Introduction to Common Logic

John F. Sowa

10 January 2011

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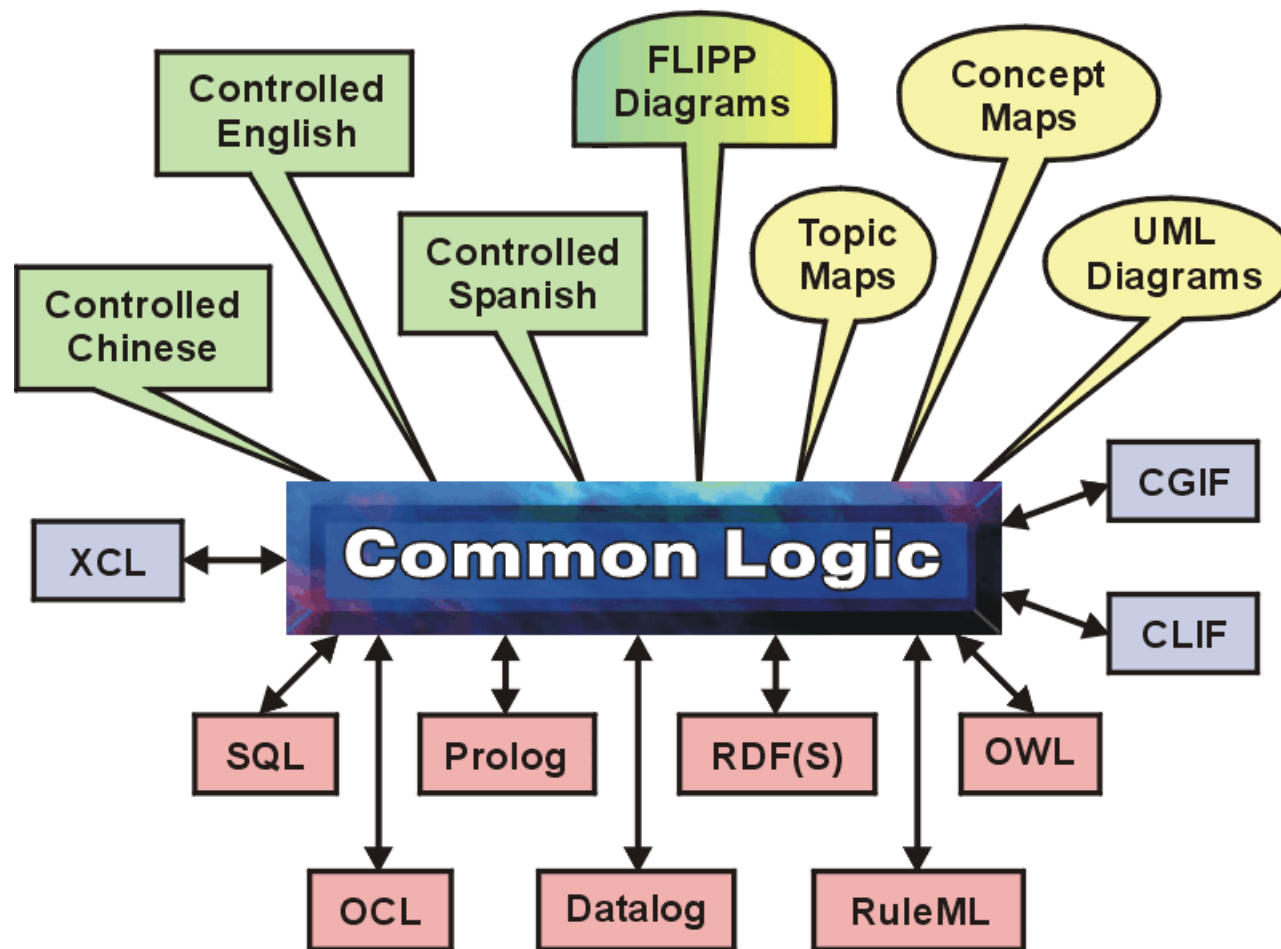
The ISO standard 24707 for Common Logic defines a very general semantic foundation for an open-ended family of dialects.

Three normative dialects are specified in ISO 24707:
- CLIF — Common Logic Interchange Format
- CGIF — Conceptual Graph Interchange Format
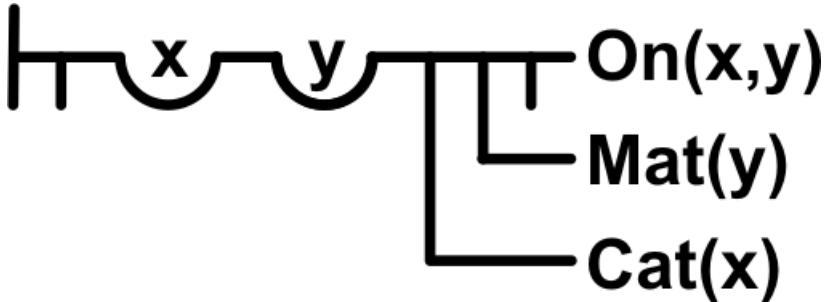- XCL — XML-based notation for Common Logic

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

Human Interfaces

Controlled English

Controlled Chinese

Controlled Spanish

FLIPP Diagrams

Topic Maps

Concept Maps

UML Diagrams

XCL

Common Logic

CGIF

CLIF

SQL

Prolog

RDF(S)

OWL

OCL

Datalog

RuleML

Machine Interfaces

How to say "A cat is on a mat."

Gottlob Frege (1879):

Charles Sanders Peirce (1885):

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

Giuseppe Peano (1895):

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

Charles Sanders Peirce (1897):

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

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):

(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.

Some Dialects of Common Logic

This tutorial illustrates the use of Common Logic as a foundation for relating multiple notations or dialects of 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

A graph notation for logic with a minimum of primitives:

Existence: ——

Negation: (oval)

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

A cat is on a mat: Cat——On——Mat

Something is under a mat: ——Under——Mat

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

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

Boolean Combinations

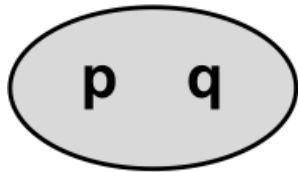**Areas nested inside an odd number of negations are shaded.**

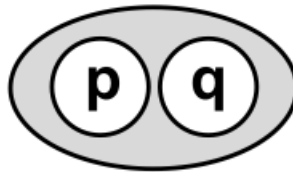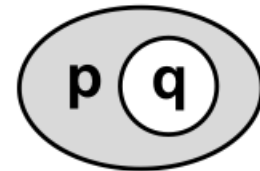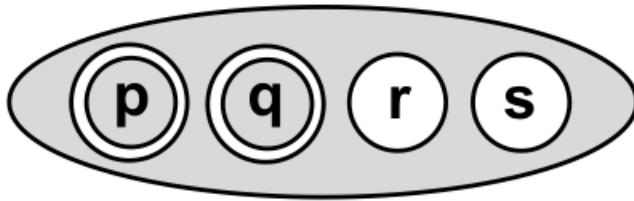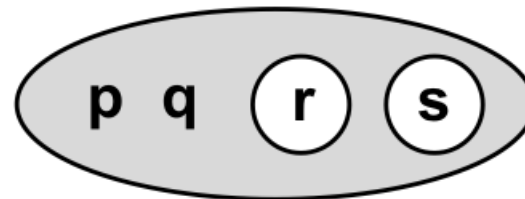# Representing Quantifiers

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.
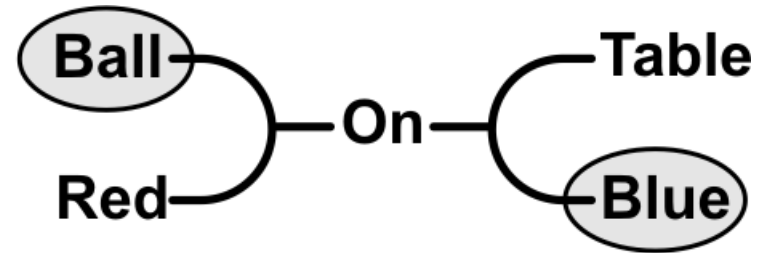
Every cat is black.

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.
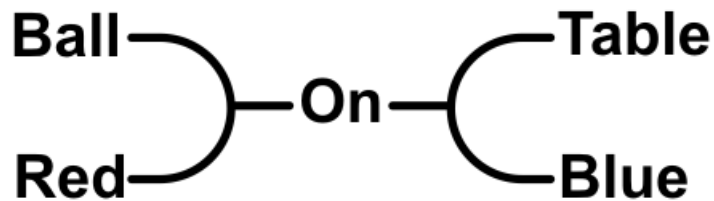
Left graph:

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.
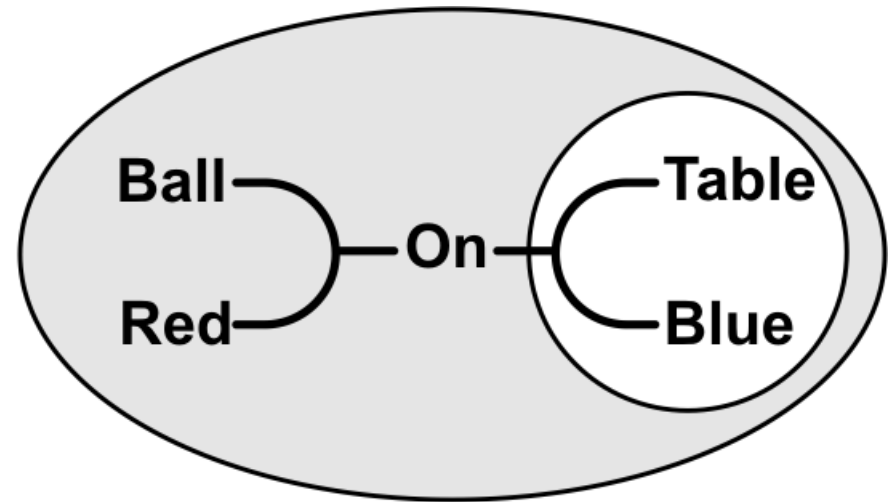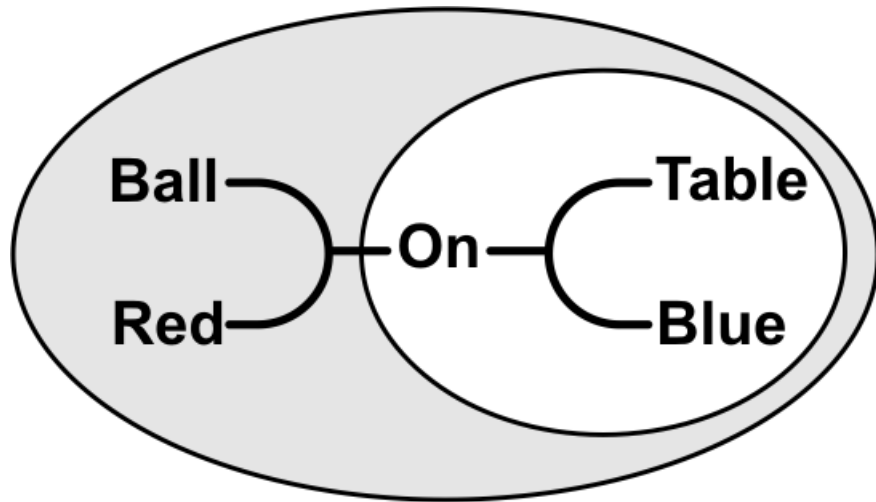
Left graph:

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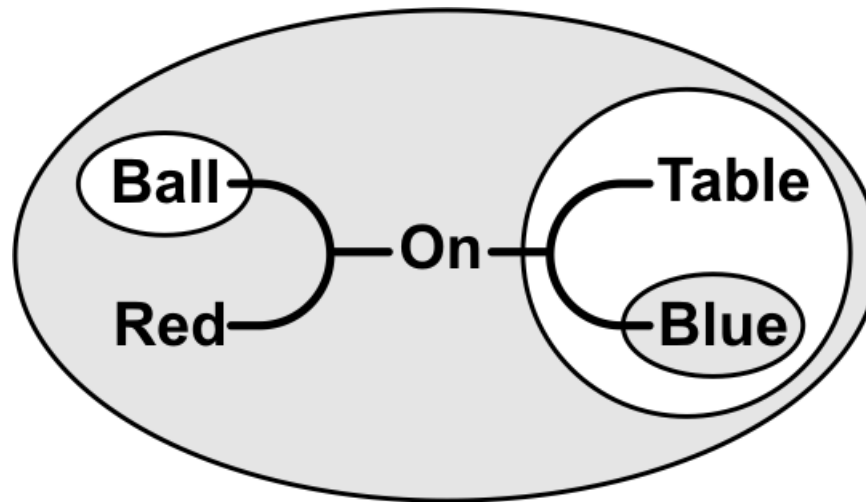
*If something red that is not a ball is on something y,
then y is a table that is not blue.*

*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: —   ($\exists$x)

Negation: ⬭   ~( )

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

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

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

Some cat is not on a mat: ($\exists$x)(Cat(x) $\wedge$ ~($\exists$y)(On(x,y) $\wedge$ Mat(y)))

Some cat is on something that is not a mat:
                    ($\exists$x)($\exists$y)(Cat(x) $\wedge$ On(x,y) $\wedge$ ~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 an EG to CGIF
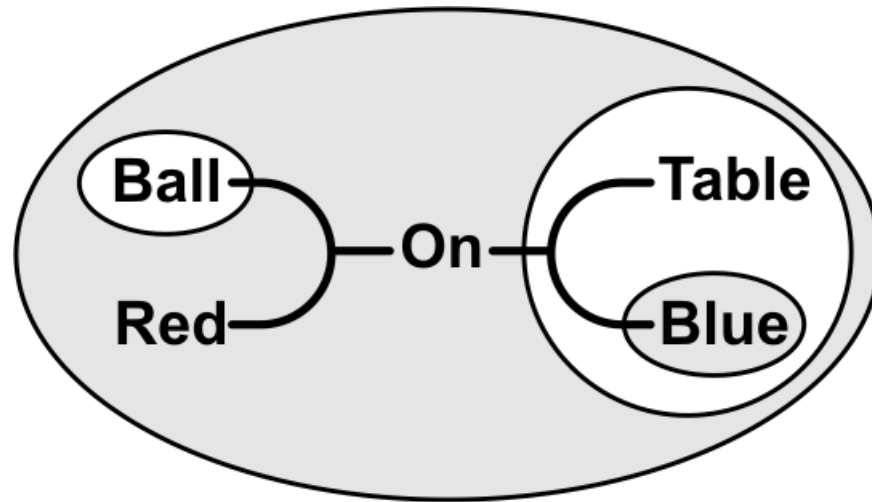
If something red that is not a ball is on something y, then y is a table that is not blue.

~[[*x] [*y] (Red ?x) ~[(Ball ?x)] (On ?x ?y) ~[(Table ?y) ~[(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 ~[ ].**
- **Five EG relation names map to five CGIF relation names.**
- **Six connections of lines to relations map to six occurrences of ?x or ?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))))

Mapping to CLIF and Predicate Calculus
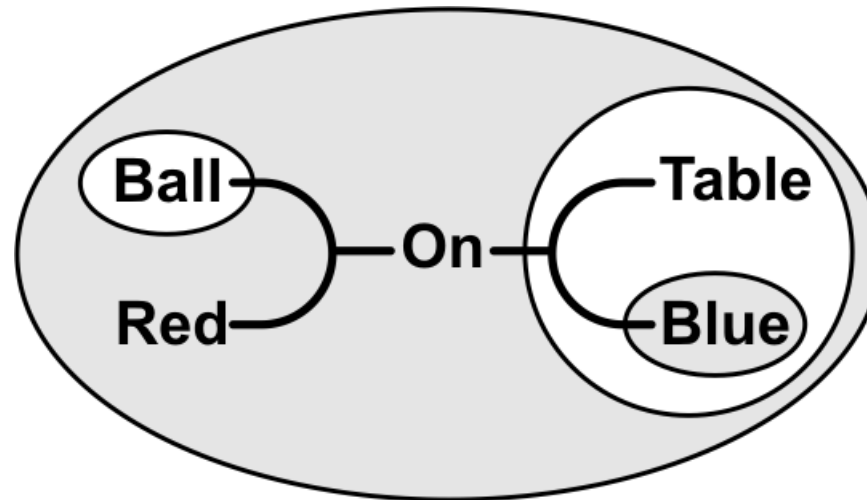
CGIF:

~[[*x] [*y] (Red ?x) ~[(Ball ?x)] (On ?x ?y) ~[(Table ?y) ~[(Blue ?y)]]]

CLIF:

(exists (x y) (and (Red x) (not (Ball x)) (On x y) (not (and (Table y) (not (Blue y)))))

Predicate calculus:

~(∃x)(∃y)(Red(x) ∧ ~Ball(x) ∧ On(x,y) ∧ ~(Table(y) ∧ ~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:

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

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 8.

Type Constraints

In an untyped version of logic: any quantifier, such as (∃x) or (∀x), can range over anything in the universe.

In a typed logic, quantifiers are restricted to a limited domain: (∃x: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: (∃x)(∃y)(Cat(x) ∧ Mat(y) ∧ On(x,y))

Typed: (∃x:Cat)(∃y:Mat)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.
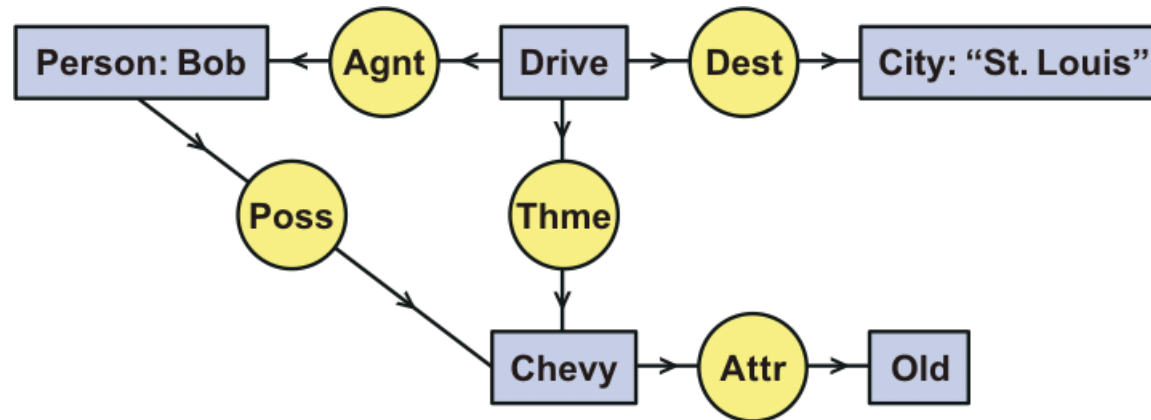
To avoid ambiguous pronouns, CLCE supports temporary names, which look like variables in predicate calculus:

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:
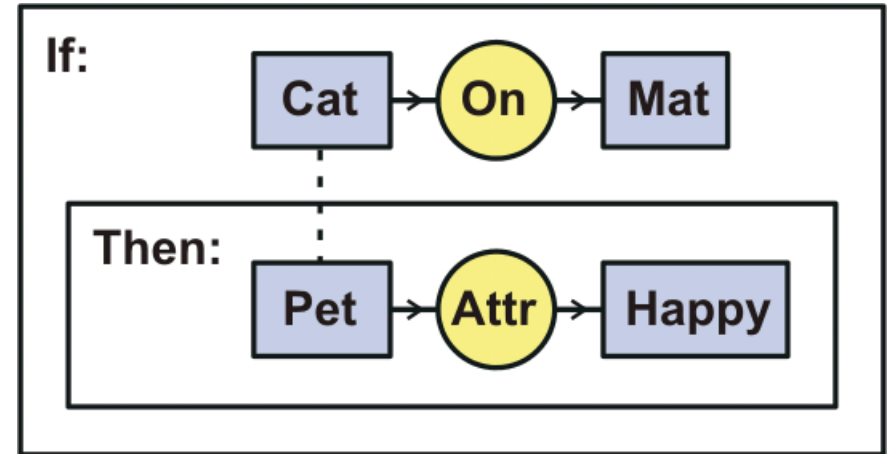
CGIF:

    [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:
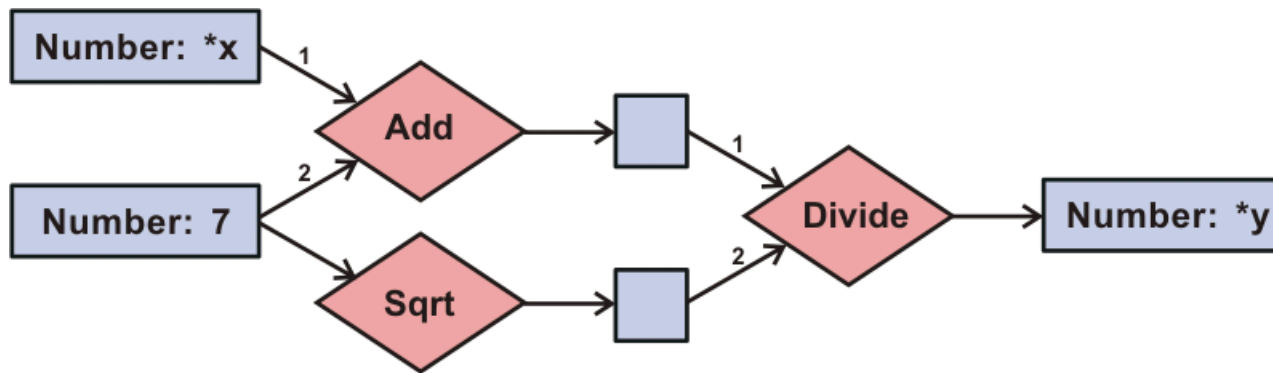
CGIF:

   [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

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 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, including logicians, can appreciate.

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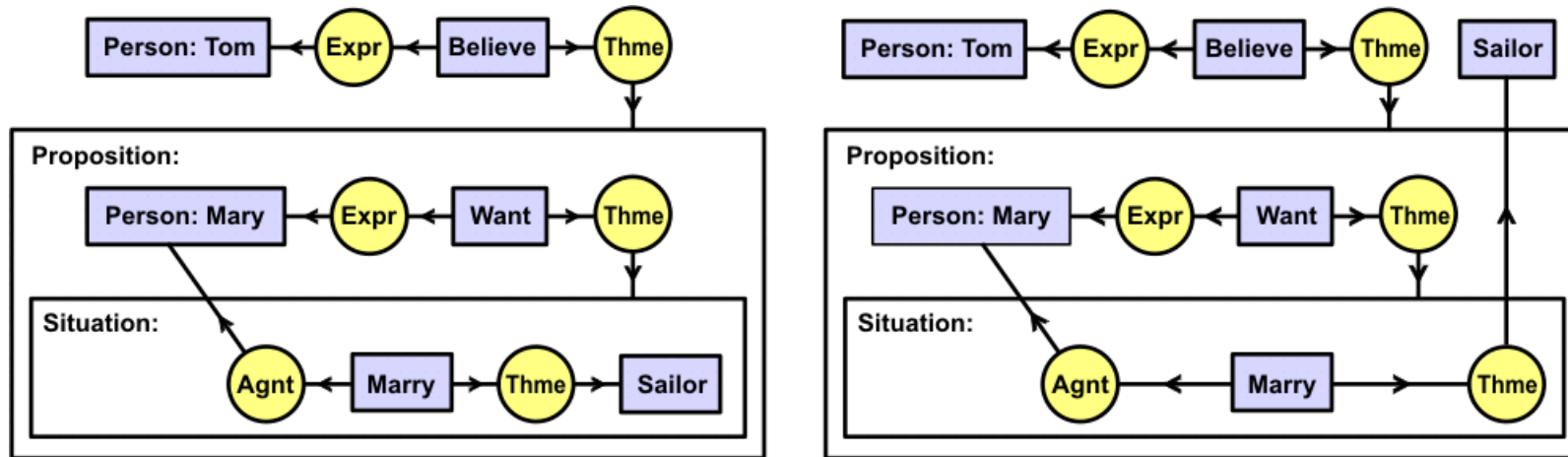
The two CGs above show two different interpretations of the sentence *Tom believes that Mary wants to marry a sailor:*

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

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

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

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

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 w2 is accessible from w1 iff every law of w1 remains true in w2 (but some laws of w1 might only be facts in w2).

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.

2014: Target date for updates and extensions for a revised version of the CL standard.

Related Readings

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

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

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

Web site for Common Logic:
http://www.common-logic.org

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