

# **Organizing and Relating The Patterns of Logic**

**John F. Sowa**

**19 March 2013**

# Classical Artificial Intelligence

**An approach to problem solving and question answering:**

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

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

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

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

# **Rule-Based Expert Systems**

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

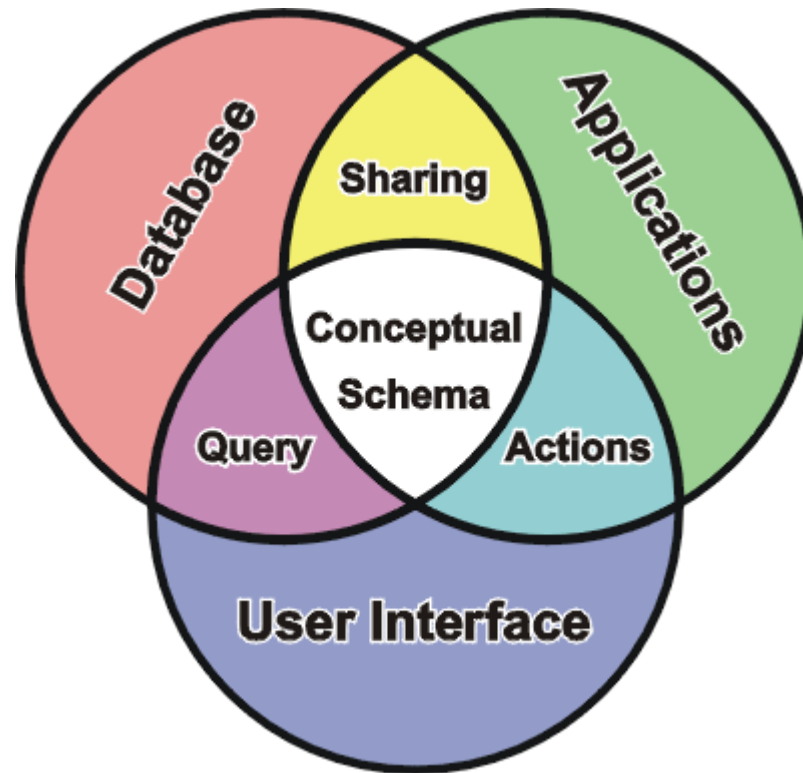
**Unfortunately, expert systems acquired a bad reputation:**

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

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

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

# Conceptual Schema



## Three-schema architecture by ANSI SPARC in 1978:

- **Conceptual schema defines the ontology of a database.**
- **Physical schema defines the storage and access methods.**
- **Application schema defines the APIs for programming languages.**

# Standardizing the Conceptual Schema

## Many database experts agreed:

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

## But other DB experts disagreed:

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

## Over thirty years of R & D and ISO proposals:

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

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

# Database and Knowledge Base Design

## Good methodologies are essential:

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

## Some tools have been designed, implemented, and used:

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

## Generic ontologies must be specialized for each context:

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

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

# Semantics of Business Vocabulary and Business Rules

**SBVR is a system for representing business rules in logic:**

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

**SBVR Controlled English about the Nobel Prize:**

- **Fact types are patterns that relate entity types and *roles*:**  
*Work was awarded in Category in Year*  
*Laureate was awarded in Category in Year*
- **Fact types combined with *quantifiers* are used to state constraints:**  
*In each Category in each Year the prize is won by at most two Works.*  
*In each Category in each Year the prize is won by at most three Laureates.*

# Logic Programming (LP)

**Logic can be used as a programming language:**

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

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

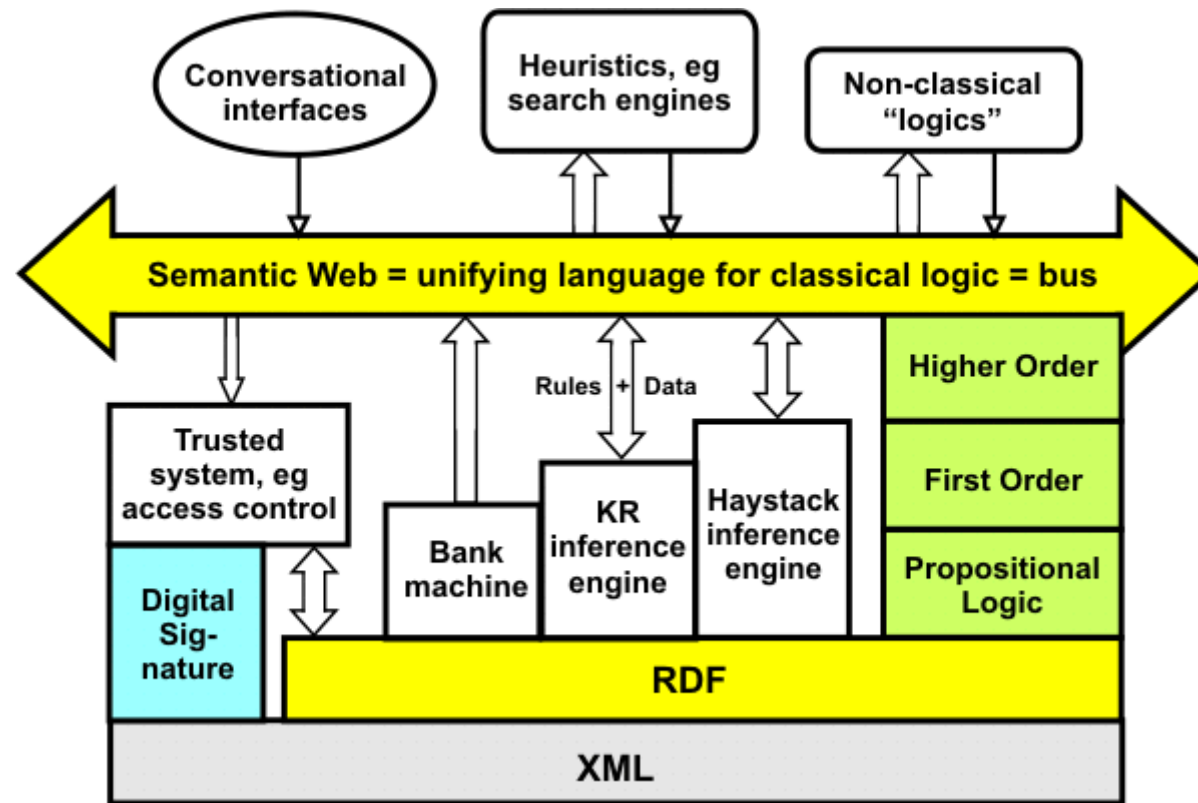
- **When Ted Codd saw Prolog, he said “I wish I had invented that.”**

**Example: The Experian credit bureau.**

- **They use rules written in Prolog to check everybody’s credit score.**
- **The amount of data they process daily is enormous.**
- **They are so heavily dependent on Prolog that they bought Prologia, the company founded by Alain Colmerauer, who invented Prolog.**



# The Original Semantic Web Proposal

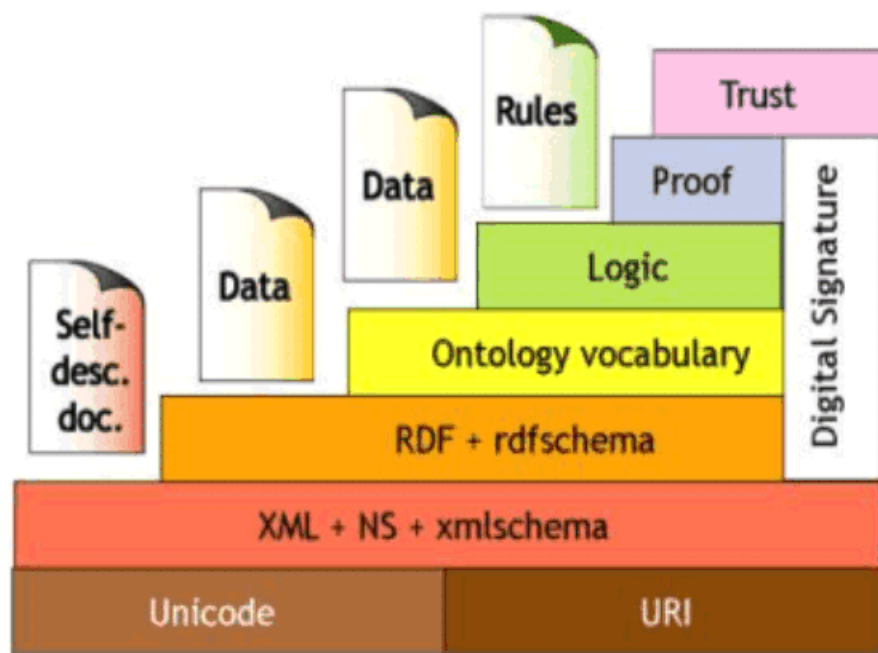


In the DAML proposal by Tim Berners-Lee in Feb. 2000:

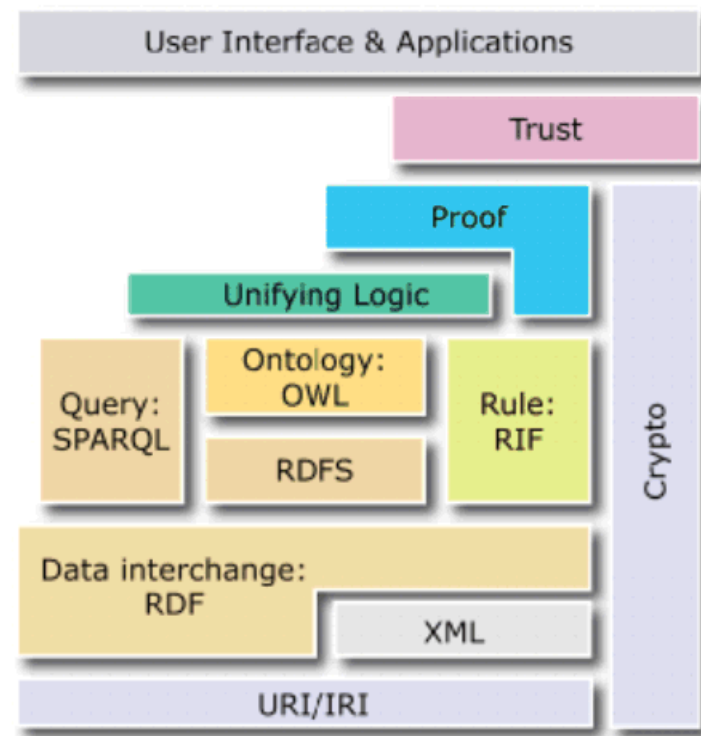
- Emphasis on diversity, heterogeneity, and interoperability.
- SweLL: Semantic Web Logic Language as a highly expressive, unifying language that can relate information in all other logics.

See <http://www.w3.org/2000/01/sw/DevelopmentProposal>

# Semantic Web as Implemented



Original Layer Cake



New Layer Cake

The “layer cake” on the left from 2001 had many good ideas. But the one on the right from 2006 omitted essential features. See the documents cited in <http://www.jfsowa.com/ikl>

# Missing Features

**The final report in 2005 was a pale shadow of Tim's vision:**

- **The highly expressive SWeLL was limited to RDF and OWL.**
- **In 2000, diversity, heterogeneity, and interoperability were repeatedly discussed throughout the proposal.**
- **But in 2005, the words 'diverse' and 'interoperable' were mentioned once, and 'heterogeneous' was never mentioned.**
- **The proposal in 2000 discussed first-order logic, higher order logic, default logic, KIF, and Prolog. The 2005 report did not mention them.**
- **The word 'heuristic' occurred 23 times in 2000, but zero in 2005.**

**The worst aspect of the 2005 report was its narrowness:**

- **Tim's vision in 2000 embraced an open-ended diversity.**
- **The 2005 report excluded everything but a tiny minority of the logics that have been widely and successfully used for AI and databases.**
- **Eight years later, no progress has been made in extending the range of logics beyond the limited few of the 2005 report.**

# Lost Opportunity

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

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

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

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

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

# A Family of Logics

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

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

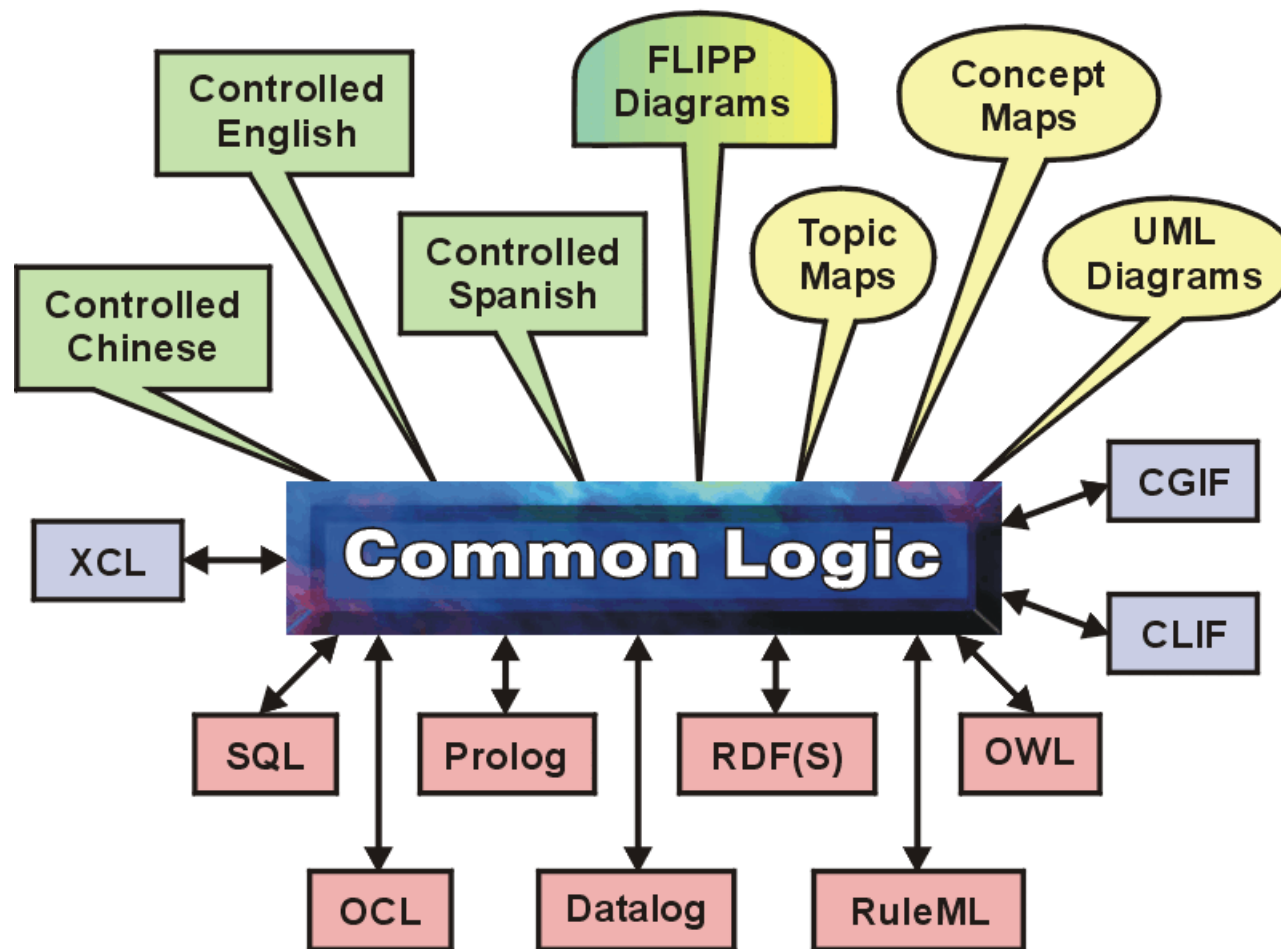
**The ISO standard 24707 for Common Logic defines a very general semantic foundation for an open-ended family of dialects.**

**Three normative dialects are specified in ISO 24707:**

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

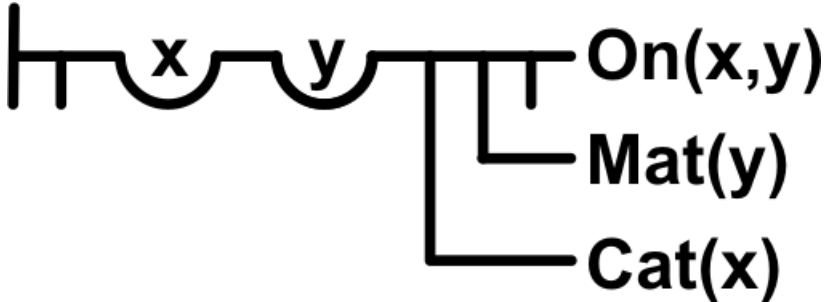
**But any notation that uses the common semantics can join the family.**

# Human Interfaces



# Machine Interfaces

# How to say “A cat is on a mat.”

Gottlob Frege (1879): 

Charles Sanders Peirce (1885):

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

Giuseppe Peano (1895):

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

Charles Sanders Peirce (1897):

$$\text{Cat} \text{ — } \text{On} \text{ — } \text{Mat}$$

All four notations have identical semantics.

# Some Modern Notations

SQL query:

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

Common Logic Interchange Format (ISO 24707):

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

Conceptual Graph Interchange Format (ISO 24707):

$[\text{Cat } *x] [\text{Mat } *y] (\text{On } ?x \ ?y)$

Conceptual Graph Display Form:



Controlled English:

*A cat is on a mat.*



# Some Dialects of Common Logic

**Two standardized dialects defined by ISO/IEC 24707:**

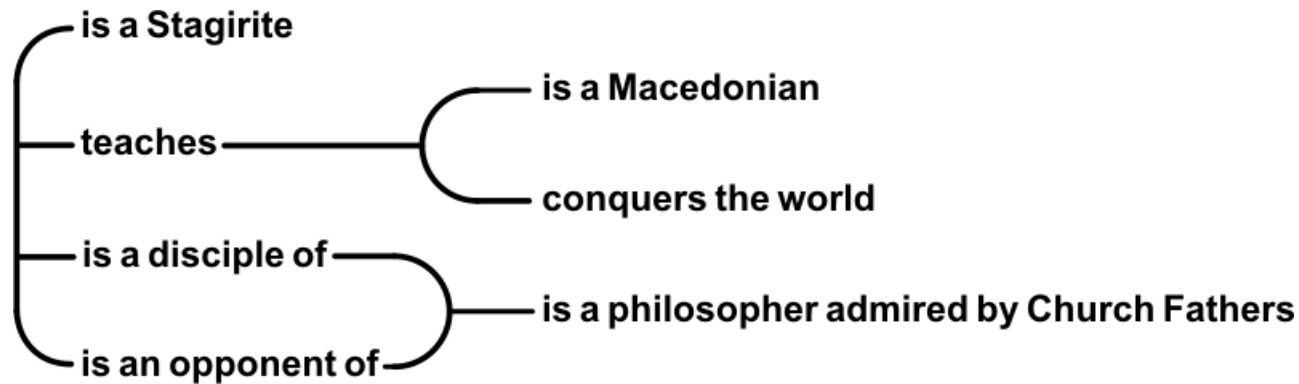
- **CLIF: Common Logic Interchange format**
- **CGIF: Conceptual Graph interchange format**

**Four notations that can be translated to and from CLIF and CGIF:**

- **Predicate calculus**
- **Existential Graphs**
- **Conceptual Graphs**
- **Common Logic Controlled English (CLCE)**

**Existential graphs will be used for the initial examples, since they are the simplest and most readable of all the notations.**

# Existential Graphs Without Negation



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

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

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

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

# FLIPP Diagrams

Readable diagrams for expressing complex Boolean combinations.

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

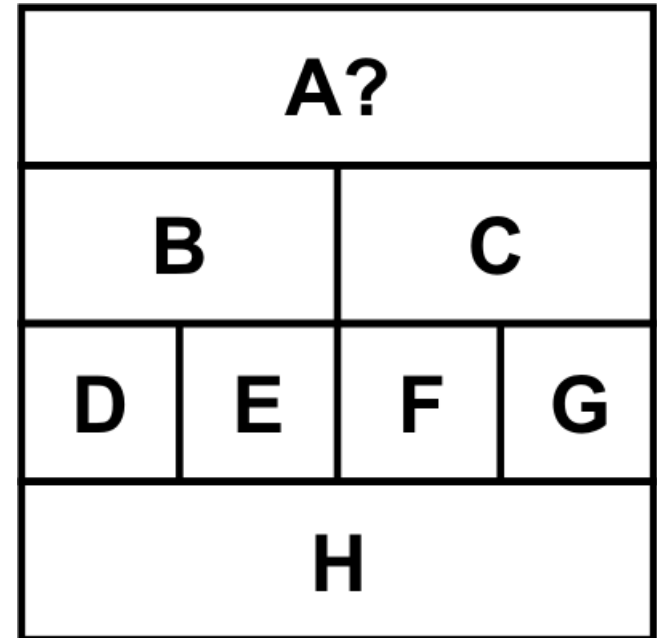
A box that contains a question begins an if-then-else statement or a case statement.

A box on top of another box represents conjunction.

Two or more adjacent boxes represent disjunctions.

The diagram on the right represents the following formula:

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



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

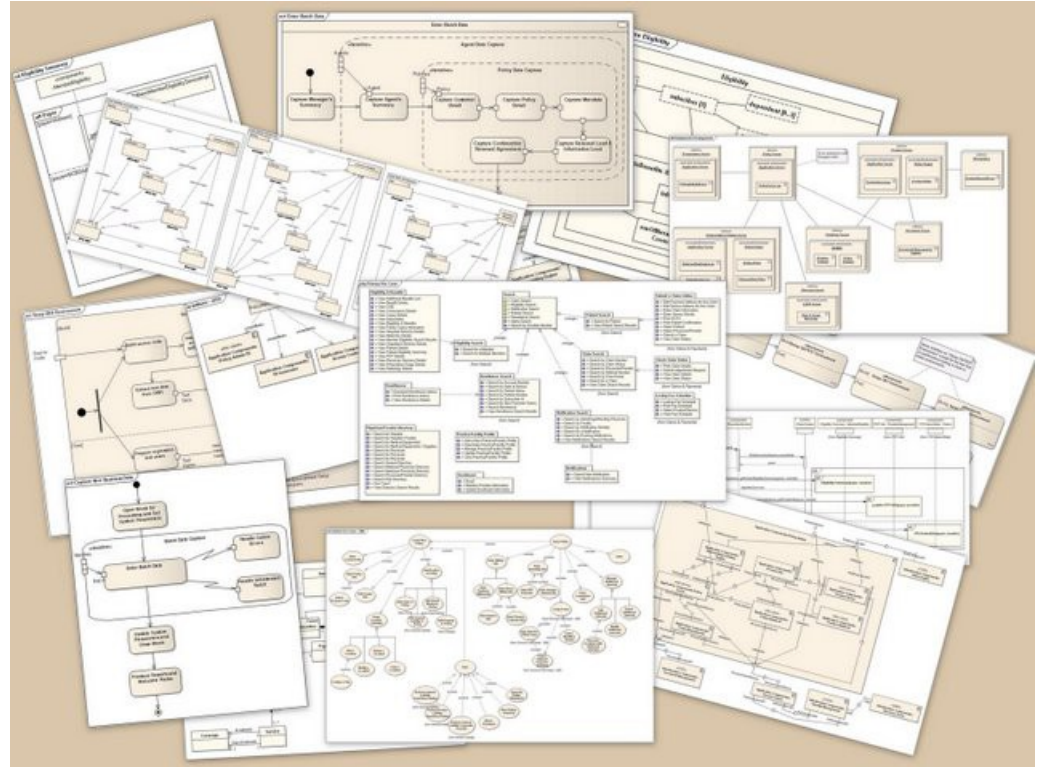
# Unified Modeling Language (UML)

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

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

Originally specified as informal notations without a foundation in logic.

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



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

# An Example of OWL

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

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

# Translation to OWL-CL

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

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

**(= ChildOfUSCitizenPost1955  
  (And (AllVals parentOf (Valuels isCitizenOf USA))  
      (AllVals dateOfBirth YearsSince1955) )**

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

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

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

# Translation to Controlled English

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

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

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

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

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

# Translating OWL to ACE

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

**The ACE “verbalization” of a hundred-line OWL 2 ontology:**

**Everything that is eaten by a goat is a leaf.**

**Everything that eats nothing but leaves is a goat.**

**Every animal is something that is a cat or that is a goat.**

**John is a man.**

**Everything is eaten by at most 1 thing.**

**Everything that is eaten by something is a food that is not an automobile.**

**Everything that is an apple or that is a leaf is a food.**

**Every human is something that is John or that is Mary.**

**Every man is a person.**

**Everything eats at most 1 thing.**

**Every human is a person that own an automobile.**

**If X eats something that eats Y then X eats Y.**

**Everything that eats something is an animal.**

**If X eats Y then Y hate X.**

**If X hate Y then Y eats X.**



# Existential Graphs

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:* Cat——Mat

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

# Boolean Combinations

Areas nested inside an odd number of negations are shaded.

p q

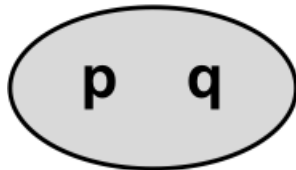
*p and q*



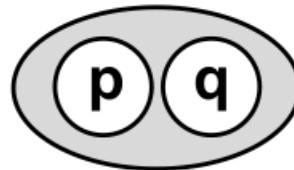
*not p and not q*



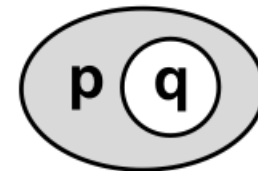
*p and not q*



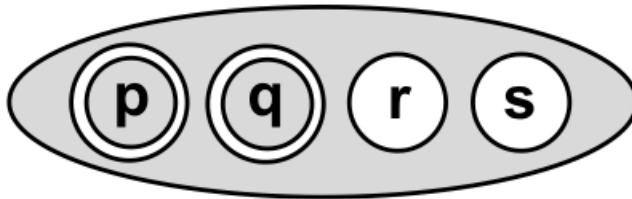
*not (p and q)*



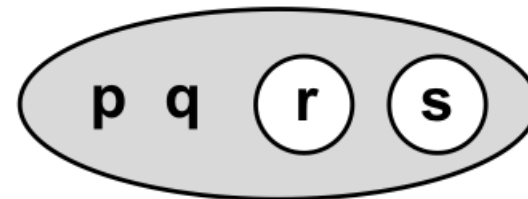
*p or q*



*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.

Cat—Black

*Some cat is black.*

Cat—Black

*Some cat is not black.*

Cat—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.*

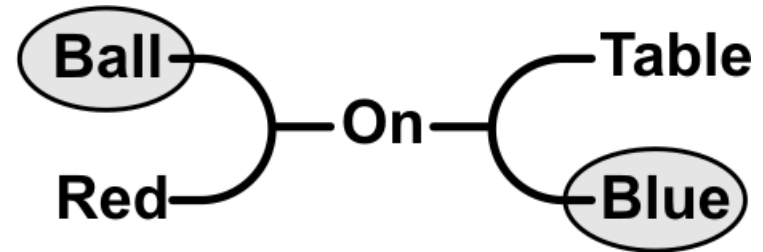
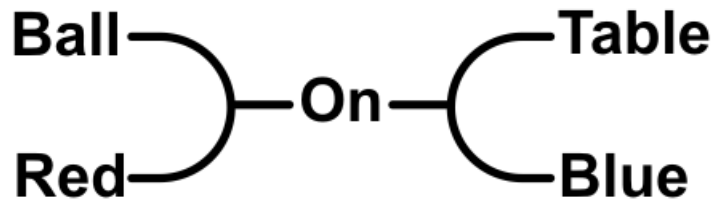
*No cat is not black.*

Cat—Black

These four sentences are synonymous (logically equivalent).

# Translating EGs to and from English

Most existential graphs can be read in several equivalent ways.



**Left graph:**

*A red ball is on a blue table.*

*Some ball that is red is on some table that is blue.*

**Right graph:**

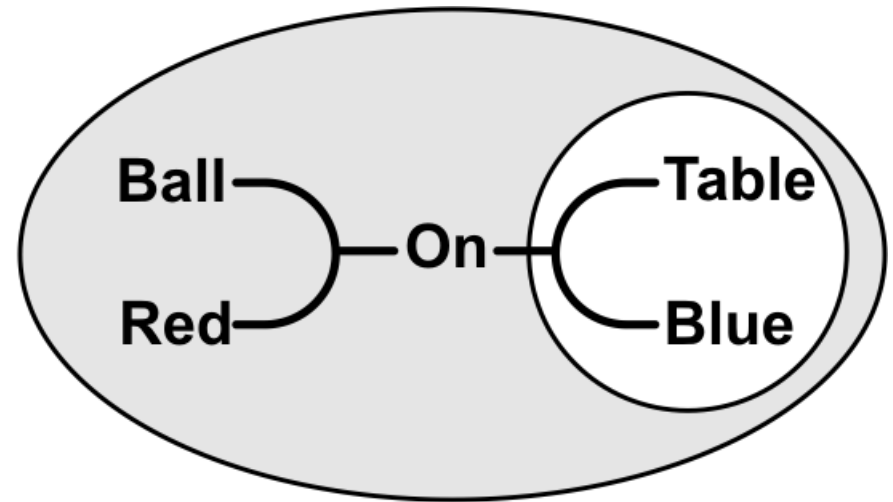
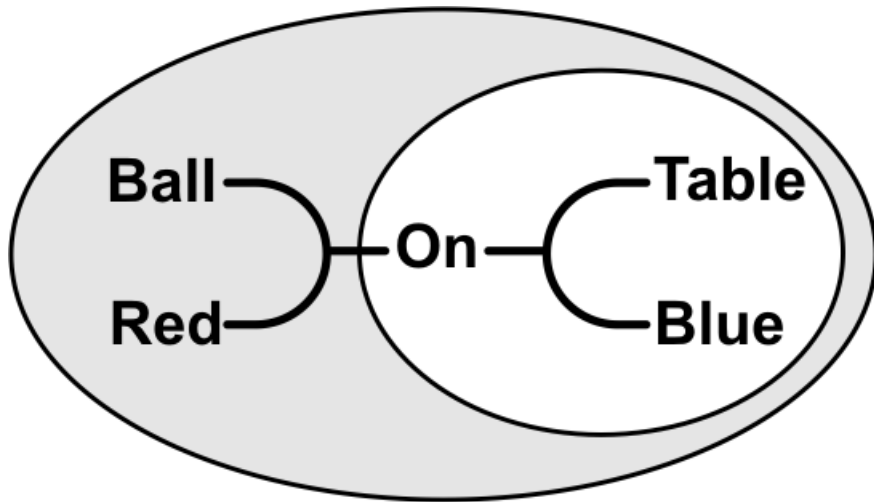
*Something red that is not a ball is on a table that is not blue.*

*A red non-ball is on a non-blue table.*

*On some non-blue table, there is something red that is not a ball.*

# Scope of Quantifiers and Negations

Ovals define the scope for both quantifiers and negations.



**Left graph:**

*If there is a red ball, then it is on a blue table.*

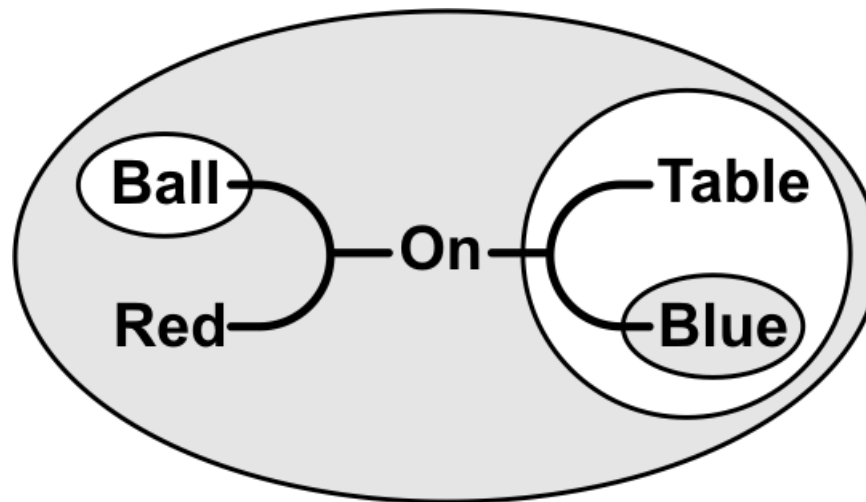
*Every red ball is on some blue table.*

**Right graph:**

*If a red ball is on something  $x$ , then  $x$  is a blue table.*

# Multiple Nested Negations

Complex patterns of negations create more variations.



*If something red that is not a ball is on something y,  
then y is a table that is not blue.*


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

*If a red thing x is on something that is not a non-blue table,  
then x is ball.*

# Predicate Calculus

The widely used Peirce-Peano algebraic notation:

Existence:  $\text{—}$   $(\exists x)$

Negation:   $\sim( )$

Relations:  $\text{Cat}(x)$   $\text{On}(x,y)$   $\text{Under}(x,y)$   $\text{Mat}(y)$

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

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

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

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

# 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 adds features (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)

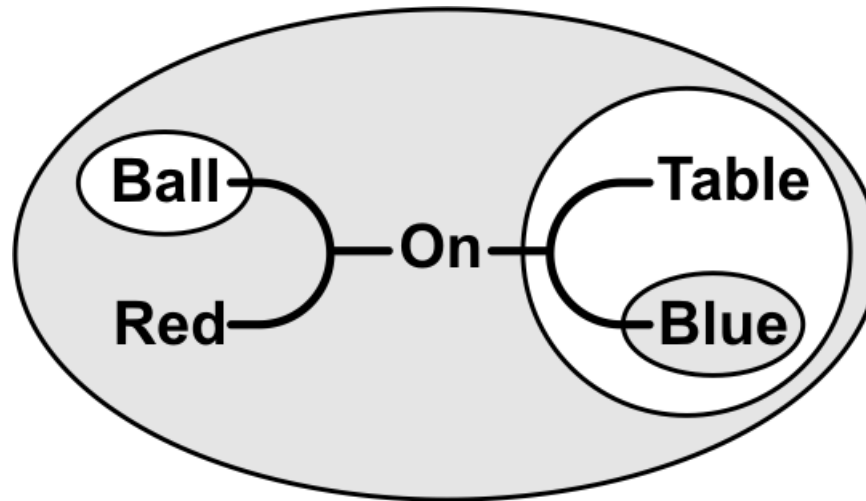
*A cat is on a mat:* [\*x] [\*y] (Cat ?x) (On ?x ?y) (Mat ?y)

*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)]

# Mapping EG to CGIF



*If something red that is not a ball is on something y, then y is a table that is not blue.*

$\sim[[[*x] [*y] (\text{Red } ?x) \sim[(\text{Ball } ?x)] (\text{On } ?x ?y) \sim[(\text{Table } ?y) \sim[(\text{Blue } ?y)]]]]]$

**Note the one-to-one mapping of EG features to CGIF features:**

- Two “ligatures” of connected lines map to  $[*x]$  and  $[*y]$ .
- Four ovals map to four negations, represented as  $\sim[ ]$ .
- Five EG relation names map to five CGIF relation names.
- Six connections of lines to relations map to six occurrences of  $?x$  or  $?y$ .

# Coreference Nodes in CGIF

Coreference nodes show how lines are extended into a nested area or how they are connected to form ligatures:

- **Defining node:**  $[*x]$  represents the start of a line.
- **Extension node:**  $[?x]$  shows an extension of the line  $x$ .
- **Identity:**  $[?x ?y]$  shows that lines  $x$  and  $y$  are connected to form a ligature; it corresponds to an equality  $x=y$ .
- **Teridentity:**  $[?x ?y ?z]$  shows a connection of lines  $x$ ,  $y$ , and  $z$ ; it corresponds to a pair of equalities,  $x=y$  and  $y=z$ .
- **Coreference nodes may connect any number of lines.**

With coreference nodes, CGIF can show how each ligature is formed by connecting lines.

Rules of inference can simplify the CGIF while preserving truth.

# Common Logic Interchange Format

Another standard dialect of Common Logic:

**Existence:**  $\text{—}$  (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)))

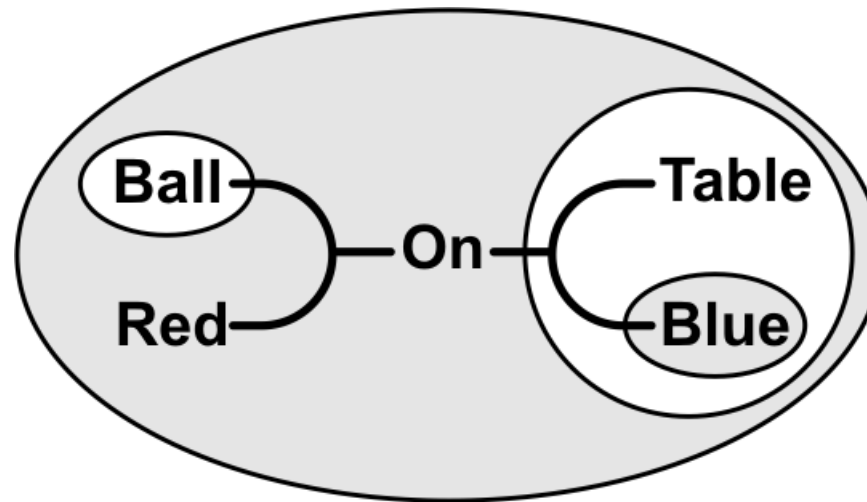
*Some cat is not on a mat:*

(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))))

# Mapping to CLIF and Predicate Calculus



## CGIF:

$\sim[[*x] [*y] (\text{Red } ?x) \sim[(\text{Ball } ?x)] (\text{On } ?x ?y) \sim[(\text{Table } ?y) \sim[(\text{Blue } ?y)]]]$

## CLIF:

$(\text{exists } (x \ y) (\text{and } (\text{Red } x) (\text{not } (\text{Ball } x)) (\text{On } x \ y) (\text{not } (\text{and } (\text{Table } y) (\text{not } (\text{Blue } y))))))$

## Predicate calculus:

$\sim(\exists x)(\exists y)(\text{Red}(x) \wedge \sim\text{Ball}(x) \wedge \text{On}(x,y) \wedge \sim(\text{Table}(y) \wedge \sim\text{Blue}(y)))$

# 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 ( $\forall x$ ), which may be read as 'for every x' or 'for all x', can be defined by an equivalence:

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

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 8.

# Type Constraints

**In an untyped version of logic: any quantifier, such as  $(\exists x)$  or  $(\forall x)$ , can range over anything in the universe.**

**In a typed logic, quantifiers are restricted to a limited domain:  $(\exists x:\text{Cat})$  restricts the variable  $x$  to entities of type Cat.**

**Two conventions for treating type mismatches:**

- 1. Strong typing: A type mismatch causes a syntax error.**
- 2. Weak typing: A type mismatch causes the expression to be false, but it does not create a syntax error.**

**Existential graphs are an untyped version of logic.**

**The base version of Common Logic is untyped.**

**The extended version of Common Logic supports weak typing by monadic relations that restrict the domain of quantifiers.**

**Any typed statement in Common Logic can be converted to a semantically equivalent untyped statement.**

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

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

**Common Logic Interchange Format (CLIF):**

Untyped:  $(\text{exists } (x \ y) (\text{and } (\text{Cat } x) (\text{Mat } y) (\text{On } x \ y)))$

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

**Conceptual Graph Interchange Format (CGIF):**

Untyped:  $[*x] [*y] (\text{Cat } ?x) (\text{Mat } ?y) (\text{On } ?x ?y)$

Typed:  $[\text{Cat } *x] [\text{Mat } *x] (\text{On } ?x ?y)$

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.

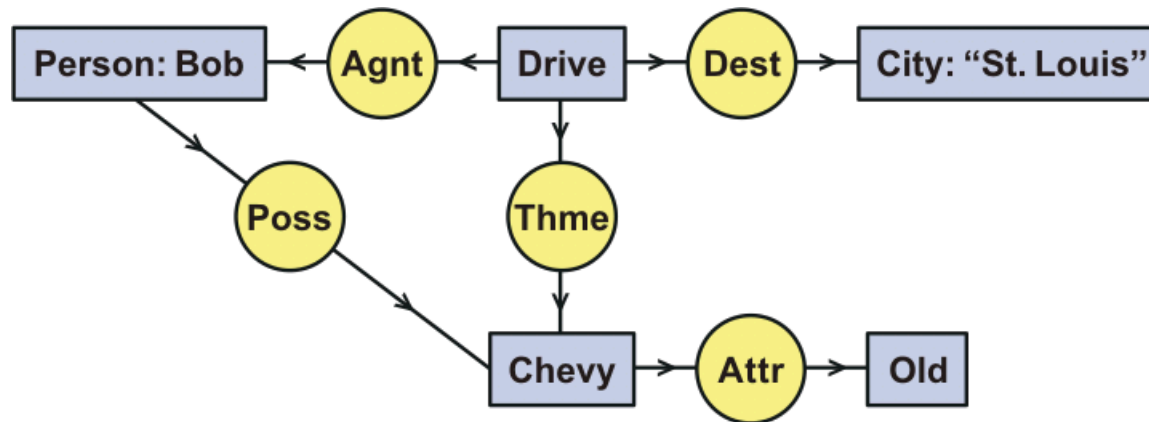
To avoid ambiguous pronouns, CLCE supports temporary names, which look like variables in predicate calculus:

*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):**

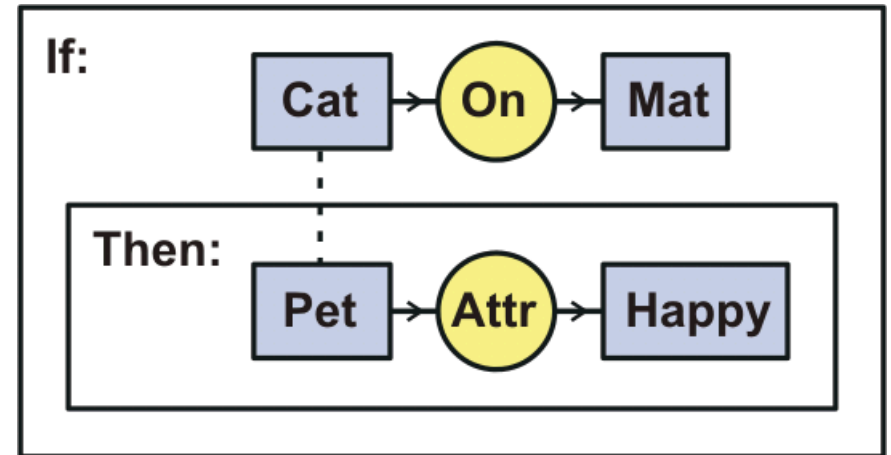
[Drive \*x] [Person Bob] [City "St. Louis"] [Chevy \*y] [Old \*z]  
(Agnt ?x Bob) (Dest ?x "St. Louis")  
(Thme ?x ?y) (Poss Bob ?y) (Attr ?y ?z)

**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:**

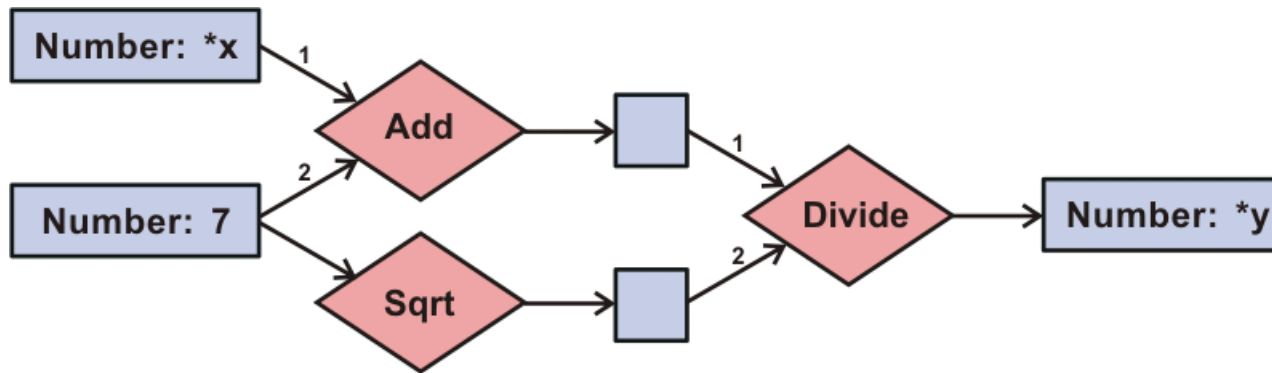
(forall ((x Cat) (y Mat))  
(if (On x y) (and (Pet x) (exists ((z Happy)) (Attr x z))))))

**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  $((x+7) / \text{sqrt}(7))$ .*

**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

**DoD-sponsored project: 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  $\rightarrow$  IKL  $\rightarrow$  CycL**
- **RPI / Booz-Allen: Multi-Sorted Logic  $\rightarrow$  IKL  $\rightarrow$  MSL**
- **Stanford/IBM/Battelle: KIF  $\rightarrow$  IKL  $\rightarrow$  KIF**
- **KIF  $\rightarrow$  IKL  $\rightarrow$  CycL  $\rightarrow$  IKL  $\rightarrow$  MSL  $\rightarrow$  IKL  $\rightarrow$  KIF**

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

# 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 a collection of documents about IKL, the IKRIS project, and their relationships to ontology, logic, and the Semantic Web, see <http://www.jfsowa.com/ikl>**

# 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))))))**

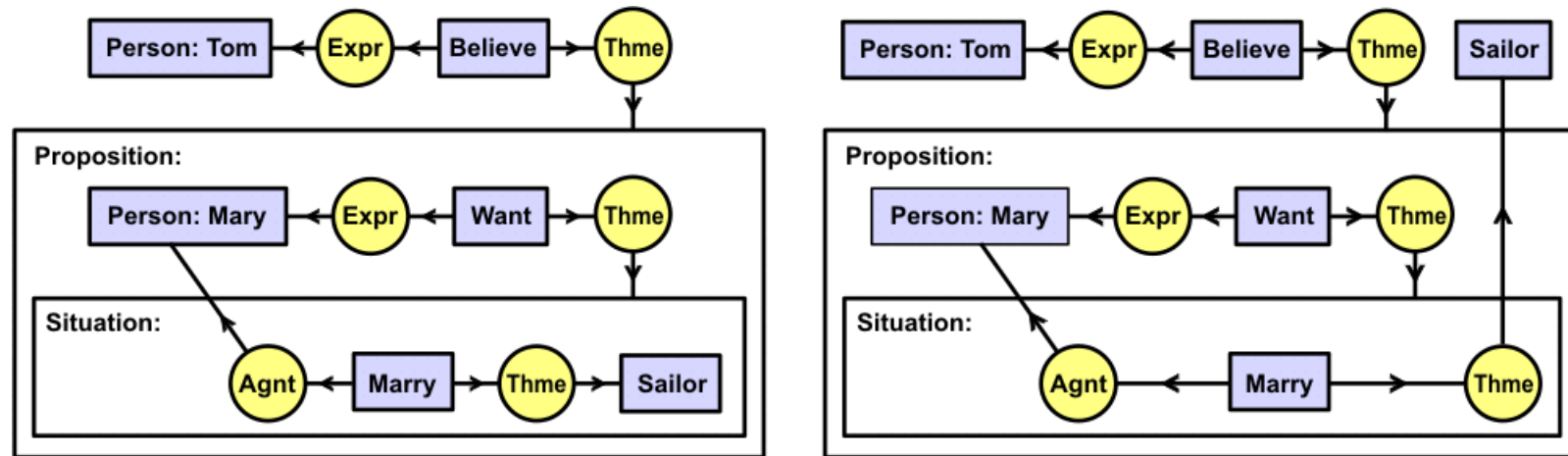
The operator 'that' is a powerful metalevel extension.

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, including logicians, can appreciate.

# Enclosures for Metalinguage



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

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

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

For further discussion, <http://www.jfsowa.com/pubs/eg2cg.pdf>

# Representing IKL in CLIF and CGIF

Following is the CGIF representation for the CG on the left 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)]))])]
```

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 of the previous slide, move the concept node [Sailor: \*x4] in front of the concept [Person: Tom] for CGIF notation. For CLIF, move (x4 Sailor) in front of (x1 Believe).

# Semantics for Modal Logic

**Saul Kripke: Possible worlds with an accessibility relation.**

- A proposition  $p$  is necessary in a world  $w$  iff  $p$  is true in every world accessible from  $w$ .
- $p$  is possible in  $w$  iff  $p$  is true in some world accessible from  $w$ .

**Michael Dunn: Each world is represented by laws and facts.**

- $p$  is necessary in  $w$  iff  $p$  is provable from the laws of  $w$ .
- $p$  is possible in  $w$  iff  $p$  is consistent with the laws of  $w$ .

**Kripke semantics and Dunn semantics are equivalent.**

- Accessibility is a derived relation from the laws and facts.
- A world  $w_2$  is accessible from  $w_1$  iff every law of  $w_1$  remains true in  $w_2$  (but some laws of  $w_1$  might only be facts in  $w_2$ ).

**Although both versions of semantics are logically equivalent, Dunn's version is easier to map to computer implementations.**

For further detail and references, see <http://www.jfsowa.com/pubs/laws.htm> and <http://www.jfsowa.com/pubs/worlds.pdf>

# Using the Metalanguage Option of IKL

**Modal terminology is used to distinguish laws from facts.**

- **Unqualified assertions are assumed to be facts.**
- **Words 'necessary', 'obligatory', or 'must' indicate laws.**

**Default logic is a metalevel specification for a family of theories.**

- **Ordinary axioms specify a base theory.**
- **Each default specifies an optional axiom added to the base theory.**
- **A nonmonotonic proof chooses one theory from the entire family.**

**Uncertainty and fuzziness are also metalevel statements.**

- **They define numeric measures over a family of theories.**

**Context theories also use metalevel markers and reasoning.**

**IKL, by itself, does not define any of these systems, but it provides the framework and primitives for stating such definitions.**

# The Development of Common Logic

**1992: ANSI projects for the Knowledge Interface Format (KIF) and Conceptual Graphs (CGs) started in the X3H4 committee.**

**1994: X3T2 inherits KIF and CG projects; sponsors workshops on ontology for an ISO project on Conceptual Schemas.**

**1999: Conceptual Schema project ends with a technical report.**

**2000: NCITS L8 merges KIF and CG projects in a proposal to SC32 for an ISO standard for Common Logic.**

**2001: Pat Hayes and Chris Menzel propose a new semantic foundation for Common Logic that is compatible with RDF.**

**2002: Guha and Hayes use the CL semantics to define the logic base (LBase) for RDF.**

**2007: ISO/IEC standard 24707 for Common Logic is approved.**

**2015: Target date for updates and extensions for a revised version of the CL standard.**



# A Migration Path to the Future

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

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

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

**Recommendation for a new generation of development tools:**

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