

# From Existential Graphs to Conceptual Graphs

John F. Sowa

VivoMind Research, LLC

**Abstract.** Existential graphs (EGs) are a simple, readable, and expressive graphic notation for logic. Conceptual graphs (CGs) combine a logical foundation based on EGs with features of the semantic networks used in artificial intelligence and computational linguistics. CG design principles address logical, linguistic, and cognitive requirements: a formal semantics defined by the ISO standard for Common Logic; the flexibility to support the expressiveness, context dependencies, and metalevel commentary of natural language; and cognitively realistic operations for reasoning by induction, deduction, abduction, and analogy. To accommodate the vagueness and ambiguities of natural language, informal heuristics can supplement the formal semantics. With sufficient background knowledge and a clarifying dialog, informal graphs can be refined to any degree of precision. Peirce claimed that the rules for reasoning with EGs generate “a moving picture of the action of the mind in thought.” Some philosophers and psychologists agree: Peirce’s diagrams and rules are a good candidate for a *natural logic* that reflects the neural processes that support thought and language. They are psychologically realistic and computationally efficient.

This is a revised preprint of an article that appeared in the *International Journal of Conceptual Structures*, vol 1, no 1. Some additional references and URLs are added at the end.

## 1. Languages and Diagrams for Logic

Existential graphs and the conceptual graphs based on them are formally defined, but they follow the long tradition of deriving logical patterns from language patterns. For the first formal logic, Aristotle developed a stylized or controlled version of natural language (NL). To clarify the references and reduce ambiguity, he replaced pronouns with letters. For his *Elements of Geometry*, Euclid followed Aristotle’s conventions as far as he could. When he needed more expressive power, Euclid added diagrams and a broader range of language patterns. Controlled Greek was the first CNL, but logicians and mathematicians translated it to controlled Latin, Arabic, and other languages. Over the centuries, they abbreviated words and phrases with various symbols and organized them in diagrams. The plus sign +, for example, is a simplified ampersand &, which abbreviated a hand-written *et* in Latin. The oldest surviving type hierarchy is the Tree of Porphyry from the third century AD.

In the 19th century, George Boole (1854) presented his *laws of thought* as an algebra for propositional logic: 1 for truth; 0 for falsehood; + for *or*;  $\times$  for *and*; and  $-$  for *not*. Thirty years later, Gottlob Frege and Charles Sanders Peirce independently developed notations for first-order logic (FOL). Frege (1879) invented tree diagrams for representing FOL, but nobody else adopted his notation. Peirce added *n*-adic relations to Boolean algebra in 1870, introduced quantifiers in 1880, and extended the algebraic notation to both first-order and higher-order logic in 1885. Giuseppe Peano (1889) adopted Peirce’s algebra and changed some of the symbols to create the modern notation for predicate calculus. But in 1896, Peirce invented *existential graphs* (EGs) as a more *diagrammatic* notation for “the atoms and molecules of logic.”

As an example, the English sentence *A cat is on a mat* and a controlled English version would be identical. Since Boolean algebra cannot represent the details of relations and quantifiers, it can only represent the full proposition by a single unanalyzed letter *p*. For his relational algebra of 1870,

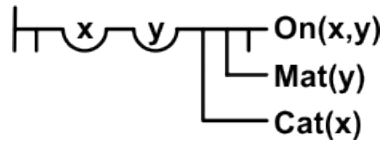
Peirce invented a notation for the details at the word and phrase level:  $Cat_i$  for a cat  $i$ ,  $Mat_j$  for a mat  $j$ , and  $On_{i,j}$  for something  $i$  on something  $j$ . Then the conjunction  $(Cat_i \cdot On_{i,j} \cdot Mat_j)$  can be read as an abbreviation for *A cat  $i$  is on a mat  $j$* , but Peirce did not have a systematic way of handling quantifiers. In 1880, while he was revising his father's book on linear algebra, Peirce noticed that the Greek letters  $\Sigma$  for repeated addition and  $\Pi$  for repeated multiplication could be adapted to logic: the existential quantifier corresponds to repeated *or*, and the universal quantifier corresponds to repeated *and*. With existential quantifiers in front, the following formula represents the sentence *There exists something  $i$ , there exists something  $j$ ,  $i$  is a cat,  $i$  is on  $j$ , and  $j$  is a mat*:

$$\Sigma_i \Sigma_j Cat_i \cdot On_{i,j} \cdot Mat_j$$

Since Peano wanted to mix mathematical and logical symbols in the same formulas, he invented new symbols by turning letters upside down or backwards. He replaced Boole's  $+$  for *or* with  $\vee$  for the Latin *vel*, and he turned  $\vee$  upside down for *and*. For the existential quantifier, he turned E backwards for  $\exists$ . With these symbols, Peano's version of Peirce's formula becomes

$$\exists i \exists j Cat(i) \wedge On(i,j) \wedge Mat(j)$$

The developments from Boole to Peirce to Peano continued the Aristotelian tradition of relating language to logic. But Frege (1879) had a low opinion of natural language. His goal was "to break the domination of the word over the human spirit by laying bare the misconceptions that through the use of language often almost unavoidably arise concerning the relations between concepts." For his *Begriffsschrift* (concept writing), Frege used only the operators that occur in his rules of inference: assertion (vertical line), negation (short vertical line), universal quantifier (cup that contains a variable), and implication (hook). Figure 1 represents *A cat is on a mat* with Frege's operators.



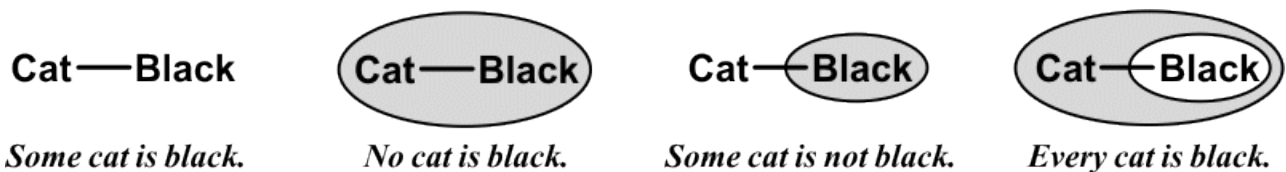
**Figure 1. Begriffsschrift for *A cat is on a mat*.**

At the left of Figure 1, the vertical line asserts the entire diagram. The short vertical lines represent *not*, the cups represent *for every*, and the hooks represent *if-then*. With these operators, the diagram may be read *It is false that for every  $x$ , for every  $y$ , if  $x$  is a cat, then if  $y$  is a mat, then  $x$  is not on  $y$ .*

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

**Figure 2. Existential graph for *A cat is on a mat*.**

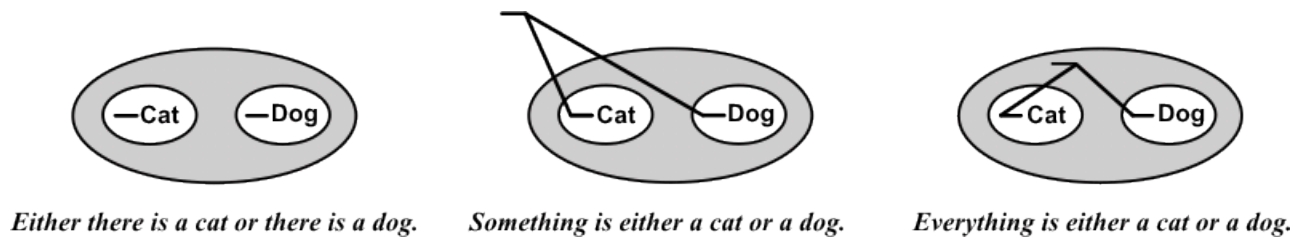
By contrast, Peirce's existential graph in Figure 2 expresses the sentence with a minimum of symbols. The character strings name the relations; the two bars, which Peirce called *lines of identity*, represent existential quantifiers; the *and* operators are implicit. For negation, Peirce used an oval enclosure, as illustrated in Figure 3. A negative area (shaded) is nested in an odd number of negations. A positive area (not shaded) is nested in an even number of negations, possibly zero.



**Figure 3. EGs for the Aristotelian sentence patterns**

Figure 3 shows how EGs represent the four sentence patterns of Aristotelian syllogisms. Without negations, the EG on the left says *Some cat is black*. An oval that negates that EG says *It is false that some cat is black*. This is equivalent to saying *No cat is black*. An oval that negates only the relation Black says *Some cat is not black*. By negating that EG, the rightmost EG says *It is false that some cat is not black*. This is equivalent to saying *No cat is not black* or *Every cat is black*. But a shaded area that contains a nested white area can be read *if-then*. With that option, the rightmost EG can also be read *If there is a cat, then it is black*. As Figure 3 shows, an EG can serve as the *canonical form* that represents the underlying logical structure of one or more English sentences.

When combined in all possible ways, conjunction, negation, and lines of identity can represent all the operators of first-order logic. One more feature is needed to represent full FOL with equality: the option of connecting two or more lines. Figure 4 shows the pattern for *either-or*: two or more white areas nested in a shaded area. The EG on the left has two lines of identity, each nested in one of the alternatives. It says *Either there is a cat or there is a dog*. The EGs in the middle and the right add *ligatures* of connected lines of identity. Each ligature could be analyzed as a connection of five separate lines, each with its own existential quantifier, and all related by equal signs. As a result of those equalities, the only quantifier that determines the meaning is the outermost one. For the middle EG in Figure 4, the line outside the nest says *There is something*. The connections to the inner areas say *It is either a cat or a dog*. For the rightmost EG, the outermost quantifier is in a negative area, where it has the effect of a universal quantifier: *Everything is either a cat or a dog*. That same EG could also be read *If there is something, then either it is a cat or it is a dog*.

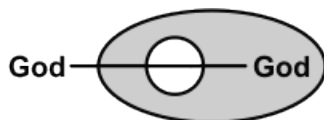


**Figure 4. Ligatures that connect lines of identity**

Different branches of a ligature might represent different individuals. Figure 5 shows a ligature that can be analyzed as three lines of identity. The outermost area, which Peirce called the *sheet of assertion*, contains a line x attached to the monadic relation God. The shaded area contains a line y and another copy of the relation God. The white area nested two levels deep contains a third line z that is connected to lines x and y. Altogether, Figure 5 can be translated to the formula

$$\exists x \text{ God}(x) \wedge \sim(\exists y \text{ God}(y) \wedge \sim(\exists z z=x \wedge z=y)).$$

The expression  $\sim(\exists z z=x \wedge z=y)$  can be simplified to just  $\sim(x=y)$ . With this simplification, the EG can be read *There is a God x, and it is false that there is a God y that is not the same as x*. Since a shaded area with a nested white area can be read *if-then*, the EG can also be read *There is a God x, and if there is a God y, then x is y*. Both sentences are equivalent to saying *There is one and only one God*.



**Figure 5. EG for the sentence *There is one and only one God***

As these examples show, EGs are based on three graphic elements: lines of identity that show existence and connectivity, character strings that name relations, and ovals that negate the enclosed graphs or subgraphs. Conjunction is represented by the convention that two or more graphs or parts

of graphs in the same area are linked by *and*. Equality is represented by connecting two or more lines. No special symbols are needed for other operators because their graphic patterns can be seen directly:

- *If-then*: A shaded area with a nested white area can be seen and read as *if... then....*
- *Either-or*: A shaded area with two or more nested white areas can be seen and read as *either... or... or....*
- *Some and every*: A line with its outermost point in a white area can be read as *some* or *something*. With its outermost point in a shaded area, it can be read as *every* or *everything*.

With practice, even more complex patterns can be seen at a glance. In his original EG notation, Peirce did not use shading. To distinguish positive areas from negative areas, the reader had to count the levels of nesting. He later added shading to heighten the contrast and make the diagrams more *iconic*: each pattern is an icon whose appearance directly shows the logical structure.

Frege's Begriffsschrift also has iconic aspects. In Figure 1, for example, a cup symbol within the scope of an odd number of negations can be read *there exists*. A sequence of hooks preceded by a negation and followed by another negation can be read as repeated *and*. With these conventions, Figure 1 can be read *There exists an x, there exists a y, x is a cat, and y is a mat, and x is on y*. Frege recognized the importance of learning and using such patterns for reasoning, but he never encouraged anyone to read his notation as a controlled NL.

The same notation can be more or less iconic with respect to different structures. Frege designed his notation to support his rules of inference. For the *Principia Mathematica*, Whitehead and Russell adopted Frege's rules, but they used the Peirce-Peano algebra, whose structure is less clearly related to those rules. For that reason, the algebra is less iconic than Begriffsschrift with respect to the rules of inference, but more iconic in its mapping to NLS. For existential graphs, Peirce discovered remarkably simple rules of inference stated in terms of the EG notation. But the EG shading can be used to highlight positive and negative areas in other notations. With shading to make them more iconic, algebraic formulas and even controlled NLS can be processed by Peirce's rules.

Conceptual graphs (CGs) inherit the semantics, operations, and iconic aspects of EGs (Sowa 1984). But they add innovations from the past century of research in cognitive science, especially linguistics and artificial intelligence. The Conceptual Graph Interchange Format (CGIF) is a linearization of CGs specified by the ISO/IEC standard 24707 for Common Logic (CL). But CGIF can also be used as a linearization of EGs and other graphic notations used in computer science and the Semantic Web. Common Logic is a highly expressive extension to FOL that supports quantification over propositions, relations, and functions while retaining a first-order style of proof theory. The Interoperable Knowledge Language (IKL) is an extension of CL that supports metalanguage. It can also be used to define versions of modal, nonmonotonic, and fuzzy logic.

For artificial intelligence, the formal aspects of a notation are the basis for computation. But the major challenge is to support reasoning with and about the informal richness, complexity, and often vagueness of human thought and language. As logics, EGs and CGs are precise, but they can be used for representing and reasoning about NLS at the object level and the metalevel. Metalevel reasoning can support a wide range of formal, informal, and heuristic methods for reasoning about statements at the object level. Even when the object level is as vague or underspecified as an NL sentence, the metalevel can represent and reason about the background knowledge necessary to make the representation as clear and precise as necessary (Majumdar and Sowa 2009).

## 2. Semantic Networks

Existential graphs have structural similarities to the semantic networks of artificial intelligence, the dependency graphs of linguistics, and the *discourse representation structures* by Hans Kamp (1981). Conceptual graphs combine EGs with features from those systems and from related research in logic and linguistics. An important experience in Peirce's career was his work as an associate editor of the *Century Dictionary*. He wrote, revised, or edited over 16,000 definitions — more than any other editor of that dictionary. A letter he wrote to the general editor, B. E. Smith, shows how that experience affected his thinking:

The task of classifying all the words of language, or what's the same thing, all the ideas that seek expression, is the most stupendous of logical tasks. Anybody but the most accomplished logician must break down in it utterly; and even for the strongest man, it is the severest possible tax on the logical equipment and faculty.

Peirce scholars believe that the two decades after his work on the *Century Dictionary* were his most profound and fertile. During that period, he invented existential graphs and used them to anticipate and often improve on theories by later logicians, linguists, and philosophers (Houser 2005; Sowa 2006a).

The *dependency grammar* by Lucien Tesnière (1959) was widely adopted in AI and computational linguistics. Figure 6 shows Tesnière's graph for an epigram by Voltaire about a certain literary critic: *L'autre jour, au fond d'un vallon, un serpent piqua Jean Fréron* (The other day, at the bottom of a valley, a snake stung Jean Fréron). At the top is the verb *piqua* (stung); each word below it depends on the word to which it's attached. The bull's eye symbol indicates an implicit preposition (*à*). The conclusion of the epigram, which Tesnière also analyzed, was *Que pensez-vous qu'il arriva? Ce fut le serpent qui creva* (What do you think happened? It was the snake that died).

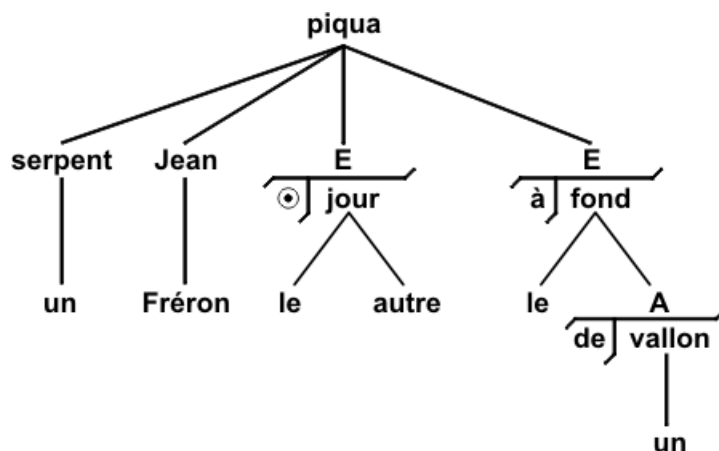
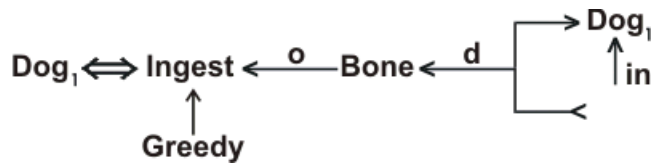


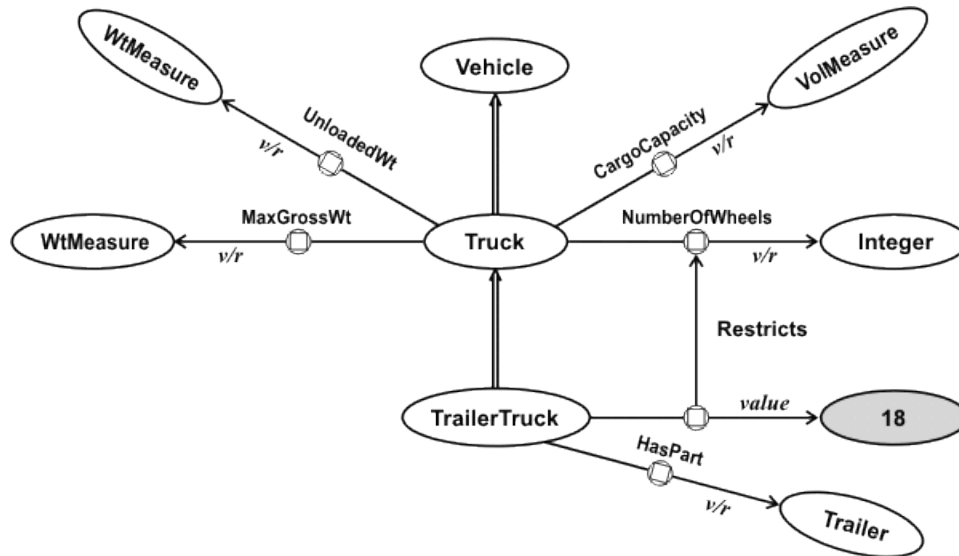
Figure 6. A dependency graph in Tesnière's notation

In Tesnière's theory, *un serpent* and *Jean Fréron* represent participants that are essential to the action. The prepositional phrases specify the circumstances of the surrounding situation. Dependency graphs can represent the same sentences as context-free grammars (Hays 1964), but their structure has a more direct mapping to the semantic networks of AI. Roger Schank (1975) adopted dependency graphs, but emphasized concepts rather than words. Figure 7 shows a *conceptual dependency graph* for the sentence *A dog is greedily eating a bone*. Instead of Tesnière's trees, Schank used graphs with various kinds of arrows:  $\Leftrightarrow$  for the agent-act relation and an arrow marked with *o* for object or *d* for direction. He also replaced the words *eat* and *greedily* with labels for the concept types *Ingest* and *Greedy*. The subscript on *Dog*<sub>1</sub> indicates *coreference*: the bone went into the same dog that ingested it.



**Figure 7. A Schankian conceptual dependency graph**

Although the early semantic networks were useful for computation, many of them did not have a formally defined semantics. Woods (1975) proposed an important family of logic-based networks that evolved into *description logics* (DLs). Brachman (1979) implemented the first version, called Knowledge Language One (KL-ONE). As an example, Figure 8 shows a KL-ONE network that defines the concepts Truck and Trailer Truck as subtypes of Vehicle.



**Figure 8. Truck and Trailer Truck concepts defined in KL-ONE**

The double arrows in the middle of Figure 8 show a branch of a concept hierarchy with Vehicle as a supertype of Truck, which is a supertype of Trailer Truck. The concept Truck is defined as a subtype of Vehicle with the roles Unloaded Weight, Maximum Gross Weight, Cargo Capacity, and Number Of Wheels. Each of those roles has a value restriction ( $v/r$ ), which may be a primitive type, such as Integer, or a type defined elsewhere in the hierarchy, such as Weight Measure or Volume Measure. The concept Trailer Truck is a subtype of Truck, which inherits all the roles of Truck and adds a new role, Has Part, whose value is restricted to the type Trailer. For Truck, the value for Number of Wheels is variable, but it is restricted to 18 for Trailer Truck.

Over the years, many versions of DLs have been designed, but their common core is based on the four sentence patterns of Aristotle's syllogisms, as illustrated in the existential graphs of Figure 3. The subtype-supertype arrows of Figure 8 can be stated in universally quantified sentences with *is* as the verb: *Every truck is a vehicle*. The roles can be stated in universally quantified sentences with *has* as the verb: *Every truck has a cargo capacity, which is a volume measure*. That complex sentence could be split in two to fit the Aristotelian patterns: *Every truck has a cargo capacity* and *Every cargo capacity is a volume measure*. Aristotelian sentences with negation can state constraints on the hierarchy: *No vehicle is an animal* and *Some vehicle is not a truck*.

The restriction arrow in Figure 8 requires a sentence pattern that goes beyond Aristotle: *Every trailer truck has a number of wheels, which is 18*. If this sentence were split in two, the connection between trailer trucks and the integer 18 would be lost. With the restriction feature, KL-ONE supports a wider

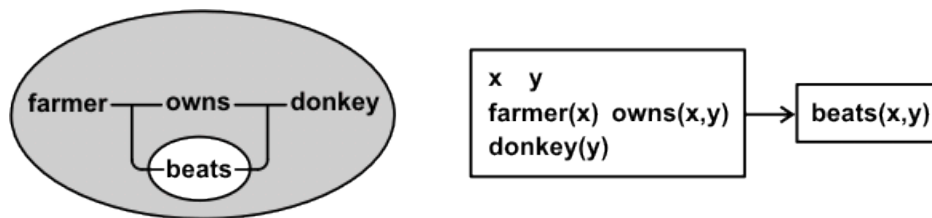
range of sentence patterns while preserving the simplicity of an Aristotelian style of reasoning. More recent DLs such as OWL use similar techniques to extend the Aristotelian paradigm.

To support more expressive logics, Brachman, Fikes, and Levesque (1983) designed a *hybrid* reasoning system called Krypton. They used KL-ONE to define the terminology (T-box) and added an inference engine for first-order logic to make assertions and reason about them (A-box). Schank and his colleagues also developed hybrid systems. They used conceptual dependency graphs to represent NL semantics, and they wrote procedures in LISP to implement a wide range of applications. Since then, hybrids that mix multiple declarative and procedural languages have been widely used for databases, knowledge bases, and the Semantic Web.

A major challenge for hybrid systems is to define, combine, and relate the patterns of heterogeneous computer languages to each other and to the natural languages that people read, write, and speak. As Figures 1 and 2 show, logics with equivalent expressive power can vary widely in the structural patterns they support. For example, the sentence *If a farmer owns a donkey, then he beats it* can be translated to the following formula in Peirce-Peano algebra:

$$\forall x \forall y (\text{farmer}(x) \wedge \text{donkey}(y) \wedge \text{owns}(x,y)) \supset \text{beats}(x,y)$$

In English, this formula can be read *For every x and y, if x is a farmer who owns a donkey y, then x beats y*. This awkward paraphrase results from the rules for the scope of quantifiers. In English and many other languages, an existential quantifier in the *if* clause of the sentence includes the *then* clause in its scope. But quantifiers on the left side of the  $\supset$  operator do not include the right side in their scope. For a more direct mapping from language to logic, Kamp (1981) defined *discourse representation structures* (DRSs) with scoping rules similar to natural languages (Figure 9). Coincidentally, the EG and DRS logical structures are isomorphic.



**Figure 9. EG and DRS for *If a farmer owns a donkey, then he beats it*.**

Kamp's primitives are the same as Peirce's: the default *and* operator is omitted, and the default quantifier is existential. DRS negation is represented by a box marked with the  $\neg$  symbol, and implication is represented by two boxes. As Figure 9 illustrates, nested EG ovals allow lines in the *if* oval to extend into the *then* oval. For DRS, Kamp made an equivalent assumption: the quantifiers for  $x$  and  $y$  in the *if* box govern  $x$  and  $y$  in the *then* box. Since their structures are isomorphic and they use the same operators with the same scoping rules, the EG and DRS in Figure 9 can be translated to the same algebraic formula and the same Conceptual Graph Interchange Format (CGIF):

$$\sim(\exists x \exists y \text{ farmer}(x) \wedge \text{ donkey}(y) \wedge \text{ owns}(x,y) \wedge \sim \text{ beats}(x,y))$$

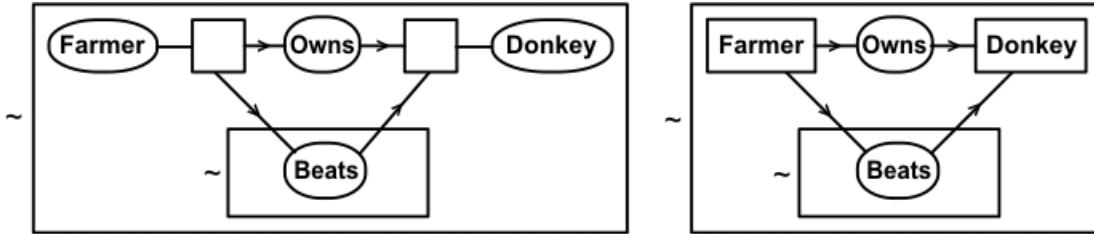
$$\sim[ [*x] [*y] (\text{ farmer } ?x) (\text{ donkey } ?y) (\text{ owns } ?x ?y) \sim[ (\text{ beats } ?x ?y) ] ]$$

Like EG and DRS, CGIF does not use a symbol for *and*. The square brackets marked with the  $\sim$  symbol represent EG ovals or DRS boxes. The symbols  $[*x]$  and  $[*y]$  represent existential quantifiers; they map to EG lines of identity or DRS variables. Since the graphs do not have variables, the letters  $x$  and  $y$  are called *identifiers* or *coreference labels*. With an asterisk,  $*x$  is called the *defining label*, and each occurrence of  $?x$  is a *bound label* within the scope of the defining label with the same identifier. To represent full first-order logic with equality, only one more feature is needed: a *coreference node* that

shows a ligature of two or more lines. For example, the following CGIF represents the EG in Figure 5:

$[*x] (\text{God } ?x) \sim [ [*y] (\text{God } ?y) \sim [ [?x ?y] ] ]$

The coreference node  $[?x ?y]$  maps to the equality  $x=y$  in DRS. But a coreference node can contain any number of bound labels:  $[?x ?y ?z]$  is the equivalent of two equalities,  $x=y$  and  $y=z$ .



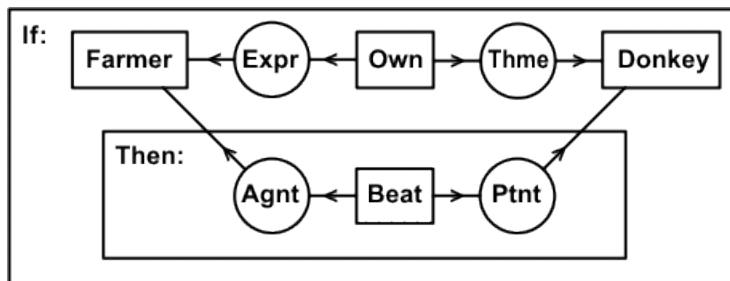
**Figure 10. Untyped and Typed CGs for *If a farmer owns a donkey, then he beats it.***

The *display form* of conceptual graphs uses nested boxes instead of ovals. As in CGIF, a ~ symbol in front of a box indicates negation. Figure 10 shows two conceptual graphs that are logically equivalent to the EG and DRS for the donkey sentence. The CG on the left is *untyped*. The two empty boxes, called *concept nodes*, represent existential quantifiers. In CGIF, they map to  $[*x]$  and  $[*y]$ . The ovals represent relations, and the lines show connections between concept and relation nodes. With these conventions, the untyped CG maps to the same formula and CGIF as the EG and DRS in Figure 9.

The CG on the right of Figure 10 is *typed*. The boxes that contain the labels *Farmer* and *Donkey* are typed concept nodes that restrict the domain of the quantifiers by the monadic relations *Farmer* and *Donkey*. In a typed algebraic notation, they map to the restricted existential quantifiers  $(\exists x:\text{Farmer})$  and  $(\exists y:\text{Donkey})$ . In CGIF, they map to  $[\text{Farmer } *x]$  and  $[\text{Donkey } *y]$ . Following are the typed algebra and the typed CGIF statements:

$\sim(\exists x:\text{Farmer})(\exists y:\text{Donkey})(\text{Owns}(x,y) \wedge \sim\text{Beats}(x,y))$   
 $\sim[ [\text{Farmer } *x] [\text{Donkey } *y] (\text{Owns } ?x ?y) \sim [ (\text{Beats } ?x ?y) ] ]$

The CGs in Figure 10 represent the verbs *owns* and *beats* as dyadic relations. Those relations, which are commonly used in DRS, can also be used in the EG or CG. But Peirce noted that the event or state expressed by a verb is an entity that can be referenced by a quantified variable. Davidson (1967) called that option *event semantics*. Figure 11 represents the state *Own* and the act *Beat* as entities linked to their participants by the linguistic *case relations* or *thematic roles*. To enhance readability, the type labels *If* and *Then* can be used as synonyms for the ~ symbols that mark negated contexts. The relations are Experiencer (Expr), Theme (Thme), Agent (Agnt), and Patient (Ptnt).



**Figure 11. CG with case relations shown explicitly**

Following is the CGIF for Figure 11:

$[ \text{If } [\text{Farmer } *x] [\text{Own } *y] [\text{Donkey } *z] (\text{Expr } ?y ?x) (\text{Thme } ?y ?z) [ \text{Then } [\text{Beat } *w] (\text{Agnt } ?w ?x) (\text{Ptnt } ?w ?y) ] ] ]$



The Common Logic Interchange Format (CLIF) uses a LISP-like notation to represent the same semantics:

(not (exists ((x Farmer) (y Own) (z Donkey)) (and (Expr y x) (Thme y z)  
(not (exists ((w Beat)) (and (Agnt w x) (Ptnt w y)))))))

CLIF and CGIF have the same model theoretic semantics, which is specified in the Common Logic standard. CLIF syntax is specified in Annex A of the standard, and CGIF syntax in Annex B.

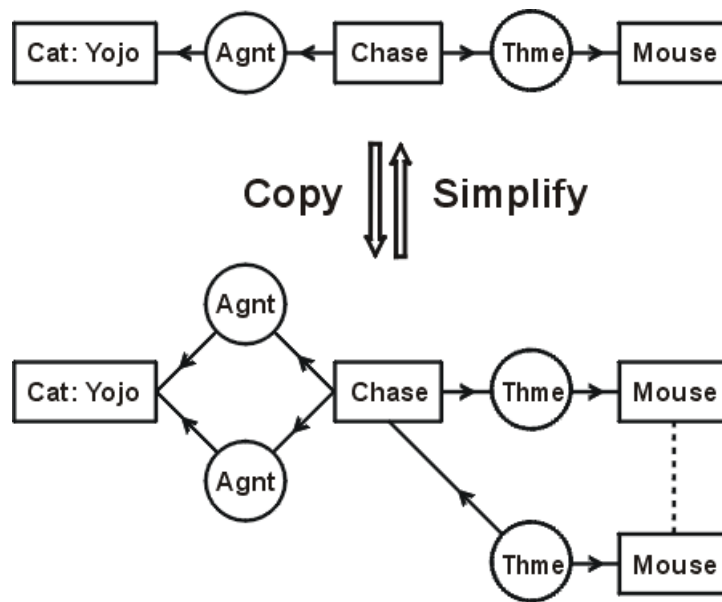
### 3. Reasoning With Graphs

Graphs have advantages over linear notations in human factors, computational efficiency, and cognitive representation. For readability, graphs show relationships at a glance that are harder to see in linear notations. They also have a highly regular structure that can simplify many algorithms for reasoning, searching, indexing, and pattern matching. With his BS degree in chemistry, Peirce was the first to recognize the similarity between chemical graphs and logical graphs. He adopted the chemical term *valence* for the number of arguments of a relation. By applying algorithms for chemical graphs to conceptual graphs, Levinson and Ellis (1992) implemented the first type hierarchy that could support classification and retrieval in logarithmic time. Further research on chemical graphs led to high-speed algorithms for indexing graphs and finding similar graphs (Rhodes et al. 2007). Such techniques can find analogous graphs in logarithmic time (Sowa & Majumdar 2003).

Conceptual graphs can be generated and transformed by a graph grammar based on three pairs of *canonical formation rules*. Each rule transforms a CG or a pair of CGs  $u$  to a CG or pair of CGs  $v$ . Each transformation has one of three semantic effects: equivalence, specialization, or generalization.

- *Equivalence*. *Copy* and *simplify* are equivalence rules, which generate a CG  $v$  that is logically equivalent to the original:  $u \supset v$  and  $v \supset u$ . Equivalent graphs are true in the same models.
- *Specialization*. *Join* and *restrict* are specialization rules, which generate a CG  $v$  that implies the original:  $v \supset u$ . Specialization rules monotonically decrease the set of models in which the result is true.
- *Generalization*. *Detach* and *unrestrict* are generalization rules, which generate a CG  $v$  that is implied by the original:  $u \supset v$ . Generalization rules monotonically increase the set of models in which the result is true.

Each rule has an inverse that can reverse any change it makes. The inverse of copy is simplify, the inverse of restrict is unrestrict, and the inverse of join is detach. Combinations of these rules, called *projection* and *maximal join*, perform larger semantic operations for answering a question or interpreting a text. Maximal joins determine the the unifiable overlap between two CGs. In language processing, they help resolve ambiguities, find the antecedents of anaphoric references, and determine the most likely connections of new information to background knowledge. The next three diagrams illustrate these rules with CGs that have no nested subgraphs.

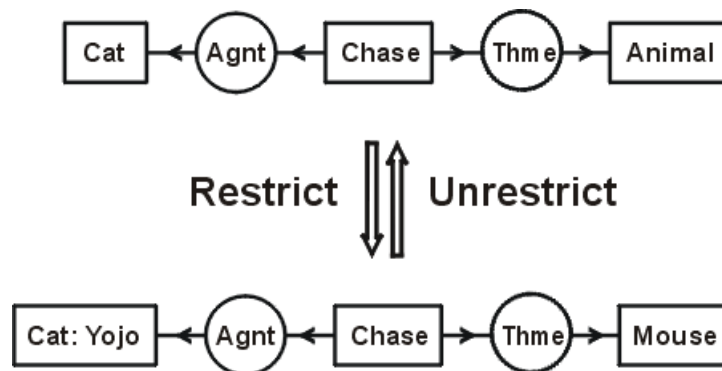


**Figure 12. Copy and simplify rules**

The CG at the top of Figure 12 represents the sentence *The cat Yojo is chasing a mouse*. The down arrow represents two applications of the copy rule. One application copies the Agnt relation, and a second copies the subgraph  $\rightarrow(\text{Thme})\rightarrow[\text{Mouse}]$ . The dotted line connecting the two [Mouse] concepts is a *coreference link*, which indicates that both concepts refer to the same individual. The up arrow represents two applications of the simplify rule, which performs the inverse operations of erasing redundant copies. Following is the CGIF for both graphs:

[Cat: Yojo] [Chase: \*x] [Mouse: \*y] (Agent ?x Yojo) (Thme ?x ?y)  
 [Cat: Yojo] [Chase: \*x] [Mouse: \*y] [Mouse: ?y]  
 (Agent ?x Yojo) (Agent ?x Yojo) (Thme ?x ?y) (Thme ?x ?y)

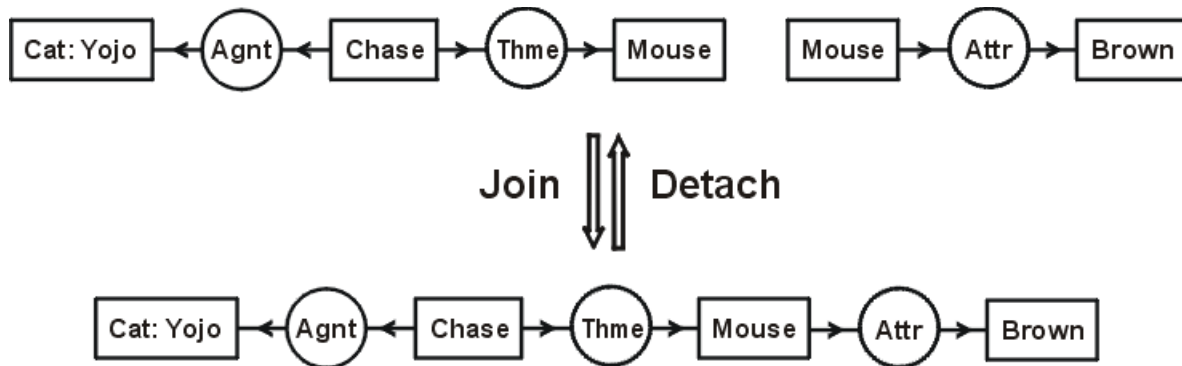
As the CGIF illustrates, the copy rule makes redundant copies, which are erased by the simplify rule. In effect, the copy rule is  $p \supset (p \wedge p)$ , and the simplify rule is  $(p \wedge p) \supset p$ .



**Figure 13. Restrict and unrestrict rules**

The CG at the top of Figure 13 says *A cat is chasing an animal*. By two applications of restrict, it is transformed to the CG for *The cat Yojo is chasing a mouse*. In the first step, the concept [Cat] is *restricted by referent* to [Cat: Yojo], which says that there exists a cat named Yojo. In the second step, the concept [Animal] is *restricted by type* to [Mouse]. The more specialized graph implies the more general one: if the cat Yojo is chasing a mouse, then a cat is chasing an animal. To show that the bottom graph  $v$  implies the top graph  $u$ , let  $c$  be a concept of  $u$  that is being restricted to a more

specialized concept  $d$ , and let  $u$  be  $c \wedge w$ , where  $w$  is the remaining information in  $u$ . By hypothesis,  $d \supset c$ . Therefore,  $(d \wedge w) \supset (c \wedge w)$ . Hence,  $v \supset u$ .



**Figure 14. Join and detach rules**

At the top of Figure 14 are two CGs for the sentences *Yojo is chasing a mouse* and *A mouse is brown*. The join rule overlays the two identical copies of the concept [Mouse] to form a single CG for the sentence *Yojo is chasing a brown mouse*. The detach rule undoes the join to restore the top graphs. Following are the CGIF sentences that represent the graphs in Figure 14:

[Cat: Yojo] [Chase: \*x] [Mouse: \*y] (Agent ?x Yojo) (Thme ?x ?y)  
 [Mouse: \*z] [Brown: \*w] (Attr ?z ?w)

[Cat: Yojo] [Chase: \*x] [Mouse: \*y] (Agent ?x Yojo) (Thme ?x ?y)  
 [Brown: \*w] (Attr ?y ?w)

As the CGIF illustrates, the effect of the join is to substitute  $y$  for every occurrence of  $z$  in the top graph and erase redundant copies. In general, every join assumes an equality of the form  $y=z$  and simplifies the result. If  $q$  is the equality and  $u$  is the top pair of graphs, then the bottom graph is equivalent to  $q \wedge u$ , which implies  $u$ . Therefore, the result of join implies the original graphs. The six canonical formation rules define a generalization hierarchy of graphs whose only logical operators are existence and conjunction. To support full first-order logic, additional rules are needed to handle negation.

## 4. Peirce's Rules of Inference

For existential graphs, Peirce defined a sound and complete proof procedure for FOL whose rules depend on whether an area is positive (unshaded) or negative (shaded). To apply the rules to conceptual graphs and other notations for logic, the version by Peirce (1911) has been restated in terms of specialization and generalization. The rules are grouped in three pairs: one rule (i) inserts a graph, and the other (e) erases a graph. The only axiom is a blank sheet of paper (an empty graph with no nodes). The empty graph and any EG derived from it by Peirce's rules is true in all models.

1. (i) Insert: In a negative area, any graph or subgraph (including the blank) may be replaced by any specialization.  
 (e) Erase: In a positive area, any graph or subgraph may be replaced by any generalization (including the blank).
2. (i) Iterate: Any graph or subgraph in any area  $c$  may be iterated (copied) in the same area  $c$  or into any area nested in  $c$ . No graph may be copied directly into itself. But it is permissible to copy a graph  $g$  in the same area  $c$  and then copy the copy into some area nested in the original  $g$ .  
 (e) Deiterate: Any graph or subgraph that could have been derived by rule 2i may be erased.

Whether or not the graph had been derived by 2i is irrelevant.

3. (i) Double negation: A double negation (nest of two negations with nothing between the inner and outer) may be drawn around any graph, subgraph, or set of graphs in any area. In an EG, lines that originate outside a double negation and pass through it without a connection to anything in the area between the two negations are not considered part of that area. In CGIF, those lines would not have any bound labels in the area between the outer  $\sim[$  and the inner  $\sim[$ .

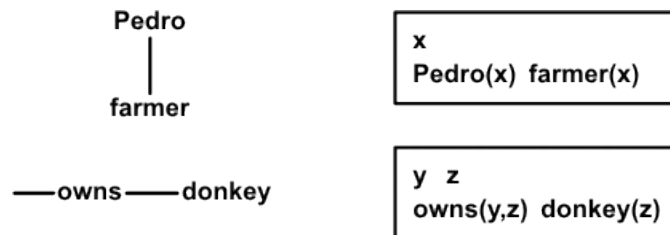
(e) Double negation: A double negation in any area may be erased.

With adaptations for the variations in syntax, these same rules can be applied to EGs, CGs, and DRS in either CGIF or the original graph notations. They can even be applied to the algebraic notation and controlled natural languages. In EGs, the operation of inserting a link between two lines has the effect of identifying them (inserting an equality); erasing a link has the inverse effect of erasing an equality. In a linear notation, the operation of inserting or erasing an equality may require an additional operation of renaming labels. Since pure graph notations have no labels, there is nothing to rename.

To illustrate the similarity between EG and DRS, consider the following pair of sentences: *Pedro is a farmer. He owns a donkey.* Kamp and Reyle (1993) observed that proper names like *Pedro* are not rigid identifiers. In DRS, proper names are represented by predicates rather than constants. That convention is similar to Peirce's practice with EGs. Following are the algebraic formulas:

$$\exists x \text{ Pedro}(x) \wedge \text{farmer}(x). \quad \exists y \exists z \text{ owns}(y,z) \wedge \text{donkey}(z).$$

In English, a pronoun such as *he* can refer to something in a previous sentence, but the variable  $y$  in the second formula cannot be linked to the variable  $x$  in the first. The structure of the EG and DRS notations facilitates that linkage. On the left of Figure 15 are the EGs for each of the sentences; on the right are the DRSs.



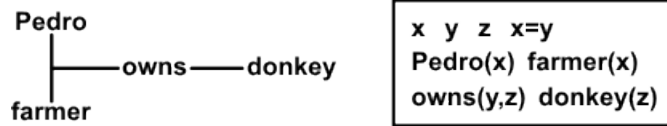
**Figure 15. EGs and DRSs for *Pedro is a farmer. He owns a donkey.***

As in the earlier examples, the same CGIF can be used to linearize EG and DRS. When two EGs are drawn in the same area, the lines of identity that represent existential quantifiers are independent of one another, but the CGIF identifiers might cause accidental name clashes. To avoid those clashes, CGIF allows brackets without the  $\sim$  symbol to limit the scope of quantifiers. Although the brackets are not required for this example, they can be used to emphasize the similarities between EG, DRS, and CGIF. The following CGIF represents the EGs and DRSs in Figure 15:

$$[ [*x] (\text{Pedro } ?x) (\text{farmer } ?x) ] [ [*y] [*z] (\text{owns } ?y ?z) (\text{donkey } ?z) ]$$

To combine the two EGs, connect the line of identity for *Pedro* to the line for *he*, as in Figure 16. To combine the two DRSs and equate the variables  $x$  and  $y$ , transfer the contents of one DRS box to the other, move the variables to the top, and insert the equality  $x=y$ . For the equivalent CGIF, erase the brackets around the CGIF for each EG, move all the defining nodes to the left, and insert the coreference node  $[?x ?y]$  after the defining nodes. Finally, insert a pair of brackets around the result to emphasize the similarity of CGIF and DRS:

$$[ [*x] [*y] [*z] [?x ?y] (\text{Pedro } ?x) (\text{farmer } ?x) (\text{owns } ?y ?z) (\text{donkey } ?z) ]$$

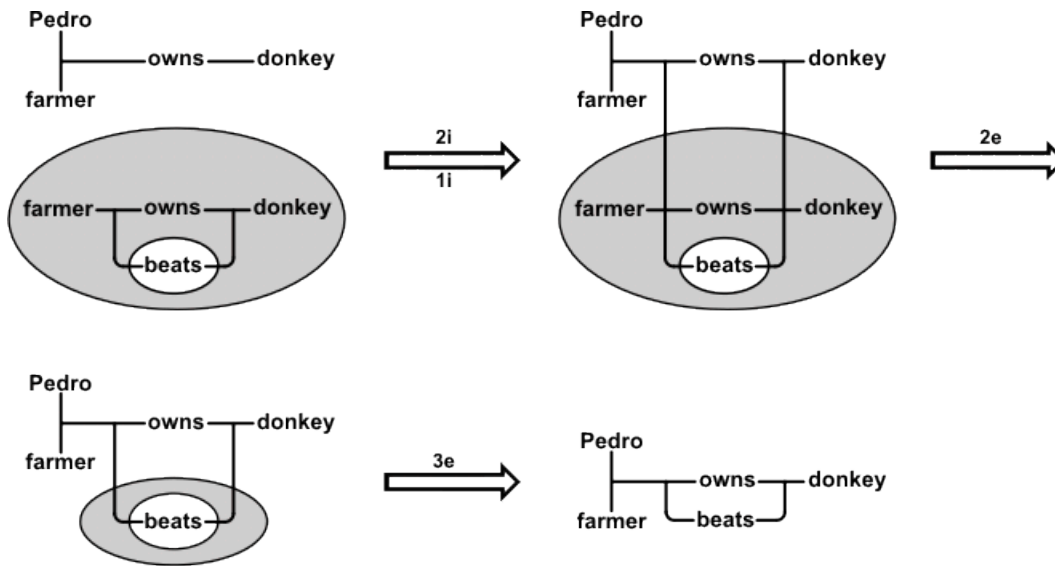


**Figure 16. Combining the EGs and DRSs in Figure 15**

No relabeling is necessary or possible for the EG. But the equality  $x=y$  allows either  $x$  or  $y$  to be deleted from the DRS or CGIF and its bound labels renamed. After deleting  $[*y]$ , relabeling every occurrence of  $?y$  to  $?x$ , and erasing the irrelevant  $[?x ?x]$ , the CGIF is simplified to

$[ [*x] [*z] (\text{Pedro } ?x) (\text{farmer } ?x) (\text{owns } ?x ?z) (\text{donkey } ?z) ]$

The isomorphism between EG, DRS, and CGIF implies that Peirce's rules can generate equivalent proofs with each notation. Figure 17 begins with the EGs for the sentences *Pedro is a farmer who owns a donkey* and *If a farmer owns a donkey, then he beats it*. Then the rules 2i, 1i, 2e, and 3e derive an EG for the conclusion *Pedro is a farmer who owns and beats a donkey*.



**Figure 17. A proof according to Peirce's rules**

In the graphic notation, rule 2i iterates (extends) two lines of identity from the outer graph into the negative area. Then rule 1i inserts links (equalities) in the negative area to connect the lines from the outer area to corresponding lines in the inner area. Following is the CGIF for the two starting EGs, but with a choice of labels that avoids conflicts:

$[*a] [*b] (\text{Pedro } ?a) (\text{farmer } ?a) (\text{owns } ?a ?b) (\text{donkey } ?b)$   
 $\sim [ [*c] [*d] (\text{farmer } ?c) (\text{owns } ?c ?d) (\text{donkey } ?d) \sim [ (\text{beats } ?c ?d) ] ]$

The step of iterating the lines into the negative area corresponds to inserting coreference nodes  $[?a]$  and  $[?b]$  inside the negation. The step of inserting equalities corresponds to adding  $?c$  to  $[?a]$  to form  $[?a ?c]$  and adding  $?d$  to  $[?b]$  to form  $[?b ?d]$ . The result is

$[*a] [*b] (\text{Pedro } ?a) (\text{farmer } ?a) (\text{owns } ?a ?b) (\text{donkey } ?b)$   
 $\sim [ [*c] [*d] [?a ?c] [?b ?d] (\text{farmer } ?c) (\text{owns } ?c ?d) (\text{donkey } ?d) \sim [ (\text{beats } ?c ?d) ] ]$

To simplify this CGIF, erase  $[*c]$  and  $[*d]$ , relabel every occurrence of  $?c$  to  $?a$  and  $?d$  to  $?b$ , and erase the irrelevant  $[?a ?a]$  and  $[?b ?b]$ . The result is

$[*a] [*b] (\text{Pedro } ?a) (\text{farmer } ?a) (\text{owns } ?a ?b) (\text{donkey } ?b)$   
 $\sim [ (\text{farmer } ?a) (\text{owns } ?a ?b) (\text{donkey } ?b) \sim [ (\text{beats } ?a ?b) ] ]$

The next step is to notice that the subgraph and corresponding CGIF for (farmer ?a) (owns ?a ?b) (donkey ?b) in the negative area is identical to the corresponding subgraph and CGIF in the outer area. Therefore, the inner copy can be deiterated (erased) by rule 2e. The result is

[\*a] [\*b] (Pedro ?a) (farmer ?a) (owns ?a ?b) (donkey ?b)  
 ~[ ~[ (beats ?a ?b) ] ]

Finally, erase the double negation by rule 3e to derive the CGIF for the conclusion:

[\*a] [\*b] (Pedro ?a) (farmer ?a) (owns ?a ?b) (donkey ?b) (beats ?a ?b)

Every CGIF statement in this proof can be translated directly to DRS, and every operation performed on the CGIF can be performed on the DRS. The most complex operations are the relabeling required for CGIF and DRS; those operations are not required for EGs. Sowa (2011) presents more detail about EG syntax, semantics, proof procedures, and their mapping to CGIF.

Kamp observed that the sentence *Every farmer who owns a donkey beats it* is equivalent to the DRS (or EG) in Figure 9. In the algebraic notation, Peirce used distinct symbols for existential and universal quantifiers. For EGs, he used only one kind of line, because a line that originates in a white area represents an existential quantifier and a line that originates in a shaded area represents a universal. Since CGIF is not as iconic as EGs, the symbol @every can be used to represent a universal quantifier. The CGIF concept [Farmer @every] represents the phrase *every farmer*. But the phrase *every farmer who owns a donkey* requires a CGIF *type expression*, which is similar to a lambda expression:

[@\*u [Farmer ?u] [Donkey \*v] (Owns ?u ?v): @every\*x] (Beats ?x ?v)

This CGIF can be read *Every farmer x who owns a donkey v beats v*. Note that the coreference label ?u cannot be used outside the type expression, but the coreference labels ?x and ?v can be. The Common Logic standard defines CGIF type expressions by their expansion to core CGIF. When this CGIF is expanded, the result is identical to the CGIF for Figure 9 (except for the names of labels). That expansion shows why the scope of the defining labels \*x and \*v includes (beats ?x ?v).

For disjunctions, Figure 18 shows the EG and DRS for an example by Kamp and Reyle (1993:210): *Either Jones owns a book on semantics, Smith owns a book on logic, or Cooper owns a book on unicorns*. The EG at the top of Figure 18 shows that the existential quantifiers for Jones, Smith, and Cooper are in the outer area, but the quantifiers for the three books are inside the alternatives.

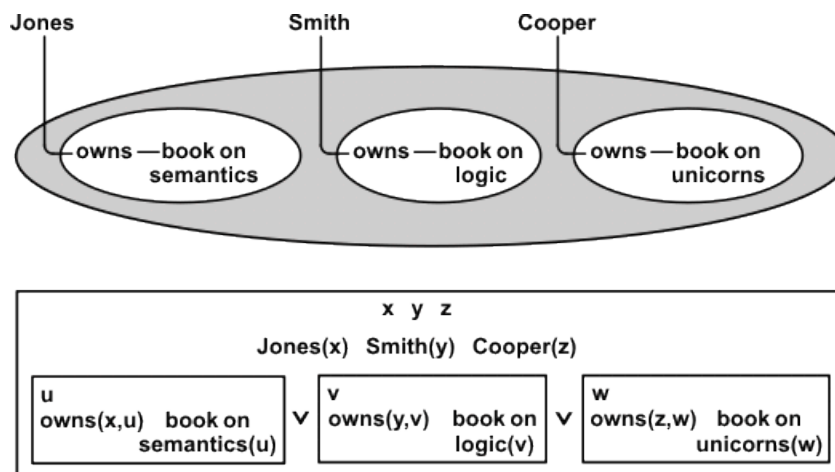


Figure 18. EG and DRS for a disjunction with three alternatives

Both Peirce and Kamp allowed spaces inside relation names, but CGIF requires names with spaces or other special characters to be enclosed in quotes:

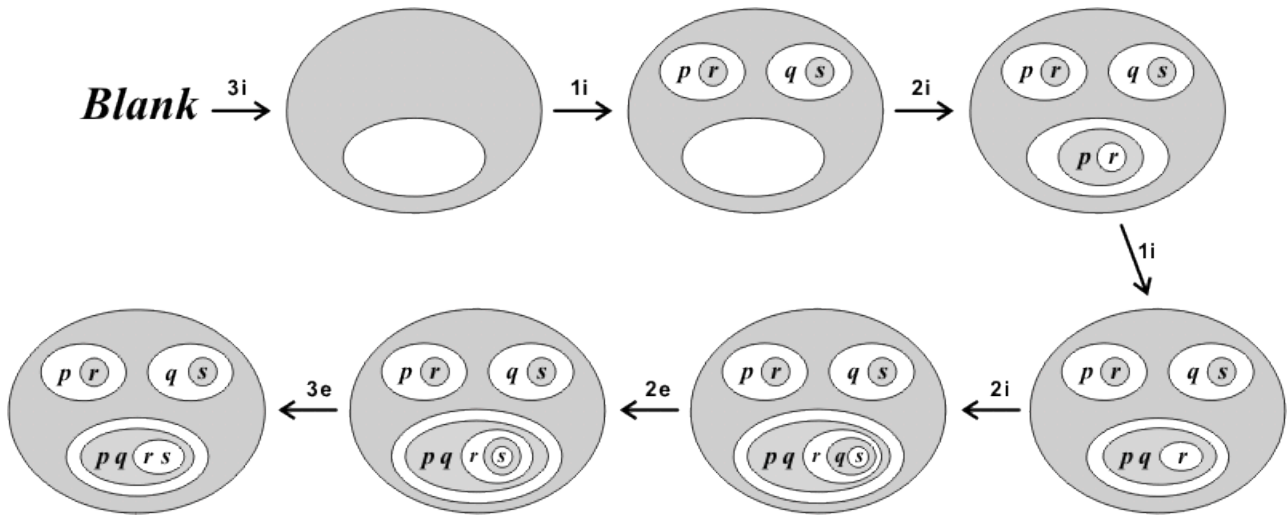
[\*x] [\*y] [\*z] (Jones ?x) (Smith ?y) (Cooper ?z)  
 [Either [Or [\*u] (owns ?x ?u) ("book on semantics" ?u)]  
   [Or [\*v] (owns ?y ?v) ("book on logic" ?v)]  
   [Or [\*w] (owns ?z ?w) ("book on unicorns" ?w) ] ]

The labels *Either* and *Or* are synonyms for the  $\sim$  symbol. Since the linear CGIF notation is not as iconic as the shading of the EG diagrams, the labels can improve its readability. Note that CGIF has a more direct mapping to DRS than to EG.

All the axioms and rules of inference for classical FOL, including the rules of the *Principia Mathematica* by Whitehead and Russell (1910), natural deduction by Gerhard Gentzen (1935), and resolution by Alan Robinson (1965) can be derived from Peirce's rules (Sowa 2011). In the *Principia*, the following statement, which Leibniz called the *Praeclarum Theorema* (Splendid Theorem), was proved in 43 steps, starting with five non-obvious axioms:

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

With Peirce's rules, this theorem can be proved in just seven steps, starting with a blank sheet of paper. Figure 19 shows the EG at each step and the rule used to derive it: Start with a blank sheet. By rule 3i, insert a double negation around an empty space. By 1i, insert the EG for the hypothesis  $((p \supset r) \wedge (q \supset s))$  in the shaded area. By 2i, iterate  $(p \supset r)$ . By 1i, insert  $q$ . By 2i, iterate  $(q \supset s)$ . By 2e, deiterate  $q$ . By 3e, erase the double negation. Each step inserts or erases one graph or subgraph, and the final graph is the statement of the theorem.



**Figure 19. Proof in 7 steps instead of 43 in the *Principia***

In CGIF, a proposition  $p$  is represented as a relation with zero arguments:  $(p)$ . The result of applying rule 3i to the blank produces the CGIF  $\sim[\sim[ ]]$ . Each of the next six steps shown in Figure 19 can be applied to the equivalent CGIF statement to produce the conclusion

$$\sim[\sim[(p) \sim[(r)]] \sim[(q) \sim[(s)]] \sim[\sim[(p) (q) \sim[(r) (s)]]]]$$

Peirce's proof procedure is a generalization and simplification of natural deduction. Unlike Gentzen's version, which uses a method of making and discharging assumptions, Peirce's proofs proceed in a straight line from a blank sheet to the conclusion: every step inserts or erases one subgraph in the immediately preceding graph. As Figure 19 illustrates, the first two steps of any proof that starts with a blank must draw a double negation around the blank and insert some graph in the negative area. Those two steps are the equivalent of making and discharging an assumption, but Peirce's rules do not require bookkeeping to keep track of the assumptions.

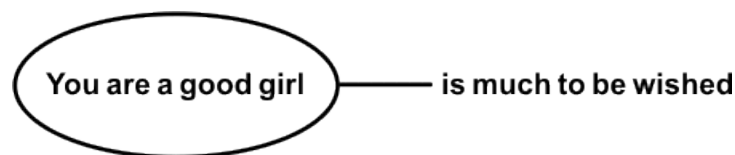
Although Peirce stated his rules in terms of EG syntax, the restatement in terms of generalization and specialization depends only on semantic criteria: the models for which a statement is true, coreference links between symbols, and whether an area is positive or negative. As the examples show, Peirce’s rules can be applied equally well to EG, DRS, or CGIF syntax. Kamp and Reyle (1996) developed a version of natural deduction for the DRS notation, but their rules can be proved as derived rules of inference in terms of Peirce’s rules. The derivation is similar to the demonstration by Sowa (2011) that Gentzen’s version of natural deduction is a special case of Peirce’s.

With adjustments to accommodate differences in syntax, Peirce’s rules can be applied to many other notations for FOL. Since the algebraic notation requires explicit  $\wedge$  symbols, those symbols would have to be inserted (for rules 1i and 2i) or erased (for rules 1e and 2e). To clarify the grouping, parentheses may be inserted or erased as needed. As an exercise, repeat the proof in Figure 19 with algebraic formulas: Use a pencil to shade the negative areas of each formula. Then perform the same seven steps by inserting or erasing algebraic expressions. Since the  $\supset$  operator for implication corresponds to a double negation, the first step of inserting a double negation by rule 3i corresponds to writing  $\supset$  on a sheet of paper and using the pencil to shade some area on the left of  $\supset$ . The next step of inserting the hypothesis  $((p\supset r) \wedge (q\supset s))$  in the shaded area would reverse the shading of that formula. For each of the next six steps, show that the formula that corresponds to each EG in Figure 19 is derived by applying the rule cited on the arrow.

Peirce’s rules can be adapted to any notation for Common Logic. CL allows quantifiers to range over functions and relations, but it preserves a first-order style of model theory and proof theory (Hayes & Menzel 2001). To support higher-order syntax, but without the computational complexity of higher-order semantics, CL model theory uses a single domain  $D$  that includes individuals, functions, and relations. The option of limiting the domain of quantifiers to a single set was suggested by Quine (1953) and used in theorem provers that allow quantifiers to range over relations and functions (Chen et al., 1993). The CL semantics is defined in an abstract syntax that is independent of any concrete notation (ISO/IEC 24707). When Peirce’s rules are stated in terms of the CL abstract syntax, they can be adapted to CL concrete notations by the same transformations used to map the syntax. Features such as shading can simplify the statement or visualization of the rules.

## 5. Extensions to Common Logic

Natural languages can say anything that can be expressed in any formal logic and much more that cannot. They can even express metalevel statements about themselves, about their relationship to other languages, and about the truth, uncertainty, modality, or intentionality of any statement. With his experience in lexicography and his pioneering work in logic, Peirce analyzed and represented the implicit logic and ontology in many aspects of language. He used the terms *Alpha graphs* for EGs that are limited to propositional logic, as in Figure 19; *Beta graphs* for EGs that use lines of identity to express FOL; and *Gamma graphs* for EGs that represent metalanguage, modal logic, and higher-order logic. Figure 20 shows one of his early Gamma graphs (Peirce 1898).



**Figure 20. An EG that makes a metalevel statement about the nested EG**

The oval in Figure 20 encloses an EG that states a proposition, which Peirce represented by a character string that contains blanks. In CGIF, that proposition is represented by a quoted string enclosed in



parentheses: ("You are a good girl"). In this EG, the oval does not represent negation because it is attached to a line of identity. The relation at the other end of the line makes a statement about the proposition stated by the nested EG. Figure 20 goes beyond the standard semantics for Common Logic, but it could be represented by an extension to CGIF:

[Proposition \*p ("You are a good girl")] ("is much to be wished" ?p)

This CGIF states that there exists a proposition p expressed by the CGIF that follows the label \*p. The relation makes a statement about the proposition p, not about something in the world. The medieval Scholastics used the term *first intention* (prima intentio) for language about the world and *second intention* for language about language.

Peirce adopted the Scholastic terminology and called his version of FOL *first-intentional logic*. He coined the term *second-intentional logic* for quantifiers that range over relations. Ernst Schröder translated those terms to German as *erste Ordnung* and *zweite Ordnung*, and Bertrand Russell translated them back to English as *first order* and *second order*. But Figure 20 uses a feature that is not available in Peirce's logic of 1885 or in many newer versions of higher-order logic: the ability to make statements about propositions stated in the logic itself.

As an example, first-order logic can represent the sentence *Bob said "The sky is blue"* by treating the quoted part as an unanalyzed character string. But first-order logic, by itself, cannot represent the sentence *Bob said that the sky is blue*. The crucial feature of this sentence is the word *that*, which enables the sentence to make a statement about the proposition expressed by the string. That ability is the defining characteristic of *metalanguage*. Common Logic can refer to propositions with quantified variables, but it cannot relate a sentence to the proposition it expresses. To support metalanguage, the Interoperable Knowledge Language (IKL) extends Common Logic with a special operator called *that* (Hayes & Menzel 2006). With the IKL extension, the sentence *Bob said that the sky is blue* can be represented in CLIF notation as

(exists (p) (and (= p (that (blue sky))) (said Bob p)))

To express the same semantics, CGIF uses a concept node with the type label Proposition:

[Proposition \*p (blue sky)] (said Bob ?p)

These statements can be read *There exists a proposition p that the sky is blue, and Bob said p*. The simpler sentence *Bob said that the sky is blue* does not require the variable p in CLIF or CGIF:

(said Bob (that (blue sky)))

(said Bob [Proposition (blue sky)])

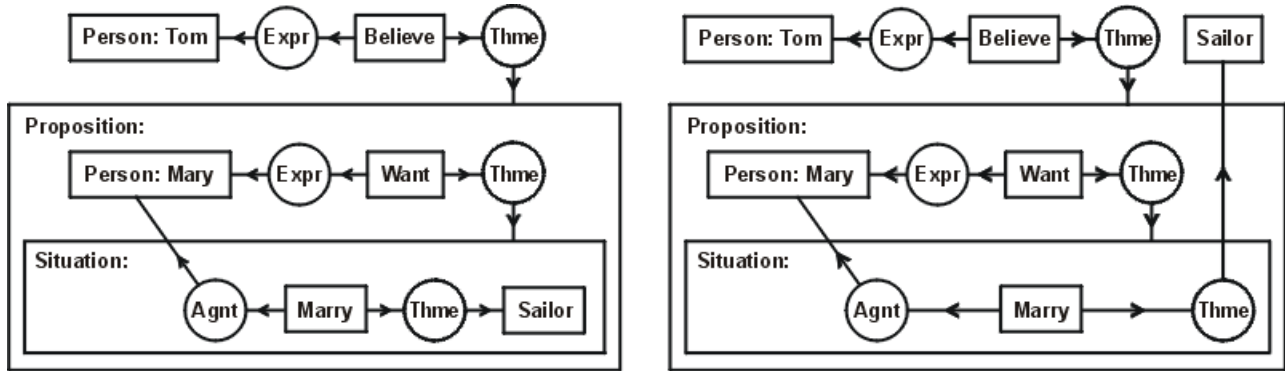
A common use of metalanguage is to talk about the beliefs, desires, and intentions of the speaker and other people. As an example, the sentence *Tom believes that Mary wants to marry a sailor*, contains three clauses, whose nesting can be marked by brackets:

*Tom believes that [Mary wants [to marry a sailor]].*

The outer clause says that Tom has a belief, which is expressed by the nested clause. That clause says Mary wants a situation described by the nested infinitive. Since each clause delimits the scope of quantifiers, questions can arise about the sailor's existence outside of Tom's belief or Mary's desire. Following are three interpretations:

1. *Tom believes that [Mary wants [to marry someone who is sailor]].*
2. *Tom believes that [there is a sailor whom Mary wants [to marry]].*
3. *There is a sailor whom Tom believes that [Mary wants [to marry]].*

The conceptual graphs in Figure 21 represent the first and third interpretations.



**Figure 21. Two interpretations of *Tom believes that Mary wants to marry a sailor*.**

In the CG on the left of Figure 21, the existential quantifier for the concept [Sailor] is nested inside the situation that Mary wants. Whether such a sailor actually exists and whether Tom or Mary knows his identity are not stated. The CG on the right explicitly states that such a sailor exists and that Tom believes Mary wants to marry him. The middle option would move the concept [Sailor] inside the concept of type Proposition. It would imply that Tom believes some such sailor exists, but not imply that Tom's belief is true.

The CGs in Figure 21 represent distinctions that are not marked in the EG of Figure 20. The theme (Thme) of belief is a proposition, but the theme of wanting is a physical situation. With *situation semantics*, Barwise and Perry (1983) addressed these issues, but the major challenge, as Devlin (1991) said, was to define situations: they “include, but are not equal to any of simply connected regions of space-time, highly disconnected space-time regions, contexts of utterance (whatever that turns out to mean in precise terms), collections of background conditions for a constraint, and so on.” Devlin finally admitted that they cannot be defined: “Situations are just that: situations. They are abstract objects introduced so that we can handle issues of context, background, and so on.”

According to Devlin, the choice of situation is subjective: “An important feature of situations that our theory reflects is that the structure of a situation is significant to the agent.” That admission undermines the hope of using situations as an objective foundation for defining beliefs, desires, and intentions. By treating situations as abstract mathematical objects, Devlin separated the logical and philosophical issues. That move transfers the hard problems from logic to ontology. In his paper, Devlin acknowledged Pat Hayes for the “motivation” to simplify the logic. Hayes, by the way, was the coauthor with Chris Menzel of the model theory for Common Logic and for the IKL extension to CL.

For conceptual graphs, the default ontology has a Description relation (Dscr), which relates a situation to a proposition. The following CGIF says that there exists a situation *s*, there exists a proposition *p* that the sky is blue, and a description of *s* is *p*:

[Situation \*s] [Proposition \*p (blue sky)] (Dscr ?s ?p)

This graph can be abbreviated by a concept node of type Situation that contains the nested CGIF:

[Situation (blue sky)]

This method of abbreviation is a *type coercion* that is common in programming languages: any situation node with a nested CG can be expanded to a CG that describes the situation by a proposition node with the same nested CG. Similar type coercions are common in natural languages.

With the IKL metalanguage, the description relation can support a semantics with propositions as primitive instead of situations. But that option requires a definition of *proposition*. According to

Peirce (CP 5.427), “The meaning of a proposition is itself a proposition. Indeed, it is no other than the very proposition of which it is the meaning: it is a translation of it.” Peirce’s criterion implies that a proposition is an *equivalence class* of sentences that can be translated from one to another while preserving meaning. Formally, a *meaning-preserving translation* (MPT) over the sentences of a first-order language L is any function  $f$  from L to L that satisfies three constraints:

1. *Truth preserving.* The sentences  $s$  and  $f(s)$  must have the same truth conditions. For any model M of L, the truth values of the sentences  $s$  and  $f(s)$  must be identical. Preserving truth is necessary for meaning preservation, but it groups too many sentences in the same equivalence class. For example,  $2+2=4$  and Fermat’s last theorem are true in every model of Peano’s axioms, but they are not synonymous. The test to determine whether two sentences mean the same should be computable by a simple and efficient algorithm.
2. *Vocabulary preserving.* A sentence  $s$  and its translation  $f(s)$  may contain different logical operators, their syntax may be rearranged, and the labels of quantified variables may be different. But  $f$  must not add or delete arbitrary terms. For example, the sentence *Every cat is a cat* should not be considered synonymous with *Every dog is a dog* or *Every unicorn is a unicorn*. This constraint may be relaxed to allow  $f$  to replace every occurrence of a term with its definition. It might replace *cat* with *domestic feline*, but not with *dog* or *unicorn*.
3. *Structure preserving.* The computational complexity of the sentences  $s$  and  $f(s)$  must be the same. For example, the sentences *Every farmer who owns a donkey beats it* and *If a farmer  $x$  owns a donkey  $y$ , then  $x$  beats  $y$*  have the same computational properties. But the sentences  $p$  and  $\sim p$ , which have the same truth values, are treated differently by many inference engines. Since negations are critical to computational complexity,  $s$  and  $f(s)$  must have the same depth of nested negations when translated to EGs.

A convenient way to specify the equivalence class is to define an MPT  $f$  that translates sentences to a *canonical form*. Two sentences  $u$  and  $v$  belong to the same equivalence class if and only if they have the same canonical form:  $f(u) = f(v)$ . The following four steps define a suitable MPT:

1. For any named entity (individual, function, or relation) that is specified by a context-free, nonrecursive definition, replace the name by its definition. Do not replace any name that is specified by a context-sensitive or recursive definition.
2. Replace the quantifiers and Boolean operators by their definitions in terms of conjunction, negation, and the existential. Delete unnecessary blanks, parentheses, or other punctuation.
3. Relabel all variables or labels bound by quantifiers to a fixed sequence, such as  $x_1, x_2, \dots$ . Within each area, sort all conjuncts by their representation as character strings.
4. After step 3, note whether any conjunct is identical to its predecessor; if so, delete it. This step is equivalent to Peirce’s rule of deiteration within a single area, but not nested areas. For  $p$  of any size, it would replace  $p \wedge p$  with  $p$ , but it would leave  $\sim(p \wedge \sim(p))$  unchanged.

This translation may be called *Peirce normal form* (PNF), since it uses the same operators as EGs. Sentences with the same PNF are logically equivalent: steps 1 and 2 replace some terms and operators with their definitions; step 3 sorts conjuncts; and step 4 deletes duplicates. The translation to PNF is efficient: the sort in step 3 may take  $(N \log N)$  time; the other steps take linear time.

For modal logics, propositions can specify or replace possible worlds. Michael Dunn (1973) showed that every possible world  $w$  in Kripke’s semantics has a one-to-one mapping to a pair of sets (L,F): the laws L of  $w$  are the necessarily true propositions, the facts F of  $w$  are the true propositions, and the possibilities P of  $w$  are the propositions consistent with L. With Dunn’s semantics, Kripke’s

accessibility relation is no longer an unexplained primitive: a world  $u$  is accessible from a world  $w$  iff every law of  $w$  is a fact of  $u$ . A logic that supports metalanguage, such as IKL, can declare appropriate conditions on the laws and facts to support versions of modality and theories of beliefs, desires, and intentions (Sowa 2003, 2006b). It can also express the “three modes of being” by Peirce (CP 1.21-23): “the being of positive qualitative possibility, the being of actual fact, and the being of law that will govern facts in the future.”

As these examples show, Common Logic with extensions for metalanguage provides a first-order framework that can eliminate the need for a multiplicity of different logics, each with its own model theory and proof theory. To avoid paradoxes of metalanguage, Tarski (1933) proposed a stratified series of first-order metalevels. In effect, he declared that certain kinds of sentences (those that violate the stratification) do not express propositions in his logic. The IKL model theory does not require stratified metalevels, but it also avoids paradoxes by eliminating troublesome propositions. The following sentence sounds paradoxical, but the IKL version is false because no such proposition exists:

*There exists a proposition  $p$ ,  $p$  is true, and  $p$  says that  $p$  is false.*

In CLIF and CGIF notation,

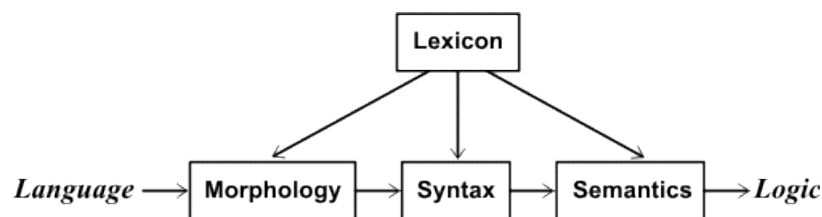
(exists ((p Proposition)) (and (p) (= p (that (not (p))))))

[Proposition \*p] (?p) [Proposition ?p ~[(?p)]]

In any notation for IKL, the equivalent of  $(\exists p)$  would be false.

## 6. Mapping Language to Logic

Frege and Russell considered their symbolic notations superior to NLS, but Richard Montague (1970) ignored the differences: “I reject the contention that an important theoretical difference exists between formal and natural languages.” Many computational linguists agreed. Terry Winograd (1972) designed SHRDLU as a compiler from English to logic and described the process as *Natural Language Understanding*. Figure 22 shows the compilation as a linear sequence: the morphology stage strips off the endings of words and retrieves their definitions from a lexicon; the syntactic stage parses sentences; and the semantic stage generates a version of logic called Microplanner. Finally, an inference engine processes the logic to answer a question, perform an action, or update the knowledge base.

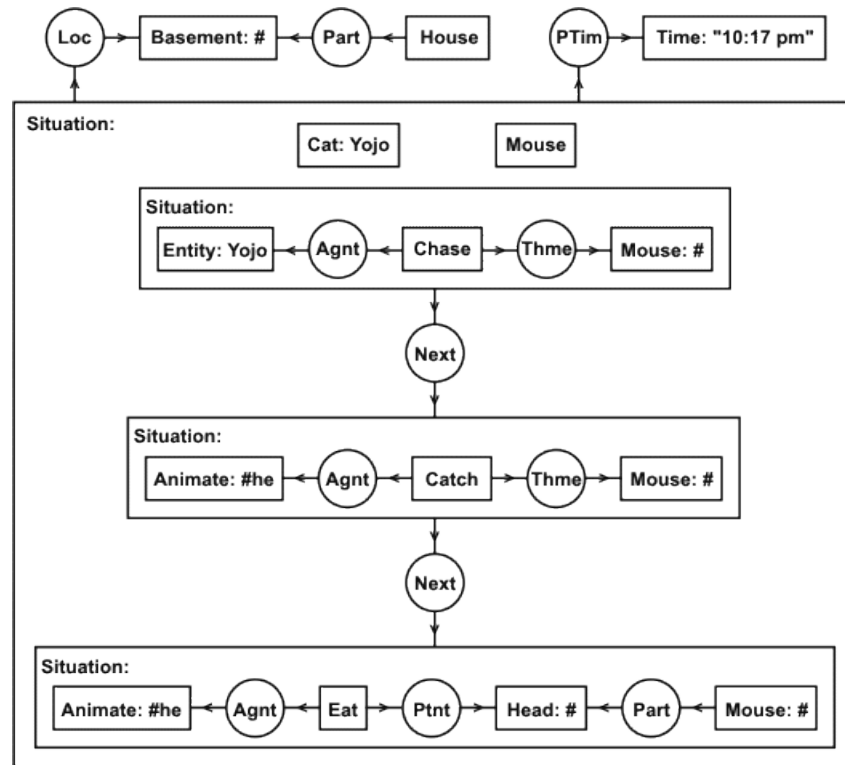


**Figure 22. Compiling language to logic**

Although SHRDLU was called a “natural language” system, its subset of English should be called a controlled NL. SHRDLU had a formal grammar, a version of logic for specifying ontology, and a theorem prover for processing the logic. For his next project, Winograd (1983) wrote a well-received textbook with the title *Language as a Cognitive System. Volume I: Syntax*. While writing the second volume, he realized the limitations of formal semantics (Winograd & Flores 1986). Since then, he has been a strong critic of attempts to map language to logic.

Even when every sentence is unambiguous, background knowledge is needed to determine the pattern of relationships. Figure 23 shows a conceptual graph derived from the following story:

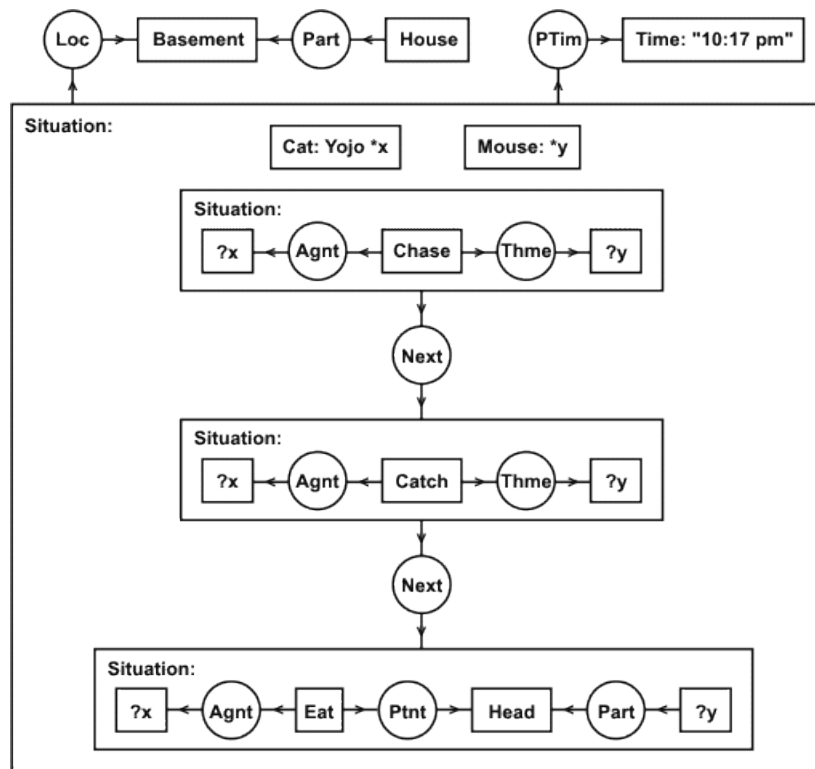
At 10:17 pm, Yojo the cat and a mouse were in the basement of a house. Yojo chased the mouse. He caught the mouse. He ate the head of the mouse.



**Figure 23. Marking indexicals in discourse**

Figure 23 uses DRS conventions to show the time sequence. The large box encloses a situation with three nested situations, each described by one of the sentences. The # symbol, which is not included in the CGIF standard, marks context-dependent *indexicals*. A considerable amount of information is required to derive Figure 23; even more is needed to replace the # markers with coreference labels:

- In a static description, the sentences have no implicit temporal order, and the default connective is the conjunction *and*. But in a narrative, the default connective is *and then*, represented by the relation (Next). For this example, the stative verb *were* in the first sentence suggests a description. The action verbs suggest a narrative sequence.
- The relations for location (Loc) and point in time (PTim) could be attached to a verb, as in Tesnière's graphs (Figure 6), or they could be attached to a situation box, as in Figure 23. Since the story did not say that the cat or the mouse left the basement, the default assumption is that the actions mentioned in the last three sentences occurred there.
- Machine-readable dictionaries and related lexical resources are needed to determine the thematic roles associated with verbs and the relations such as (Part) that connect [House] to [Basement] and [Mouse] to [Head].
- Other kinds of background knowledge are necessary to determine which markers or symbols should be used for names like *Yojo*, time references like *10:17 pm*, pronouns like *he*, and the definite articles in the phrases *the basement*, *the mouse*, *the head*, and *the cat Yojo*.



**Figure 24. Resolving the indexicals**

Figure 24 shows the results of resolving indexicals by replacing # symbols with bound labels. Since the nested boxes in the CG and DRS are isomorphic, similar methods can be used. For this example, the two main *discourse referents* are marked by defining labels: [Cat Yojo \*x] and [Mouse \*y]. References to them are marked by bound labels ?x and ?y. But the indexical markers in [Basement #] and [Head #] in Figure 23 were not replaced with bound labels in Figure 24. Those # markers were erased because background knowledge shows that houses often have basements and mice always have heads. The concepts [Basement] and [Head] can therefore be assumed as default discourse referents.

In DRS, proper names are represented by monadic relations, but the name *Yojo* seems to be a constant. To show that names are not rigid identifiers, the default ontology for CGs defines [Cat Yojo] as an abbreviation for the following CGIF:

[Cat \*x] [String 'Yojo'] (HasName ?x 'Yojo')

This CGIF can be read *There exists a cat x, 'Yojo' is a character string, and x has 'Yojo' as its name.* In the book *Moby Dick*, Queequeg had an ebony idol named Yojo. If the cat Yojo and the idol Yojo are mentioned in the same context, the full CGIF can be used to distinguish them:

[Cat \*x] [Idol \*y] [String 'Yojo'] (HasName ?x 'Yojo') (HasName ?y 'Yojo')

Names for points in time are treated like names for cats and people. Names with embedded blanks or special characters are enclosed in double quotes: [Time "10:17 pm"]. For computer applications, the recommended standards for dates, times, and measures can be used as names in Common Logic.

As this story shows, background knowledge is needed to determine the patterns of relations among the words and phrases. Formally defined knowledge can be stored in the lexicon and used to compile language to logic, as in Figure 22. But Alan Perlis (1982) warned that it's impossible to translate an informal language to a formal language by any formal algorithm. What makes NLS informal is the unpredictable amount and variety of background knowledge. Missing or implicit knowledge is the

source of vagueness in language, but the many questions about finding it and integrating it with syntax and semantics are still open research problems.

Long before Winograd abandoned his book on semantics, Wittgenstein (1953) recognized the “grave errors” (*schwere Irrtümer*) in the framework he had adopted from his mentors, Frege and Russell. He realized that a vague statement can sometimes be more useful than a precise one:

Frege compares a concept to a region, and says that a region without clear boundaries can’t be called a region at all. This presumably means that we can’t do anything with it. – But is it senseless to say: “Stand roughly here”? (§71)

Peirce used logic to analyze language, not replace it. As he said, the vocabulary of a language embodies “all the ideas that seek expression,” and defining them precisely “is the most stupendous of logical tasks.” He also recognized the importance of vagueness:

It is easy to speak with precision upon a general theme. Only, one must commonly surrender all ambition to be certain. It is equally easy to be certain. One has only to be sufficiently vague. It is not so difficult to be pretty precise and fairly certain at once about a very narrow subject. (CP 4.237)

Vagueness is essential to the flexibility of language. A finite vocabulary cannot have unique words or even unique phrases (patterns of words) for a continuous infinity of possible variations. Most words must therefore have an open-ended variety of senses or microsenses, each with a dynamically varying collection of patterns that map to patterns in the world. Peirce emphasized the continuous transformations that generalize patterns from one “narrow subject” and adapt them to others. For the patterns of those narrow subjects, Wittgenstein coined the term *language game* (Sprachspiel). Words and patterns may be defined precisely for a narrow subject or game, but the transformations can mix or blend patterns from different sources (Sowa 2010).

When patterns are taken out of context and mixed with patterns from sources with different constraints, vagueness is inevitable. Almost any paragraph taken at random from a daily newspaper has patterns that cannot be analyzed by a fixed algorithm. Readers who are familiar with the subject would understand it, but anyone without the background would be confused. A formal logic can represent the patterns of any language game, but a language processor would require heuristic and statistical methods that can find, learn, invent, and adapt new patterns dynamically (Majumdar & Sowa 2009).

## 7. A Moving Picture of Thought

Peirce defined all his versions of logic and their rules of inference in purely formal terms, but he also discussed their linguistic and psychological implications. Among his many intriguing insights are the term *mental diagram* and the claim that existential graphs “put before us moving pictures of thought... in its essence free from physiological and other accidents” (CP 4.8). But he added, “Please note that I have not called it a perfect picture. I am aware that it is not so: indeed, that is quite obvious. But I hold that it is considerably more nearly perfect than it seems to be at first glance, and quite sufficiently so to be called a portraiture of Thought” (CP 4.11).

Pietarinen (2006) showed that Peirce’s mental diagrams and moving pictures are intimately connected to every aspect of his logic and semiotics. The psychologist Johnson-Laird (2002), who had written extensively about mental models, supported Peirce’s claims:

Peirce’s existential graphs... establish the feasibility of a diagrammatic system of reasoning equivalent to the first-order predicate calculus. They anticipate the theory of mental models in many respects, including their iconic and symbolic components, their eschewal of

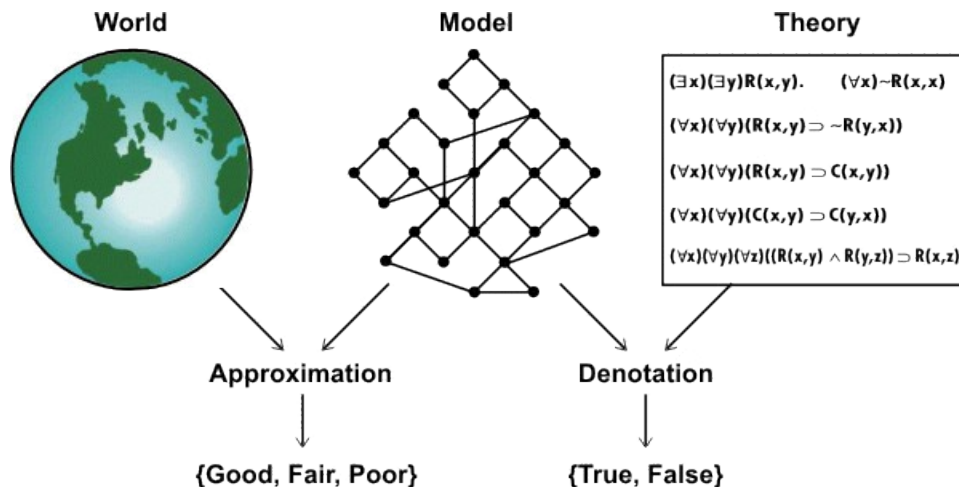
variables, and their fundamental operations of insertion and deletion.

Much is known about the psychology of reasoning... But we still lack a comprehensive account of how individuals represent multiply-quantified assertions, and so the graphs may provide a guide to the future development of psychological theory.

Today, neuroscience has confirmed the psychological evidence for mental models. The neuroscientist Damasio (2010) avoided the word *mental*, but his maps and images refer to the same phenomena:

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

The images form mental models of the real world or the imaginary worlds in our hopes, fears, plans, and desires. Neural processes map them to and from language more efficiently than any computer. Like Tarski's models, they determine the truth of a sentence, but they are flexible, dynamic, and situated in the daily drama of life. Peirce's diagrams can represent Tarski's models, but they can also represent the mental models. Figure 25 shows a Peircean diagram as a Janus-like model with a cognitive side that faces the world and a formal side that faces a theory about the world.



**Figure 25. A model that relates a theory to the world**

On the left is the world, which contains more detail and complexity than any humanly conceivable theory can represent. In the middle is a diagram of a Tarski model that represents a domain  $D$  of individuals and a set  $R$  of relations over  $D$ . If the world had a unique decomposition into discrete objects, the world itself would be a universal model, of which all correct models would be subsets. But the choice of objects and relations depends on the intentions of some agent and the limitations of the agent's measuring instruments. As Peirce observed, all theories are fallible, but some can be "a pretty good" approximation to a "very narrow" aspect of the world. Engineers express that point in a pithy slogan: All models are wrong, but some are useful.

In his *Summa Logicae*, William of Ockham (1323) presented a model-theoretic semantics for a version of controlled Latin. For the logical operators, he stated truth conditions that are equivalent to Tarski's. But he limited the syntax to the sentence patterns of the traditional categorical and hypothetical syllogisms. To relate the truth value of a complex sentence to the truth values of its simple clauses, Ockham's rules were not as general as Tarski's, by they were adequate for him to demonstrate the soundness of the rules of inference for syllogisms.

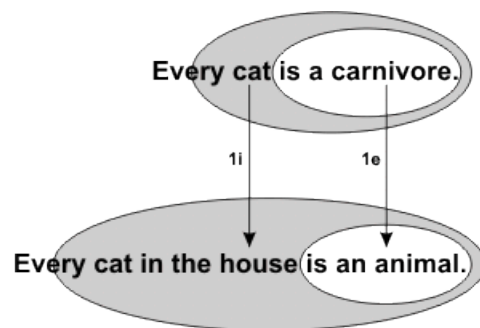


Peirce had studied Ockham in detail, and he used similar semantic arguments to prove the soundness of his own rules of inference. For EGs, he developed a version of model theory he called *endoporeutic* (outside-in evaluation). To evaluate the truth of a graph  $g$  in terms of a model  $M$ , Peirce invented a procedure that is as precise and general as Tarski's, but psychologically more realistic. He defined endoporeutic as a dialog between Graphist, who asserts a graph, and Grapheus, who doubts its truth. Most logicians ignored Peirce's unconventional notation and strange terminology, but Hilpinen (1982) recognized that endoporeutic is a version of *game theoretical semantics* (GTS):  $g$  is true of  $M$  if Graphist has a winning strategy;  $g$  is false if Grapheus has a winning strategy.

Although Tarski's definition is more familiar, Hintikka (1973) showed that GTS is more flexible. By using context-dependent background knowledge, GTS can be adapted to informal languages. To determine the truth value of a sentence  $s$  in terms of a model  $M$ , Tarski defined an *evaluation function*  $\Phi(s,M)$  that considers all possible mappings of individuals in  $M$  to variables in  $s$ . That definition implies a huge amount of unnecessary computation. But GTS can prune irrelevant branches of the game tree with the  $\alpha$ - $\beta$  algorithm used in chess programs (Knuth & Moore 1975). For an introduction to model theory, Barwise and Etchemendy (1993) chose the title *Tarski's World*, but GTS was the method they taught. It's easier to explain, and it allows students to use perception and background knowledge in the same way as a human chess player.

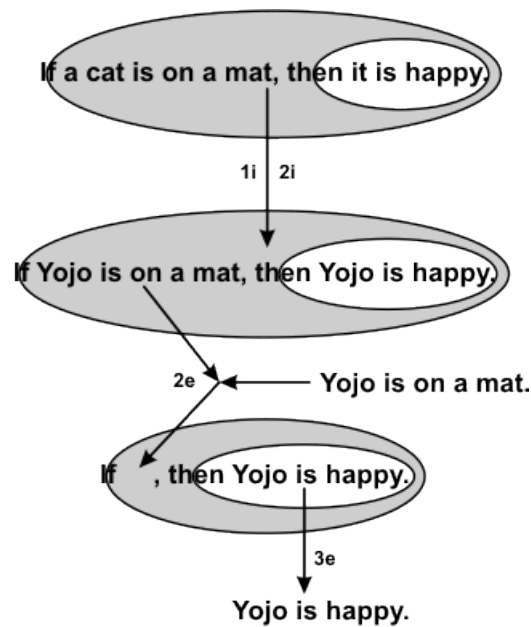
As a theory of mental models, Peirce's EGs and endoporeutic are quite plausible: both the logic and the model are represented as graphs; the neural mechanisms for pattern recognition can support the rules of inference and the model theory; and the GTS evaluation corresponds to an imaginary dialog between a proposer (Graphist) and a skeptic (Grapheus). If a graph  $g$  has no negations, endoporeutic corresponds to a graph mapping (homomorphism) of  $g$  into the model  $M$ :  $g$  is true if it can be mapped to  $M$ ; otherwise,  $g$  is false. If  $g$  has negations, Graphist and Grapheus would take turns peeling off negations and mapping subgraphs of  $g$  to  $M$  (Sowa 2011).

The simplicity and generality of the EG structure, rules of inference, and dialog for evaluating truth or falsity makes EGs a good candidate for a mental logic. As Damasio said, images are "the main currency of our minds." Those images or their neural representations could be mapped to the lines, ovals, and character strings of EGs. As an example, Figure 26 applies Peirce's rules to images of English sentences overlaid with shaded ovals.



**Figure 26. Specializing and generalizing English phrases**

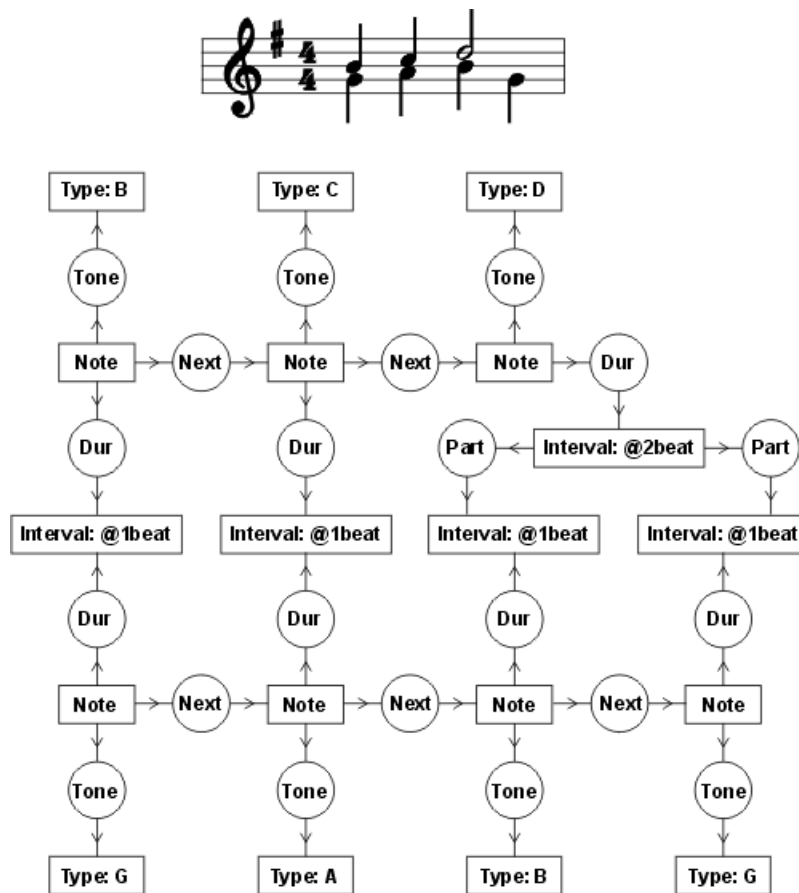
In Figure 26, the word *every*, which is on the boundary of the ovals, creates the same pattern of nested negations as *if-then*. By the rule of insertion in a negative area (1i), the word *cat* can be specialized to *cat in the house*. By the rule of erasure in a positive area (1e), the word *carnivore* can be generalized to *animal*. In this example, the images are printed words, but the mental models could link them to images of animals and houses. Shading could be represented by inhibitory connections in neurons.



**Figure 27. A proof in controlled English by Peirce's rules**

Figure 27 shows a short proof. By the rule of insertion in a negative area (1i), the clause *a cat is on a mat* is specialized to *Yojo is on a mat*. By iteration (2i), a copy of the name Yojo replaces the coreferent pronoun *it*. The replacement of a pronoun with a name corresponds to replacing an indexical marker with a bound label in Figure 24. By deiteration (2e), the clause *Yojo is on a mat* is erased because it is identical to a sentence in the outer area. Finally, the shaded area and the operators *if* and *then* are erased by the rule of double negation (3e). The rules illustrated in Figures 26 and 27 are sound for a version of controlled English. But as the story about Yojo and the mouse showed, unrestricted English requires more background knowledge and more ways of using it.

Although EGs can represent a moving picture of the mind in thought, they are not a perfect picture, as Peirce admitted. Many kinds of diagrams are better tailored to specialized subjects. For music, Figure 28 shows one measure in the usual notation. Below it is a note-by-note translation to the boxes and circles of a conceptual graph. An experienced musician can read music notation at sight and play it at full speed. By contrast, the CG reflects the laborious analysis of a beginner who examines each note to determine its tone, duration, and relationship to other tones on the same beat or the next beat.




**Figure 28. Two diagrams for representing the logic and ontology of music**

The diagrams in Figure 28 illustrate issues about language, logic, ontology, and diagrams of any kind. Ancient flutes and stringed instruments show that people understood the logic and ontology of music thousands of years before they had any notation for recording them. The masters taught their students how to play those instruments by example and by explanations in whatever language they spoke. For beginners, those explanations probably used as many words as the boxes and circles in the CG. The logic and ontology of that language would have a direct mapping to and from CGs. For more advanced students, the master would continue to make comments in language and other signs. But the signs could be brief because the students would use their newly acquired knowledge to fill in the implicit detail. Similar principles apply to diagrams of any kind:

- *Logic.* The two most common logical operators are existence and conjunction. Any mark on a page indicates the existence of something represented by that mark. Conjunction is implicit: any marks or patterns of marks in the same area are related by *and*. Other logical operators are much less common.
- *Ordered relations.* A direction in space can represent an ordered relation. In music, time flows from left to right, and two notes in sequence are related by Next with an implicit *and*. Relative pitch is represented vertically, and absolute pitch is marked by lines on the staff. For a partial ordering, such as a type hierarchy, vertical or horizontal position can show the direction, and alternatives can be shown by branching lines.
- *Unordered relations.* Most relations don't have an ordering, linear or partial. They can be marked by a word or other symbol and connected to whatever they relate by lines, adjacency, or some enclosure. Ordered relations can also be marked by symbols. In music, for example,

tempo is a continuous ordering represented by discrete words such as *allegro* or *andante*.

- *Coreference*. Diagrams can eliminate variables by linking all references to a single node for each individual. If an individual is shown in more than one area, coreference can be indicated by lines, as in EGs, or by a choice of lines or labels, as in CGs.
- *Uniqueness*. In perception, two objects seen in different locations in the same view are considered distinct, but similar objects seen at different times could be the same or different. In logic, two existential quantifiers with different variables, such as  $(\exists x)$  and  $(\exists y)$ , do not imply  $x \neq y$ . Diagrams may have various defaults and conventions for overriding them.
- *Precision*. Some diagrams are formally defined, but others are as ambiguous as any NL text. As Figure 28 shows, music notation leaves much of the ontology implicit, but it is sufficiently precise to be compiled to logic. Some diagrams that look “simple” omit so much background knowledge that nobody but the author could interpret them.
- *Images*. Photographs and drawings are the most exact icons. Some words, such as *ring*, are auditory icons of the sounds they represent. But as Peirce said, symbols evolve from icons. The symbol  no longer resembles a typical telephone, whose ring tone rarely sounds like the word *ring* and is usually far more complex than a single tone.

Diagrams lie on a continuum from a photograph or phonograph, to a drawing or transcription, and then to the stylized versions in Figure 28. They all preserve some aspects of the original structure, but they differ in the amount of detail, the selection of detail, the accuracy of the mapping, and the suitability for various purposes. For a musician with long years of practice, the traditional notation has become “a moving picture of the action of the mind in thought.” But it can be mapped to and from a CG. Neither notation is identical to anything in the brain, but the structure that is common to both could serve as a hypothesis about the neural patterns.

## References

- Barwise, Jon, & John Etchemendy (1993) *Tarski's World*, Stanford: CSLI.
- Barwise, Jon, J. M. Gawron, G. Plotkin, & S. Tutiya, eds. (1991) *Situation Theory and its Applications*, vol. 2, San Francisco, CA: CSLI. For a summary, [http://www.stanford.edu/~kdevlin/Papers/HHL\\_SituationTheory.pdf](http://www.stanford.edu/~kdevlin/Papers/HHL_SituationTheory.pdf)
- Barwise, Jon, & John Perry (1983) *Situations and Attitudes*, Cambridge, MA: MIT Press.
- Boole, George (1854) *An Investigation into the Laws of Thought*, reprinted, New York: Dover, 1958.
- Brachman, Ronald J. (1979) On the epistemological status of semantic networks, in N. V. Findler, *Associative Networks: Representation and Use of Knowledge by Computers*, New York: Academic Press, pp. 3-50.
- Brachman, Ronald J., Richard E. Fikes, & Hector J. Levesque (1983) KRYPTON: A functional approach to knowledge representation, *IEEE Computer* **16:10**, 67-73.
- Carnap, Rudolf (1947) *Meaning and Necessity*, Chicago: University of Chicago Press. Second edition 1956.
- Chen, Weidong, Michael Kifer, & David S. Warren (1993) Hilog: A Foundation for Higher-Order Logic Programming, *Journal of Logic Programming* **15:3**, 187-230.
- Damasio, Antonio R. (2010) *Self Comes to Mind: Constructing the Conscious Brain*, New York: Pantheon Books.
- Davidson, Donald (1967) The logical form of action sentences, reprinted in D. Davidson (1980) *Essays on Actions and Events*, Oxford: Clarendon Press, pp. 105-148.
- Devlin, Keith (1991a) Situations as mathematical abstractions, in Barwise et al. (1991) pp. 25-39.
- Frege, Gottlob (1879) *Begriffsschrift*, English translation in J. van Heijenoort, ed. (1967) *From Frege to Gödel*, Cambridge, MA: Harvard University Press, pp. 1-82.
- Frost, Robert (1963) *A Lover's Quarrel with the World*, filmed interview, Boston: WGBH Educational Foundation.

- Gentzen, Gerhard (1935) Untersuchungen über das logische Schließen, translated as Investigations into logical deduction in *The Collected Papers of Gerhard Gentzen*, ed. and translated by M. E. Szabo, Amsterdam: North-Holland, 1969, pp. 68-131.
- Hayes, Patrick, & Chris Menzel (2001) A semantics for the Knowledge Interchange Format, *Proc. IJCAI 2001 Workshop on the IEEE Standard Upper Ontology*, Seattle: IJCAI.
- Hayes, Patrick, & Chris Menzel (2006) IKL Specification Document, <http://www.ihmc.us/users/phayes/IKL/SPEC/SPEC.html>
- Hays, David G. (1964) Dependency theory: a formalism and some observations, *Language* **40:4**, 511-525.
- Hintikka, Jaakko (1973) *Logic, Language Games, and Information*, Oxford: Clarendon Press. For a summary and review of the many variations of logic games, <http://plato.stanford.edu/entries/logic-games/>
- Hilpinen, Risto (1982) On C. S. Peirce's theory of the proposition: Peirce as a precursor of game-theoretical semantics, *The Monist* **65**, 182-88.
- Houser, Nathan (2005) Peirce in the 21st century, *Transactions of the Charles S. Peirce Society* **41:4**, p. 729.
- ISO/IEC (2007) *Common Logic (CL) — A Framework for a family of Logic-Based Languages*, IS 24707, Geneva: International Organisation for Standardisation. <http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html>
- Johnson-Laird, Philip N. (1983) *Mental Models*, Cambridge, MA: Harvard University Press. For related publications, <http://mentalmodels.princeton.edu/publications/>
- Johnson-Laird, Philip N. (2002) Peirce, logic diagrams, and the elementary operations of reasoning, *Thinking and Reasoning* **8:2**, 69-95. <http://mentalmodels.princeton.edu/papers/2002peirce.pdf>
- Kamp, Hans (1981) A theory of truth and semantic representation, in *Formal Methods in the Study of Language*, ed. by J. A. G. Groenendijk, T. M. V. Janssen, & M. B. J. Stokhof, Amsterdam: Mathematical Centre Tracts, pp. 277-322. For related publications, <http://www.ims.uni-stuttgart.de/institut/mitarbeiter/hans/>
- Kamp, Hans, & Uwe Reyle (1993) *From Discourse to Logic*, Dordrecht: Kluwer.
- Kamp, Hans, & Uwe Reyle (1996) A calculus for first order Discourse Representation Structures, *Journal of Logic, Language, and Information* **5:3-4**, 297-348.
- Knuth, Donald E., & Ronald W. Moore (1975), An analysis of alpha-beta pruning, *Artificial Intelligence* **6:4**, 293-326.
- Majumdar, Arun K., & John F. Sowa (2009) Two paradigms are better than one and multiple paradigms are even better, in S. Rudolph, F. Dau, and S.O. Kuznetsov, eds., *Proceedings of ICCS'09*, LNAI 5662, Berlin: Springer, pp. 32-47. <http://www.jfsowa.com/pubs/paradigm.pdf>
- Levinson, R. A., & G. Ellis (1992) Multilevel hierarchical retrieval, *Knowledge Based Systems* **5:3**, pp. 233-244.
- Peano, Giuseppe (1889) *Aritmetices principia nova methoda exposita*, Torino: Bocca.
- Peirce, Charles Sanders (1870) Description of a notation for the logic of relatives, reprinted in *Writings of Charles S. Peirce*, Bloomington: Indiana University Press, vol. 2, pp. 359-429.
- Peirce, Charles Sanders (1880) On the algebra of logic, *American Journal of Mathematics* **3**, 15-57.
- Peirce, Charles Sanders (1885) On the algebra of logic, *American Journal of Mathematics* **7**, 180-202.
- Peirce, Charles Sanders (1898) *Reasoning and the Logic of Things*, The Cambridge Conferences Lectures of 1898, ed. by K. L. Ketner, Cambridge, MA: Harvard University Press, 1992.
- Peirce, Charles Sanders (1902) Vague, in J. M. Baldwin, ed., *Dictionary of Philosophy and Psychology*, New York: MacMillan, p. 748. For a quotation and discussion, <http://plato.stanford.edu/entries/vagueness/>
- Peirce, Charles Sanders (1906) Manuscripts on existential graphs, *Collected Papers of Charles Sanders Peirce*, vol. 4, Harvard University Press, Cambridge, MA, pp. 320-410.
- Peirce, Charles Sanders (1911) Letter to J. H. Kehler, in Peirce (NEM) 3:159–210.
- Peirce, Charles Sanders (CP) *Collected Papers of C. S. Peirce*, ed. by C. Hartshorne, P. Weiss, & A. Burks, 8 vols., Cambridge, MA: Harvard University Press, 1931-1958.
- Peirce, Charles Sanders (NEM) *The New Elements of Mathematics*, edited by Carolyn Eisele, 4 vols., The Hague: Mouton & Co., 1976.
- Perlis, Alan J. (1982) Epigrams in Programming, *SIGPLAN Notices*, Sept. 1982, ACM. <http://www.cs.yale.edu/homes/perlis-alan/quotes.html>



- Pietarinen, Ahti-Veikko (2006) *Signs of Logic: Peircean Themes on the Philosophy of Language, Games, and Communication*, Synthese Library, vol. 329, Berlin: Springer. For publications, <http://www.helsinki.fi/~pietarin/>
- Quine, Willard Van Orman (1953) Reduction to a dyadic predicate, *J. Symbolic Logic* **19**, reprinted in Quine, *Selected Logic Papers*, Cambridge, MA: Harvard University Press, 1995, pp. 224-226.
- Rhodes, James, Stephen Boyer, Jeffrey Kreulen, Ying Chen, & Patricia Ordonez (2007) Mining patents using molecular similarity search, *Pacific Symposium on Biocomputing* **12**, 304-315.
- Robinson, J. Alan (1965) A machine oriented logic based on the resolution principle, *Journal of the ACM* **12**, 23-41.
- Schank, Roger C., ed. (1975) *Conceptual Information Processing*, North-Holland Publishing Co., Amsterdam.
- Sowa, John F. (1984) *Conceptual Structures: Information Processing in Mind and Machine*, Reading, MA: Addison-Wesley. For related publications, <http://www.jfsowa.com/pubs>
- Sowa, John F. (1992) Semantic networks, in S. C. Shapiro (ed), *Encyclopedia of Artificial Intelligence*, second edition, New York: Wiley, pp. Revised and extended version: <http://www.jfsowa.com/pubs/semnet.htm>
- Sowa, John F. (2000) *Knowledge Representation: Logical, Philosophical, and Computational Foundations*, Brooks/Cole Publishing Co., Pacific Grove, CA. For excerpts, <http://www.jfsowa.com/ontology>
- Sowa, John F. (2003) Laws, facts, and contexts: Foundations for multimodal reasoning, in *Knowledge Contributors*, V. F. Hendricks, K. F. Jørgensen, & S. A. Pedersen, eds., Dordrecht: Kluwer, pp. 145-184. <http://www.jfsowa.com/pubs/laws.htm>
- Sowa, John F. (2006a) Peirce's contributions to the 21st Century, in H. Schärfe, P. Hitzler, & P. Øhrstrøm, eds., *Conceptual Structures: Inspiration and Application*, LNAI 4068, Berlin: Springer, pp. 54-69. <http://www.jfsowa.com/pubs/csp21st.pdf>
- Sowa, John F. (2006b) Worlds, models, and descriptions, *Studia Logica*, Special Issue *Ways of Worlds II* **84:2**, 323-360. <http://www.jfsowa.com/pubs/worlds.pdf>
- Sowa, John F. (2010) The role of logic and ontology in language and reasoning, in R. Poli & J. Seibt, eds., *Theory and Applications of Ontology: Philosophical Perspectives*, Berlin: Springer, pp. 231-263. <http://www.jfsowa.com/pubs/rolelog.pdf>
- Sowa, John F. (2011) Peirce's tutorial on existential graphs, *Semiotica* **186:1-4**, 345-394. <http://www.jfsowa.com/pubs/egtut.pdf>
- Sowa, John F., & Arun K. Majumdar (2003) Analogical reasoning, in A. de Moor, W. Lex, & B. Ganter, eds. (2003) *Conceptual Structures for Knowledge Creation and Communication*, LNAI 2746, Berlin: Springer, pp. 16-36. <http://www.jfsowa.com/pubs/analog.htm>
- Stewart, John (1996) *Theorem Proving Using Existential Graphs*, MS Thesis, Computer and Information Science, University of California at Santa Cruz.
- Tarski, Alfred (1933) The concept of truth in formalized languages, in A. Tarski, *Logic, Semantics, Metamathematics*, Second edition, Indianapolis, Hackett, pp. 152-278. For summary, <http://plato.stanford.edu/entries/tarski-truth/>
- Tesnière, Lucien (1959) *Éléments de Syntaxe structurale*, corrected edition, Paris: Klincksieck, 1988. For an introduction based on a summary by Tesnière, <http://www.home.uni-osnabrueck.de/bschwisc/archives/tesniere.pdf>
- Turing, Alan M. (1939) Systems of logic defined by ordinals, reprinted in M. Davis, ed., *The Undecidable*, New York: Raven Press, pp. 154-222. For more about oracles, <http://www.people.cs.uchicago.edu/~soare/History/turing.pdf>
- Whitehead, Alfred North, & Bertrand Russell (1910) *Principia Mathematica*, 2nd edition, Cambridge: University Press, 1925.
- Whitehead, Alfred North (1937) Analysis of meaning, *Philosophical Review*, reprinted in A. N. Whitehead, *Essays in Science and Philosophy*, New York: Philosophical Library, pp. 122-131.
- Winograd, Terry (1972) *Understanding Natural Language*, New York: Academic Press.
- Winograd, Terry (1983) *Language as a Cognitive Process. Volume I: Syntax*, Reading, MA: Addison-Wesley.
- Winograd, Terry, & Fernando Flores (1986) *Understanding Computers and Cognition*, Norwood, NJ: Ablex.
- Wittgenstein, Ludwig (1953) *Philosophische Untersuchungen*, translated by G. E. M. Anscombe, P. M. S. Hacker, & Joachim Schulte as *Philosophical Investigations*, fourth edition, Oxford: Blackwell, 2009.
- Woods, William A. (1975) What's in a link: foundations for semantic networks, in D. G. Bobrow & A. Collins, eds. (1975) *Representation and Understanding*, New York: Academic Press, pp. 35-82.