Controlled Natural Languages For Representing Ontology

John F. Sowa

14 September 2011

Outline of This Tutorial

1. What is a controlled natural language (CNL)?
2. Aristotle’s CNLs for expressing logic and ontology.
3. Design patterns for expressing logic in modern logics.
4. Controlled natural languages as a bridge.
5. Some applications of controlled natural languages.
6. Processing documents in unrestricted natural language.

Note: This outline and the section summaries have a green background, and the detailed slides are in white.

1. Controlled Natural Language

A subset of a natural language that has a well-defined mapping to and from a computable form.

First CNL: Aristotle's subset of Greek for expressing logic and the patterns of *syllogisms* for reasoning about it.

CNLs support precise communication:

- **For stating requirements and specifications by humans to humans.**
- **For commands and assertions by humans to computers.**
- **For answers, explanations, and help from computers to humans.**

Advantages of controlled natural languages:

- **More readable than typical computer languages.**
- **Less training for people who write CNLs.**
- **No training for people who read CNLs.**

Logic Patterns

First-order logic is a subset or superset of most knowledge representation languages.

FOL is also a subset of all natural languages.

English expresses FOL with the following operators:

- Two quantifiers: *some* and *every*.
- Boolean operators: *and, or, not, if-then, if-and-only-if*.
- Relations represented by English words.
- Pronouns for cross references.
- English syntax for combining these operators.

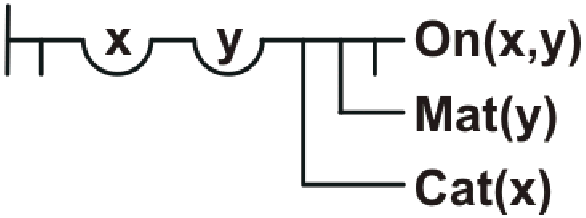
Predicate calculus uses special symbols instead of words.

Other logics use other patterns for various purposes.

A controlled natural language can express these patterns.

How to say “A cat is on a mat.”

Gottlob Frege (1879):



Charles Sanders Peirce (1885):

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

Giuseppe Peano (1895):

$$\exists x\ \exists y\ \mathrm{Cat}(x) \land \mathrm{Mat}(y) \land \mathrm{On}(x,y)$$

Frege and Peirce developed their notations independently.

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

But all three notations have identical expressive power.

Computer-Oriented Logics

Most computer logics avoid symbols not on the keyboard.

But they often have other features:

- SQL: Links to tables that store the data.
- RDF: XML notation for embedding in web pages.
- CLIF: A syntax that is easy to parse.
- Conceptual graphs: A graphic notation for logic.
- CGIF: A syntax that has a direct mapping to conceptual graphs.
- Controlled English: An English-like notation that is easy to parse.
- Prolog: Horn-clause subset of FOL with procedural extensions.

Some logics add special-purpose ontology:

- RDFS and OWL: XML plus a metalevel ontology about ontology.
- UML diagrams: Graphic notations that add ontology for software design and specification.

Some Computer-Oriented Logics

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
```

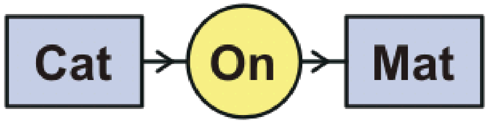
Common Logic Interchange Format (ISO 24707):

```
(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.
```

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 represented in 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' that 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 Common Logic 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 permissible in CLCE, but simpler words would make the statement more readable:

```
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.
```

Some Examples of CNLs

1. Intellect keyword system

2. Transformational Question Answering System (TQA)

3. Executable English

4. Attempto Controlled English (ACE)

5. REL, ASK, and families of sublanguages

All these systems satisfy the definition of a CNL, but #4 is the only one whose designers call a CNL.

The designers advertised #1 as “true natural language,” but a better description would be “keyword system.”

Intellect

A database query system developed in the 1970s.

- Translated English sentences to SQL queries.
- Advertised as “true natural language.”
- For simple queries, it was much easier to use than SQL.
- But for complex queries, it was highly unreliable.
- The Intellect manuals did not explain how it worked.

Examples in the following slides were tested on a sample database that was delivered with the Intellect system.

They show how the Intellect system actually worked.

Intellect Customization

Before Intellect can access a database, someone at the user’s location must customize the dictionary.

Customization maps words or phrases to the values or the columns of the database tables.

For convenience, multiple terms (synonyms) can be mapped to the same column or value.

Those words may be any character strings – they can be taken from English or any other language.

Some special words, such as *not*, are recognized.

But many words are ignored: *a, the, is, some, every, all, in, on, of...*

Warning: The words *some* and *every* are critical to logic.

Intellect Grammar

Advertising says that Intellect can handle incorrect grammar.

But that’s because it ignores grammar.

Intellect gave the same response to the following 3 questions:

1. Who are all of the exempt female managers?
2. Managers female exempt the of all are who?
3. Who the be of females all manager on exempts of of?

Note that question #2 has all the words of #1 in reverse order.

Question #3 deletes the plural marker *s* on some words and adds it to other words.

Logical Weaknesses

Some questions for Intellect and the answers it returned:

Q: How many managers are exempt?
A: 36

Q: How many managers are nonexempt?
A: 5

Q: How many managers are not exempt?
A: 5

Q: How many exempt managers are not exempt?
A: Your request contains a logical inconsistency.

Q: How many exempt managers are nonexempt?
A: 41

Note: the first four answers are correct; the last answer is 36+5.

So-called Logical Form

Intellect heuristics for translating English to logic:

- **If two terms refer to different database columns, connect them with a logical *and*.**
- **If two terms refer to the same database column, connect them with a logical *or*.**
- **Assume that *not* negates the next database term.**
- **Treat all quantifier words as equivalent to *some*.**

For the question “How many exempt managers are nonexempt?” the words *exempt* and *nonexempt* refer to the same column.

Therefore, the question was treated as equivalent to “How many managers are exempt or nonexempt?”

More Logical Weaknesses

Q: Are all managers men?
A: Yes

Q: Are all managers women?
A: Yes

Q: How many men are also women?
A: 441

According to the Intellect rules for translating English to logic:

- **“all” is equivalent to “some”;**
- **“man” is defined as “male employee” ;**
- **“woman” is defined as “female employee”;**
- **“also” is ignored.**

Therefore, the last question above is equivalent to

Q: How many employees are male or female?

Evaluation of Intellect

Definitely not "true natural language."

Too loose to be considered a "controlled natural language."

The logical rules are hopelessly unreliable for complex queries.

But the rules are adequate for existential-conjunctive logic:

- **The only implicit quantifier is *some* (existential).**
- **The only implicit Boolean operator is *and* (conjunction).**
- **There are no negations or disjunctions, explicit or implicit.**

For this logic, Intellect acts like a keyword retrieval system.

With its customizing tools, a database administrator can easily define meaningful keywords as synonyms for the DB columns.

TQA System

Transformational Question Answering System at IBM Research

- Based on a version of Chomsky’s transformational grammar.
- Translated English questions to first-order logic.
- Translated the logic to SQL for a relational database.
- Verified its operations by translating the SQL to controlled English.

Some observations:

- TQA was called an English query system, but its formal grammar could be considered a version of controlled English.
- TQA had a rich grammar that allowed users to ask their questions in a wide range of commonly used expressions.
- But customizing TQA required more knowledge of grammar and logic than just defining synonyms for the columns of a DB table.

Testing the TQA System

IBM decided to test TQA with a typical customer.

A problem faced by White Plains city officials:

> During the 1974 gasoline shortage, the land use file was searched by hand on the "use" code to find the locations of all gas stations so that police could go there to direct traffic.

At Yorktown Research, IBM installed a relational database for all the buildings and parcels of land in the city of White Plains.

Then they connected the TQA system at Yorktown to a terminal in the office of land use planning in the White Plains city hall.

The IBM researchers gave the city planners and officials a short tutorial and a demo on how to type queries and get answers.

Typical Questions Asked by Users

What is the total area of the parcels in ward 6 block 72?

What parcels in the R5 zone on Stevens St. have greater than 5000 sq. ft.?

How many parking spaces are there in ward 1 block 2?

What is the zone of the vacant parcels in subplanning area 410?

How many two family houses are there in the Oak Ridge Residents Assn. ?

Where are the apartment dwellings which have more than 50 units which are more than 6 stories high on Lake St. ?

What is the assessment of the parcels in the Battle Hill Assn. having more than 3 dwelling units?

Source: Operating statistics for the Transformational Question Answering System, by Fred J. Damerau, http://acl.ldc.upenn.edu/J/J81/J81-1003.pdf

Echo Generated by Q-Trans

Original English question, as typed by the user:

What parcels in the R5 zone on Stevens St. have greater than 5000 sq. ft.?

Translation of English to SQL by TQA:

```
SELECT UNIQUE A.JACCN, B.PARAREA
FROM ZONEF A, PARCFL B
WHERE A.JACCN = B.JACCN
AND B.STN = 'STEVENS ST'
AND B.PARAREA > 5000
AND A.ZONE = R5;
```

Translation of SQL to controlled English by Q-Trans:

Find the account numbers and parcel areas for lots that have the street name STEVENS ST, a parcel area of greater than 5000 sq. ft., and zoning code R5.

Evaluation of TQA

During the one-year trial, the users typed 788 questions.

65% of the questions were processed correctly.

The remaining 35% had various syntactic and semantic errors.

In most cases, the users rephrased the sentence to get a version that TQA could process.

In rare cases, they called the developers for help.

The users loved it. They were unhappy when the trial period ended, and IBM unplugged the terminal.

But IBM management decided that the cost of customizing and supporting TQA would be too expensive to make it profitable.

Executable English™

A system for writing and running question answering applications in a language called Open Vocabulary Executable English.

Example:

some-person does research into some-subject
that-subject is a sub topic of some-topic

--

that-person does research into that-topic

The first two lines represent predicates in the if-part of some rule.

The line at the bottom represents the conclusion.

The prefix *some-* introduces a variable, and the prefix *that-* refers back to such a variable.

Mapping Executable English to Logic

Following is a translation of the previous rule to the CLIF dialect of Common Logic:

```
(forall (person-x subject-y topic-z)
   (if (and (does_research_into person-x subject-y)
                (is_a_sub_topic_of subject-y topic-z)
       (then (does_research_into person-x topic-z)) ))
```

This example illustrates the methods for mapping Executable English to more traditional notations for logic.

The result is so readable that the author or reader can treat it as English with some special formatting conventions.

The notation is as precise as TQA, but the parsing is simple, the customization is easy, and the implementation is straightforward.

Rules and Explanations

The Executable English system can explain its answers by displaying instances of the rules that were used:

```
estimated demand 523 in NJ is for 1000 gallons of product-y in October of 2005
for estimated demand 523 0.19 of the order will be product-x from Shell Canada One
1000 * 0.19 = 190
---------------------------------------------------------------------------------------------------------
for demand 523 NJ for 1000 product-y we use 190 product-x from Shell Canada One
```

Note that mathematical expressions can be written on any line of a rule or explanation, with * for multiplication and = for equality.

The syntactic convention of writing predicate names with arbitrary English phrases or sentences makes comments unnecessary.

Mapping to Relational Databases

Tables of a relational database, either stored or computed, can be mapped to or from Executable English by writing a single line:

```
we have this-method transportation from refinery this-name to region this-region
====================================================================
         truck                                    Shell Canada One     NJ
         rail                                     Shell Canada One     NJ
```

The prefix *this*- indicates a mapping to one column of a table.

With its inference engine and mapping to SQL, Executable English can support deductive databases.

Evaluation of Executable English

With its highly flexible and readable notation, Executable English demonstrates a novel method for designing user interfaces.

The lack of special grammars, dictionaries, and ontologies makes the rule formats easy to learn, read, and write.

That same lack, which is a benefit for a single developer, may make it difficult to coordinate and integrate the contributions of a large development team over an extended period of time.

But the system makes suggestions to help authors find or remember the exact phrasing of a previous sentence.

For further information, see

http://www.reengineeringllc.com/

For more examples, see

http://www.reengineeringllc.com/Oil_Industry_Supply_Chain_by_Kowalski_and_Walker.pdf

Attempto Controlled English (ACE)

ACE is a controlled natural language that has been developed and used at the University of Zurich for over 15 years.

Like both TQA and Executable English, ACE reliably maps English sentences to a subset of first-order logic.

Like TQA, but unlike Executable English, ACE has a large built-in grammar of English.

ACE has a dictionary of about 100,000 words of English, but it allows those words to be redefined as needed.

The complete ACE system with many associated tools is available for free download and use under the LGPL license.

See http://attempto.ifi.uzh.ch/site/

Verbalizing OWL in ACE

One of the applications of ACE is a verbalizer that translates OWL 2 ontologies to statements in ACE.

See http://attempto.ifi.uzh.ch/site/docs/owl_to_ace.html for a hundred-line OWL 2 ontology that is verbalized in ACE as

> 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.

These ACE statements can then be translated back to OWL 2.

Technology for Controlled NLs

Two views about the nature of CNLs:

- Bridge: A controlled natural language is an intermediate step between a natural language and a formal notation for logic.
- Formal: A controlled natural language is a formal notation for logic.

Two ways of enforcing control:

- Syntactic: Strict constraints on the permissible grammar.
- Semantic: Strict constraints on the semantics, but an allowance for any grammatical pattern that has a unique semantic interpretation.

Advantages and disadvantages:

- Syntactic: More predictable, but harder for people to write.
- Semantic: More natural and easier to write, but requires an "echo" that shows exactly how each sentence is interpreted.

See Naturalness vs. Predictability: A Key Debate in Controlled Languages, by P. Clark, P. Harrison, W. R. Murray, & J. Thompson, http://www.cs.utexas.edu/users/pclark/papers/cnl09.pdf

2. Aristotle’s CNL

Aristotle’s design patterns:

- **Four sentence patterns: Stated in controlled natural language.**
- **Syllogisms: Logic based on those sentence patterns.**
- **Ontology: Ten categories in answer to ten basic questions.**
- **Definitions: Specifying new categories by genus and differentiae.**

Propositional logic developed in late antiquity:

- **Natural language patterns that use *and, or, not, if*.**
- **Hypothetical syllogism: Reasoning patterns based on *if-then*.**
- **Disjunctive syllogism: Reasoning patterns based on *either-or*.**

Medieval synthesis by Scholastic philosophers:

- **Mnemonics and diagrams for remembering the patterns.**
- **Extensions to modal logic and temporal logic.**
- **Methods of conceptual analysis for knowledge representation.**

Aristotle’s Logic

Aristotle represented logic in a controlled natural language.

He used a subset of Greek to represent four sentence patterns:

A	**Universal affirmative:**	**Every X is Y.**
I	**Particular affirmative:**	**Some X is Y.**
E	**Universal negative:**	**No X is Y.**
O	**Particular negative:**	**Some X is not Y.**

Boethius (6th century AD) introduced the letters A, I, E, and O as mnemonics for naming and remembering the combinations:

- **A and I are the first two vowels in the Latin *affirmo* (I affirm).**
- **E and O are the first two vowels in *nego* (I deny).**

The letters X and Y represent words or phrases that specify categories of the ontology or differentiae that distinguish them.

Aristotle’s Syllogisms

The first and most famous pattern has three type A sentences, whose letters become the vowels in the name Barbara:

A Every animal is a living thing.
A Every human is an animal.
A Therefore, every human is a living thing.

The syllogism named Darii has sentence types A, I, I:

A Every beast is irrational.
I Some animal is a beast.
I Therefore, some animal is irrational.

Syllogisms of the pattern Barbara support the inheritance of properties from a supertype (Animal) to a subtype (Human).

Syllogisms of the pattern Darii support the inheritance of properties from a type to instances of the type.

Negative Syllogisms

The first negative syllogism is named Celarent:

E No body is a spirit.
A Every animal is a body.
E Therefore, no animal is a spirit.

The negative syllogism Ferio applies to particular individuals:

E No plant is rational.
I Some living thing is a plant.
O Therefore, some living thing is not rational.

Negative syllogisms express constraints on the type hierarchy.

Syllogisms with Other Verbs

Sentences with the verb *has* and an arbitrary object:

A Every quadruped has four legs.
A Every elephant is a quadruped.
A Therefore, every elephant has four legs.

But any verb or verb phrase can be used:

A Every mammal breathes oxygen.
I Some sea creature is a mammal.
I Therefore, some sea creature breathes oxygen.

With a suitable paraphrase, all sentences can be converted to a pattern with *is* as the main verb:

- **Every quadruped is an animal with four legs.**
- **Every mammal is an oxygen-breathing animal.**

Other Paraphrases

The pattern Cesare has the same predicate in both premises:

E No oyster has lungs.
A Every mammal has lungs.
E Therefore, no mammal is an oyster.

With a paraphrase, Cesare can be converted to Celarent:

E No animal with lungs is an oyster.
A Every mammal is an animal with lungs.
E Therefore, no mammal is an oyster.

Negating some verbs requires the auxiliary verb *does*:

E No herbivore eats meat.
I Some mammal is a herbivore.
O Therefore, some mammal does not eat meat.

The CNL for Categorial Syllogisms

Each sentence affirms or denies a type-subtype relationship between two categories.

The verb *is* links terms that can be either subjects or predicates.

Terms that use other verbs can only occur as predicates.

A Every quadruped has four legs.
I Some sea creature breathes oxygen.
E No oyster has lungs.
O Some mammal does not eat meat.

There are 256 possible patterns of 3-sentence syllogisms.

But only 19 of those patterns are logically valid.

The other 237 patterns are fallacious: they could derive a false conclusion from true premises.

The 19 Valid Patterns

Aristotle chose Barbara and Celarent as the two basic patterns.

He defined *conversion rules* for deriving all other valid patterns.

Barbara, Celarent, Darii, and Ferio belong to the *first schema*.

A total of 19 valid patterns in four schemata or *figures*.

The Scholastics used mnemonic poems to remember them:

> *Barbara, Celarent, Darii, Ferioque prioris*
> *Cesare, Camestres, Festino, Baroco secundae*
> *Tertia grande sonans recitat Darapti, Felapton*
> *Disamis, Datisi, Bocardo, Ferison. Quartae*
> *Sunt Bamalip, Calames, Dimatis, Fesapo, Fresison.*

The consonants in the names indicate the conversion rules.

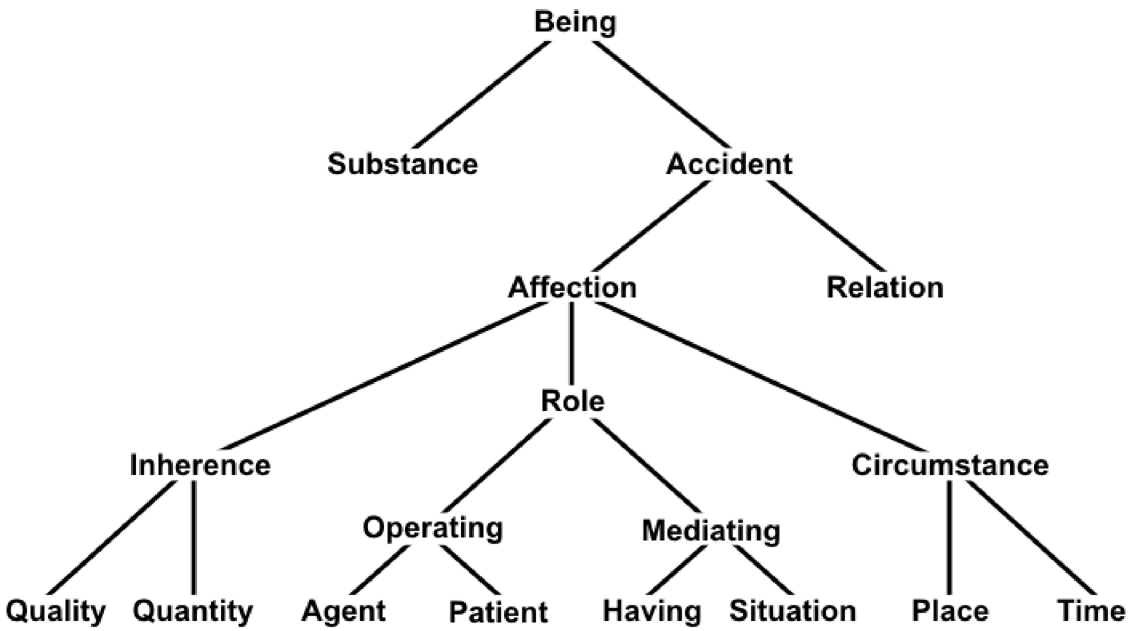
Aristotle’s Ontology

Ten categories based on the semantics of Greek.

For each category, there is a corresponding question:

- Substance – What is it?
- Relation – Toward what?
- Quantity – How much?
- Quality – What kind?
- Agent – Doing what?
- Patient – Undergoing what?
- Having – Having what?
- Situation – Situated how?
- Place – Where?
- Time – When?

Tree of Aristotle’s Categories



Aristotle’s categories, as arranged by Franz Brentano (1862).

The ten categories are at the leaves of the tree.

Brentano chose labels for the branching nodes from other words and phrases that Aristotle used.

Aristotle’s Method of Definition

Define categories by *genus* and *differentiae*:

- Genus – a supertype of the category to be defined.
- Species – the subtype that is being defined.
- Differentia – a property or attribute that distinguishes the species to be defined from other species of the same genus.

This method is still widely used for dictionaries today.

Ideally, the differentiae should state necessary and sufficient conditions that uniquely distinguish the species.

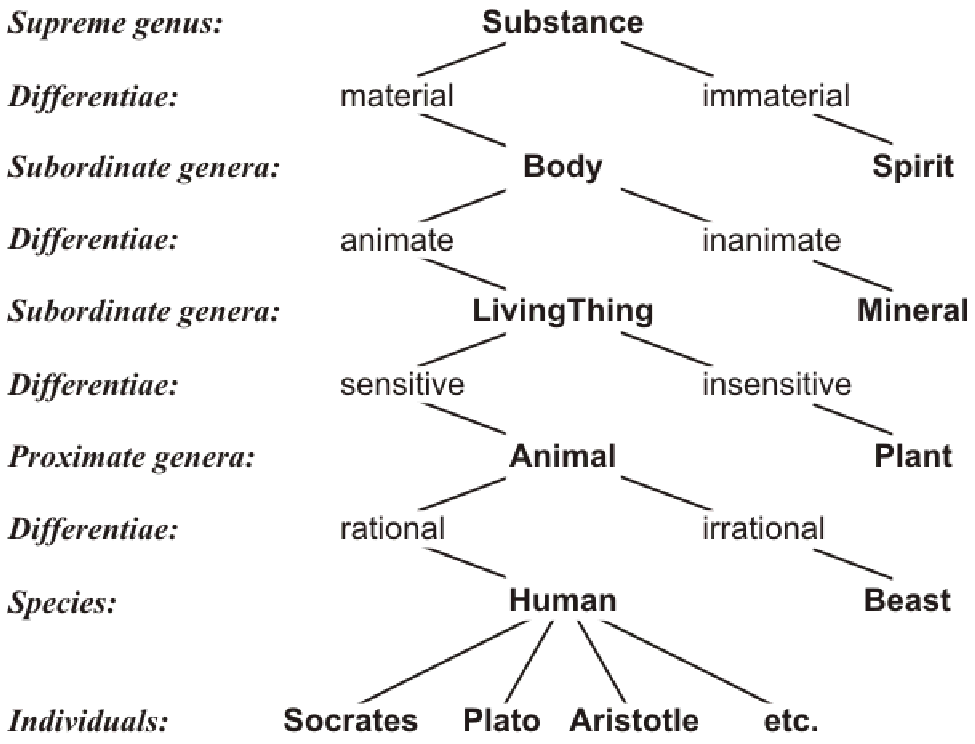
That goal is achieved in mathematics.

But it is rarely possible for artifacts or natural kinds.

In his writings on biology, Aristotle suggested an alternative:

- For each species, describe a type specimen in detail.
- Use looser criteria of similarity to distinguish various species.

Tree of Porphyry



Supreme genus: Substance

Differentiae: material immaterial

Subordinate genera: Body Spirit

Differentiae: animate inanimate

Subordinate genera: LivingThing Mineral

Differentiae: sensitive insensitive

Proximate genera: Animal Plant

Differentiae: rational irrational

Species: Human Beast

Individuals: Socrates Plato Aristotle etc.

A tree of categories and differentiae was found in a commentary on Aristotle by the philosopher Porphyry (3rd century AD).

The above tree is based on a version by Peter of Spain (1239).

Propositional Logic

Aristotle made a few comments about propositional logic.

He even used variables to represent propositions:

- **“If the conclusion is false, the premises of the argument must be false, either all or some of them.”**
- **“If when A is, B must be, then when B is not, A cannot be.”**

The Stoic philosophers made important contributions.

In the 6th century AD, Boethius summarized propositional logic:

- **When a formula requires two propositions, he used the phrase *hoc est* (this is) for the first and *illud est* (that is) for the second.**
- **But when he needed more propositions, he referred to them with variables: A est, B est, C est, D est.**
- **Example: “Sicut cum A est, B est, C est.”**
  If (when A is, B is), C is.
  $(A \supset B) \supset C$.

Hypothetical Syllogisms

A hypothetical syllogism uses *if-then* rules with arbitrary sentence patterns and two rules of inference.

The rule *modus ponens* (mode of putting) makes an assertion:

Given, If A, then B.
But, A.
Therefore, B.

The rule *modus tollens* (mode of taking away) denies the premise:

Given, If A, then B.
But, not B.
Therefore, not A.

In modern rule-based systems, repeated modus ponens is called *forward chaining.*

Repeated modus tollens is called *backward chaining*.

Disjunctive Syllogisms

A disjunctive syllogism is based on *either-or* patterns.

The basic disjunctive syllogism allows arbitrary sentences:

Given, Either A or B.
But, not A.
Therefore, B.

A disjunctive syllogism can also be applied to the predicate of a sentence, as in the following extended argument:

Given, Every philosopher is either sleeping or debating.
But, Socrates is a philosopher.
Therefore, Socrates is either sleeping or debating.
But, Socrates is not sleeping.
Therefore, Socrates is debating.

Medieval Synthesis

The Scholastic logicians organized the contributions from Greek, Latin, and Arabic philosophers:

- Aristotle’s syllogisms, categories, and definitions.
- Propositional logic.
- Diagrams for showing relationships.
- Controlled Latin for the notation.

They also made many innovations and extensions:

- Ontology of suppositions for reasoning about fictional, hypothetical, and mental entities.
- Modal and temporal logics.
- Model-theoretic semantics for Latin by William of Ockham.
- Methodology of conceptual analysis.

Observations

Traditional logic was a brilliant achievement.

A superset of RDFS and the most widely used parts of OWL.

And the notation is much more readable.

But traditional logic is not sufficiently expressive:

- The controlled NL for syllogisms is a small subset of full NL.
- Euclid tried to follow Aristotle for his textbook on mathematics.
- But math requires more than syllogisms and propositional logic.
- Computer science also requires much richer logics.

The Aristotelians discovered many useful patterns.

But their conversion rules were too complex.

With simpler rules, they wouldn’t need the mnemonic poems.

3. Design Patterns in Modern Logics

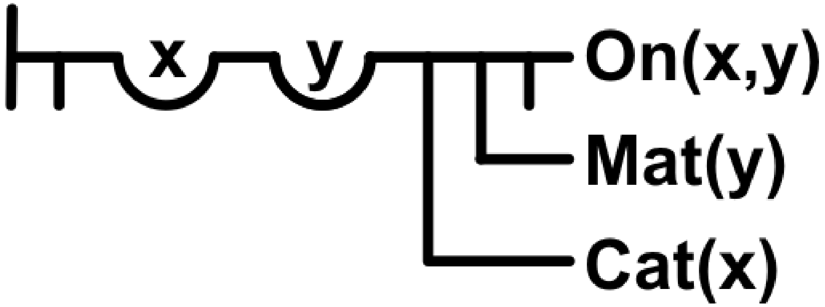
Several kinds of notations:

- Algebraic: Boolean algebra and predicate calculus.
- Special purpose diagrams: Type hierarchies, UML diagrams
- General purpose diagrams: Existential graphs.
- Controlled natural languages
- Miscellaneous: Mixed notations for a variety of purposes.

Design choices:

- Optimize readability.
- Optimize expressive power.
- Tailor the notation for the reasoning method.
- Tailor the notation for special kinds of applications.
- Mix some special-purpose ontology with the logic.

Choice of Design Patterns



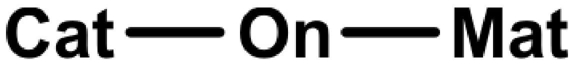
Frege sacrificed readability in order to tie his Begriffsschrift to the axioms and rules of inference. But nobody else liked his notation.

Peirce extended Boolean algebra with quantifiers and relations. Peano followed Peirce, but introduced new symbols:

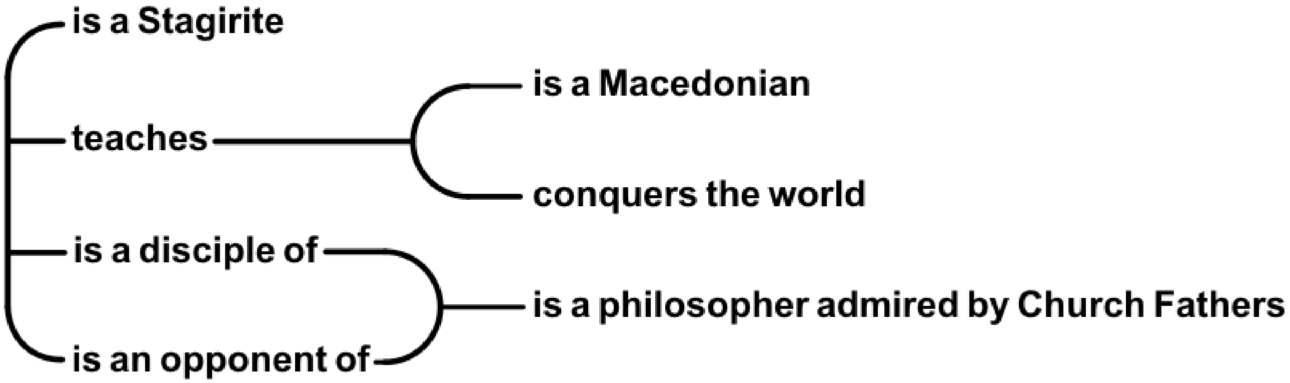
$$\Sigma_x\ \Sigma_y\ \mathrm{Cat}_x \bullet \mathrm{Mat}_y \bullet \mathrm{On}_{x,y}$$

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

But in 1897, Peirce discovered existential graphs as a simple, readable notation with extremely simple rules of inference:


Cat — On — Mat

Existential Graphs Without Negation



A simple version of logic that can represent the content of many graphical notations, including RDF, Concept Maps, and Topic Maps.

The above example by Peirce represents the following formula:

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

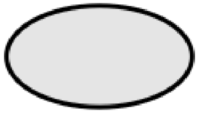
EGs without negation can represent the content stored in a database, but more logical operators are needed to represent DB queries.

Existential Graphs for Full FOL

A graph notation for logic with a minimum of primitives:

Existence: —

Negation:

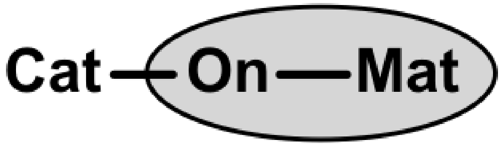


Relations: Cat- -On- -Under- -With- -Mat

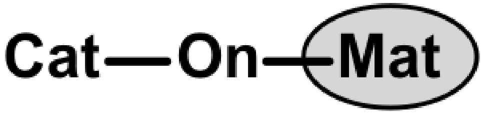
A cat is on a mat: Cat—On—Mat

Something is under a mat: —Under—Mat

Some cat is not on a mat:

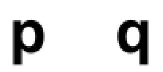


Some cat is on something that is not a mat:

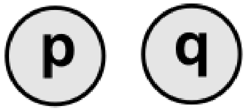


Boolean Combinations

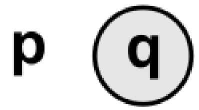
Areas nested inside an odd number of negations are shaded.



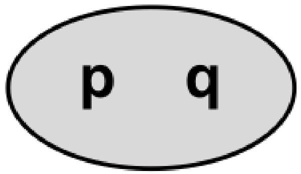
p and q



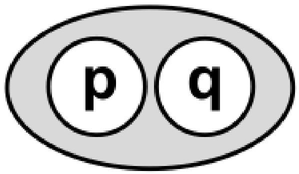
not p and not q



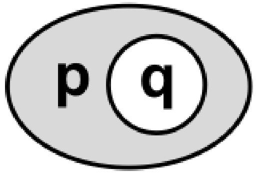
p and not q



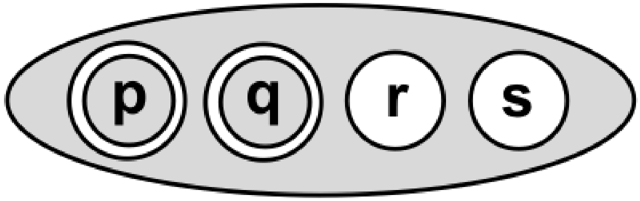
not (p and q)



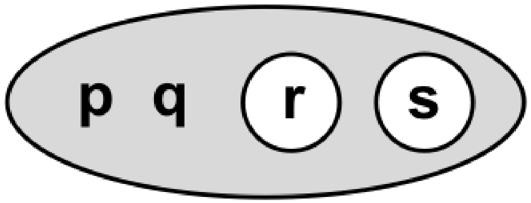
p or q



if p, then q



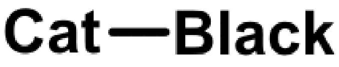
not p or not q or r or s



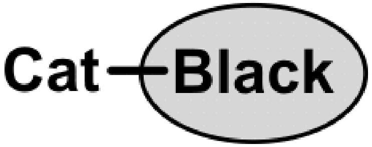
if p and q, then r or s

Representing Quantifiers

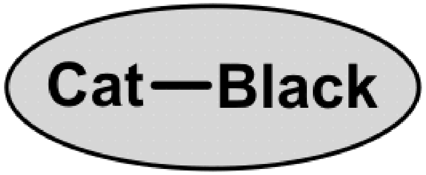
The outermost point of a line defines the scope of quantification.



Some cat is black.



Some cat is not black.



No cat is black.

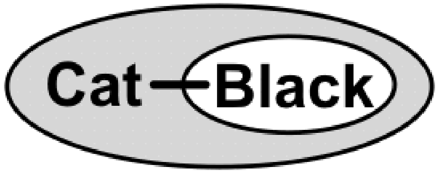
With more negations, the number of readings increases:

It is false that some cat is not black.

If there is a cat, then it is black.

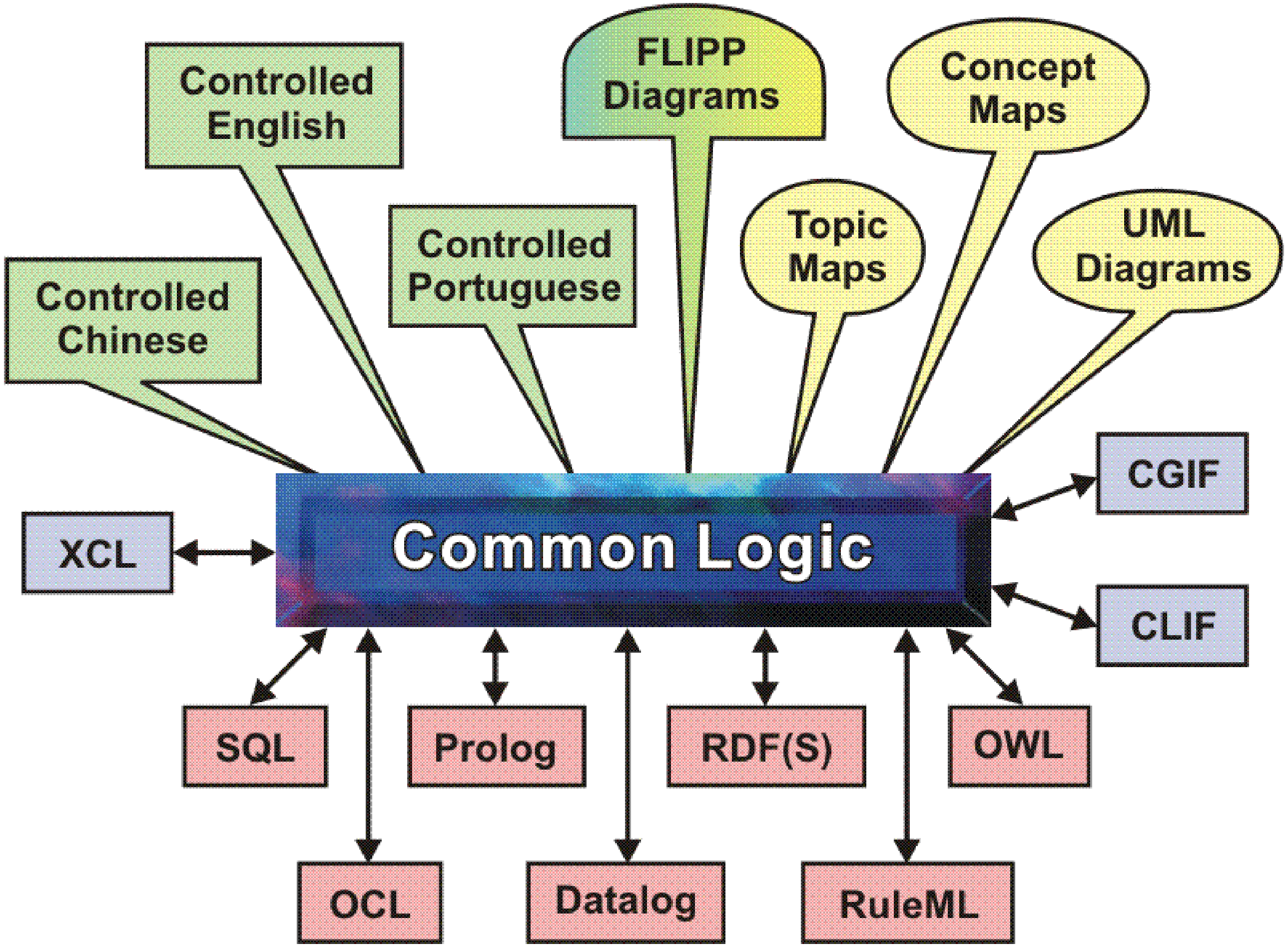
Every cat is black.

No cat is not black.



These four sentences are synonymous (logically equivalent).

Human Interfaces



Controlled Chinese

Controlled English

Controlled Portuguese

FLIPP Diagrams

Topic Maps

Concept Maps

UML Diagrams

Common Logic

XCL

CGIF

CLIF

SQL

Prolog

RDF(S)

OWL

OCL

Datalog

RuleML

Machine Interfaces

Common Logic

A highly expressive version of logic that was designed as a superset of many widely used notations.

Standardized by ISO/IEC 24707 as a formal semantics with an abstract syntax that can be mapped to many different notations.

Three concrete dialects specified in the standard:

- CLIF: Common Logic Interchange Format.
- CGIF: Conceptual Graph Interchange Format.
- XCL: An XML-based notation (but it requires some revision).

CLIF and CGIF can express the full CL semantics.

Any notation that has a formal mapping to the CL semantics can be called a CL dialect.

Translating EGs to Common Logic

Existential graphs can be translated to or from many different notations for logic.

The simplest mapping is to a subset of CGIF, which is designed to support graph notations:

- Each feature of an EG maps to exactly one feature of CGIF.
- Like EGs, CGIF has no implicit ordering of nodes.
- Like EGs, the conjunction 'and' is implicit.

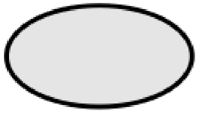
But as a linear notation, CGIF uses labels to represent the cross links of a graph.

Those labels have a direct mapping to variables in other notations, such as CLIF.

Conceptual Graph Interchange Format

A standard dialect of Common Logic:

Existence: — [*x]

Negation:  ~[ ]

Relations: (Cat ?x) (On ?x ?y) (Under ?x ?y) (Mat ?y)

A cat is on a mat: [*x] [*y] (Cat ?x) (On ?x ?y) (Mat ?y)

Something is under a mat: [*x] [*y] (Under ?x ?y) (Mat ?y)

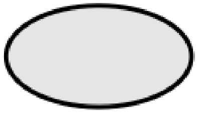
Some cat is not on a mat: [*x] (Cat ?x) ~[[*y] (On ?x ?y) (Mat ?y)]

Some cat is on something that is not a mat:
[*x] [*y] (Cat ?x) (On ?x ?y) ~[(Mat ?y)]

Common Logic Interchange Format

Another standard dialect of Common Logic:

Existence: — (exists (x))

Negation:  (not)

Relations: (Cat x) (On x y) (Under x y) (Mat y)

***A cat is on a mat*:** (exists (x y) (and (Cat x) (On x y) (Mat y)))

***Something is under a mat*:** (exists (x y) (and (Under x y) (Mat y)))

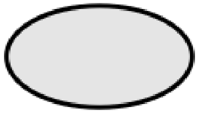
***Some cat is not on a mat*:**
(exists (x) (and (Cat x) (not (exists (y) (and (On x y) (Mat y))))))

***Some cat is on something that is not a mat*:**
(exists (x y) (and (Cat x) (On x y) (not (Mat y))))

Predicate Calculus

The widely used Peirce-Peano algebraic notation:

Existence: — $(\exists x)$

Negation:  $\sim(\quad)$

Relations: Cat(x) On(x,y) Under(x,y) Mat(y)

***A cat is on a mat*:** $(\exists x)(\exists y)(\mathrm{Cat}(x) \wedge \mathrm{On}(x,y) \wedge \mathrm{Mat}(y))$

***Something is under a mat*:** $(\exists x)(\exists y)(\mathrm{Under}(x,y) \wedge \mathrm{Mat}(y))$

***Some cat is not on a mat*:** $(\exists x)(\mathrm{Cat}(x) \wedge \sim(\exists y)(\mathrm{On}(x,y) \wedge \mathrm{Mat}(y)))$

***Some cat is on something that is not a mat*:**

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

Additional Operators

Existential graphs represent full first-order logic with just 3 operators.

Those operators are also sufficient for other notations.

But most notations add more symbols for more operators.

In predicate calculus, the universal quantifier (∀x), which may be read 'for every x' or 'for all x', can be defined by an equivalence:

$$(\forall x)P(x) \quad \text{is defined as} \quad \sim(\exists x)\sim P(x)$$

In this definition, P(x) can be a single predicate, such as P, or any expression of any complexity that contains a free variable x.

CLIF uses the symbol 'forall' for the universal quantifier, and CGIF uses the symbol '@every'.

Predicate calculus, CLIF, and CGIF also introduce symbols to represent 'or', 'if', and 'if and only if'.

These symbols are defined by Boolean combinations as in slide 53.

Typed and Untyped Statements

Type constraints in predicate calculus, CLIF, and CGIF can be expressed by a monadic relation that restricts the quantifier.

Predicate calculus:

Untyped: $(\exists x)(\exists y)(\mathrm{Cat}(x) \wedge \mathrm{Mat}(y) \wedge \mathrm{On}(x,y))$

Typed: $(\exists x{:}\mathrm{Cat})(\exists y{:}\mathrm{Mat})\mathrm{On}(x,y)$

Common Logic Interchange Format (CLIF):

Untyped: (exists (x y) (and (Cat x) (Mat y) (On x y)))

Typed: (exists ((x Cat) (y Mat)) (On x y))

Conceptual Graph Interchange Format (CGIF):

Untyped: [*x] [*y] (Cat ?x) (Mat ?y) (On ?x ?y)

Typed: [Cat *x] [Mat *x] (On ?x ?y)

All six of these examples are semantically equivalent: they are true if and only if a cat is on a mat.

CL uses weak typing: type mismatches cause the expression to be false, but not ungrammatical.

Common Logic Controlled English

CLCE is a formally defined language that uses English syntax.

Every CLCE sentence can be read as if it is ordinary English.

But CLCE can be translated automatically to and from CGIF or CLIF.

In fact, every “English” sentence in the preceding slides that was translated to or from an existential graph was written in CLCE.

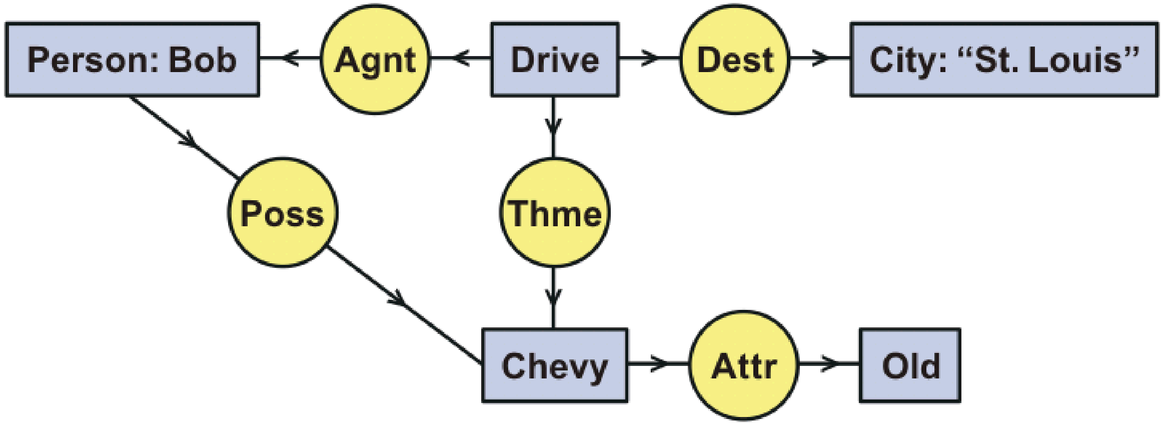
To avoid ambiguous pronouns, CLCE supports temporary names, which look like variables in predicate calculus:

> *If a red thing x is on something y,*
> *then either x is a ball or y is a table that is not blue.*

In these slides, all sentences *written in italics* **are CLCE examples.**

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

Conceptual graph display form:



Conceptual Graph Interchange Format (CGIF):

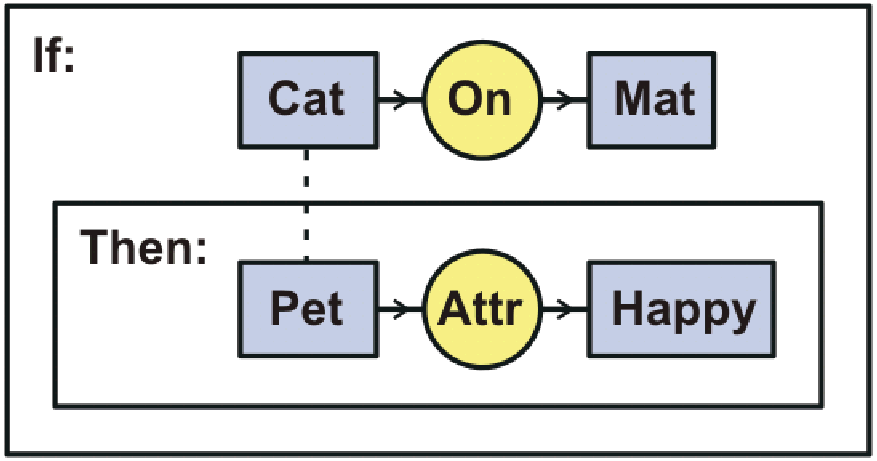
```
[Drive *x] [Person Bob] [City "St. Louis"] [Chevy *y] [Old *z]
(Agnt ?x Bob) (Dest ?x "St. Louis")
(Thme ?x ?y) (Poss Bob ?y) (Attr ?y ?z)
```

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 Cat) (y Mat)) (and (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:

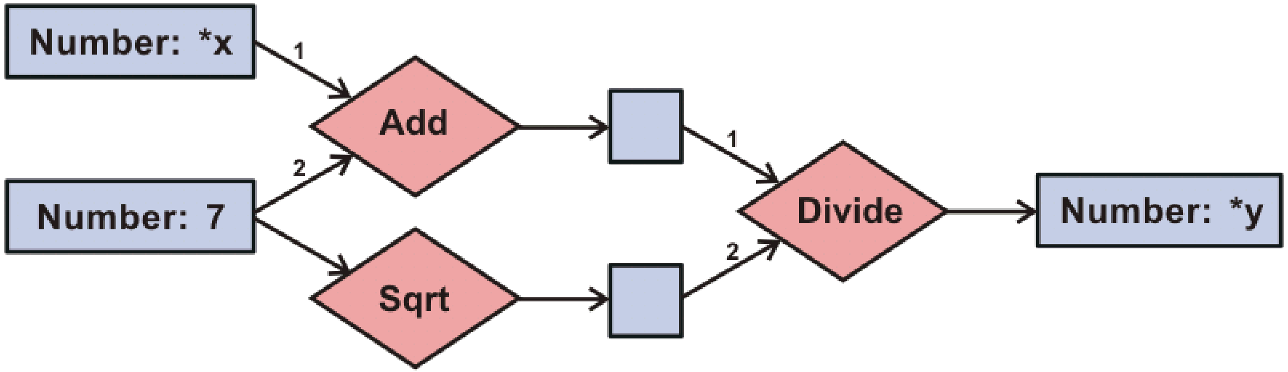
```
(forall ((x Cat) (y Mat))
    (if (On x y) (and (Pet x) (exists ((z Happy)) (Attr x z)))))
```

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.

CLCE: *For a number x, a number y is* ((*x*+7) / *sqrt*(7)).

Conceptual graph display form:



CGIF:

```
[Number: *x]  [Number: *y]
(Add ?x 7 | *u)  (Sqrt 7 | *v)  (Divide ?u ?v | ?y)
```

CLIF:

```
(exists ((x Number) (y Number))
       (= y (Divide (Add x 7) (Sqrt 7))))
```

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)))
```

Defining New Words in CLCE

The word “relation” is not a reserved word in CLCE.

But CLCE allows new words to be defined by their mapping to CGIF, CLIF, or other languages, such as SQL:

Define "familial relation r" as (and (Familial r) (Relation r)).

Define "relation r between x and y" as (and (Relation r) (r x y)).

With these definitions, the following CLCE sentence can be translated to the CLIF and CGIF sentences in the previous slide:

> ***There is a familial relation between Bob and Sue.***

Wish List of Extensions to CL

Metalanguage.

- Names for propositions and statements about propositions.
- Statements that relate propositions to other propositions.

Nonmonotonic reasoning.

- Default logics, negation as failure (e.g., SQL and Prolog).
- Belief or theory revision methods for classical FOL theories.

Uncertainty, vagueness, and fuzziness.

Modality.

- Alethic (necessity and possibility).
- Epistemic (knowledge and belief).
- Deontic (obligation and permission).

Contexts and microtheories.

Can all these features be represented by an extension to CL?

IKRIS Project

Goal: Design an Interoperable Knowledge Language (IKL) as an extension to Common Logic.

Goals:

- **Enable interoperability among advanced reasoning systems.**
- **Test that capability on highly expressive AI languages.**

Show that semantics is preserved in round-trip mapping tests:

- **Cycorp: Cyc Language → IKL → CycL**
- **RPI / Booz-Allen: Multi-Sorted Logic → IKL → MSL**
- **Stanford/IBM/Battelle: KIF → IKL → KIF**
- **KIF → IKL → CycL → IKL → MSL → IKL → KIF**

Conclusion: "IKRIS protocols and translation technologies function as planned for the sample problems addressed."

***Interoperable Knowledge Representation for Intelligence Support (IKRIS)*, Evaluation Working Group Report, prepared by David A. Thurman, Alan R. Chappell, and Chris Welty, Mitre Public Release Case #07-1111, 2007.**
http://nrrc.mitre.org/NRRC/Docs_Data/ikris/IKRIS_Evaluation_Report_31Dec06.doc

The IKL Extension to Common Logic

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

Only one new operator is needed: a metalanguage enclosure, which uses the keyword 'that' to mark the enclosed statement.

- **The enclosed statement denotes a proposition.**
- **That proposition could 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 extension, see http://www.ihmc.us/users/phayes/IKL/SPEC/SPEC.html
and http://www.ihmc.us/users/phayes/ikl/guide/guide.html

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 for IKL:

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

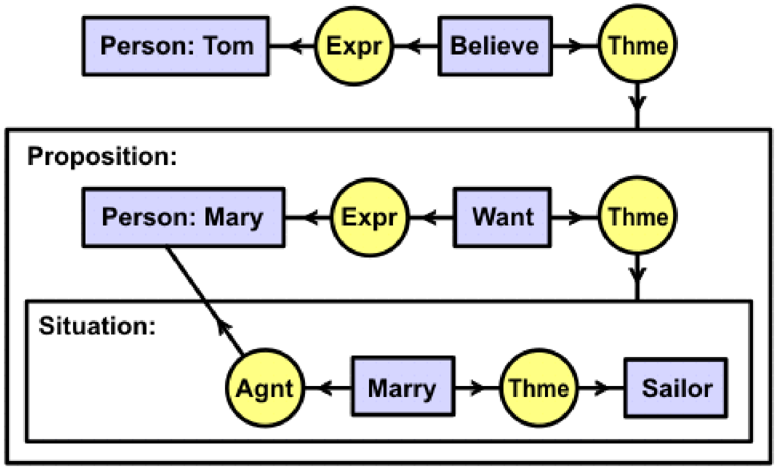
The operator 'that' is a powerful metalevel extension.

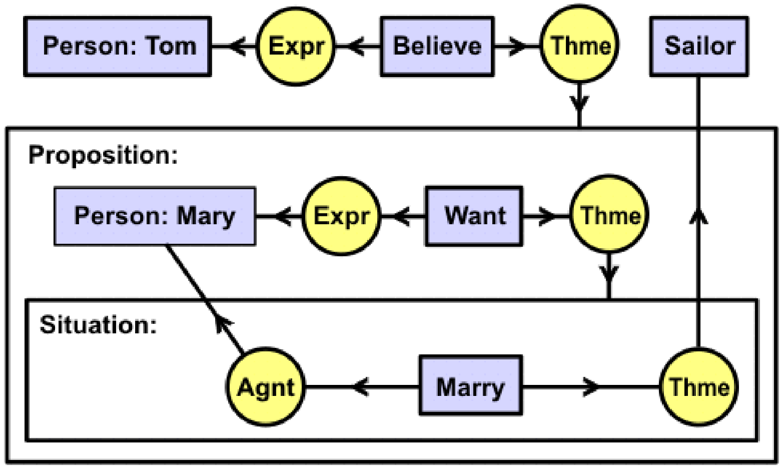
It enables IKL to specify languages, define their semantics, and specify transformations from one language to another.

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 can appreciate.

Situations and Propositions





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 is a sailor, and Tom believes a proposition that Mary wants a situation in which she marries that sailor.***

Sentences about propositions and situations require metalevel language about language.

They can be expressed by the IKL extensions to CL.

Mapping IKL to CGIF and CLIF

Following is the CGIF notation for the CG on the left side of the previous slide:

```
[Person: Tom] [Believe: *x1] (Expr ?x1 Tom) (Thme ?x1 [Proposition:
   [Person: Mary] [Want: *x2] (Expr ?x2 Mary) (Thme ?x2 [Situation:
      [Marry: *x3] [Sailor: *x4] (Agnt ?x3 Mary) (Thme ?x3 ?x4)])])
```

To represent the CG on the right of the previous slide, move the concept node [Sailor: *x4] in front of the concept [Believe: *x1].

In CLIF notation, the operator *that* applied to a CL or IKL sentence denotes the proposition stated by the sentence:

```
(exists ((x1 Believe)) (and (Person Tom) (Expr x1 Tom) (Thme x1 (that
   (exists ((x2 Want) (s Situation)) (and (Person Mary) (Expr x2 Mary)
      (Thme x2 s) (Dscr s (that
         (exists ((x3 Marry) (x4 Sailor)) (and (Agnt x3 Mary) (Thme x3 x4)
))))))))))
```

To represent the CG on the right, begin the CLIF statement with

```
(exists ((s Situation) (x1 Believe)) ...
```

Tradeoffs in the Choice of Patterns

Controlled natural languages:

- Readable without special training by anyone who knows the NL.
- But they require some training to write correctly.
- They can be extended to highly expressive logics.
- But the NL syntax can sometimes obscure the logical operators.

Artificial notations for logic:

- Usually designed for some specific purpose.
- They can often be very easy to use for that purpose.
- But they can be difficult or impossible to use for other purposes.

Peirce's existential graphs:

- A minimalist version of logic that is surprisingly easy to read.
- They show the implicit logical operators very clearly.
- Good foundation for defining and explaining all other operators.

4. CNLs as a Bridge

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

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 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.

Translating English to Controlled English

Requires translators with some training and good tools:

- **A basic knowledge of the subject and its terminology.**
- **Some training in logic and the ability to write clear English.**
- **The tools can provide dictionaries for the subject terminology.**
- **They can help the translator stay within the controlled dialect.**

Not necessary for the translator from English to controlled English to be an expert in the subject matter.

But it’s essential for a subject matter expert to read and verify the translation to controlled English.

With typical computer languages, the translator is rarely, if ever, a SME and the verification step is impossible.

An Example of Medical English

Find the percentage of patients with AMI2 receiving persistent beta blockers (for 135 of 180 days following discharge).

Numerator: Of patients in denominator, those prescribed a beta blocker following date of discharge with supply for at least 135 of next 180 days

Denominator: Age >= 35 years. All AMI cases except those transferred to another facility during the hospitalization.

Exclude patients with a history of Asthma, COPD3, Hypotension, Bradycardia (heart block > 1st degree or sinus bradycardia) or prescription of inhaled corticosteroids.

CLCE Definitions

Define "the age of x" as (– (Year CurrentDate) (Year (DoB x))).

Define "x is at least y" as (ge x y).

Define "x is transferred" as (Transferred x).

Define "AMI case" as "patient who has AMI".

Define "x is prescribed y on z for w days" as (Prescribed x y z w).

Define "x is a y" as (Type x y).

Define "x is discharged on y" as (DateOfDischarge x y).

Define "x is after y" as (gt x y).

Define "x has a history of y" as (HistoryOf x y).

Define "Bradycardia x" as
(or (and (HeartBlock x) (gt (Degree x) 1)) (SinusBradycardia x)).

Define "x is excluded" as (Excluded x).

Translation from English to CLCE

The denominator D is the set of every AMI case x
where the age of x is at least 35, and x is not transferred;

The numerator N is the set of every patient x in the denominator D
where x is prescribed a drug y on date z for w days, and y is
a beta blocker, and x is discharged on z2, and z is after z2,
and w is at least 135;

If a patient x has a history of asthma, or x has a history of COPD3,
or x has a history of hypotension, or x has a history of bradycardia,
or (x is prescribed a drug y, and y is inhaled, and y is a
corticosteroid), then x is excluded;

The ratio R is (the count of every patient in the numerator N
who is not excluded) divided by (the count of every patient
in the denominator D who is not excluded).

Translation from CLCE to CLIF

```
(Set D)  (Set N)  (Set N1)  (Set D1)  (Number R)

(forall ((x Patient))
        (if (and (Has x AMI) (not (Transferred x)
                 (ge (– (Year CurrentDate) (Year (DoB x))) 35) ))))
           (In x D) )))

(forall ((x Patient))
        (if (and (In x D) (exists ((y Drug) (z Date) (w Number))
                            (and (Prescribed x y z w) (Type y BetaBlocker)
                                 (DateOfDischarge x z2) (gt z z2) (ge w 135) )))
           (In x N) ))

(forall ((x Patient))
        (if (or (HistoryOf asthma x) (HistoryOf COPD3 x)
                (HistoryOf hypotension x) (HistoryOf Bradycardia x)
                (exists ((y Drug) (z Date) (w Number))
                        (and (Prescribed x y z w) (Inhaled y) (Type y Corticosteroid)) )))
           (Excluded x) ))

(and (forall ((x Patient))
             (if (and (In x N) (not (Excluded x))) (In x N1) ))
     (forall ((x Patient))
             (if (and (In x D) (not (Excluded x))) (In x D1) )))
     (= R (/ (Count N1) (Count D1)) )
```

Translation from CLCE to CGIF

```
[Set D]  [Set N]  [Set N1]  [Set D1]  [Number R]

[If  [Patient *x]  (has ?x AMI)  ~[ (transferred ?x) ]
     (year CurrentDate | *y1)  (DoB x | *y2)  (– ?y1 ?y2 | *a)  (ge ?a 35)
     [Then  (In ?x D) ] ]

[If  [Patient *x]  (in ?x D)  [Drug *y]  [Date *z]  [Number *w]  (Prescribed ?x ?y ?z ?w)
     [BetaBlocker ?y]  (DateOfDischarge ?x [Date *z2])  (gt ?z ?z2)  (ge ?w 135)
     [Then  (in ?x N) ] ]

[If  [Patient *x]
     [Either  [Or  (HistoryOf asthma ?x) ]  [Or  (HistoryOf COPD3 ?x) ]
              [Or  (HistoryOf hypotension ?x) ]  [Or  (HistoryOf Bradycardia ?x) ]
              [Or  [Drug *y]  [Date *z]  [Number *w]  (Prescribed ?x ?y ?z ?w)
                   (Inhaled ?y)  (Corticosteroid ?y) ] ]
     [Then  (excluded x) ] ]

[If  [Patient *x]  (In ?x N)  ~[ (Excluded ?x) ]
     [Then  (in ?x N1) ] ]
[If  [Patient *x]  (In ?x D)  ~[ (Excluded ?x) ]
     [Then  (in ?x D1) ] ]
[ (Count N1 | *x)  (Count D1 | *y)  (/ ?x ?y | R) ]
```

Choice of Representation

Two critical design points:

1. Translator interface: Mapping English to controlled English.
2. IT interface: Choosing the machine-oriented representations.

Options at the first interface are determined by subject matter experts and their colleagues.

Options at the second interface are determined by the IT professionals and the hardware-software developers.

Ease of use and accurate translation requires professionals at both levels to collaborate in the design choices.

Computational Complexity

CLCE imposes no restriction on the expressive power of the logic.

But most practical applications (including this medical example) can be translated to an efficiently computable subset of logic.

Many tools and techniques can automatically

1. Check the subset of logic required for the CLCE statements,
2. Determine the computability for various kinds of problems,
3. Convert the statements to a suitable form for the application.

Computational complexity depends on the problem, not on the language in which it is expressed.

Subject matter experts should not discard or distort their familiar forms of expression because of concerns about computability.

See "Fads and Fallacies about Logic," http://www.jfsowa.com/pubs/fflogic.pdf

5. Applications of CNLs

Good technology can lead to major breakthroughs.

But even the best technology requires good methodologies and tools to enable people to use it effectively.

This section shows some useful examples, but more research is needed to simplify the development tools.

To be commercially successful, the technology must be integrated with mainstream software development tools.

Solving Problems Stated in English

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

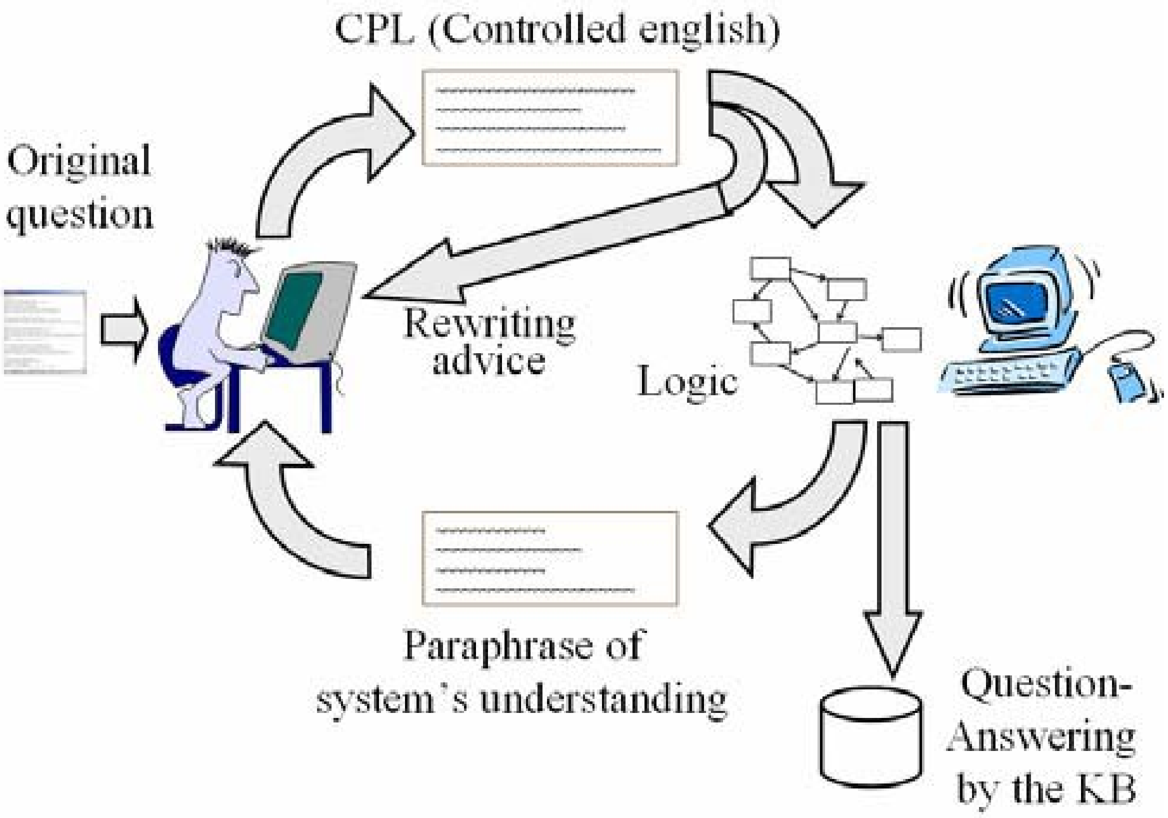
A question from the Advanced Placement Exam in physics:

> 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?

Helping a Translator Map NL to CNL


CPL (Controlled english)
Original question
Rewriting advice
Logic
Paraphrase of system's understanding
Question-Answering by the KB

From a paper by members of Project Halo: 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

Mean Tries to Completion (MTC)

Users often fail to stay within the limits of a controlled NL.

The MTC rate measures the mean number of tries to rephrase a sentence before the user succeeds or gives up.

Clark et al. found that for questions in physics taken from the Advanced Placement Exams, the MTC was 6.3.

For AP chemistry questions, the MTC was 6.6.

But for AP biology questions, the MTC was 1.5.

The reason why the biology performance was so much better is that many of the questions could be approximated by simple queries of the form “What is an X?” or “What is the Y of X?”

Given the nature of the problems, these rates were not bad.

Business Rules in Retail

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

Rules in controlled English, automatically generated from statements in controlled English typed by Tesco employees:

- 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.

Applying RDR in Medicine

A medical application was the first to use the RDR methodology.

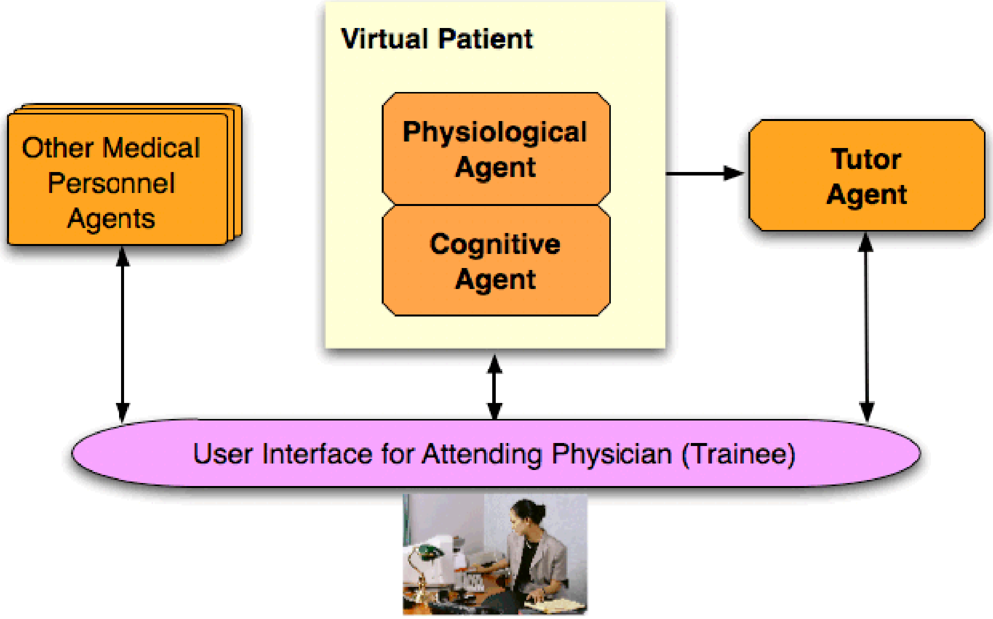
Compton et al.* described a larger pathology application, called Labwizard, which was in routine commercial use:

- During a 29-month period, 16,000 rules were added.
- A total of over 6 million cases were interpreted by RDR rules.
- The total time for several different pathologists to add rules was 353 person hours – an average of 77 seconds per rule.
- The pathologists who used Labwizard and added new rules to it were very satisfied with its performance and functionality.

For more detail, see

* P. Compton, L. Peters, G. Edwards, T. G. Lavers, Experience with Ripple-Down Rules, http://www.cse.unsw.edu.au/%7Ecompton/publications/2005_SGAI.pdf

Maryland Virtual Patient


Virtual Patient
Physiological Agent
Cognitive Agent
Other Medical Personnel Agents
Tutor Agent
User Interface for Attending Physician (Trainee)

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.

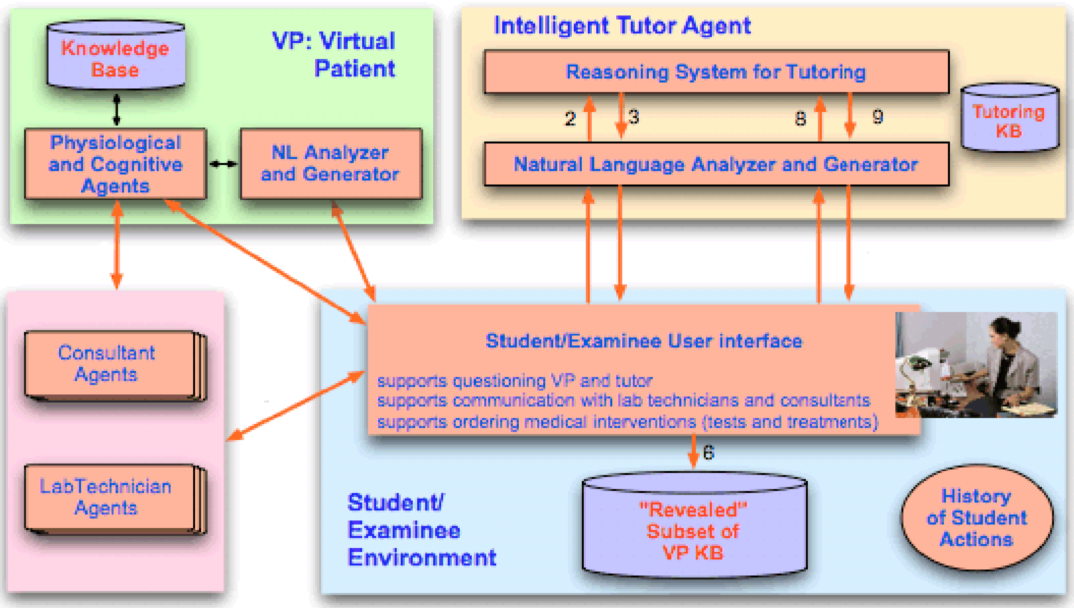
Source: Maryland virtual patient: a knowledge-based, language-enabled simulation and training system, by M. McShane, S. Nirenburg, B. Jarrell, S. Beale, & G. Fantry, http://bams.cm-uj.krakow.pl/bams3_pdf/bams%209.pdf

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


Knowledge Base
VP: Virtual Patient
Physiological and Cognitive Agents
NL Analyzer and Generator
Intelligent Tutor Agent
Reasoning System for Tutoring
2
3
8
9
Tutoring KB
Natural Language Analyzer and Generator
Consultant Agents
LabTechnician Agents
Student/Examinee User interface
6
Student/ Examinee Environment
"Revealed" Subset of VP KB
History of Student Actions

supports questioning VP and tutor
supports communication with lab technicians and consultants
supports ordering medical interventions (tests and treatments)

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 promising combination of technologies.

6. Processing Unrestricted NLs

Natural languages are the normal means for people to express their meaning, but they are not the most computable.

Full natural language understanding involves unsolved research problems, but there are many useful ways of processing NLs short of total understanding.

Finding relevant background knowledge is critical to resolving ambiguities and determining the correct interpretation.

A high-speed analogy engine is necessary to make such methods efficient and practical.

Some Observations

History of natural language processing:

- 1949: Warren Weaver wrote a highly influential memo to suggest that computers could be used to translate natural languages.
- 1950s: Early research on machine translation (MT).
- 1963: Research terminated on the GAT translator, which was converted to a commercial system for MT called SYSTRAN.
- 1970s: Logic-based natural language query systems, such as TQA.
- 1980s: New logic-based semantic systems: Montague grammar, Discourse Representation Theory, Situation Semantics, conceptual graphs...
- 1990s: Statistical processing of language becomes popular.
- 2000s: SYSTRAN is still one of the most widely used MT systems.

Pure logic-based NLP systems are useful for controlled NLs, but they have been fragile and inflexible for unrestricted NLs.

Systems that combine formal and informal methods have been more robust and practical.

Four Views of Analogy

1. **By logicians:**

   Deduction is reasoning from “first principles.”

2. **By psychologists:**

   Analogy is fundamental to human and animal cognition.

   All aspects of language understanding depend on analogy.

3. **Theoretical:**

   All methods of formal logic — deduction, induction, and abduction — are disciplined special cases of analogy.

4. **Computational:**

   A powerful and flexible technique with important applications in reasoning, learning, and language processing.

   But practicality depends on finding analogies efficiently.

Computational Complexity

Research by Falkenhainer, Forbus, & Gentner:

Pioneers in finding analogies with their Structure Mapping Engine.

Demonstrated that the SME algorithms take time proportional to N-cubed, where N is the number of 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™.

Find the closest matching signatures in logarithmic time.

Use structure mapping only on a very small number of graphs.

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 important application 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:

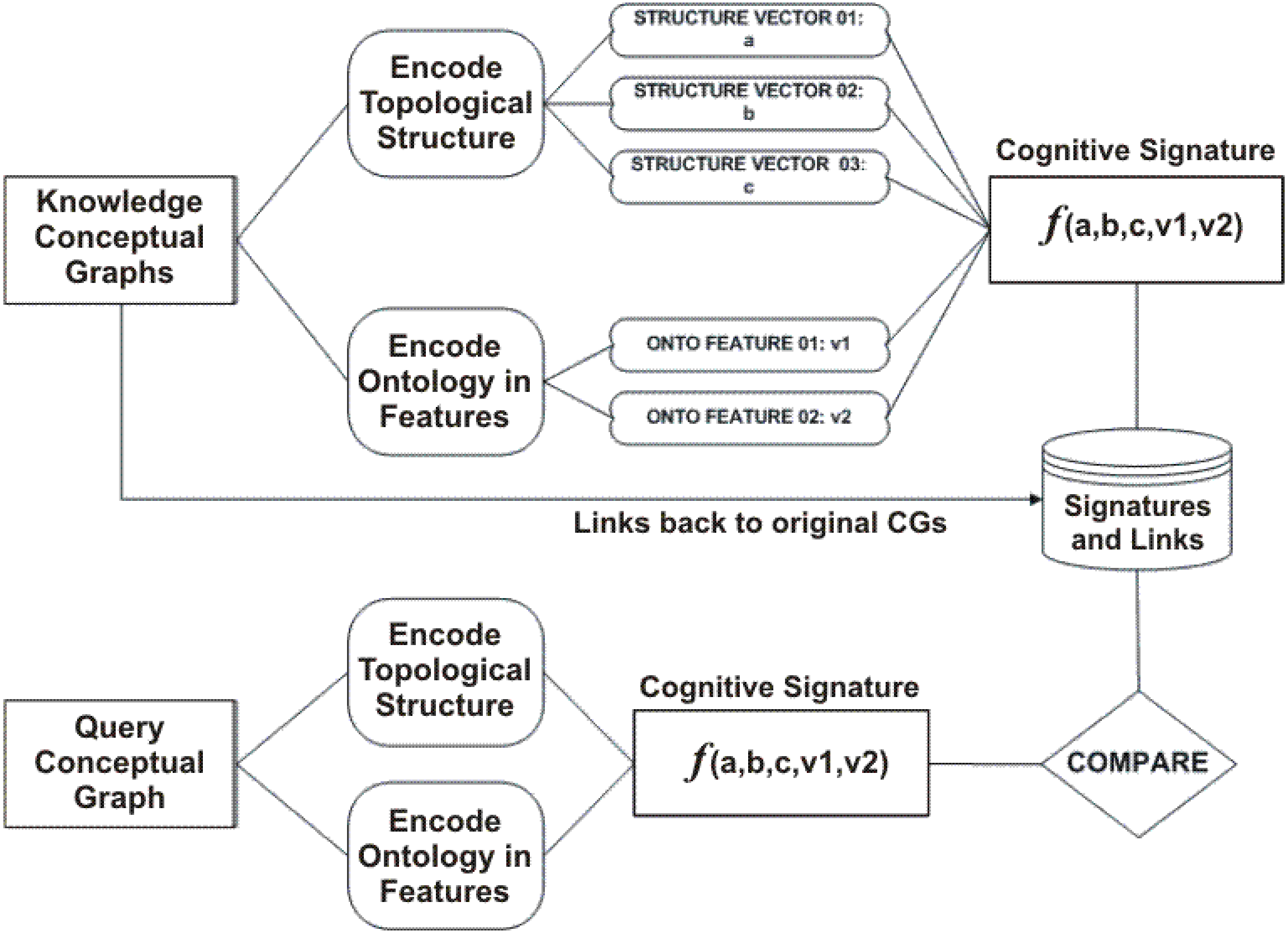
- Represent each graph by its unique International Chemical Identifier (InChI).
- Map the InChI codes to numeric vectors that encode both the graph structure and the labels of the atoms and bonds.
- Index the vectors by a locality-sensitive hashing (LSH) algorithm.
- Estimate the semantic distance between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

Similar techniques can be adapted to conceptual graphs.

For a description 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

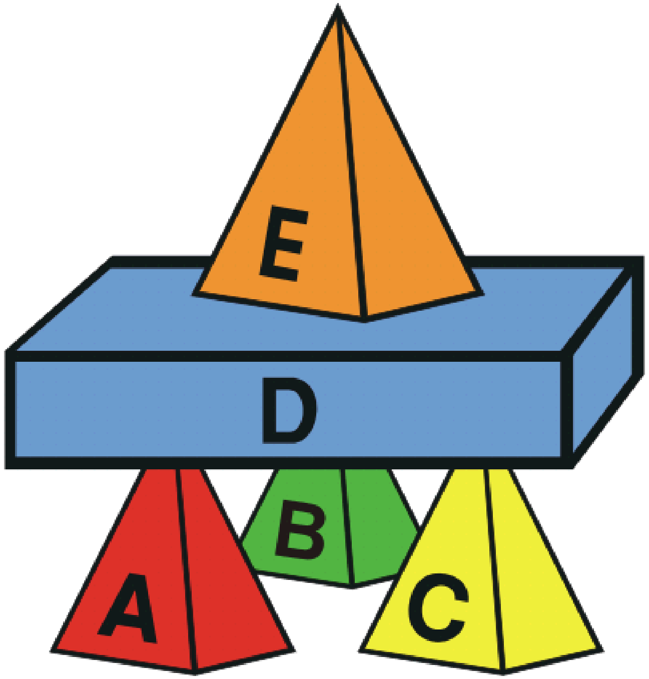
Cognitive Memory™



Basis for the VivoMind Analogy Engine (VAE)

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?

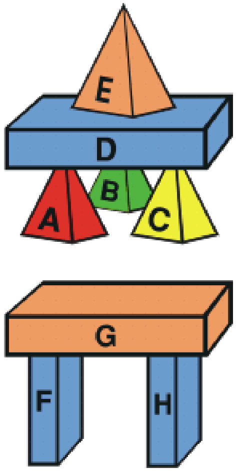
In a programming language?

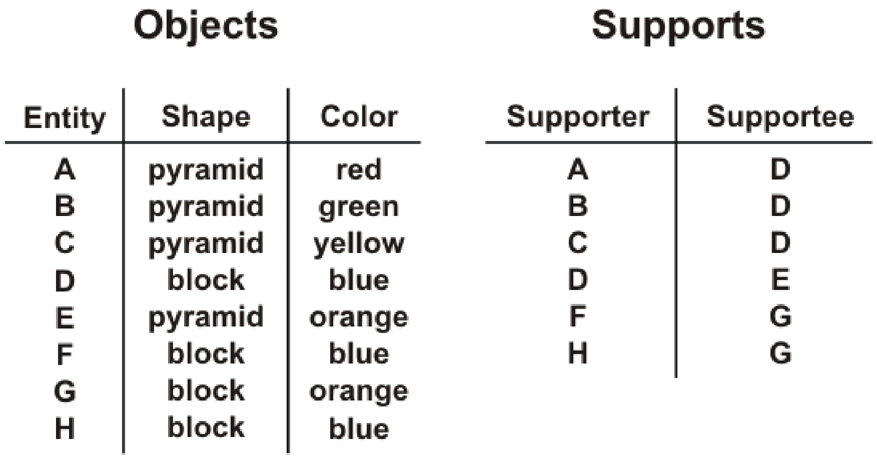
Even when people use the same language,
they use different words and expressions.

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

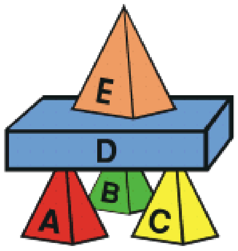
A description in English:

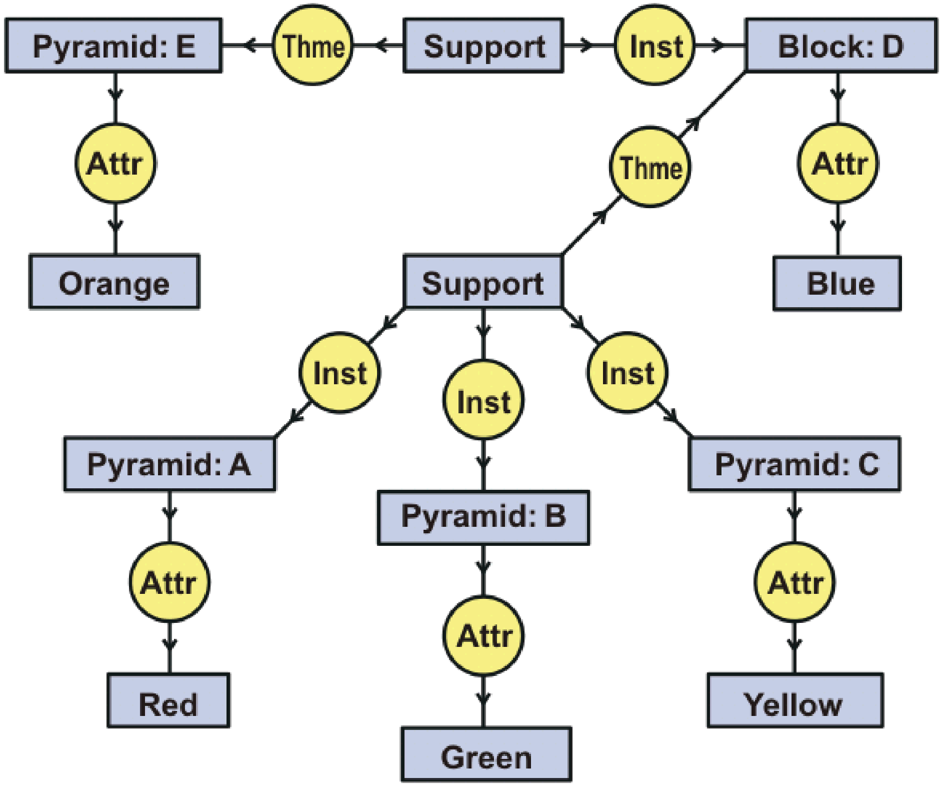
> “A red pyramid A, a green pyramid B, and a yellow pyramid C support a blue block D, which supports an orange pyramid E.”

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

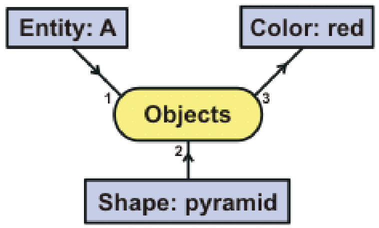
Objects

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

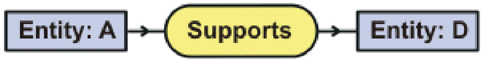
Supports

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

⇕

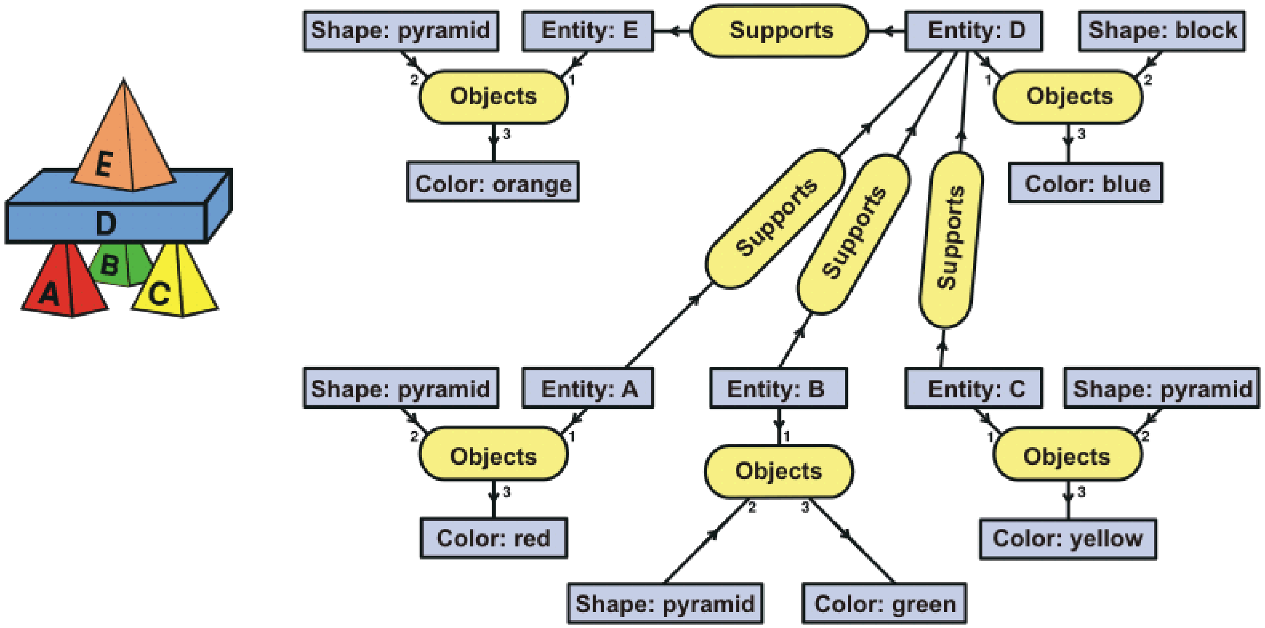


⇕



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

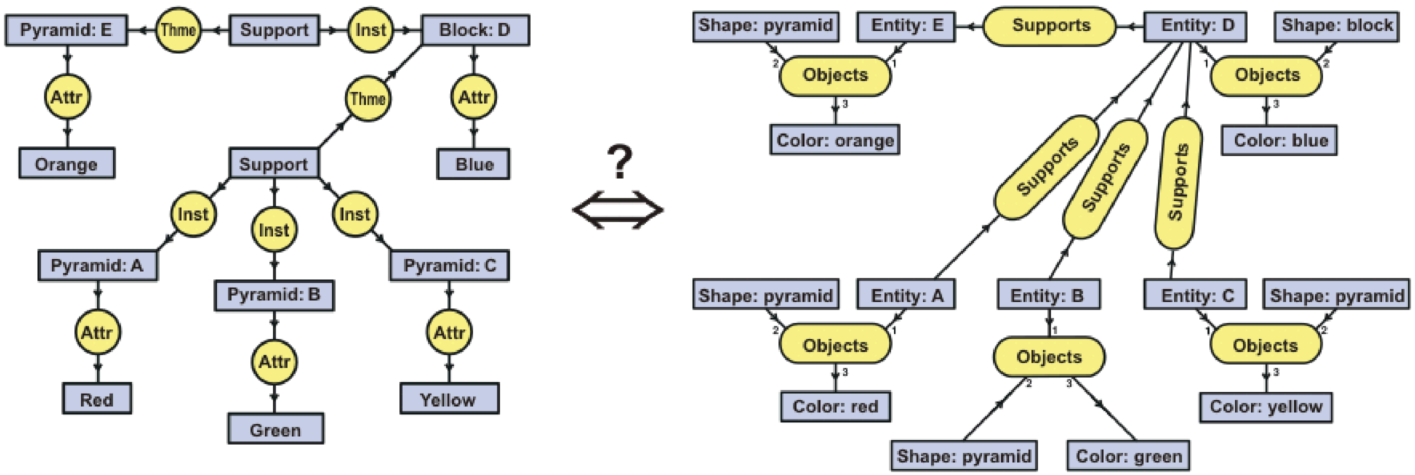
Mapping an Entire Database to Conceptual Graphs



Join concept nodes that refer to the same entities.

Closely related entities are described by connected graphs.

Mapping the Two Graphs to One Another



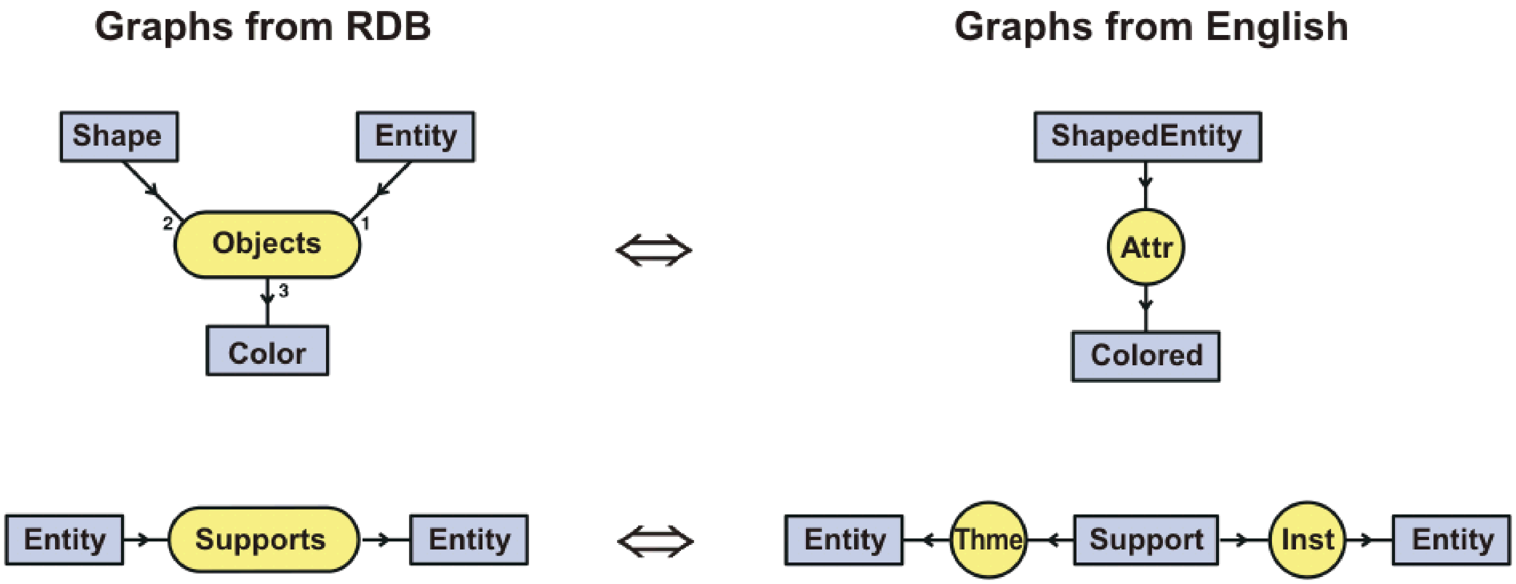
Very different ontologies: 12 concept nodes vs. 15 concept nodes, 11 relation nodes vs. 9 relation nodes, no similarity in type labels.

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?

Mapping the Graphs by Aligning the Ontologies



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.

Three Applications for an Analogy Engine

1. Educational Software

   Evaluating student answers written in free-form English.

2. Legacy Re-engineering

   Comparing structured data to English documentation, finding which sentences describe which data, and detecting errors and inconsistencies between the data and the sentences.

3. Oil and gas exploration

   Reading English documents about oil and gas exploration and answering English questions written by a geologist.

These applications were implemented by VivoMind to satisfy requirements stated by the clients.

1. 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.***

How could a computer system evaluate such answers?

Determine whether they are correct, incorrect, or partially correct?

And make helpful suggestions about the incorrect answers?

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.***

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

How can a computer system evaluate them?

How can 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.

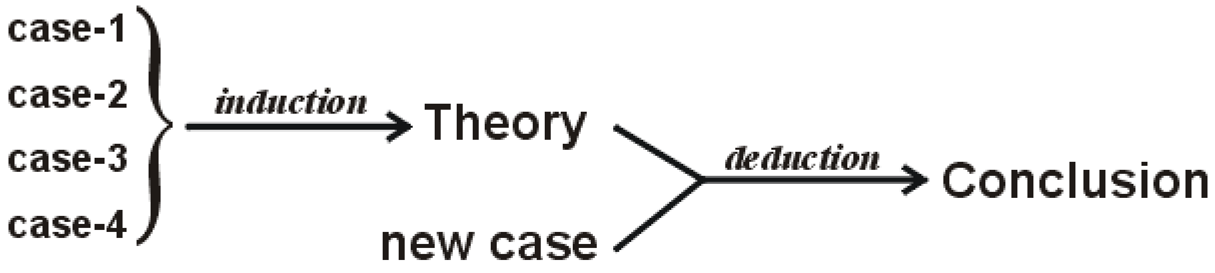
- Some are partially correct
  — and the teacher says what is missing.

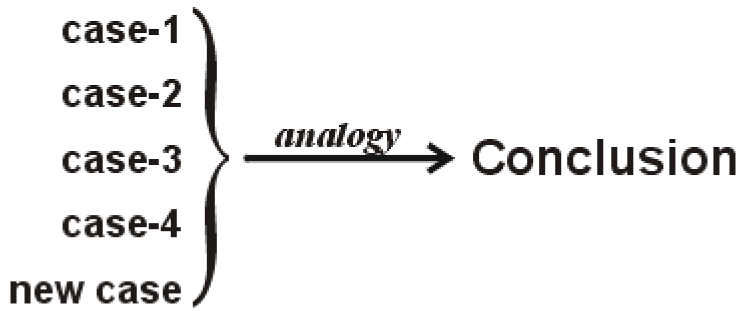
- Others are wrong
  — in many different ways.

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 VAE to Evaluate Student Answers

Translate answers to CGs & store in Cognitive Memory™.

VAE compares each new answer to the 50 cases:

1. VivoMind Language parser (VLP) translates each new answer to a CG.
2. Compare the new CG to the CGs in Cognitive Memory.
3. If there is a good match, print out the corresponding response.
4. Otherwise, send the new student answer to some teacher to evaluate.

Results:

- VAE found a good match for nearly all student answers.
- For each good match, the previous teacher’s response was appropriate.
- When VAE failed to find a good match, the new case would be added to Cognitive Memory in order to improve its coverage.
- No need for teachers to use any language other than English.

2. Problem for Legacy Re-engineering

Analyze the software and documentation of a large corporation.

Programs in daily use, some of which were 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 index to the software.

  Structure diagrams of the programs, files, and data.

  List of discrepancies between the programs 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.

Interpreting Novel Patterns

Many texts contain unusual or ungrammatical patterns.

They may be elliptical forms that could be stored in tables.

But some authors write them as phrases in a sentence:

- ***32 — loss on unbilled***
- ***72 — gain on uncollected***
- ***85 — loss on uncollected***

Intellitex generated a CG with a default relation (Link):

[Number: 32]→(Link)→[Punctuation: "–"]→(Link)→[Loss]→(On)→[Unbilled]

The value 32 was stored as a constant in a COBOL program.

The phrase "loss on unbilled" was written as a comment.

The value and the comment from COBOL were translated to a CG that was the closest match to the CG derived from the text.

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 Intellitex + 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:

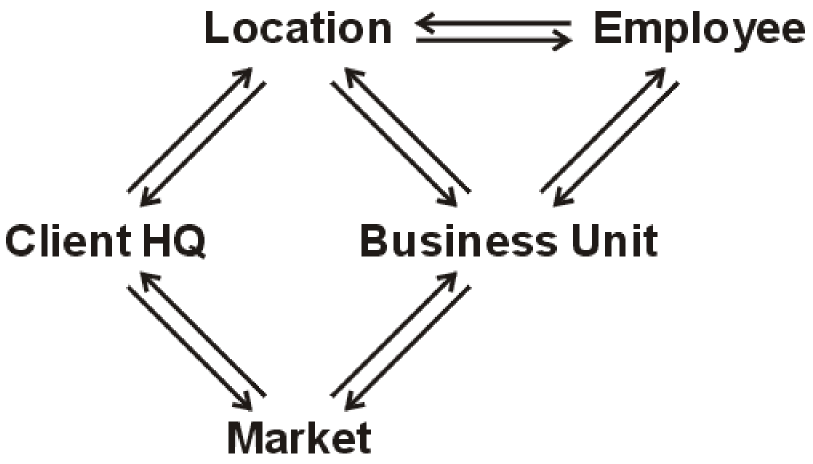
  Glossary, data dictionary, data flow diagrams, process architecture, system context diagrams.

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

With VivoMind software, it was completed 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 the ontology used for interpreting English:

- 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.”

3. Application to Oil and Gas Exploration

Source material:

- 79 documents, ranging in length from 1 page to 50 pages.
- Some are reports about oil or gas fields, and others are chapters from a textbook on geology used as background information.
- English, as written for human readers (no semantic tagging).
- Additional data from relational DBs and other structured sources.
- Lexical resources derived from WordNet, CoreLex, IBM-CSLI Verb Ontology, Roget's Thesaurus, and other sources.
- An ontology for the oil and gas written in CLCE by geologists from the Energy and Geoscience Institute at the University of Utah.

Queries:

- One or more sentences that describe a potential oil or gas field.
- Analogies compare the query to passages in the documents.

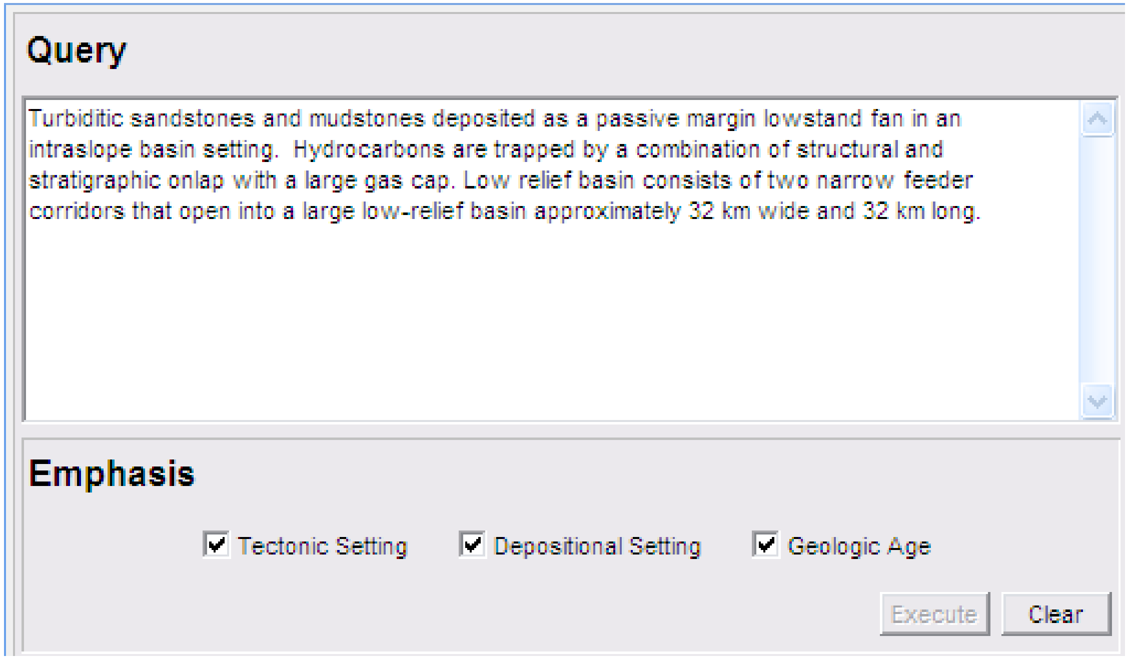
Answering Queries with VLP and VAE

For each sentence or structured data item in the sources,

- Translate the sentence or data item to a conceptual graph.
- Translate the CGs to Cognitive Signatures™ in time proportional to (N log N), where N is the total number of CGs.
- Store each Cognitive Signature in Cognitive Memory™ with a pointer back to the original source.
- Use previously translated CGs to help interpret new sentences.

For a query stated as an English sentence or paragraph,

- Translate the query to conceptual graphs.
- Find matching patterns in the source data and rank them in order of semantic distance. (Zero distance means an exact match.)
- For each match within a given threshold, use structure mapping to verify which parts of the query CG match the source CG.
- As answer, return the English phrases from the source document from which the best matches were derived.



Query

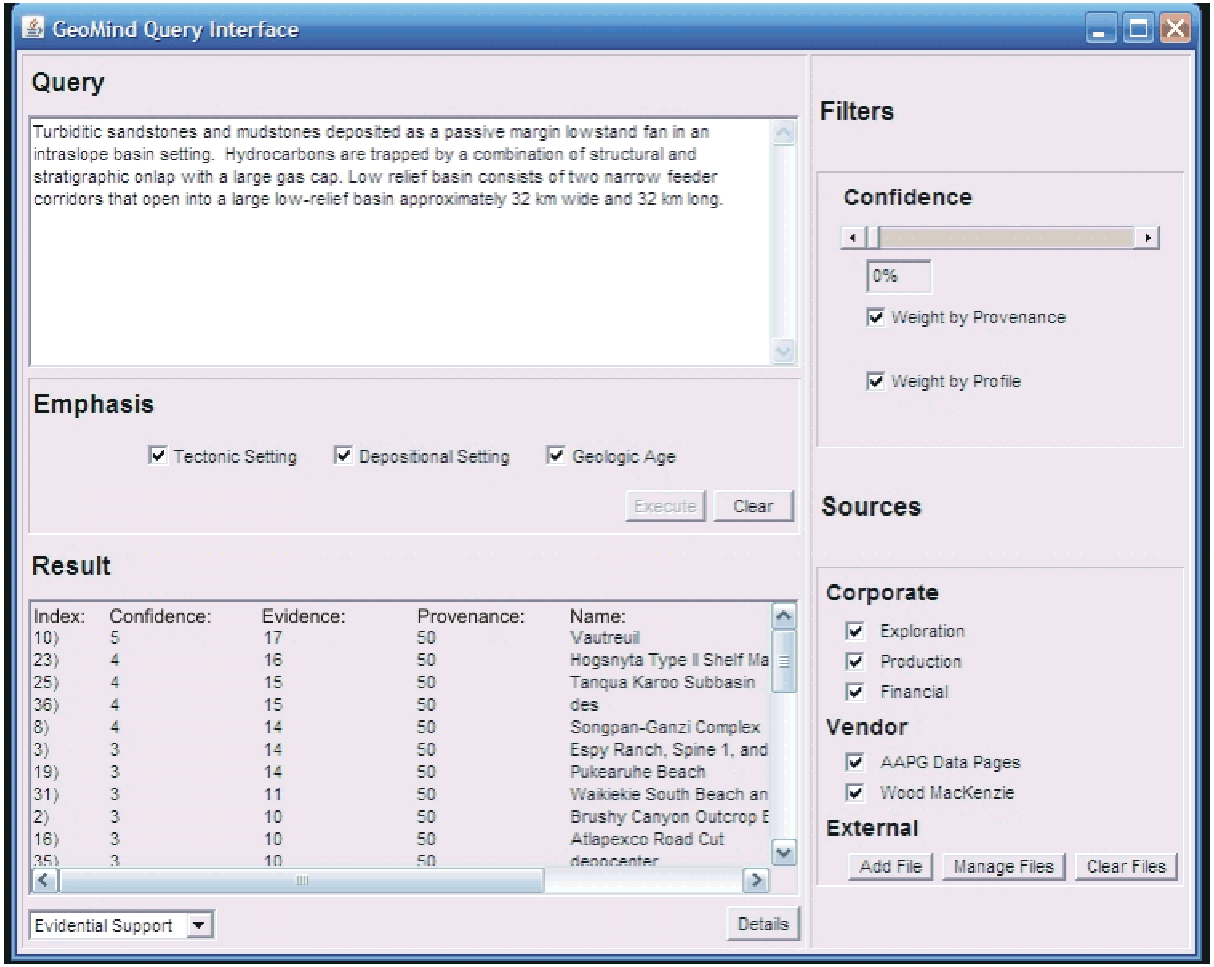
Turbiditic sandstones and mudstones deposited as a passive margin lowstand fan in an intraslope basin setting. Hydrocarbons are trapped by a combination of structural and stratigraphic onlap with a large gas cap. Low relief basin consists of two narrow feeder corridors that open into a large low-relief basin approximately 32 km wide and 32 km long.

Emphasis

☑ Tectonic Setting ☑ Depositional Setting ☑ Geologic Age

Turbiditic sandstones and mudstones deposited as a passive margin lowstand fan in an intraslope basin setting. Hydrocarbons are trapped by a combination of structural and stratigraphic onlap with a large gas cap. Low relief basin consists of two narrow feeder corridors that open into a large low-relief basin approximately 32 km wide and 32 km long.

THE QUERY



GeoMind Query Interface

Query

Turbiditic sandstones and mudstones deposited as a passive margin lowstand fan in an intraslope basin setting. Hydrocarbons are trapped by a combination of structural and stratigraphic onlap with a large gas cap. Low relief basin consists of two narrow feeder corridors that open into a large low-relief basin approximately 32 km wide and 32 km long.

Emphasis

☑ Tectonic Setting ☑ Depositional Setting ☑ Geologic Age

Result

Index:	Confidence:	Evidence:	Provenance:	Name:
10)	5	17	50	Vautreuil
23)	4	16	50	Hogsnyta Type II Shelf Ma
25)	4	15	50	Tanqua Karoo Subbasin
36)	4	15	50	des
8)	4	14	50	Songpan-Ganzi Complex
3)	3	14	50	Espy Ranch, Spine 1, and
19)	3	14	50	Pukearuhe Beach
31)	3	11	50	Waikiekie South Beach an
2)	3	10	50	Brushy Canyon Outcrop E
16)	3	10	50	Atlapexco Road Cut
35)	3	10	50	depocenter

Evidential Support

Filters

Confidence

0%

☑ Weight by Provenance

☑ Weight by Profile

Sources

Corporate

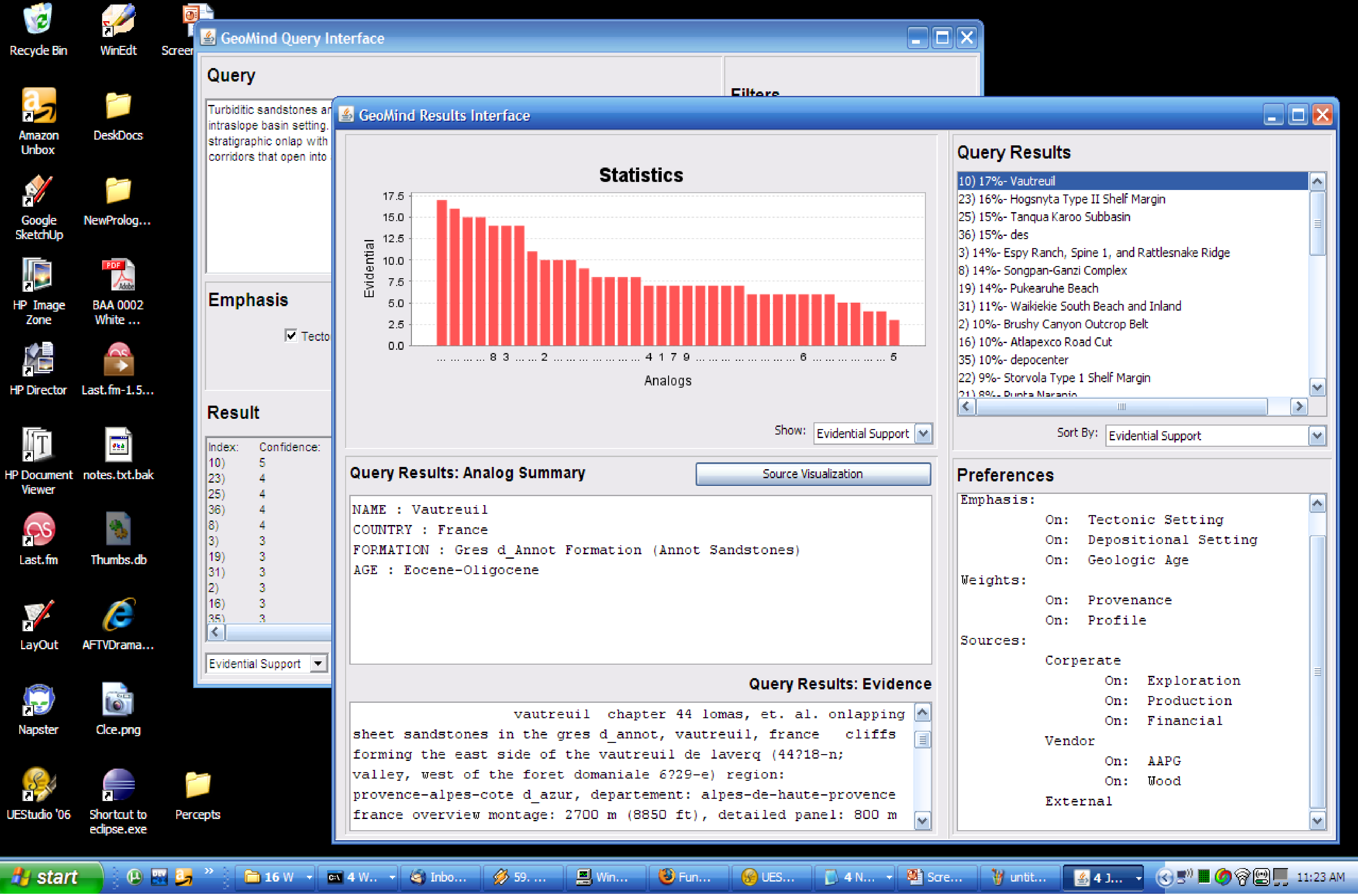
☑ Exploration
☑ Production
☑ Financial

Vendor

☑ AAPG Data Pages
☑ Wood MacKenzie

External

RESULTS, ranked by evidence (Dempster-Shafer) & confidence factors



Turbiditic sandstones ar
intraslope basin setting.
stratigraphic onlap with
corridors that open into

☑ Tecto

Index:	Confidence:
10)	5
23)	4
25)	4
36)	4
8)	4
3)	3
19)	3
31)	3
2)	3
16)	3
35)	3

Evidential Support

Show: Evidential Support

Query Results

10) 17%- Vautreuil
23) 16%- Hogsnyta Type II Shelf Margin
25) 15%- Tanqua Karoo Subbasin
36) 15%- des
3) 14%- Espy Ranch, Spine 1, and Rattlesnake Ridge
8) 14%- Songpan-Ganzi Complex
19) 14%- Pukearuhe Beach
31) 11%- Waikiekie South Beach and Inland
2) 10%- Brushy Canyon Outcrop Belt
16) 10%- Atlapexco Road Cut
35) 10%- depocenter
22) 9%- Storvola Type 1 Shelf Margin
21) 8%- Punta Naranjo

Sort By: Evidential Support

Query Results: Analog Summary

```
NAME : Vautreuil
COUNTRY : France
FORMATION : Gres d_Annot Formation (Annot Sandstones)
AGE : Eocene-Oligocene
```

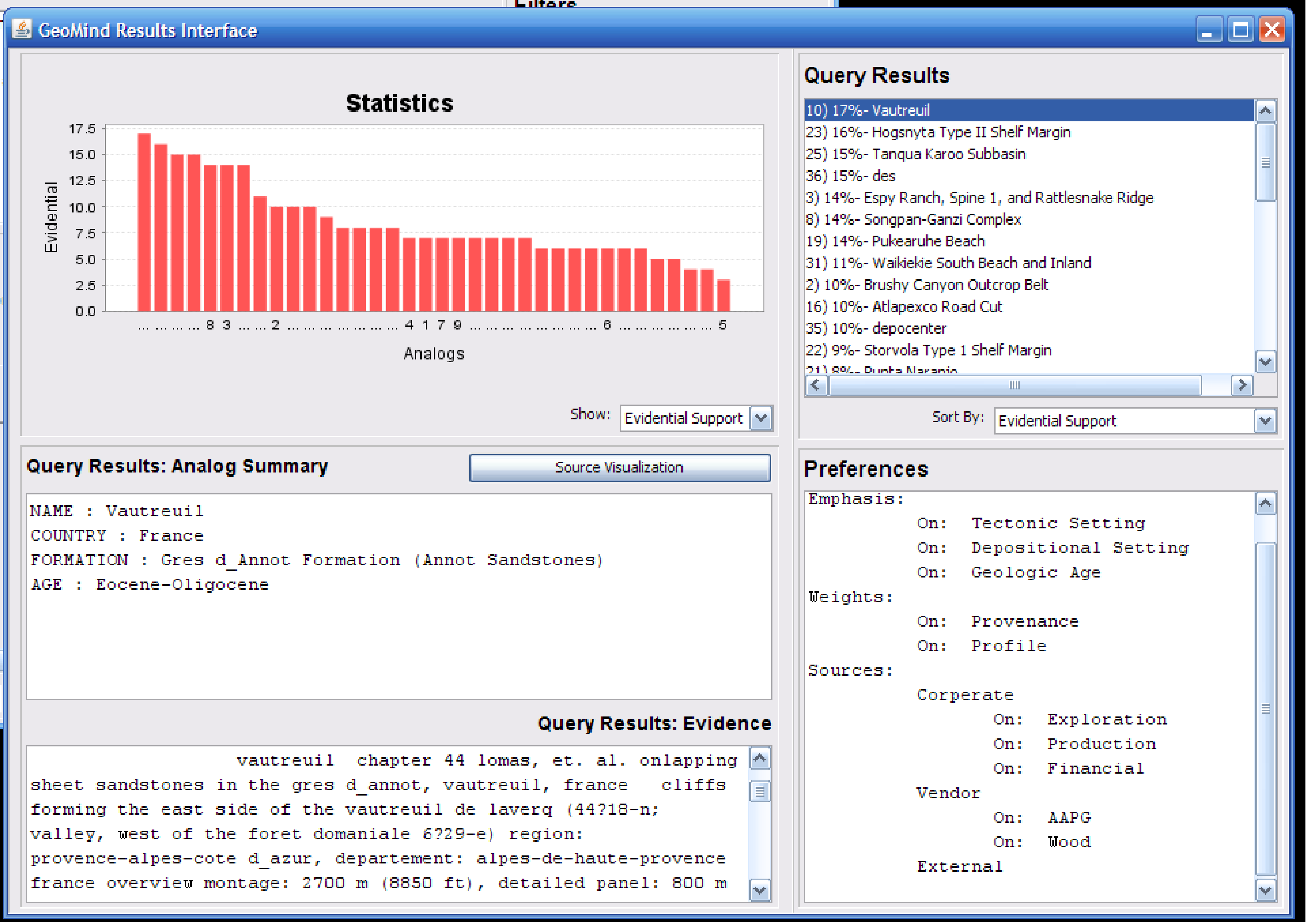
Query Results: Evidence

```
                    vautreuil  chapter 44 lomas, et. al. onlapping
sheet sandstones in the gres d_annot, vautreuil, france   cliffs
forming the east side of the vautreuil de laverq (44?18-n;
valley, west of the foret domaniale 6?29-e) region:
provence-alpes-cote d_azur, departement: alpes-de-haute-provence
france overview montage: 2700 m (8850 ft), detailed panel: 800 m
```

Preferences

```
Emphasis:
          On:  Tectonic Setting
          On:  Depositional Setting
          On:  Geologic Age
Weights:
          On:  Provenance
          On:  Profile
Sources:
          Corperate
                 On:  Exploration
                 On:  Production
                 On:  Financial
          Vendor
                 On:  AAPG
                 On:  Wood
          External
```

After clicking the “Details” button on the previous window



Query Results

10) 17%- Vautreuil
23) 16%- Hogsnyta Type II Shelf Margin
25) 15%- Tanqua Karoo Subbasin
36) 15%- des
3) 14%- Espy Ranch, Spine 1, and Rattlesnake Ridge
8) 14%- Songpan-Ganzi Complex
19) 14%- Pukearuhe Beach
31) 11%- Waikiekie South Beach and Inland
2) 10%- Brushy Canyon Outcrop Belt
16) 10%- Atlapexco Road Cut
35) 10%- depocenter
22) 9%- Storvola Type 1 Shelf Margin
21) 8%- Punta Naranjo

Query Results: Analog Summary

NAME : Vautreuil
COUNTRY : France
FORMATION : Gres d_Annot Formation (Annot Sandstones)
AGE : Eocene-Oligocene

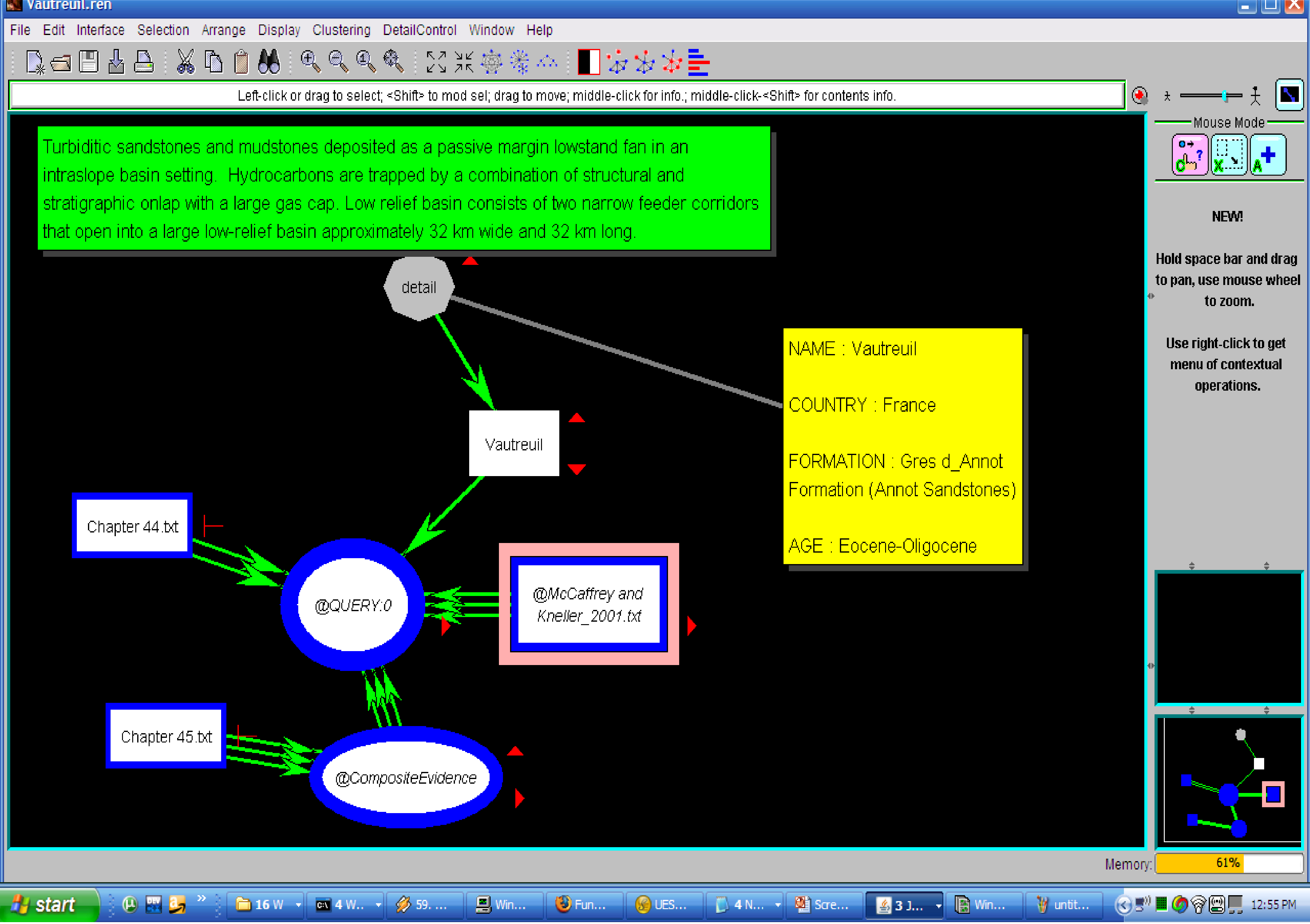
Query Results: Evidence

vautreuil chapter 44 lomas, et. al. onlapping sheet sandstones in the gres d_annot, vautreuil, france cliffs forming the east side of the vautreuil de laverq (44?18-n; valley, west of the foret domaniale 6?29-e) region: provence-alpes-cote d_azur, departement: alpes-de-haute-provence france overview montage: 2700 m (8850 ft), detailed panel: 800 m

Preferences

```
Emphasis:
          On:   Tectonic Setting
          On:   Depositional Setting
          On:   Geologic Age
Weights:
          On:   Provenance
          On:   Profile
Sources:
          Corperate
                 On:   Exploration
                 On:   Production
                 On:   Financial
          Vendor
                 On:   AAPG
                 On:   Wood
          External
```

DETAILS – Next, click the “Source Visualization” button



Left-click or drag to select; <Shift> to mod sel; drag to move; middle-click for info.; middle-click-<Shift> for contents info.

Turbiditic sandstones and mudstones deposited as a passive margin lowstand fan in an intraslope basin setting. Hydrocarbons are trapped by a combination of structural and stratigraphic onlap with a large gas cap. Low relief basin consists of two narrow feeder corridors that open into a large low-relief basin approximately 32 km wide and 32 km long.

NAME : Vautreuil

COUNTRY : France

FORMATION : Gres d_Annot Formation (Annot Sandstones)

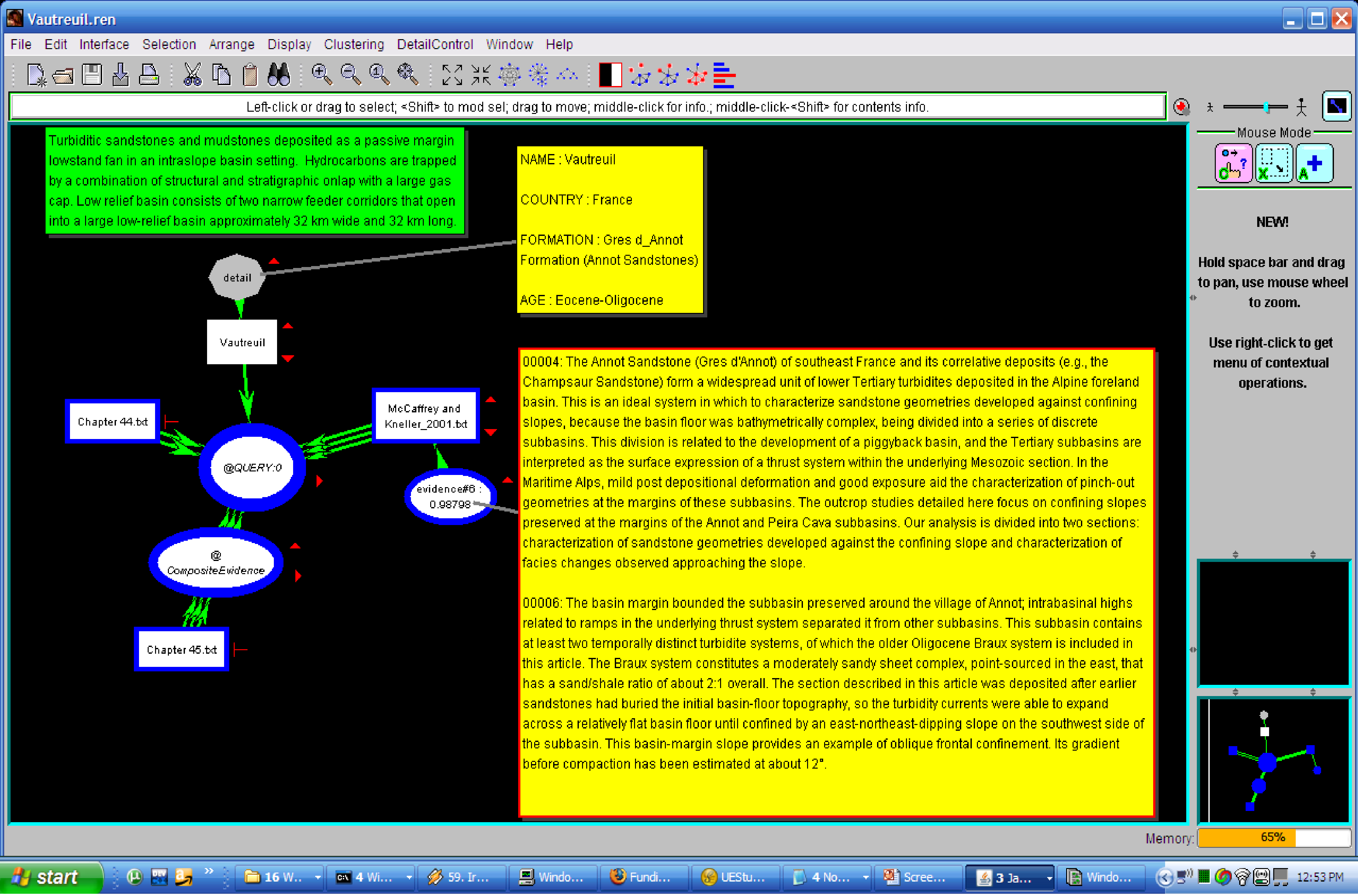
AGE : Eocene-Oligocene

NEW!

Hold space bar and drag to pan, use mouse wheel to zoom.

Use right-click to get menu of contextual operations.

Top-level source visualization and the highest-ranked result



Left-click or drag to select; <Shift> to mod sel; drag to move; middle-click for info.; middle-click-<Shift> for contents info.

Turbiditic sandstones and mudstones deposited as a passive margin lowstand fan in an intraslope basin setting. Hydrocarbons are trapped by a combination of structural and stratigraphic onlap with a large gas cap. Low relief basin consists of two narrow feeder corridors that open into a large low-relief basin approximately 32 km wide and 32 km long.

NAME : Vautreuil

COUNTRY : France

FORMATION : Gres d_Annot Formation (Annot Sandstones)

AGE : Eocene-Oligocene

00004: The Annot Sandstone (Gres d'Annot) of southeast France and its correlative deposits (e.g., the Champsaur Sandstone) form a widespread unit of lower Tertiary turbidites deposited in the Alpine foreland basin. This is an ideal system in which to characterize sandstone geometries developed against confining slopes, because the basin floor was bathymetrically complex, being divided into a series of discrete subbasins. This division is related to the development of a piggyback basin, and the Tertiary subbasins are interpreted as the surface expression of a thrust system within the underlying Mesozoic section. In the Maritime Alps, mild post depositional deformation and good exposure aid the characterization of pinch-out geometries at the margins of these subbasins. The outcrop studies detailed here focus on confining slopes preserved at the margins of the Annot and Peira Cava subbasins. Our analysis is divided into two sections: characterization of sandstone geometries developed against the confining slope and characterization of facies changes observed approaching the slope.

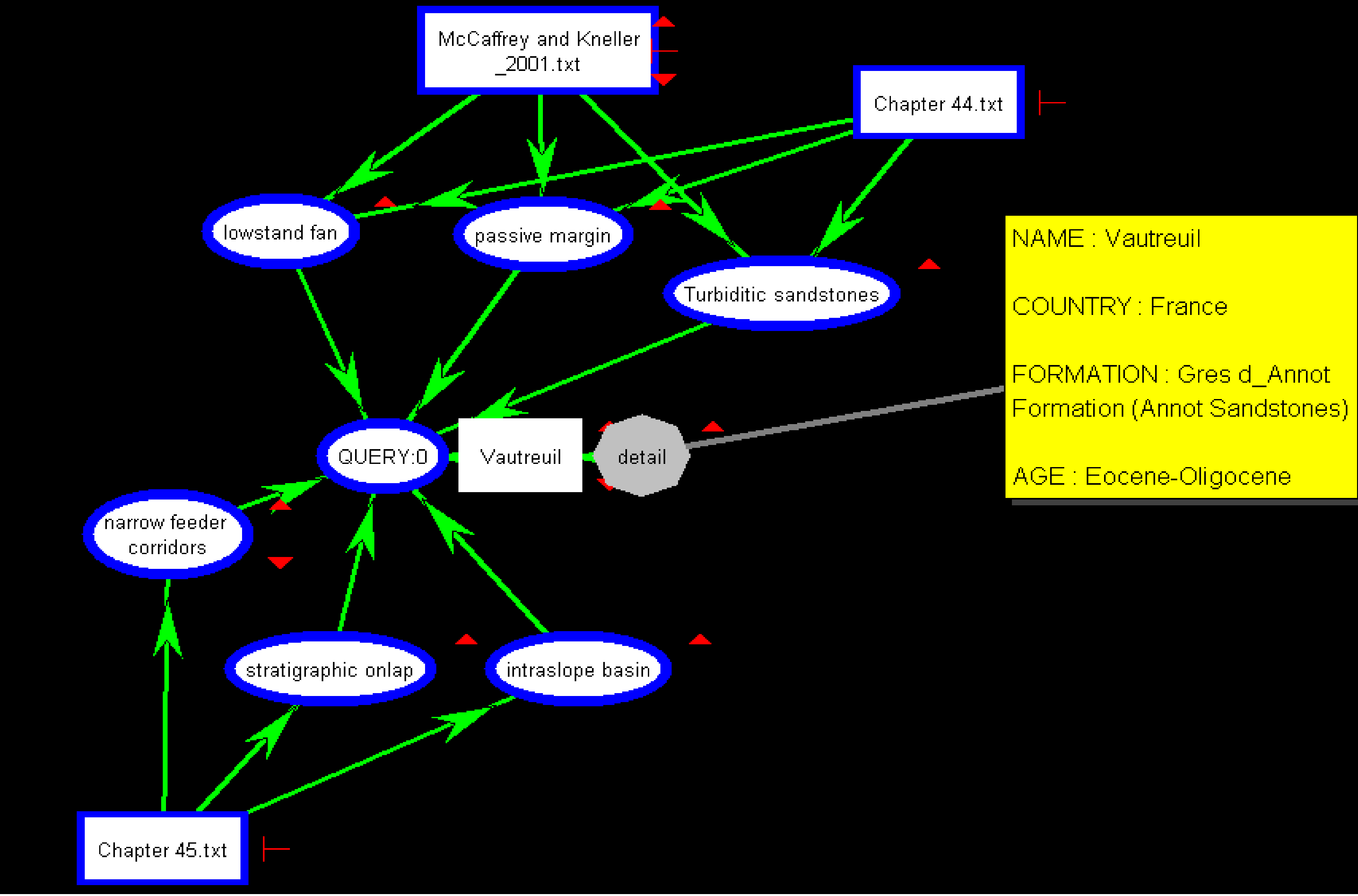
00006: The basin margin bounded the subbasin preserved around the village of Annot; intrabasinal highs related to ramps in the underlying thrust system separated it from other subbasins. This subbasin contains at least two temporally distinct turbidite systems, of which the older Oligocene Braux system is included in this article. The Braux system constitutes a moderately sandy sheet complex, point-sourced in the east, that has a sand/shale ratio of about 2:1 overall. The section described in this article was deposited after earlier sandstones had buried the initial basin-floor topography, so the turbidity currents were able to expand across a relatively flat basin floor until confined by an east-northeast-dipping slope on the southwest side of the subbasin. This basin-margin slope provides an example of oblique frontal confinement. Its gradient before compaction has been estimated at about 12°.

NEW!

Hold space bar and drag to pan, use mouse wheel to zoom.

Use right-click to get menu of contextual operations.

Drill down to one of the documents for the human readers



NAME : Vautreuil

COUNTRY : France

FORMATION : Gres d_Annot Formation (Annot Sandstones)

AGE : Eocene-Oligocene

Drill down into the query and its relationships to the source documents

Emergent Knowledge

When reading the 79 documents,

- VLP translates the sentences and paragraphs to CGs.
- But it does not do any further analysis of the documents.

When a geologist asks a question,

- The VivoMind system may find related phrases in many sources.
- To connect those phrases, it may need to do further searches.
- The result is a conceptual graph that relates the question to multiple passages in multiple sources.
- Some of those sources might contribute information that does not have any words that came from the original question.
- That new CG can be used to answer further questions.

By a “Socratic” dialog, the geologist can lead the system to explore novel paths and discover unexpected patterns.

Diagnosing Cancer Patients

The same technology can be applied to language about any topic.

As an example, the documents might describe cancer patients, and the query could describe another patient.

The analogies could highlight any aspect of interest: patient description, medical history, therapy, results, etc.

The source documents could include unstructured reports in any natural language and structured data from a database.

With appropriate parsers and translators, any of that data could be translated to conceptual graphs, indexed, and processed by the analogy engine.

Ontologies with detailed definitions would be useful, but not required.

A global alignment of the ontologies would be useful, but not required.

Operational decisions would be made by a physician, who could examine the source documents to evaluate any hypotheses generated by the system.

Conclusions

Analogy is the foundation for human reasoning.

Without analogy, language understanding is impossible.

Logic is a highly disciplined special case of analogical reasoning:

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

Conceptual graphs support both logical and analogical methods:

- **They are defined by the ISO/IEC standard 24707 for Common Logic.**
- **But they also support semantic distance measures for analogy.**
- **They provide a bridge between informal language and formal logic.**

Related Readings

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

The Role of Logic and Language in Ontology, by J. F. Sowa
http://www.jfsowa.com/pubs/rolelog.pdf

Future Directions in Semantic Systems,
http://www.jfsowa.com/pubs/futures.pdf

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

Peirce's own tutorial on existential graphs:
http://www.jfsowa.com/peirce/ms14.htm

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