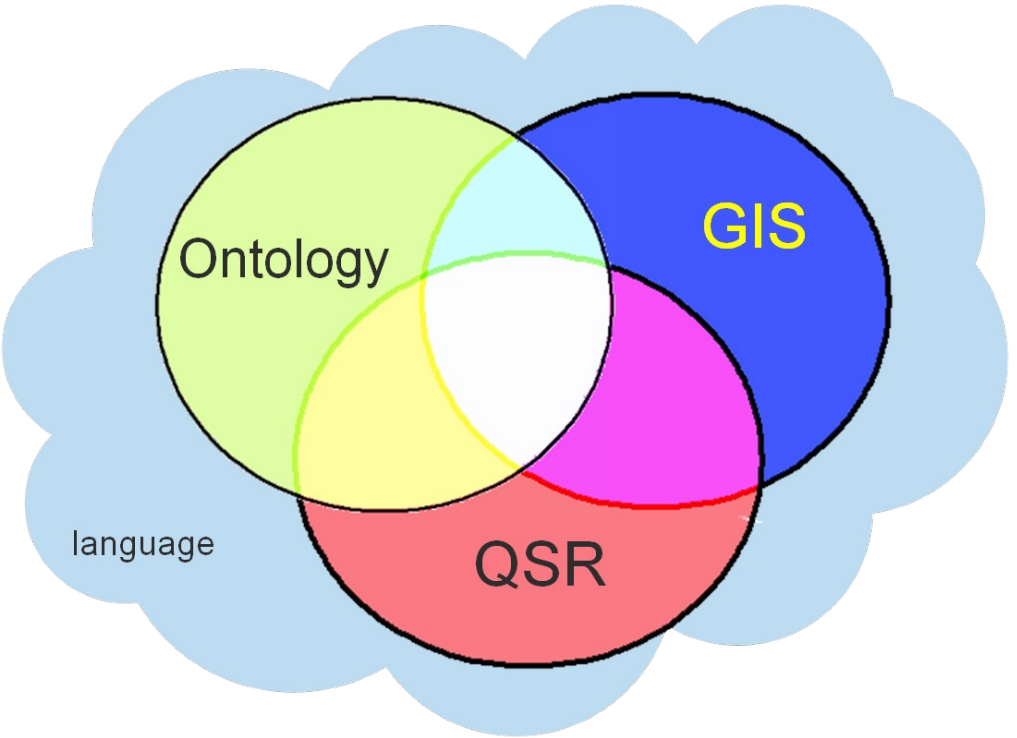
http://www.fb10.uni-bremen.de/ontology/

Spatial Representation and Reasoning

Ontology

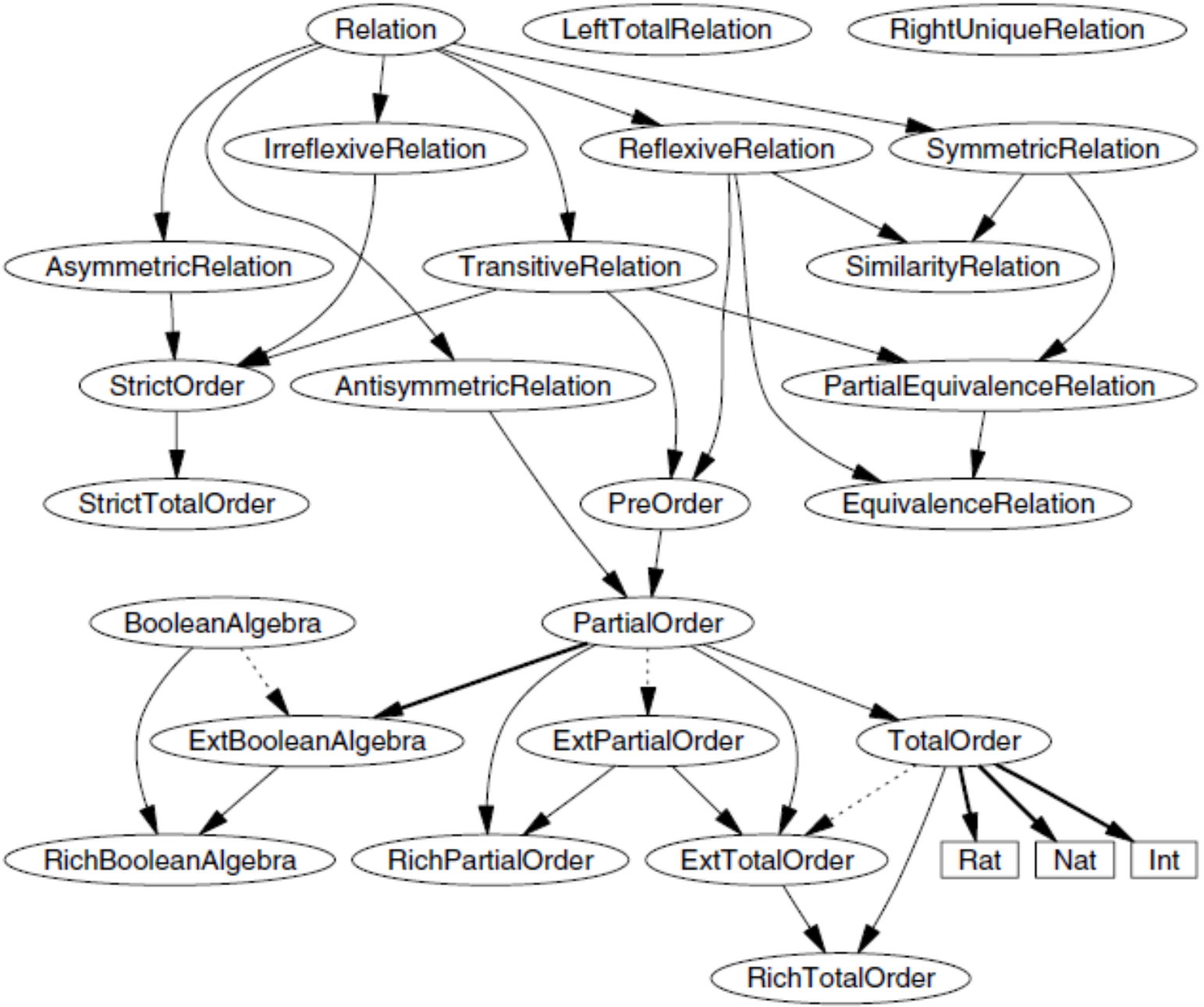
Geographic Information Science (GIS)

Qualitative Spatial Reasoning and Representation (QSR)

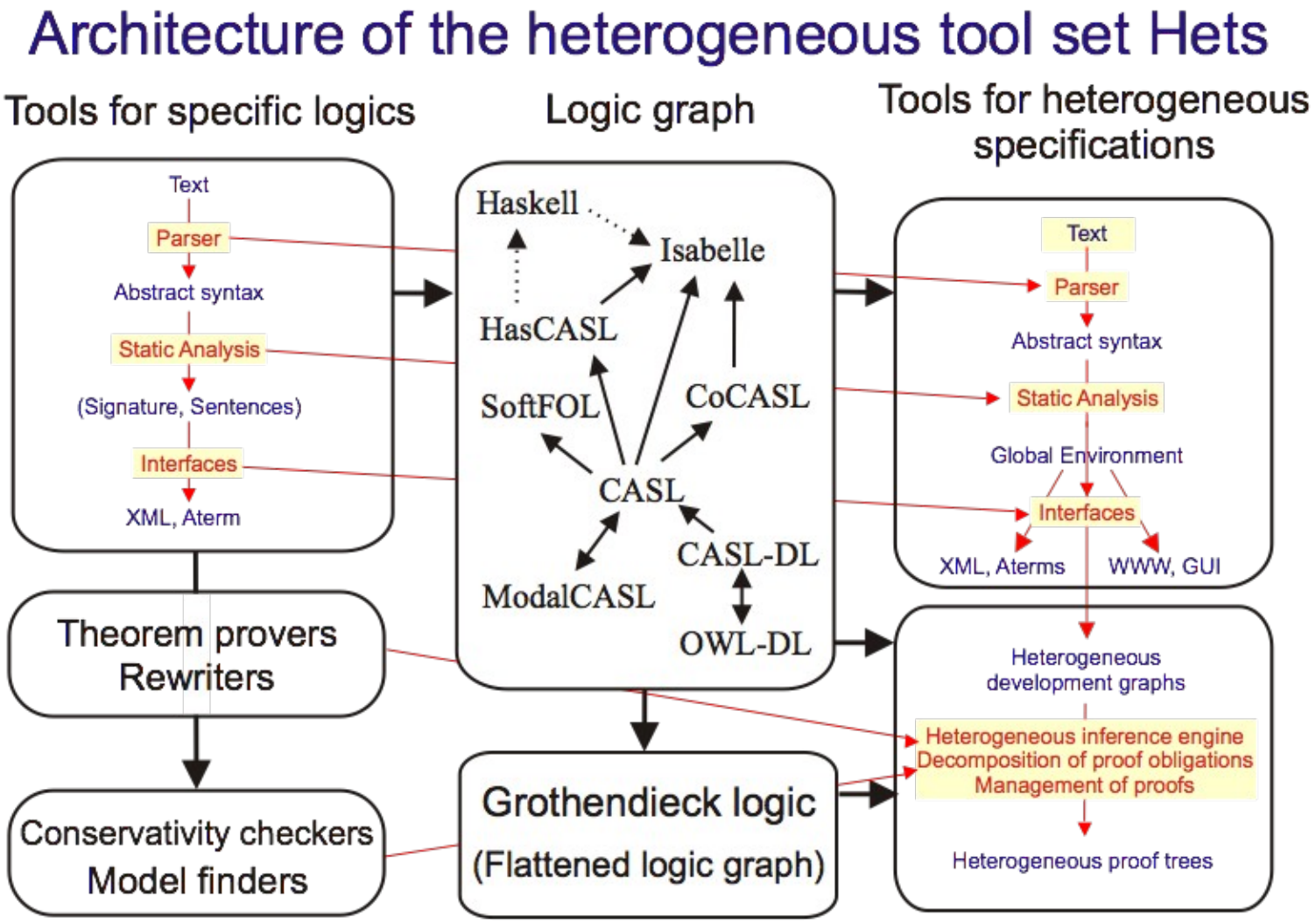


Mapping to and from Natural Languages

A Hierarchy of Mathematical Theories



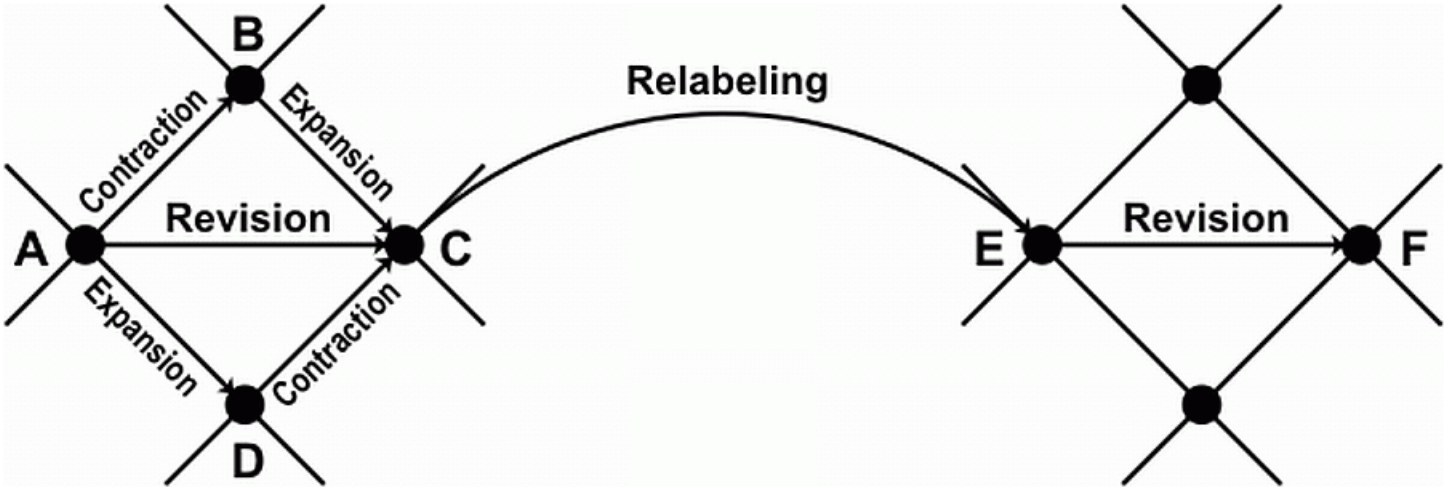
Tools for Processing Multiple Logics



http://www.informatik.uni-bremen.de/agbkb/forschung/formal_methods/CoFI/hets/index_e.htm

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.

Sharing and Integrating Ontologies

The COLORE and Bremen projects have compatible tool kits that can be used with the Open Ontology Repository.

Their tools and theoretical frameworks can support lattices.

The full lattice of theories is infinite, but only a subset, called the Hierarchy of Ontologies, will ever be stored in the OOR.

The tools can perform lattice operations on the hierarchy:

- **Combine ontologies, as illustrated in slides 74 and 77.**
- **Check for inconsistencies, as described in slide 69.**
- **Find mappings between ontologies with different terminology.**
- **Combine consistent ontologies to form a larger ontology.**
- **Find a common consistent subset of inconsistent ontologies.**
- **Save metadata about all tests and operations that are performed.**

http://ontolog.cim3.net/file/work/OOR/OOR_presentations_publications/OOR-SemTech_Jun2010.pdf

A Migration Path to the Future

Any declarative notation, graphic or linear, can be mapped to some version of logic.

Common Logic, possibly with the IKL extension, is upward compatible with all the systems discussed in these slides.

Good development methodologies supported by controlled natural languages can be used effectively by nonprogrammers.

Recommendation for a new generation of development tools:

- **Integrate all systems, including legacy systems, with logic-based methodologies.**
- **Enable subject-matter experts to review, update, and extend their knowledge bases with little or no assistance from IT specialists.**
- **Provide tools that support collaboration, review, and testing by people with different levels and kinds of expertise.**

6. Supporting New Tools for the Future

Many research prototypes have demonstrated feasibility of advanced technology and methodologies.

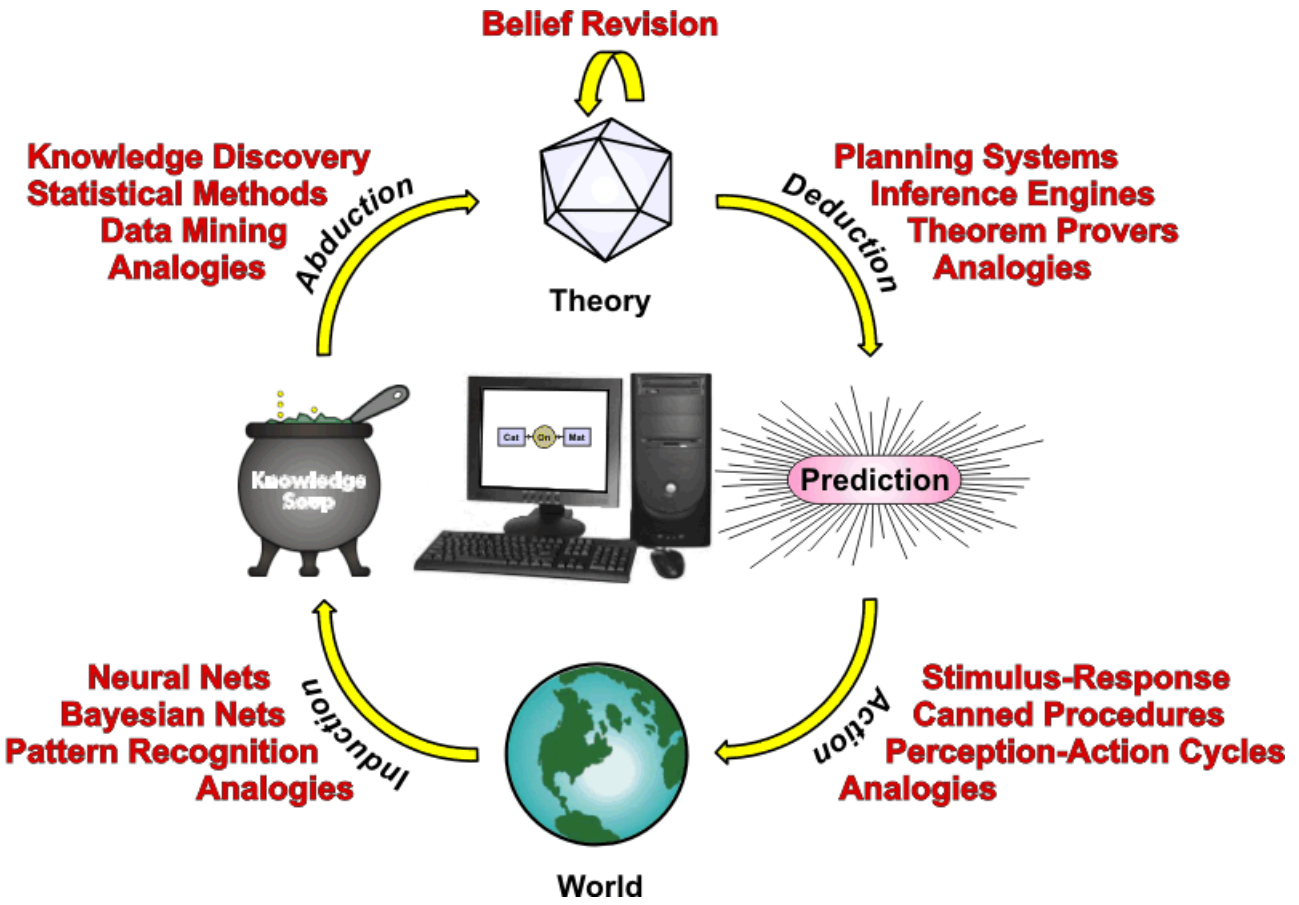
But they use tools, techniques, and languages that are very different from the mainstream of commercial IT.

Cyc, for example, has been available for a quarter of a century, but it has been hard to integrate with commercial tools.

But recent developments with semantic systems, ontologies, and logic-based tools can create new opportunities for using such systems.

Those tools can revolutionize the way applications are designed, implemented, used, and integrated with legacy systems.

Peirce's Cycle of Pragmatism



There are many different ways of using knowledge.
New kinds of useful tools are constantly being invented.
Analogy is a general and powerful reasoning method.

Describing Things in Different Ways

How can we describe what we see?

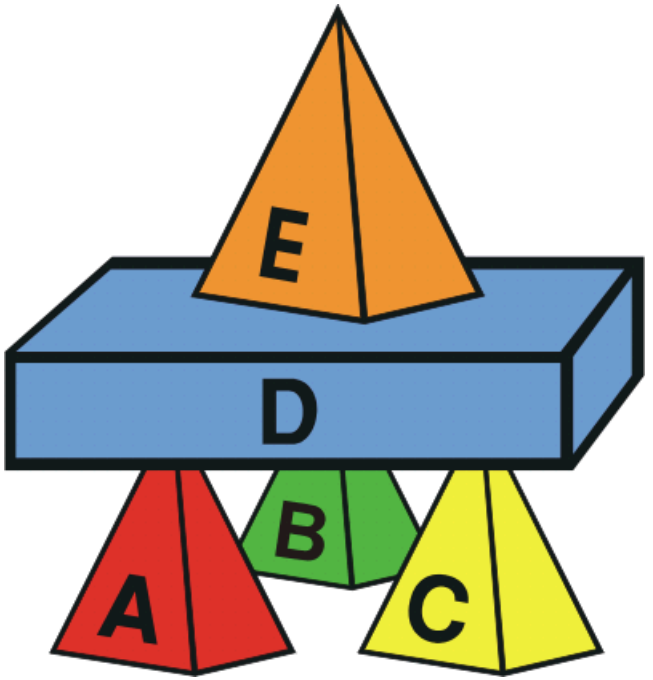
In ordinary language?

In some version of logic?

In a relational database?

In the Semantic Web?

In a programming language?

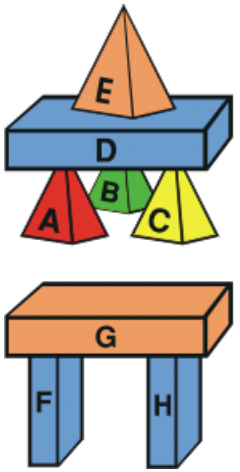


Even when people use the same language, they use different words and expressions.

How could humans or computers relate different descriptions to one another?

Structured and Unstructured Representations

A description in tables of a relational database:



Objects

Entity	Shape	Color
A	pyramid	red
B	pyramid	green
C	pyramid	yellow
D	block	blue
E	pyramid	orange
F	block	blue
G	block	orange
H	block	blue

Supports

Supporter	Supportee
A	D
B	D
C	D
D	E
F	G
H	G

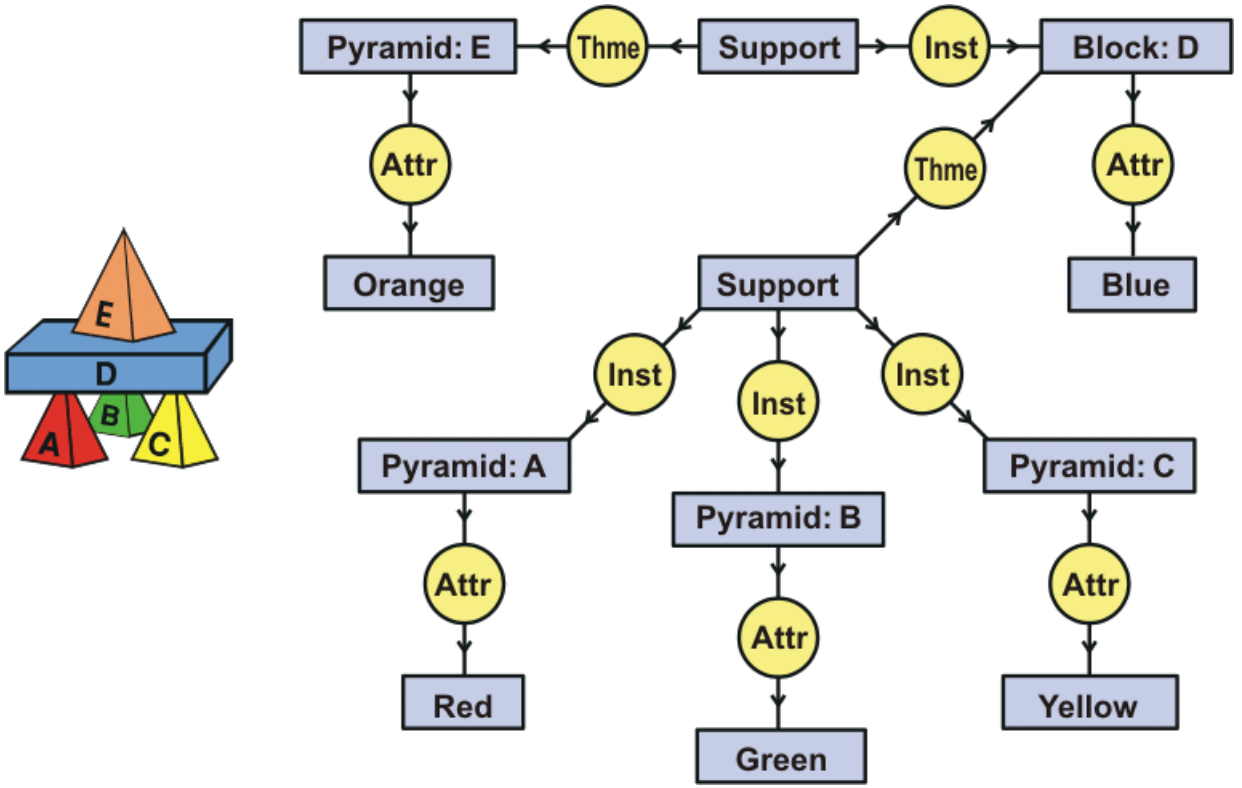
A description in English:

> **"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.

Mapping English to a Conceptual Graph



"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.

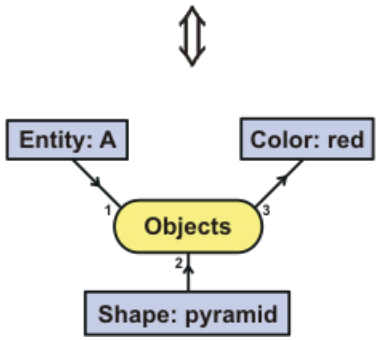
Mapping Database Relations to Conceptual Relations

Objects

Entity	Shape	Color
A	pyramid	red
B	pyramid	green
C	pyramid	yellow
D	block	blue
E	pyramid	orange
F	block	blue
G	block	orange
H	block	blue

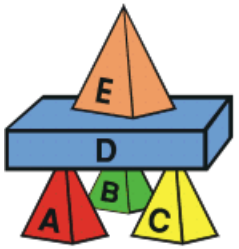
Supports

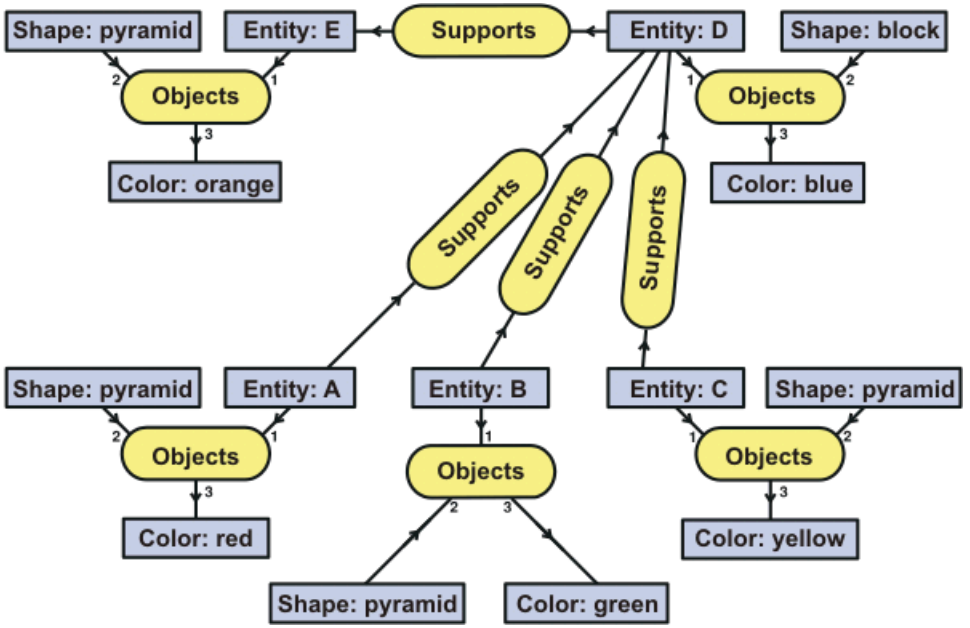
Supporter	Supportee
A	D
B	D
C	D
D	E
F	G
H	G



Each row of each table maps to one conceptual relation, which is linked to as many concepts as there are columns in the table.

Mapping an Entire Database to Conceptual Graphs

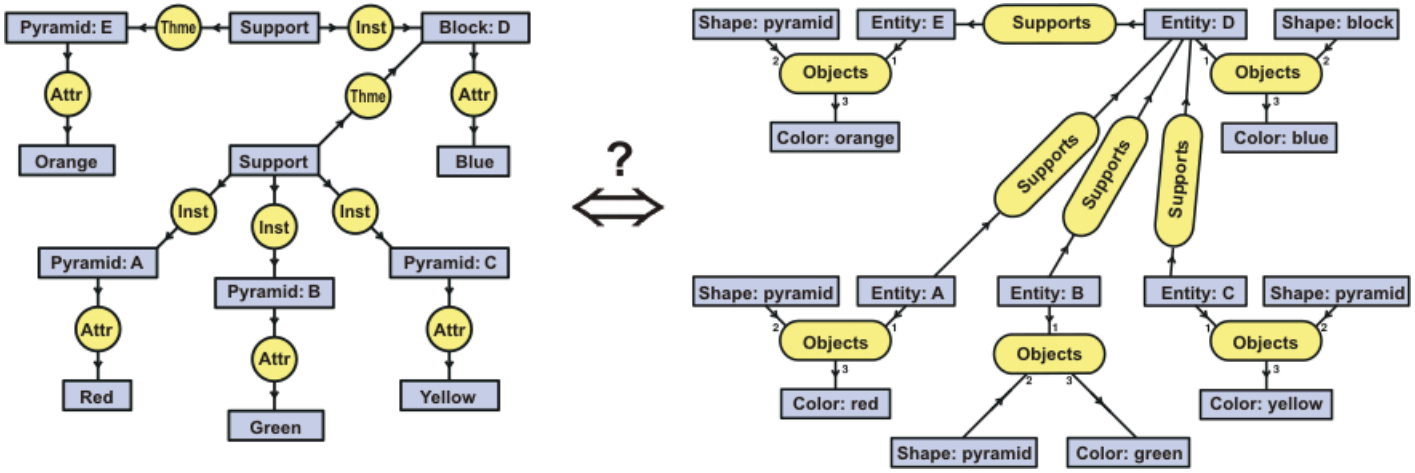




Join concept nodes that refer to the same entities.

Closely related entities are described by connected graphs.

Mapping the Two Graphs to One Another



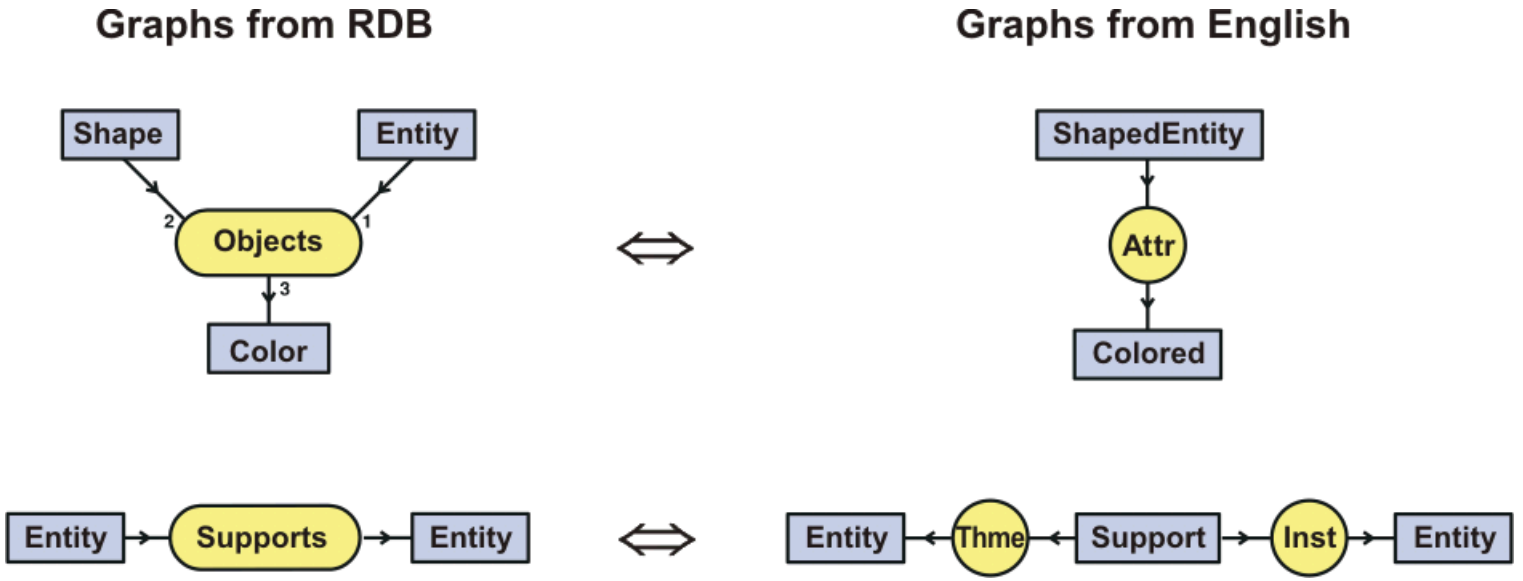
Very different ontologies: 12 concept nodes vs. 15 concept nodes, 11 relation nodes vs. 9 relation nodes, no similarity in type labels.

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.

Problem for Legacy Re-engineering

Analyze software and documentation of a large corporation.

Programs in daily use, some of which up to 40 years old:

- **1.5 million lines of COBOL programs.**
- **100 megabytes of English documentation — reports, manuals, e-mails, Lotus Notes, HTML, and program comments.**

Goal:

- **Analyze the COBOL programs.**
- **Analyze the English documentation.**
- **Compare the two to generate**

  English glossary of all terms with pointers to the software,

  Structure diagrams of the programs, files, and data,

  List of discrepancies between software and documentation.

An Important Simplification

An extremely difficult, still unsolved problem:

- Translate English specifications to executable programs.

Much easier task:

- Translate the COBOL programs to conceptual graphs.
- Use the conceptual graphs from COBOL to interpret the English.
- Use the analogy engine to compare the graphs derived from COBOL to the graphs derived from English.
- Record the similarities and discrepancies.

Excerpt from the Documentation

The input file that is used to create this piece of the Billing Interface for the General Ledger is an extract from the 61 byte file that is created by the COBOL program BILLCRUA in the Billing History production run. This file is used instead of the history file for time efficiency. This file contains the billing transaction codes (types of records) that are to be interfaced to General Ledger for the given month.

For this process the following transaction codes are used: 32 — loss on unbilled, 72 — gain on uncollected, and 85 — loss on uncollected. Any of these records that are actually taxes are bypassed. Only client types 01 — Mar, 05 — Internal Non/Billable, 06 — Internal Billable, and 08 — BAS are selected. This is determined by a GETBDATA call to the client file.

Note that none of the files or COBOL variables are named.

By matching the English graphs to the COBOL graphs, VAE identified all the file names and COBOL variables involved.

Results

Job finished in 8 weeks by two programmers, Arun Majumdar and André LeClerc.

- **Four weeks for customization:**

  Design, ontology, and additional programming for I/O formats.

- **Three weeks to run English parser + VAE + extensions:**

  VAE handled matches with strong evidence (close semantic distance).

  Matches with weak evidence were confirmed or corrected by Majumdar and LeClerc.

- **One week to produce a CD-ROM with integrated views of the results:**

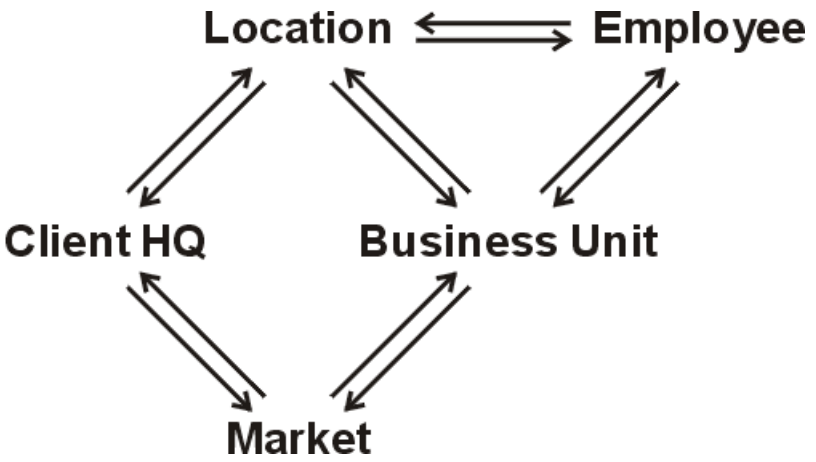
  Glossary, data dictionary, data flow diagrams, process architecture, system context diagrams.

A major consulting firm had estimated that the job would take 40 people two years to analyze the documentation and determine the cross references.

With VAE, it was done in 15 person weeks.

Mismatch Found by VAE

A diagram of relationships among data types in the database:



Question: Which location determines the market?

According to the documentation: Business unit.

According to the COBOL programs: Client HQ.

Management had been making decisions based on incorrect assumptions.

Contradiction Found by VAE

From analyzing English documentation:

- **Every employee is a human being.**
- **No human being is a computer.**

From analyzing COBOL programs:

- **Some employees are computers.**

What is the reason for this contradiction?

Quick Patch in 1979

A COBOL programmer made a quick patch:

- Two computers were used to assist human consultants.
- But there was no provision to bill for computer time.
- Therefore, the programmer named the computers Bob and Sally, and assigned them employee ids.

For more than 20 years:

- Bob and Sally were issued payroll checks.
- But they never cashed them.

VAE discovered the two computer “employees.”

Some Observations

These examples used semantic technology that is not yet widely available in commercial IT.

The VAE system was “consultant-ware.”

- **Two superprogrammers could use it effectively.**
- **But it wasn't “shrink-wrapped” software.**
- **And it required special skills to use.**

But many such systems demonstrate feasibility.

As Common Logic becomes more widely available, tools based on CL can be integrated with the IT mainstream.

Controlled NLs and diagrams can support the user interface.

Knowledge Discovery

Observation by the philosopher Immanuel Kant:

> **Socrates said he was the midwife to his listeners, i.e., he made them reflect better concerning that which they already knew, and become better conscious of it. If we always knew what we know, namely, in the use of certain words and concepts that are so subtle in application, we would be astonished at the treasures contained in our knowledge...**
>
> **Platonic or Socratic questions drag out of the other person's cognitions what lay within them, in that one brings the other to consciousness of what he actually thought.**
>
> **From his *Vienna Logic***

We need tools that can play the role of Socrates.

They should help us discover the implicit knowledge and use it to process the huge volumes of digital data.

For other views of semantics, see http://vimeo.com/3829682

Related Readings

Fads and Fallacies About Logic, by J. F. Sowa,
http://www.jfsowa.com/pubs/fflogic.pdf

Conceptual Graphs, by J. F. Sowa,
http://www.jfsowa.com/cg/cg_hbook.pdf

Two paradigms are better than one, but multiple paradigms are even better, by A. K. Majumdar & J. F. Sowa, http://www.jfsowa.com/pubs/paradigm.pdf

Pursuing the goal of language understanding, by A. K. Majumdar, J. F. Sowa, & J. Stewart, http://www.jfsowa.com/pubs/pursuing.pdf

Papers from a recent workshop on controlled natural languages,
http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-448/

Web site for controlled natural languages,
http://sites.google.com/site/controllednaturallanguage/

ISO/IEC standard 24707 for Common Logic,
http://standards.iso.org/ittf/PubliclyAvailableStandards/c039175_ISO_IEC_24707_2007(E).zip