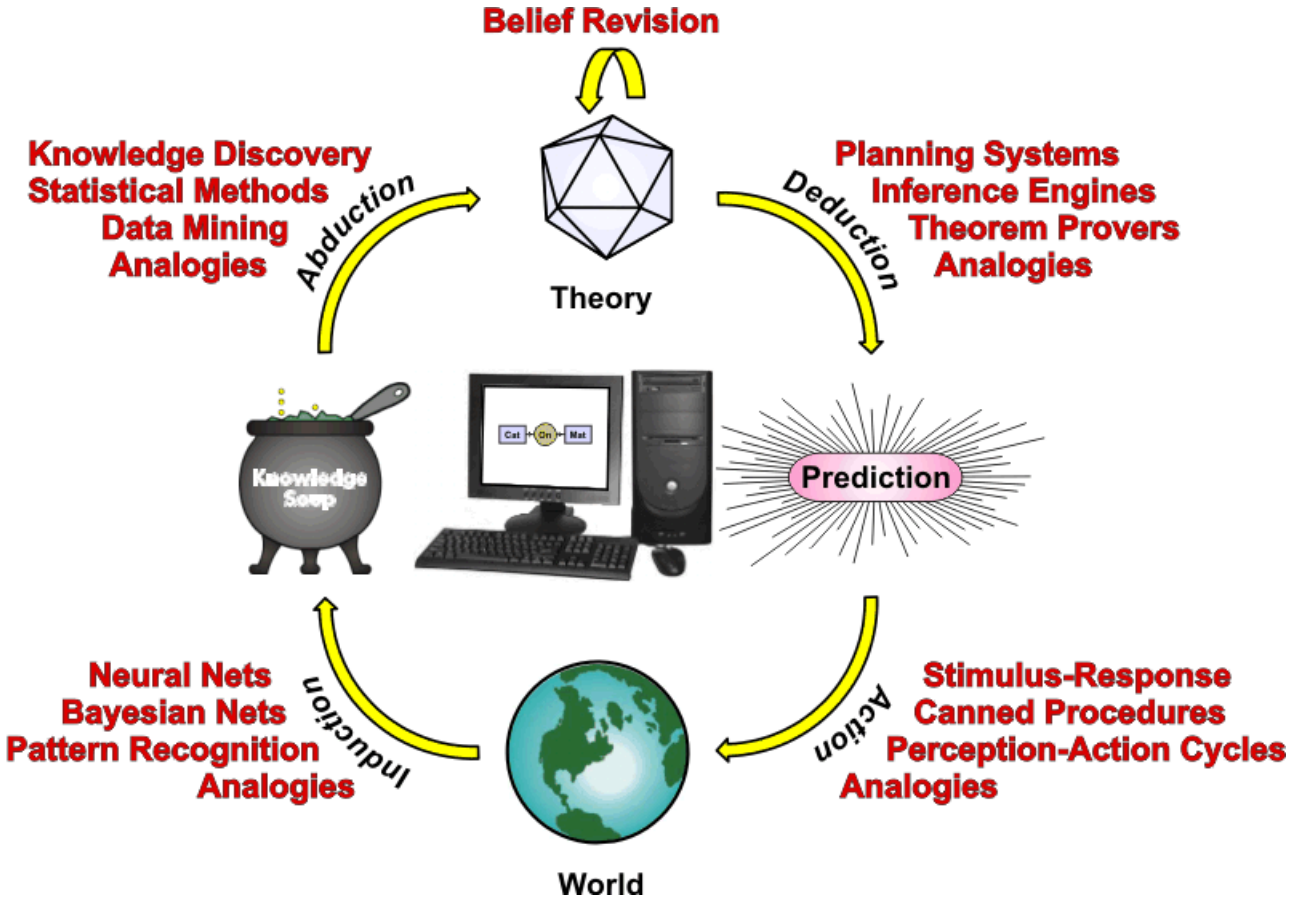
Methods of Reasoning

John F. Sowa

20 March 2013

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.

Existential Graphs

EGs for full first-order logic are based on three primitives:

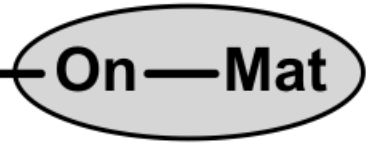
Existence: —

Negation: 

Relations: Cat- -On- -Under- -With- -Mat

A cat is on a mat: Cat—On—Mat

Something is under a mat: —Under—Mat

Some cat is not on a mat: Cat—  On—Mat

Some cat is on something that is not a mat: Cat—On—  Mat

Peirce's Rules of Inference

Peirce's rules support the simplest, most general reasoning method ever invented for any logic.

Three pairs of rules, which insert or erase a graph or subgraph:

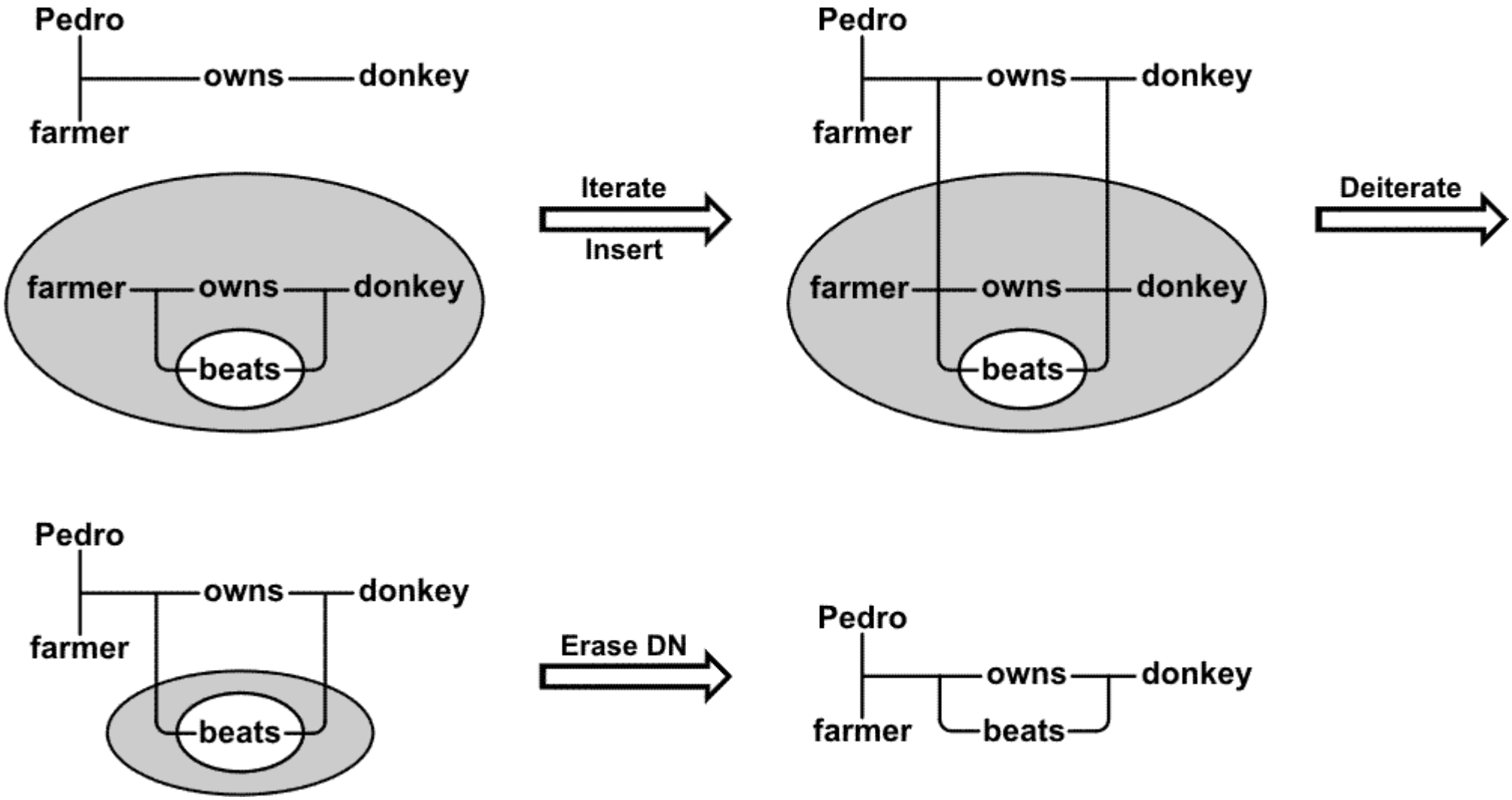
1. **Insert/Erase: Insert anything in a negative area; erase anything in a positive area.**
2. **Iterate/Deiterate: Iterate (copy) anything in the same area or any nested area; deiterate (erase) any iterated copy.**
3. **Double negation: Insert or erase a double negation (pair of ovals with nothing between them) around anything in any area.**

These rules are stated in terms of EGs.

But they can be adapted to many syntaxes, including CGs, predicate calculus, frame notations, and even natural language.

For more detail, see Peirce's own explanation of these rules:
http://www.jfsowa.com/pubs/egtut.pdf

A Proof by Peirce's Rules



Conclusion: ***Pedro is a farmer who owns and beats a donkey*.**

Proving a Theorem

Peirce's only axiom is the empty graph – a blank sheet of paper.

- **The empty graph cannot say anything false.**
- **Therefore, the empty graph is always true.**
- **Silence is golden.**

A theorem is a proposition that is proved from the empty graph.

- **For the first step, only one rule can be applied: draw a double negation around a blank area.**
- **The next step is to insert the hypothesis into the negative area.**

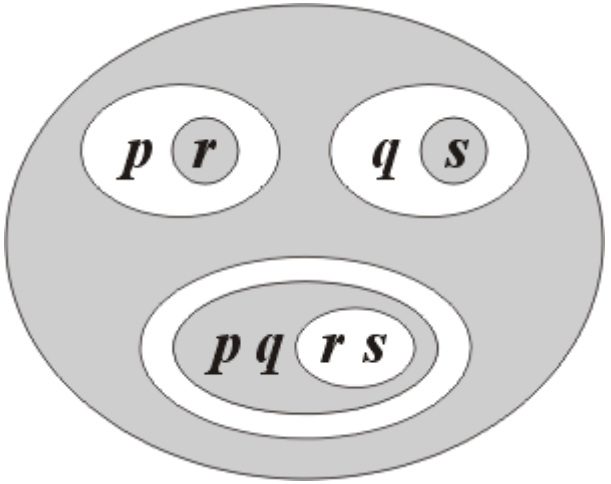
The Praeclarum Theorema (splendid theorem) by Leibniz:

$$((p \supset r) \wedge (q \supset s)) \supset ((p \wedge q) \supset (r \wedge s))$$

In the *Principia Mathematica*, Whitehead and Russell took 43 steps to prove this theorem.

With Peirce's rules, the proof takes only 7 steps.

Praeclarum Theorema

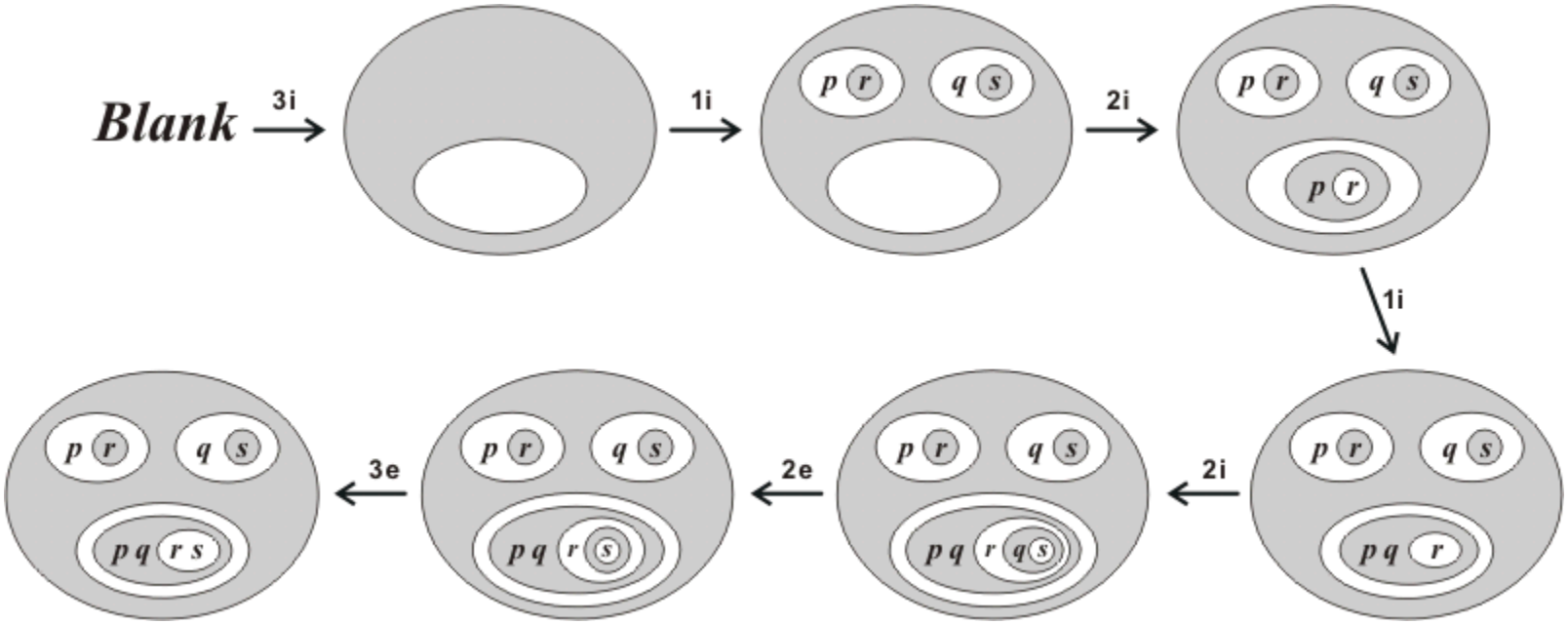


$$((p \supset r) \wedge (q \supset s)) \supset ((p \wedge q) \supset (r \wedge s))$$

Note that the if-parts of ($p \supset r$) and ($q \supset s$) are white, because those areas are nested two levels deep.

But the if-part of ($p \wedge q$) $\supset$ ($r \wedge s$) is shaded, because that area is nested three levels deep.

Proof of the Praeclarum Theorema



Each step is labeled with the number of the rule:

3i, insert double negation. 1i, insert ((*p*⊃*r*) ∧ (*q*⊃*s*)). 2i, iterate (*p*⊃*r*). 1i, insert *q*. 2i, iterate (*q*⊃*s*). 2e, deiterate q. 3e, erase double negation.

For humans, perception determines which rule to apply.

Look ahead to the conclusion to see which rule would make the current graph look more like the target graph.

Proof of the Praeclarum Theorema in CGIF

1. By 3i, draw a double negation around the blank: ~[~[ ]]

2. By 1i, insert the hypothesis in the negative area:
   ~[~[(p) ~[(r)]] ~[(q) ~[(s)]] ~[ ]]

3. By 2i, iterate the left part of the hypothesis into the conclusion:
   ~[~[(p) ~[(r)]] ~[(q) ~[(s)]] ~[~[(p) ~[(r)]]]]

4. By 1i, insert (q):
   ~[~[(p) ~[(r)]] ~[(q) ~[(s)]] ~[~[(p) (q) ~[(r)]]]]

5. By 2i, iterate the right part of the hypothesis into the innermost area:
   ~[~[(p) ~[(r)]] ~[(q) ~[(s)]] ~[~[(p) (q) ~[(r)] ~[(q) ~[(s)]]]]]

6. By 2e, deiterate (q):
   ~[~[(p) ~[(r)]] ~[(q) ~[(s)]] ~[~[(p) (q) ~[(r)] ~[~[(s)]]]]]

7. By 3e, erase the double negation to generate the conclusion:
   ~[~[(p) ~[(r)]] ~[(q) ~[(s)]] ~[~[(p) (q) ~[(r)] (s)]]]

Applying Peirce's Rules to Other Notations

With minor changes, Peirce's rules can be used with many logic notations, including controlled subsets of natural languages.

Definition: Proposition X is more general (or specialized) than Y iff the models for X are a proper superset (subset) of the models for Y.

Modified version of Peirce's first pair of rules:

- **Insert: In a negative context, any propositional expression may be replaced by a more specialized expression.**
- **Erase: In a positive context, any propositional expression may be replaced by a more general expression.**

The rules of Iterate/Deiterate and Double Negation are unchanged.

This modification holds for existential graphs, since erasing any subgraph makes a graph more general.

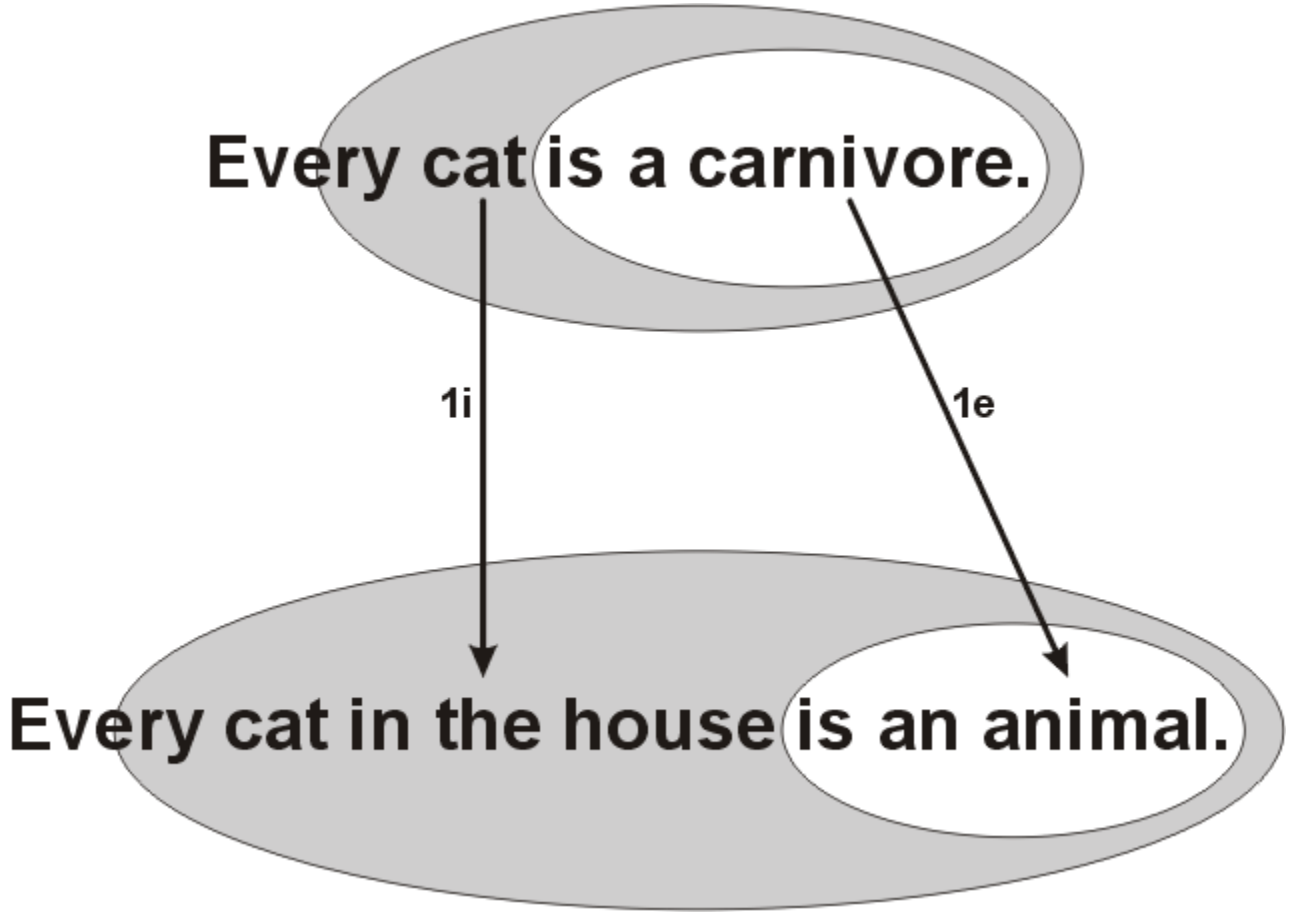
But this version can be easier to apply to other notations.

Peirce's Rules Applied to English

Use shading to mark the positive and negative parts of each sentence.

Rule 1i specializes 'cat' to 'cat in the house'.

Rule 1e generalizes 'carnivore' to 'animal'.



This method of reasoning is sound for sentences that can be mapped to a formal logic. It can also be used on propositional parts of sentences that contain some nonlogical features.

A Proof in English

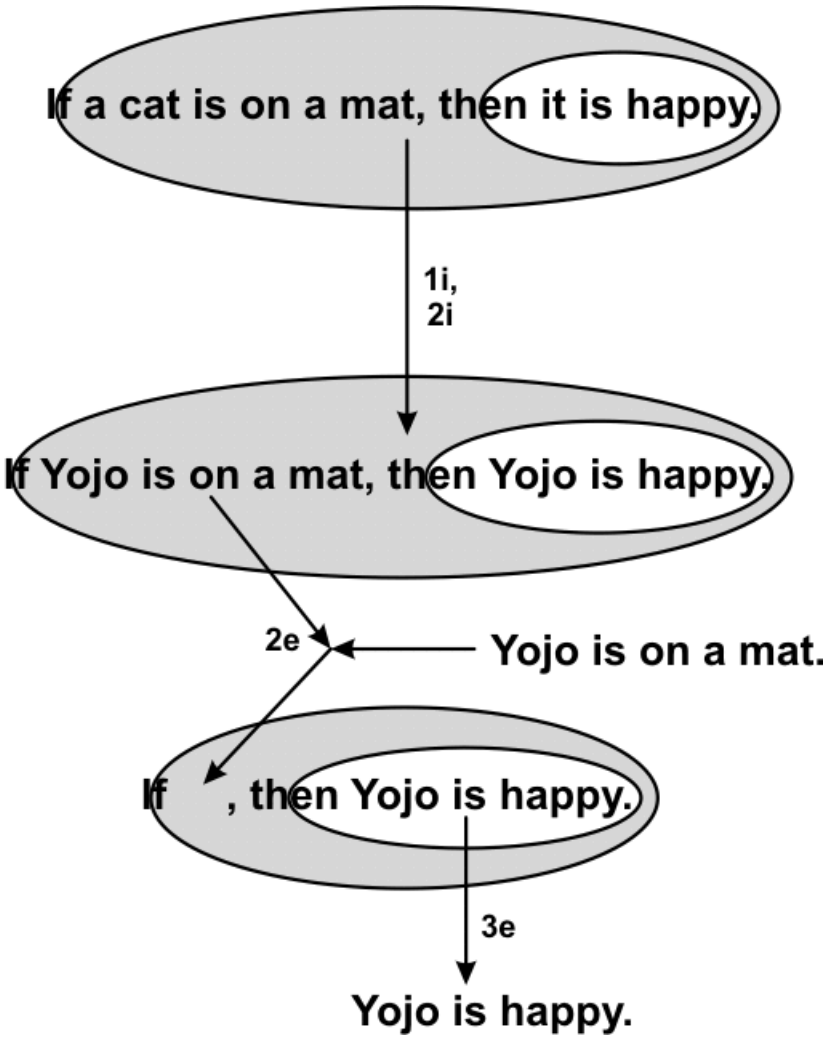
Use shading to mark positive and negative parts of each sentence.

Rule 1i specializes 'a cat' to 'Yojo', and Rule 2i iterates 'Yojo' to replace the pronoun 'it'.

Rule 2e deiterates the nested copy of the sentence 'Yojo is on a mat'.

As a result, there is nothing left between the inner and outer negation of the if-then nest.

Finally, Rule 3e erases the double negation to derive the conclusion.



Theoretical Issues

Peirce's rules have some remarkable properties:

- **Simplicity: Each rule inserts or erases a graph or subgraph.**
- **Symmetry: Each rule has an exact inverse.**
- **Depth independence: Rules depend on the positive or negative areas, not on the depth of nesting.**

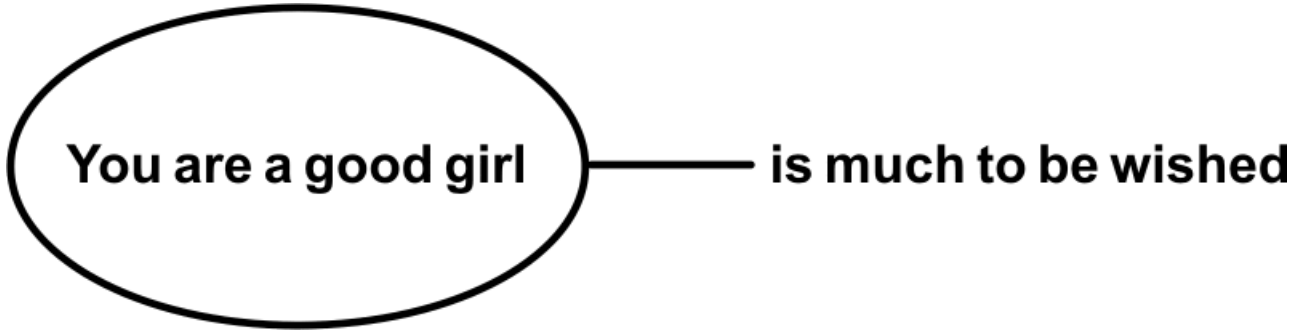
They allow short proofs of remarkable theorems:

- **Reversibility Theorem. Any proof from *p* to *q* can be converted to a proof of ~*p* from ~*q* by negating each step and reversing the order.**
- **Cut-and-Paste Theorem. If *q* can be proved from *p* on a blank sheet, then in any positive area where *p* occurs, *q* may be substituted for *p*.**
- **Resolution and natural deduction: Any proof by resolution can be converted to a proof by Peirce's version of natural deduction by negating each step and reversing the order.**

For proofs of these theorems and further discussion of the issues, see Section 6 of http://www.jfsowa.com/pubs/egtut.pdf

Metalanguage

A relation attached to an oval makes a metalevel statement about the proposition expressed by the nested graph. *



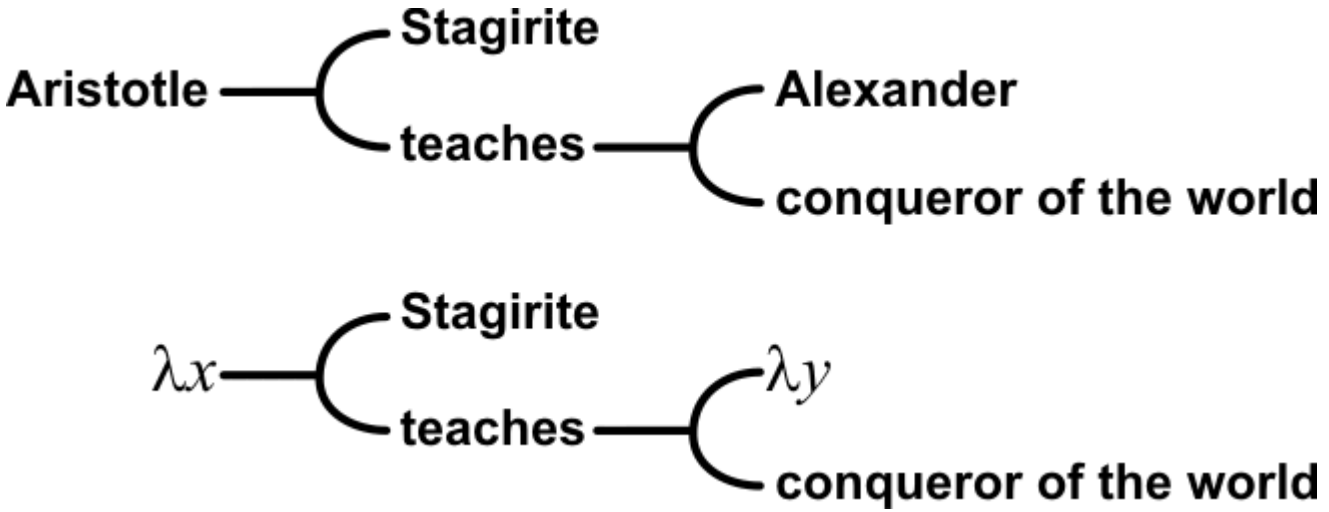
Peirce allowed the names of relations to contain blanks.

The relation named 'You are a good girl' has zero arguments. It is an existential graph that express a proposition.

The relation named 'is much to be wished' has one argument. It makes a statement about the proposition inside the oval.

*** From Charles Sanders Peirce, *Reasoning and the Logic of Things*, The Cambridge Conferences Lectures of 1898, Harvard University Press, p. 151.**

Lambda Abstraction



The top EG says *Aristotle is a Stagirite who teaches Alexander who conquers the world*.

In the EG below it, the names Aristotle and Alexander are erased, and their places are marked with the Greek letter λ.

That EG represents a dyadic relation: ___ *is a Stagirite who teaches* ___ *who conquers the world*.

Peirce used an underscore to mark those empty places, but Alonzo Church marked them with λ.

Psychological Issues

Endorsement by the psychologist Philip Johnson-Laird (2002):

> **"Peirce's existential graphs... establish the feasibility of a diagrammatic system of reasoning equivalent to the first-order predicate calculus."**
>
> **"They anticipate the theory of mental models in many respects, including their iconic and symbolic components, their eschewal of variables, and their fundamental operations of insertion and deletion."**
>
> **"Much is known about the psychology of reasoning... But we still lack a comprehensive account of how individuals represent multiply-quantified assertions, and so the graphs may provide a guide to the future development of psychological theory."**

Johnson-Laird published many papers about mental models.

His comments on that topic are significant, especially in combination with the other properties of the graphs.

Mental Maps, Images, and Models

The neuroscientist Antonio Damasio (2010):

> **"The distinctive feature of brains such as the one we own is their uncanny ability to create maps... But when brains make maps, they are also creating images, the main currency of our minds. Ultimately consciousness allows us to experience maps as images, to manipulate those images, and to apply reasoning to them."**

The maps and images form mental models of the real world or of the imaginary worlds in our hopes, fears, plans, and desires.

Words and phrases of language can be generated from them.

They provide a "model theoretic" semantics for language that uses perception and action for testing models against reality.

Like Tarski's models, they define the criteria for truth, but they are flexible, dynamic, and situated in the daily drama of life.

Role of Analogy

The basis for human reasoning and language understanding.

Logic is a disciplined special case of analogical reasoning:

- **Essential for precise reasoning in mathematics and science.**
- **Important for precision in any field.**
- **But even in science and engineering, analogy is necessary for knowledge discovery and innovation.**

Conceptual graphs support logical and analogical methods:

- **CGs derived from English can be used for analogies.**
- **But CGs used for formal logic should be derived from formal languages or be corrected by comparison to formal CGs.**

Describing Things in Different Ways

How can we describe what we see?

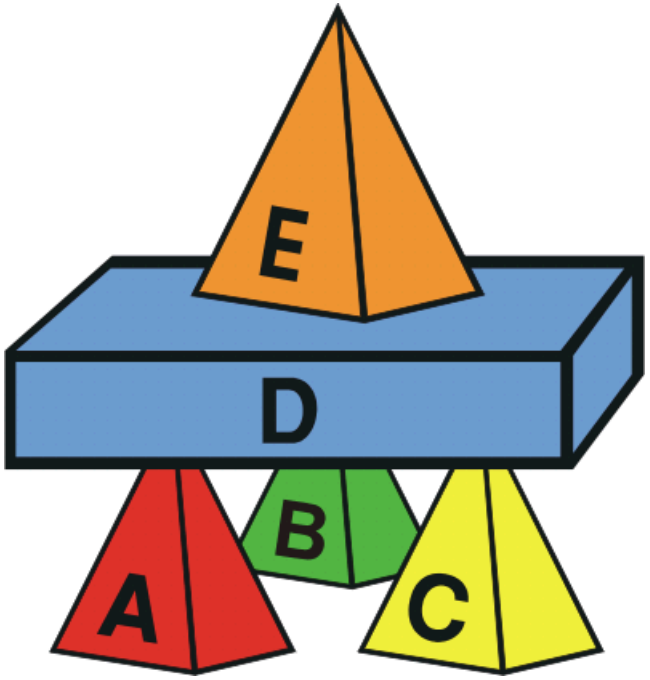
In ordinary language?

In some version of logic?

In a relational database?

In the Semantic Web?

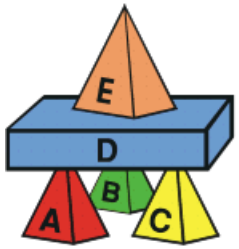
In a programming language?

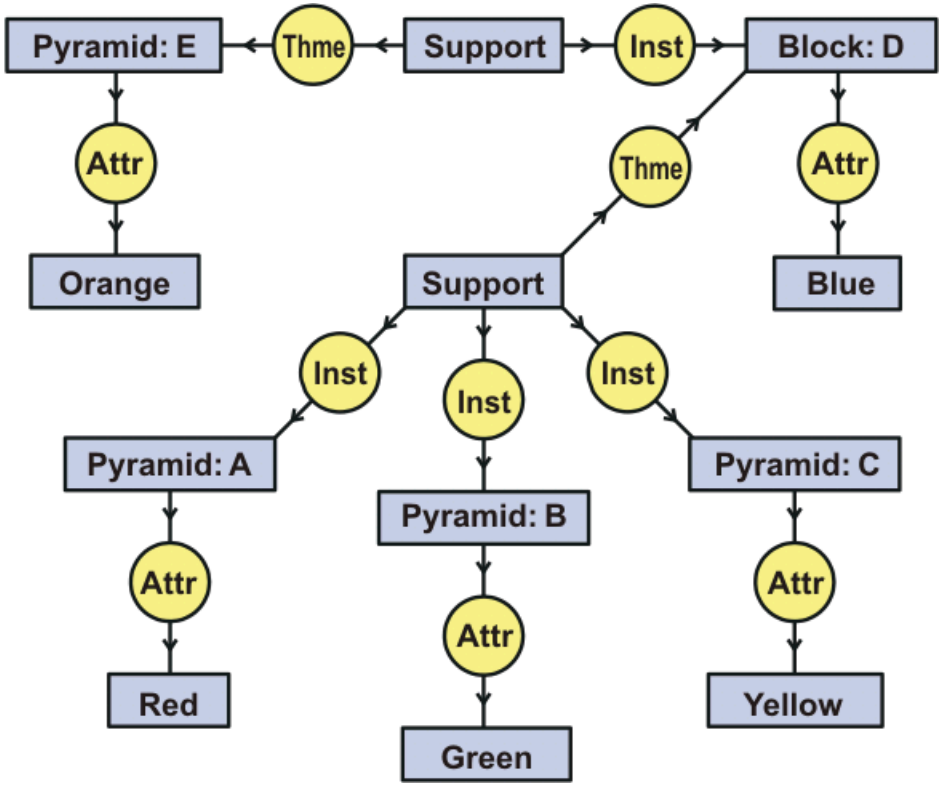


Even when people use the same language, they use different words and expressions.

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

Mapping English to a Conceptual Graph



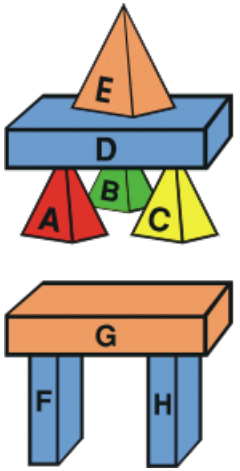


"A red pyramid A, a green pyramid B, and a yellow pyramid C support a blue block D, which supports an orange pyramid E."

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

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.

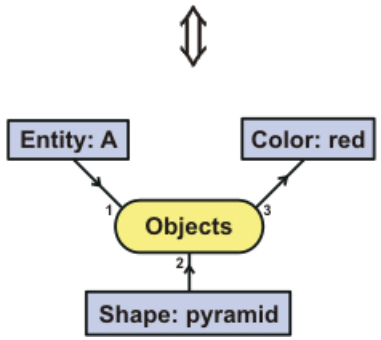
Mapping Database Relations to Conceptual Relations

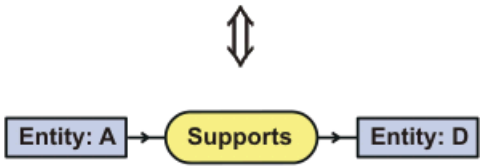
Objects

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

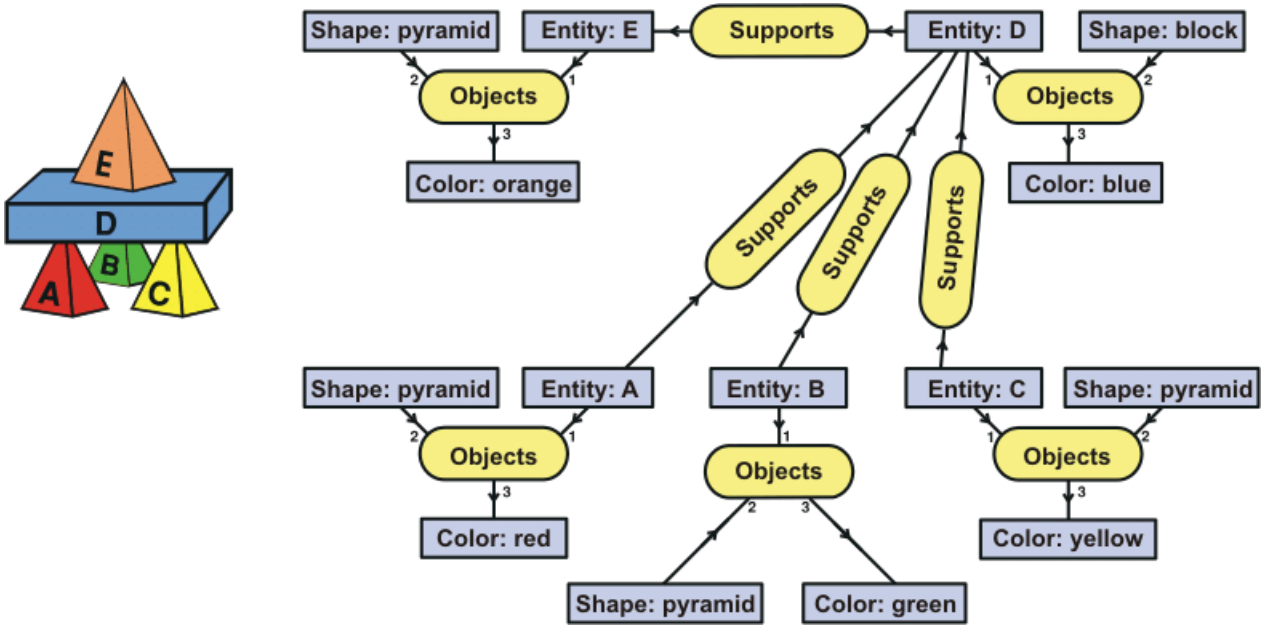
Supporter	Supportee
A	D
B	D
C	D
D	E
F	G
H	G





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

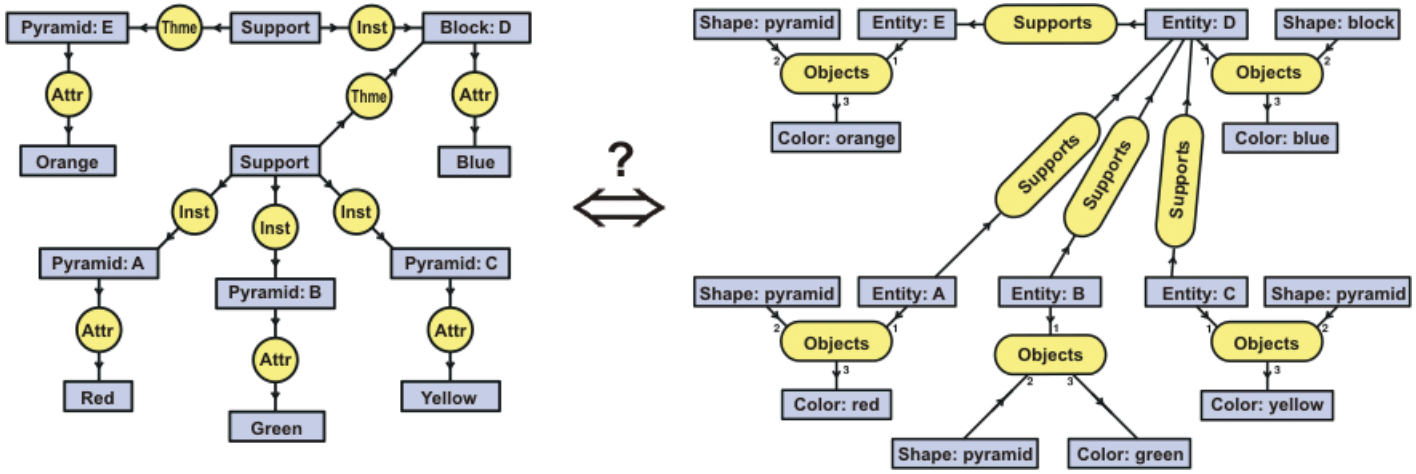
Mapping an Entire Database to Conceptual Graphs



Join concept nodes that refer to the same entities.

Closely related entities are described by connected graphs.

Mapping the Two Graphs to One Another



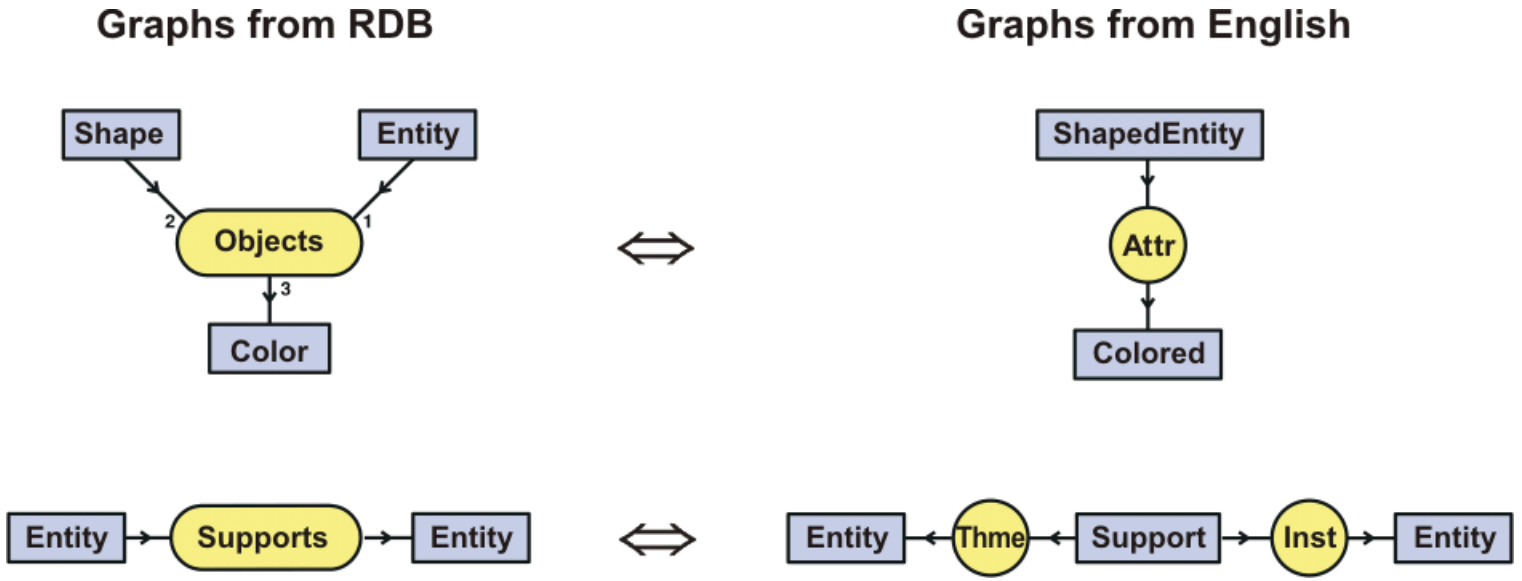
Very different structures: 12 concept nodes vs. 15 concept nodes, 11 relation nodes vs. 9 relation nodes, no similarity in type labels.

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 Structure Mapping



Repeated application of these two transformations perform an exact mapping of all the 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.

Approximate Mapping

Example: ***How is a cat like a car?***

Cat	Car
Head	Hood
Eye	Headlight
Cornea	Glass plate
Mouth	Fuel cap
Stomach	Fuel tank
Bowel	Combustion chamber
Anus	Exhaust pipe
Skeleton	Chassis
Heart	Engine
Paw	Wheel
Fur	Paint

Metaphor and other aspects of language require approximations.

Methods of Approximate Matching

Concept types are related by ontology or common associations:

- Eyes and headlights are related to light.
- Heart and engine are internal parts with a regular beat.
- Skeleton and chassis are structures for attaching parts.
- Paws and wheels support the body, and there are four of each.

One-to-one structure matching is preferred:

- Head → Eyes → Cornea.
- Hood → Headlights → Glass plate.

Approximate matches may skip some nodes (marked in red):

- Mouth → Esophagus → Stomach → Bowel → Anus.
- Fuel cap → Fuel tank → Combustion chamber → Muffler → Exhaust pipe.

Two factors determine the semantic distance between graphs:

- Ontology: How similar are the concept and relation types?
- Structure: How many nodes and arcs are added or deleted?

Computational Complexity

Research by Falkenhainer, Forbus, & Gentner:

- **Pioneers in finding analogies with their Structure Mapping Engine.**
- **Showed that SME algorithms take time proportional to N^3, where N is the number of frames (or graphs) in the knowledge base.**
- **MAC/FAC approach: Use a search engine to narrow down the number of likely candidates before using SME.**

VivoMind approach:

- **Encode graph structure and ontology in a Cognitive Signature™.**
- **For any graph, find closely matching signatures in log(N) time.**
- **Only graphs with similar signatures are likely candidates.**

For papers by Falkenhainer, Forbus, Genter, and their colleagues, see
http://www.qrg.northwestern.edu/papers/papers.html

For references to VivoMind technology, see the last slide in this presentation.

Algorithms for Chemical Graphs

Graphs of organic molecules are similar to conceptual graphs:

- **Atoms ⇒ concept nodes labeled by the name of the element.**
- **Chemical bonds ⇒ relation nodes labeled by the name of the bond type.**
- **But conceptual graphs have many more types of concepts and relations.**

Chemical graphs inspired Peirce's existential graphs as representations of "the atoms and molecules of logic."

Some of the largest and most sophisticated systems for graph processing were developed by chemists, not computer scientists.

An early example was the use of chemical graph algorithms for building and searching hierarchies of conceptual graphs:

Robert A. Levinson, & Gerard Ellis (1992) Multilevel hierarchical retrieval, *Knowledge Based Systems* 5:3, pp. 233-244.

Chemical Graph Search Engine

Find similar chemical graphs in logarithmic time:

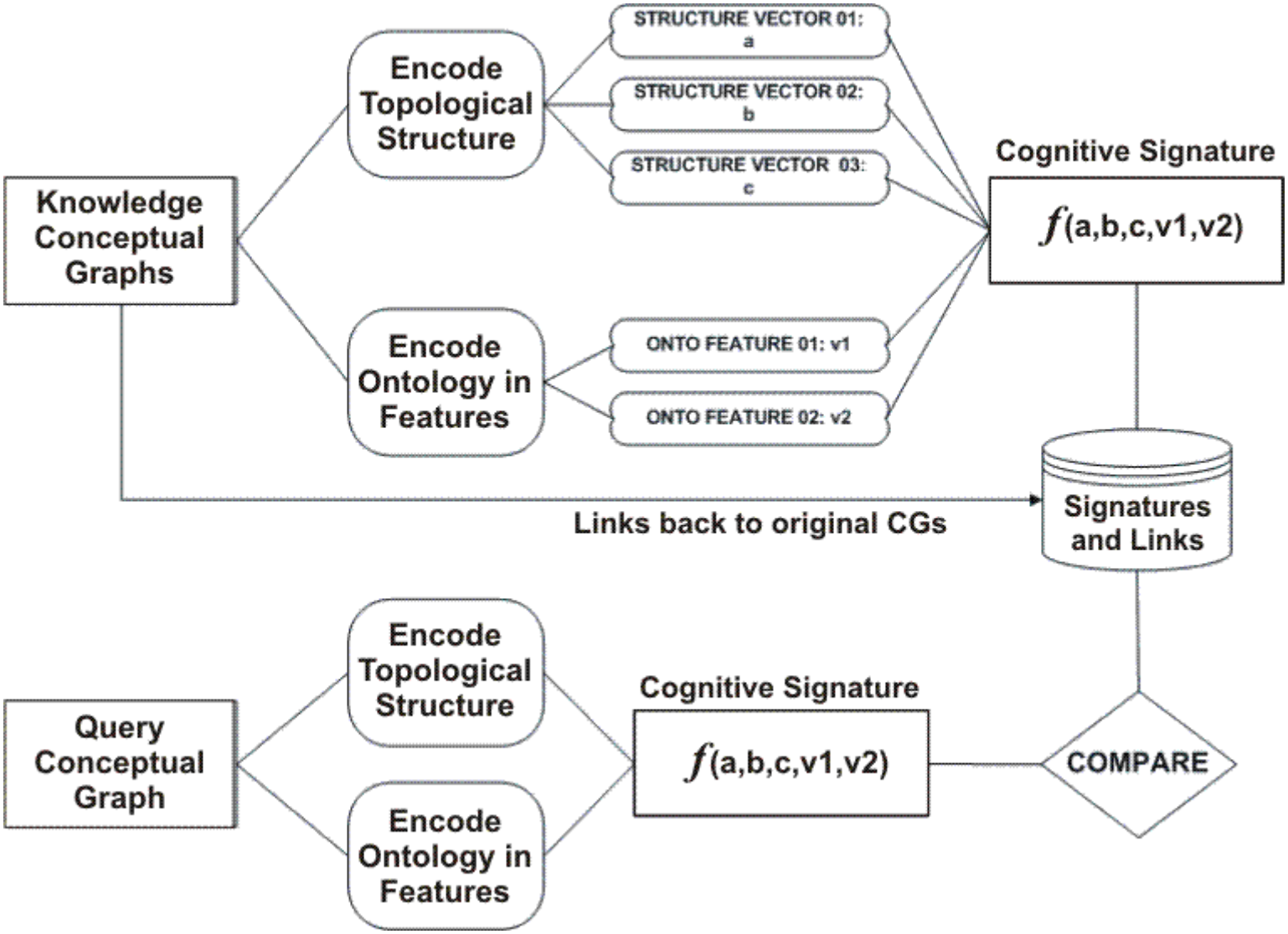
- Represent each graph by its unique International Chemical Identifier (InChI).
- Map the InChI codes to numeric vectors that encode both the graph structure and the labels of the atoms and bonds.
- Estimate the semantic distance between graphs by a measure based on both the graph structure and the labels on the nodes and arcs (atoms and bonds).
- Index the vectors by a locality-sensitive hashing (LSH) algorithm.
- Use the semantic distance measure to find the most similar graphs.

Similar techniques can be adapted to conceptual graphs.

For details of the chemical algorithms, see

Mining Patents Using Molecular Similarity Search, by James Rhodes, Stephen Boyer, Jeffrey Kreulen, Ying Chen, & Patricia Ordonez, http://psb.stanford.edu/psb-online/proceedings/psb07/rhodes.pdf

Cognitive Memory™



Basis for the VivoMind Analogy Engine (VAE)

Evaluating Student Answers

Multiple-choice questions are easy to evaluate by computer.

Long essays are often evaluated by statistical methods.

But short answers about mathematics are very hard to evaluate.

Sample question:

> ***The following numbers are 1 more than a square: 10, 37, 65, 82.***
> ***If you are given an integer N that is less than 200,***
> ***how would you determine whether N is 1 more than a square?***
>
> ***Explain your method in three or four sentences.***

Even experienced teachers must spend a lot of time checking and correcting the answers to such questions.

Many Possible Answers

An example of a correct answer:

> ***To show that N is 1 more than a square, show that N−1 is a square.***
>
> ***Find some integer x whose square is slightly less than N−1.***
> ***Compare N−1 to the squares of x, x+1, x+2, x+3, ...,***
> ***and stop when some square is equal to or greater than N−1.***
> ***If the last square is N−1, then N is one more than a square.***

How could a computer system evaluate such answers?

How could it make helpful suggestions for incorrect answers?

Publisher's Current Procedure

To evaluate new exam questions, the publisher normally gives the exam to a large number of students.

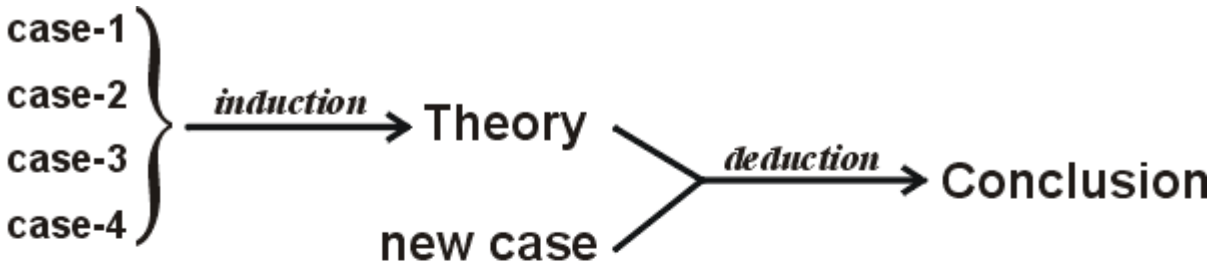
For each problem, they would get about 50 different answers:

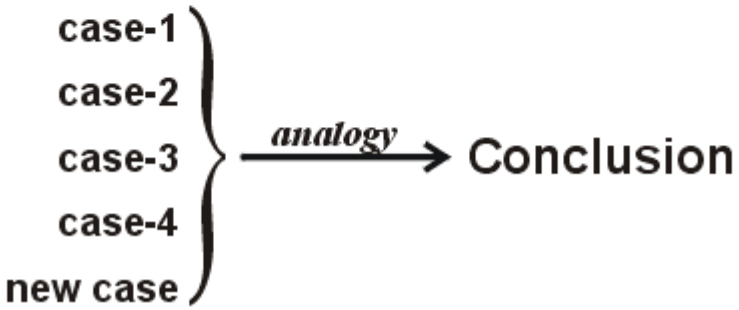
- **Some are completely correct**
  — but stated in different ways.
- **Some are partially correct**
  — and the teacher says what is missing.
- **Others are wrong**
  — in many different ways.

Result: 50 pairs of student answer and teacher's response.

Each answer-response pair is a case for case-based reasoning.

Case-Based Reasoning





Given the same cases, analogy takes one step to derive an answer that can take many steps by induction and deduction.

Analogy is usually more flexible, but a theory would be valuable if the same theory can be used and reused in multiple applications.

Using the VivoMind Language Processor

VLP translates all answers to conceptual graphs (CGs):

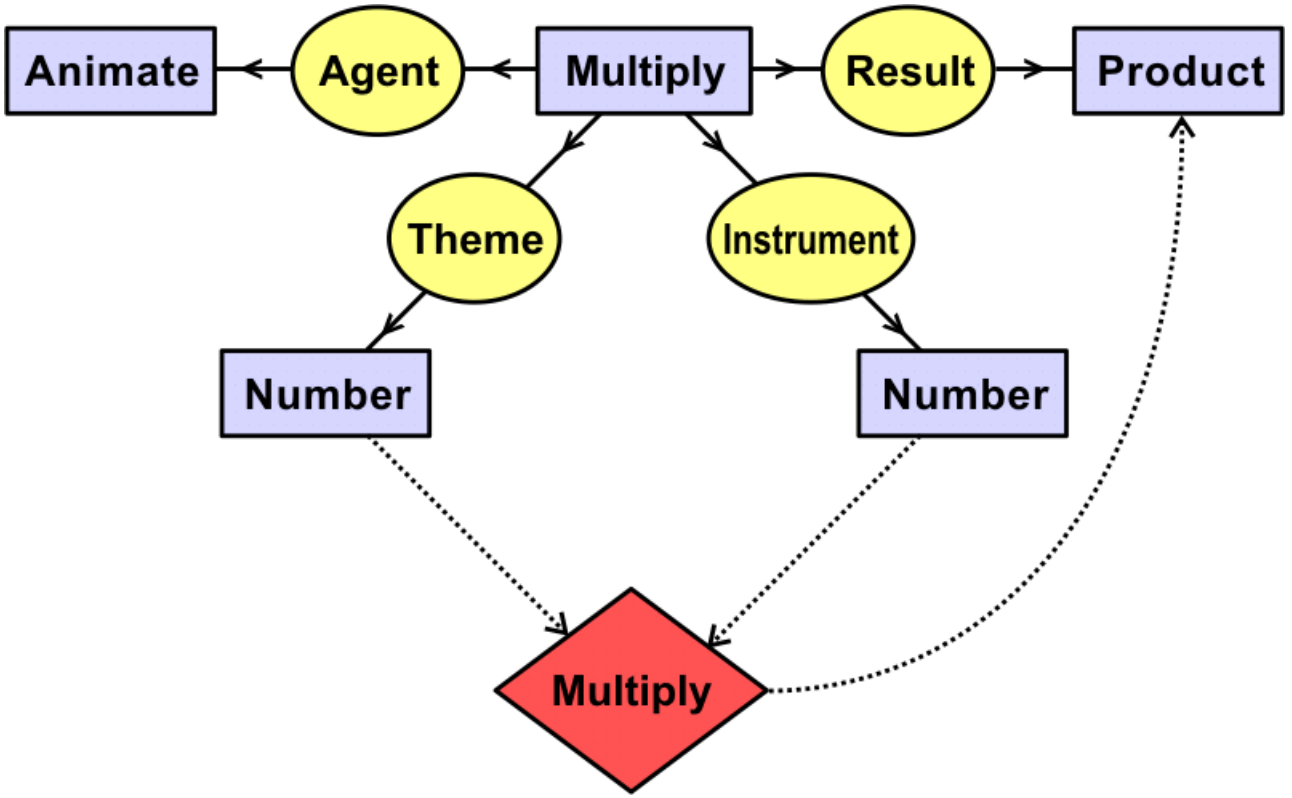
1. VLP uses a link grammar and a collection of lexical resources.
2. Canonical graphs for verbs are based on the IBM-CSLI verb ontology.
3. For this application, a small ontology of arithmetic was added.
4. The next slide shows a canonical graph for Multiply.

VAE compares each new answer to the 50 cases:

1. CGs for the answers of all 50 cases are stored in Cognitive Memory.
2. Compare the CG for each new answer to the CGs in Cognitive Memory.
3. If there is a good match, print out the teacher's previous response.
4. Otherwise, send the new student answer to some teacher to evaluate.
5. Add the new answer-response pair to the collection of cases.

This method combines a formal ontology for arithmetic with a very informal method of case-based reasoning.

Canonical Graph for Multiply



The boxes and circles represent an English sentence pattern:

[*Someone*] *multiplies a number by a number to get a product.*

The diamond node, called an actor, represents a function that computes the result of multiplying values inside the concept boxes.

Results

VAE found a good match for nearly all student answers.

For good matches, the stored response was appropriate.

Student answers are often incomplete or ungrammatical.

- **But canonical graphs make the parsing more robust.**
- **Resolve ambiguities by showing expected combinations.**
- **Use semantics to compensate and correct errors in syntax.**
- **Provide defaults for missing arguments.**
- **Show how information from multiple fragments can be related.**

The CGs derived from student answers were incomplete and unreliable for precise reasoning and calculation.

But they were adequate for approximate matching by VAE.

IBM Watson System

Beat human experts in answering Jeopardy! questions.

Shows how analogies can find relevant background knowledge:

- **Watson and VAE implement a version of structure mapping similar to the SME by Falkenhainer, Forbus, and Gentner (1989). ***
- **Both systems use analogies and case-based reasoning to relate natural language to a large volume of unstructured information.**
- **They can also use structured information stored in conventional databases and knowledge bases.**
- **But Watson does not have a search and retrieval method that can find analogies in logarithmic time.**
- **Instead, it uses a supercomputer with 2880 CPUs.**

*** See "Structure Mapping for Jeopardy! Clues" by J. William Murdock. *Proceedings of the 19th International Conference on Case Based Reasoning* (ICCBR'11), London. 2011. http://bill.murdocks.org/iccbr2011murdock_web.pdf**

Controlled Natural Language

Controlled NLs are not magic.

They 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 successful.

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).

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

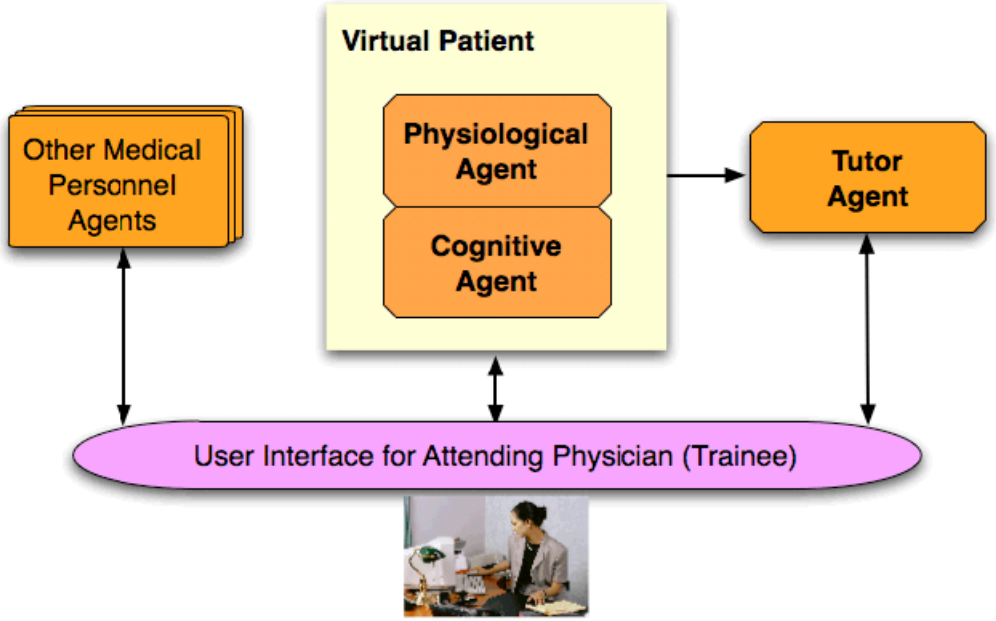
The three technologies have complementary strengths:

- **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.

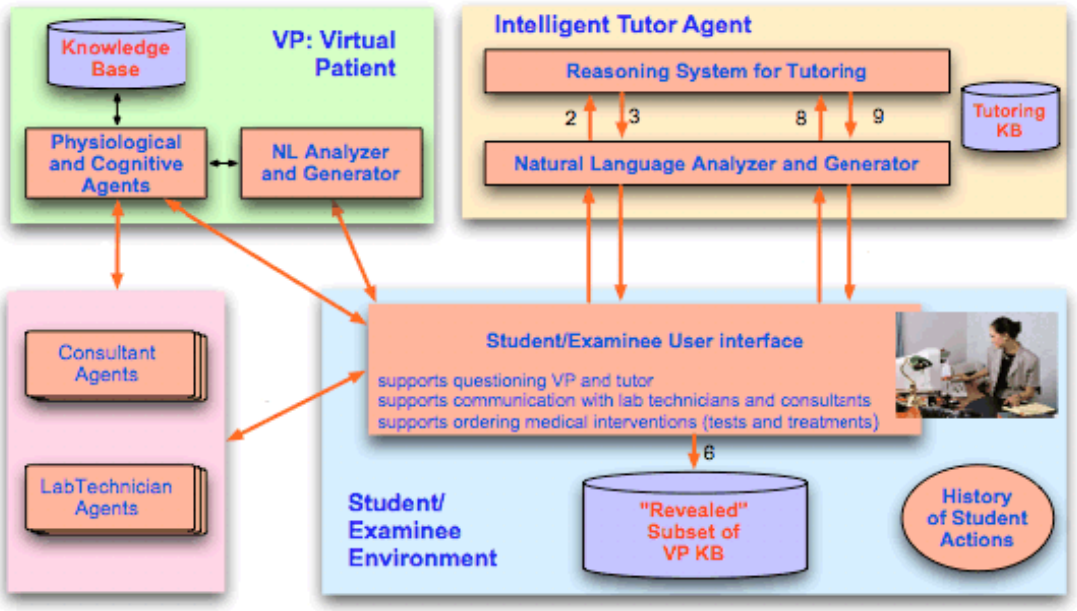
Source: Maryland virtual patient: a knowledge-based, language-enabled simulation and training system, by M. McShane, S. Nirenburg, B. Jarrell, S. Beale, & G. Fantry, http://bams.cm-uj.krakow.pl/bams3_pdf/bams%209.pdf

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.

Heterogeneous agents support multiple paradigms.

Reasoning by a Society of Agents

The Cyc approach to "classical" artificial intelligence:

- **Design a very large ontology with millions of axioms.**
- **Build a few dozen inference engines to process the axioms.**
- **Use deductive logic as the primary paradigm for reasoning.**

Marvin Minsky's Society of Mind:

- **Support reasoning by an open-ended number of agents.**
- **Allow different agents to use different reasoning paradigms.**
- **Use learning mechanisms to adapt the society to diverse problems.**
- **A powerful set of ideas that haven't been fully exploited.**

Society of Mind

Quotations by Marvin Minsky:

- What magical trick makes us intelligent? The trick is that there is no trick.
- The power of intelligence stems from our vast diversity, not from any single, perfect principle.
- Our species has evolved many effective although imperfect methods, and each of us individually develops more on our own.
- Eventually, very few of our actions and decisions come to depend on any single mechanism. Instead, they emerge from conflicts and negotiations among societies of processes that constantly challenge one another.
- To develop this idea, we will imagine first that this Mental Society works much like any human administrative organization.

Organizing a Society of Agents

1. Pandemonium. Selfridge (1959) designed a system of agents called demons. Each demon could observe aspects of the current situation or workspace, perform some computation, and put its results back into the workspace.

2. Rational agents. At the opposite extreme from simple demons are rational agents that simulate a human-like level of beliefs, desires, intentions, and the ability to reason about them.

3. Reactive agents. For designing robots, Brooks (1991) noted that the major challenge was not in deliberative planning and reasoning, but in the seemingly simpler insect-like functions of perception, locomotion, and goal seeking.

Instead of these three kinds of organizations, Minsky recommended a hierarchy of managers and employees.

A business organization enables tighter control over resources while accommodating agents with a wide variety of talents.

Flexible Modular Framework (FMF)

A platform for agents that communicate by passing messages.

Each message has 6 fields:

1. **Language. Identifier of the language used in the message.**
2. **Source. Identifier of the agent that sent the message.**
3. **Message ID. Identifier generated by the sender.**
4. **Destination. Identifier of the intended receiver, if known.**
5. **Pragmatics. The speech act or other purpose for the message.**
6. **Message. Any sentence or list of sentences in language #1.**

If the destination is 0, the message is sent to an associative blackboard, where suitable agents can find it and answer it.

Agents can be anywhere on the Internet.

Implementing a Society of Agents

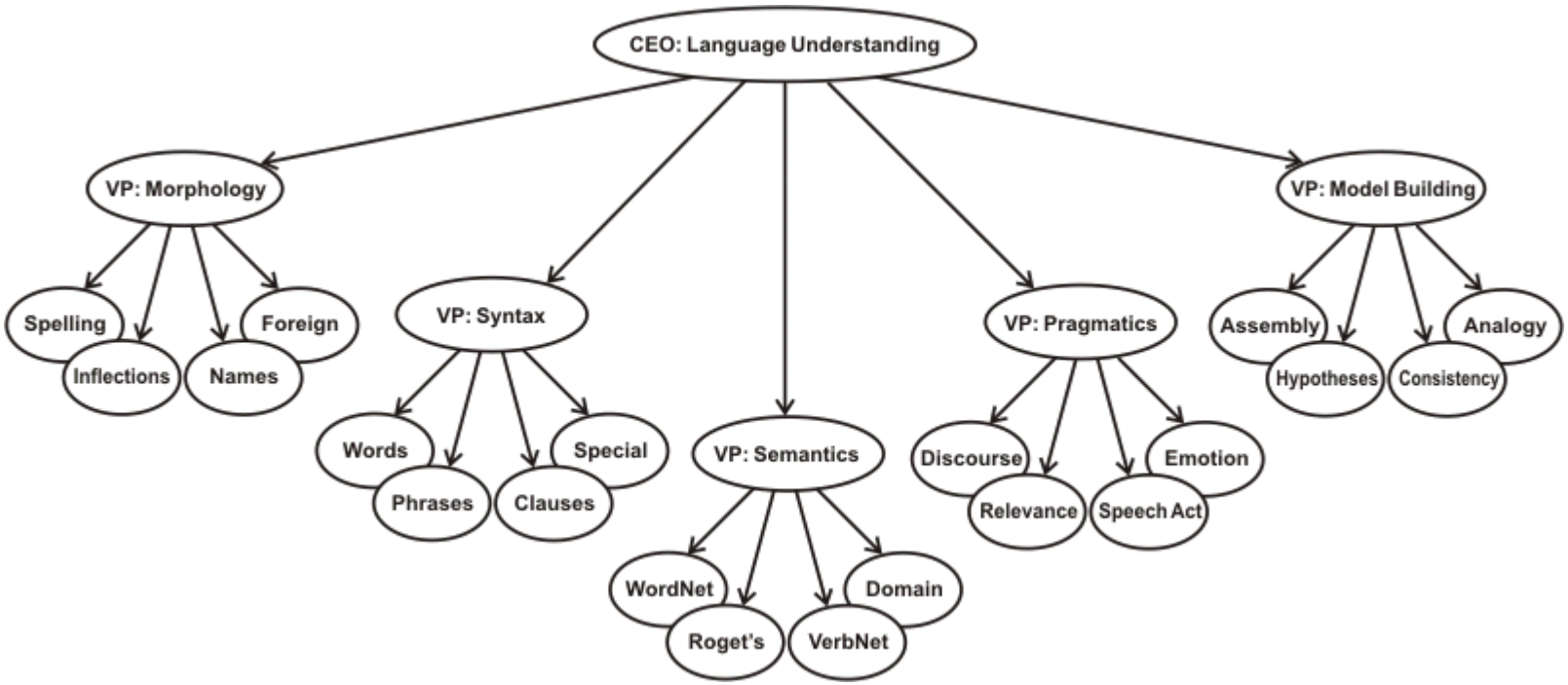
FMF protocols are simple and general:

- Based on message passing among agents of any kind.
- No restrictions on the message types or the language they use.
- No built-in or preferred organization among agents.
- Minimum space for a simple agent is 8K bytes.
- But agents can be arbitrarily large and complex.

Using the FMF to implement Minsky's hierarchy of agents:

- One agent is the Chief Executive Officer (CEO).
- Every employee except the CEO reports to exactly one manager.
- All employees report directly or indirectly to the CEO.
- All employees perform tasks assigned by their managers.
- Managers reward employees with space and time resources.

A Hierarchy of Agents for NLP



The CEO determines the goals and allocates resources.

Managers and employees are rewarded with resources (space and computer time) depending on how well they meet the goals.

Agents on all branches and levels work in parallel.

Suggested Readings

Two paradigms are better than one, and multiple paradigms are even better:
http://www.jfsowa.com/pubs/paradigm.pdf

A description of the VivoMind Analogy Engine:
http://www.jfsowa.com/pubs/arch.pdf

The Flexible Modular Framework used to implement the VivoMind software:
http://www.jfsowa.com/pubs/arch.pdf

The "Challenge of Knowledge Soup" for any approach to general AI:
http://www.jfsowa.com/pubs/challenge.pdf

A 22-page overview of conceptual graphs and the Common Logic standard:
http://www.jfsowa.com/cg/cg_hbook.pdf

Sowa, John F. (2011) Peirce's tutorial on existential graphs, *Semiotica* 186:1-4, 345-394.
http://www.jfsowa.com/pubs/egtut.pdf

Johnson-Laird, Philip N. (2002) Peirce, logic diagrams, and the elementary operations of reasoning,
***Thinking and Reasoning* 8:2, 69-95. http://mentalmodels.princeton.edu/papers/2002peirce.pdf**

Pietarinen, Ahti-Veikko (2003) Peirce's magic lantern of logic: Moving pictures of thought,
http://www.helsinki.fi/science/commens/papers/magiclantern.pdf