ISSN PRINT 2055-3706
ISSN ONLINE 2055-3714

Journal of Applied Logics

The IfCoLog Journal of Logics and their Applications

Volume 5 • Issue 5 • August 2018

Special Issue

Logical Foundations of Strategic Reasoning

Guest Editor

Woosuk Park
John Woods

Available online at
www.collegepublications.co.uk/journals/ifcolog/
Free open access

Reasoning with Diagrams and Images

John F. Sowa

sowa@bestweb.net

Abstract

Visualization and analogy are the heart and soul of mathematics. Long before they write a formal proof, mathematicians train their intuition on diagrams, visualize novel patterns, and discover creative analogies. For over two millennia, Euclid's diagrammatic methods set the standard for rigorous proof. But the abstract algebra of the 19th century led many mathematicians to claim that all formal reasoning must be algebraic. Yet Peirce and Polya recognized that Euclid's diagrammatic reasoning is a better match to human thought patterns than the algebraic rules. A combination of Peirce's graph logic, Polya's heuristics, and Euclid's diagrams is a better candidate for a natural logic than any algebraic formalization. This combination justifies Peirce's claim that his graph logic generates "a moving picture of the action of the mind in thought." Formally, it can be specified as rigorously as any algebraic proof in mathematics or computer science.

With his existential graphs (EGs), Peirce developed a graphic version of logic that was as precise and expressive as his linear notation for predicate calculus. He also hinted at the possibility of generalizing EGs to accommodate complex diagrams in more than two dimensions. This article develops Peirce's hints by adding two rules of inference: observation and imagination. By observation, any fact that may be observed in a diagram may be asserted by a formula in any version of logic, linear or graphic. By imagination, a statement of the form "If diagram, then formula" may be asserted if the formula is true of the diagram. For diagrams as simple and precise as Euclid's, these rules are sound: they preserve truth. For more complex diagrams, the rules are as dependable or fallible as the methods of observation and testing.

1 Languages, Logics, and Imagery

Is language based on logic? Is logic based on language? Or are they both based on imagery? These questions have been debated since antiquity, and modern cognitive science is beginning to provide some answers. To illustrate the issues, consider sentences spoken by a child named Laura shortly before her third birthday [32]:

> Here's a seat. It must be mine if it's a little one.
> I want this doll because she's big.
> When I was a little girl I could go "geek-geek" like that. But now I can go "this is a chair."

Laura's content words express concrete images, directly related to her actions. But her syntax and function words express a surprising amount of complex logic: possibility, necessity, tenses, indexicals, conditionals, causality, quotations, and metalanguage about her own language. As another example, a mother was talking with her son, who was about the same age as Laura [27]:

> Mother: Which of your animal friends will come to school today?
> Son: Big Bunny, because Bear and Platypus are eating.

The mother looked in his room, where the stuffed bear and the platypus were sitting in a chair and "eating". The boy had imagined a situation, built a model of it, and based his reasoning on it: The bear and the platypus are eating. They can't eat and go to school at the same time. Big Bunny isn't doing anything. Therefore, Big Bunny is available.

The reasoning abilities of children have challenged the syntax-based theories of logic and linguistics. The semantic aspects of imagery, action, and feelings appear to be more important than syntax. This point is supported by studies of infants with one deaf parent and one speaking parent. At every stage of development, they have equal ability to express themselves in one-dimensional speech or in moving, three-dimensional gestures [49]. In fact, infants with two deaf parents babble with their hands, not with vocal sounds. The neuroscientist Antonio Damasio [2]) summarized the issues:

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

The maps and images form mental models of the real world or of the imaginary worlds in our hopes, fears, plans, and desires. They provide a "model theoretic" semantics for language that uses perception and action for testing models against reality. Like Tarski's models, they define the criteria for truth, but they are flexible, dynamic, and situated in the daily drama of life.

The logician, mathematician, scientist, and philosopher Charles Sanders Peirce would agree. Although he invented the algebraic notation for predicate calculus [43],

Peirce claimed that all reasoning is based on a "a concrete, but possibly changing, mental image" that may be aided by "a drawing or a model":

> All necessary reasoning without exception is diagrammatic. That is, we construct an icon of our hypothetical state of things and proceed to observe it. This observation leads us to suspect that something is true, which we may or may not be able to formulate with precision, and we proceed to inquire whether it is true or not. For this purpose it is necessary to form a plan of investigation, and this is the most difficult part of the whole operation. We not only have to select the features of the diagram which it will be pertinent to pay attention to, but it is also of great importance to return again and again to certain features. [47, Vol 2, p.21]
>
> The word *diagram* is here used in the peculiar sense of a concrete, but possibly changing, mental image of such a thing as it represents. A drawing or model may be employed to aid the imagination; but the essential thing to be performed is the act of imagining. Mathematical diagrams are of two kinds; 1st, the geometrical, which are composed of lines (for even the image of a body having a curved surface without edges, what is mainly seen by the mind's eye as it is turned about, is its generating lines, such as its varying outline); and 2nd, the algebraical, which are arrays of letters and other characters whose interrelations are represented partly by their arrangement and partly by repetitions. If these change, it is by instantaneous metamorphosis. [48, Vol 4, p. 219]

For informal reasoning, stuffed animals or objects of any kind can aid the imagination in building models and reasoning about them. For mathematical reasoning, Euclid's diagrams are the classical examples. But moving or changing images can be even more suggestive. Figure 1 shows a diagram that inspired Archimedes to imagine polygons converging to a circle. As the number of sides increases, the perimeter of the inner polygon expands, and the perimeter of the outer polygon shrinks. The limit for both is the circumference of the circle. If the diameter of the circle is 1.0, the limit is π. By using 96-agons, Archimedes approximated π as $22/7 = 3.142857...$, an error of 0.004%.

Figure 1 embodies two creative insights. First, Archimedes recognized that the perimeters of the outer polygon and the inner polygon are upper and lower bounds on the circumference of the circle. Second, the two bounds come closer together as the number of sides increases. With those two insights, a good mathematician with enough time and enough paper could compute π to any desired approximation.

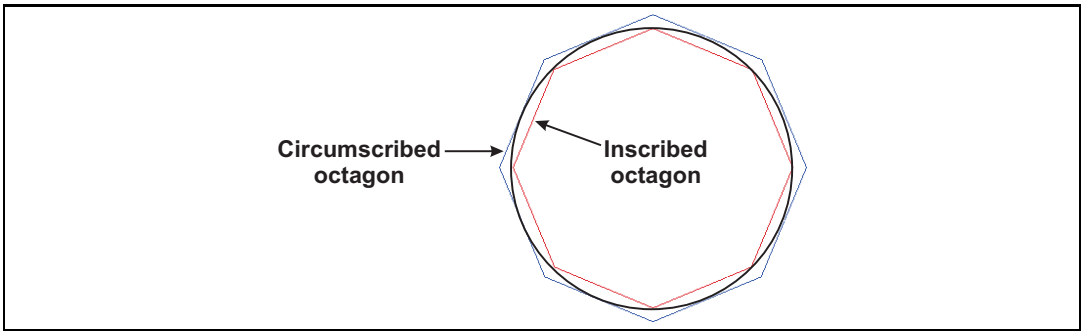

Figure 1: Approximating π by polygons converging to a circle

There were many good mathematicians before Archimedes, but none of them had those two insights. The mathematician Paul Halmos [13] said that the essence of creativity consists of visualizing novel analogies and guessing which ones are the most promising to pursue:

> Mathematics — this may surprise or shock some — is never deductive in its creation. The mathematician at work makes vague guesses, visualizes broad generalizations, and jumps to unwarranted conclusions. He arranges and rearranges his ideas, and becomes convinced of their truth long before he can write down a logical proof... the deductive stage, writing the results down, and writing its rigorous proof are relatively trivial once the real insight arrives; it is more the draftsman's work not the architect's.

To study and compare the thought processes of mathematicians, Jacques Hadamard [12] asked some of the most creative to answer a few questions. Their responses were similar to the comments by Peirce and Halmos. Einstein emphasized visual and muscular images and their combinations:

> The words or the language, as they are written or spoken, do not seem to play any role in my mechanism of thought. The psychical entities which seem to serve as elements in thought are certain signs and more or less clear images which can be voluntarily reproduced and combined... The above-mentioned elements are, in my case, of visual and some of muscular type. Conventional words or other signs have to be sought for laboriously only in a secondary stage, when the mentioned associative play is sufficiently established and can be reproduced at will.

For teaching students how to think, visualize, and guess, George Polya [51] presented *heuristics* or methods of plausible reasoning that promote the insight:

> You have to guess a mathematical theorem before you prove it; you have to guess the idea of a proof before you carry through the details. You have to combine observations and follow analogies; you have to try and try again. The results of the mathematician's creative work is demonstrative reasoning, a proof; but the proof is discovered by plausible reasoning, by guessing.

Since Peirce understood the importance of visualization for discovering a proof, he experimented with graph notations that used visual methods in the proof itself. His first version, the *relational graphs* of 1882, could express a subset of first-order logic. But like the early semantic networks in artificial intelligence, Peirce's relational graphs couldn't delimit the scope of quantifiers and negations. A dozen years later, he extended his relational graphs to *existential graphs* (EGs) by using nested ovals to represent scope. Following is an EG for the sentence "A cat is on a mat" and its translation to predicate calculus (Peirce's algebra with Peano's choice of symbols):

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

In existential graphs, a line by itself represents something that exists. The labels 'Cat', 'On', and 'Mat' represent relations. In combination, the EG states "Some cat is on some mat." The predicate calculus states "There is an x, there is a y, x is a cat and y is a mat and x is on y." For the image of a cat on a mat, English, EG, and predicate calculus could all be called diagrammatic because their mapping is simple and direct: the labels map to the two objects and the relation between them. But the predicate calculus is less digrammatic than the others because its syntactic details are more complex.

Before Peirce published his algebraic notation, Frege [9] had published a tree notation for FOL, which he called *Begriffsschrift* (concept writing). His goal was to show the reasoning steps explicitly, not to represent imagery or natural language. Figure 2 shows the Begriffsschrift for the cat sentence.

The symbols in Figure 2, reading from left to right, consist of a vertical bar for assertion, a short vertical bar for negation, two cup symbols for universal quantifiers with the variables x and y, two hooks for implication, another negation, and the same three relation labels as the EG or the predicate calculus. In English, the Begriffsschrift may be read "Assert: it is false that for every x, for every y, if x is a cat then if y is a mat, then it is false that x is on y." It is equivalent to the following predicate calculus: $\sim \forall x \forall y(\mathrm{Cat}(x) \supset (\mathrm{Mat}(y) \supset \sim \mathrm{On}(x, y)))$

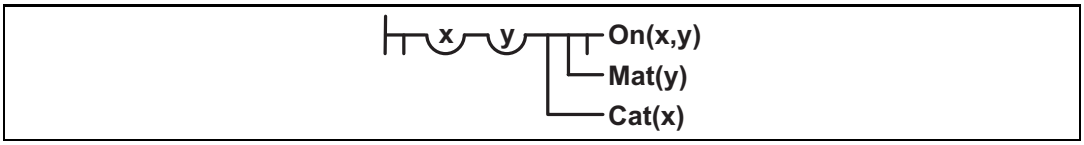

Figure 2: Frege's Begriffsschrift for "A cat is on a mat."

Although many logicians considered Begriffsschrift unreadable, it could be considered diagrammatic by Peirce's criteria. Frege designed the notation to highlight the relationship between his logical operators and his rules of inference. Occurrences of the three operators $\forall$, $\supset$, and $\sim$ trigger the three rules of inference: universal instantiation, modus ponens, and double negation. But the operators of existence and conjunction have a more direct mapping to imagery and ordinary language. If Frege had added a square cup for existence and the symbol & for conjunction, an extended Begriffsschrift would resemble predicate calculus (Figure 3).

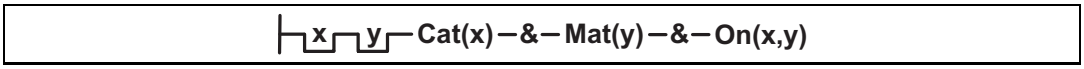

Figure 3: Extended Begriffsschrift with operators for existence and conjunction

But Frege refused to add more operators to Begriffsschrift, which he regarded as "a device invented for certain scientific purposes, and one must not condemn it because it is not suited to others." He had no desire to simplify the mapping to the language of everyday life (*Sprache des Lebens*). Instead, he hoped "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."

Peirce, however, considered language and logic as aspects of the broader field of semiotic. Instead of a sharp dichotomy, he emphasized the continuum of formal and informal methods. The logic-based methods of induction, abduction, and deduction are based on the same kinds of analogies used in observation and imagination. That view led Peirce to a better balance of readability and proof theory. His existential graphs not only have a simpler mapping to and from language and imagery, they also have a direct mapping to simpler rules of inference with an elegant version of model theory. For these reasons, Peirce called EGs his "chef d'oeuvre" and claimed that their rules of inference generate "a moving picture of the mind in thought." After a detailed comparison of Peirce's EGs to current theories about mental models, the psychologist Johnson-Laird [22] agreed:

> Peirce's existential graphs are remarkable. They 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.

The remainder of this article develops these ideas. Section 2 summarizes the syntax of existential graphs and their mapping to the linear Existential Graph Interchange Format (EGIF). Section 3 presents the EG rules of inference, and Section 4 uses Peirce's *endoporeutic* to show that the rules are sound. Section 5 shows how arbitrary images can be inserted in any EG and used in a proof. As an example, Euclid's proofs can be translated line-by-line to EGs that use Peirce's rules of inference to construct Euclid's diagrams. Section 6 presents research issues that are raised and sometimes solved by these methods. Finally, Section 7 presents evidence for the claim that EGs supplemented with images are a promising candidate for a natural logic.

2 Existential Graph Syntax

Charles Sanders Peirce was a pioneer in logic. Although Frege published the first complete system of first-order logic in 1879, no one else adopted his notation. Peirce published the algebraic version of FOL and HOL in 1885. With a change of symbols by Peano and some extensions by Whitehead and Russell, Peirce-Peano algebra is still the most widely used logic today [52]. But as early as 1882, Peirce experimented with graph notations to express "the atoms and molecules of logic." In 1897, he developed existential graphs (EGs) as a notation that expressed the semantics of first-order predicate calculus with equality. For the first-order subset of EGs, Peirce used the same structure and rules of inference in every manuscript from 1897 to 1911. But he experimented with variations, extensions, and semantic foundations [54]. For the graphics, this article uses the notation he preferred after 1906. Unless otherwise cited, all quotations by Peirce are from the *New Elements of Mathematics*, pages 3:162 to 3:169.

As an example, the EG on the left of Figure 4 asserts that there is a phoenix. The line, which he called a *line of identity*, represents existence. By itself, the line asserts "There is something." The word phoenix is the *name* of a relation type. The line attached to the name asserts "There exists a phoenix." As Peirce explained,

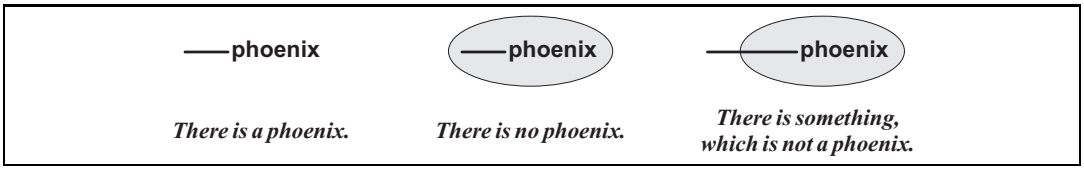

Figure 4: Three existential graphs

	EGIF	Predicate Calculus
Left	`(phoenix *x)`	$\exists x$ phoenix(x)
Middle	∼`[(phoenix *x)]`	$\sim \exists$ phoenix(x)
Right	`[*x]` ∼ `[(phoenix x)]`	$\exists x \sim$ phoenix(x)

Table 1: EGIF and predicate calculus for Figure 4

> To deny that there is any phoenix, we shade that assertion which we deny as a whole [EG on the left of Figure 4]. Thus what I have just scribed [EG in the middle] means "It is false that there is a phoenix." But the [EG on the right] only means "There is something that is not identical with any phoenix."

To indicate negation in his early EGs, Peirce used an unshaded oval enclosure, which he called a *cut* or a *sep* because it cut or separated the *sheet of assertion* into a positive (outer) area and a negative (inner) area. In the later versions, he added shading to highlight the distinction between positive and negative areas. Table 1 shows the Existential Graph Interchange Format (EGIF) for each of the graphs in Figure 4 and the corresponding formula in predicate calculus (Peirce-Peano algebra).

In the EGIF for the graph on the left of Figure 4, the parentheses enclose the name `phoenix` of a *monad* (monadic relation) and a *defining label* `*x`. The defining label represents the beginning of a line of identity in the graphic EG. It is mapped to an existentially quantified variable $\exists x$ in predicate calculus. For the EG in the middle, the shaded oval is represented in EGIF by a tilde ∼ for negation and a pair of brackets ∼`[ ]`. For the EG on the right, the beginning of the line of identity is outside the shaded oval. Since there is no relation outside the negation, the defining label is contained in a *coreference node* `[*x]` in front of the negation. Inside the parentheses of the relation, the label `x` without an asterisk is a *bound label* that is within the *scope* of the defining label `*x` on the outside.

When an oval is drawn inside another oval, the doubly nested area is positive (unshaded), as in Figure 5. Any area nested inside an odd number of ovals is shaded, and any area inside an even number of ovals (possibly zero) is unshaded. As Peirce

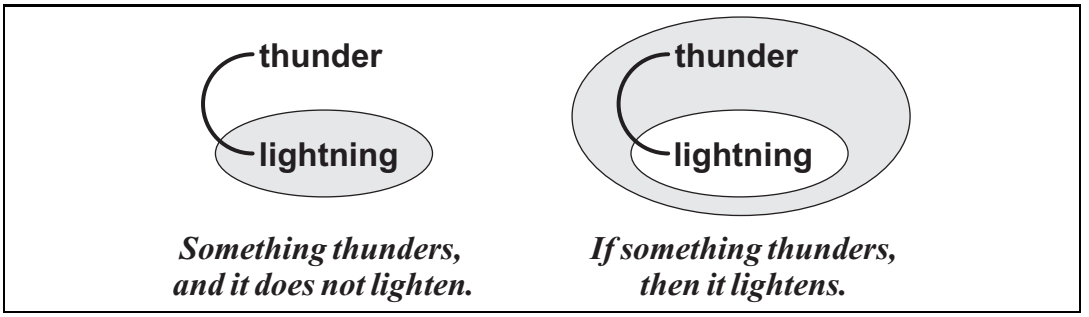

Figure 5: An EG with one negation and an EG with two negations

	EGIF	Predicate Calculus
Left	`(thunder *x)` `~[(lightning x)]`	$\exists x(\text{thunder}(x) \wedge$ $\sim \text{lightning}(x))$
Right	`~[(thunder *x)` `~[(lightning x)]]`	$\sim \exists x(\text{thunder}(x) \wedge$ $\sim \text{lightning}(x))$
Optional	`[If (thunder *x)` `[Then (lightning x)]]`	$\forall x(\text{thunder}(x) \supset \text{lightning}(x))$

Table 2: EGIF and Predicate Calculus for Figure 5

said, "The graph [on the left] asserts that it thunders without lightning... a denial shades the unshaded and unshades the shaded. Consequently [the graph on the right] means 'If it thunders, it lightens'."

In Table 2 the conjunction (AND operator) is implicit in EGIF; no symbol is required. But the formula in predicate calculus requires the symbol $\wedge$. In the last line of the table, the EGIF uses the option of replacing the two negations with the keywords `If` and `Then`; the predicate calculus uses the universal quantifier $\forall$ and the implication $\supset$.

In EG and EGIF, a nest of two negations represents an if-then statement in English. Peirce called that combination a *scroll*. To improve readability, the optional line in the table shows how the tildes for the two negations may be replaced by the keywords `If` and `Then`. In predicate calculus, $\sim \exists$ is equivalent to $\forall \sim$. With that conversion, the formula for Figure 5 becomes $\forall x \sim (\texttt{thunder}(x) \wedge \sim \texttt{lightning}(x))$. By the definition of the implication operator $\supset$, the formula $\sim (p \wedge \sim q)$ is equivalent to $p \supset q$. Therefore, the formula for the EG on the right of Figure 5 may be rewritten as $\forall x(\texttt{thunder}(x) \supset \texttt{lightning}(x))$. In English, this formula may be read "For every x, if x thunders, then x lightens." This example shows that the EG and EGIF for if-then, has a more direct mapping to English than the operator $\supset$ in predicate

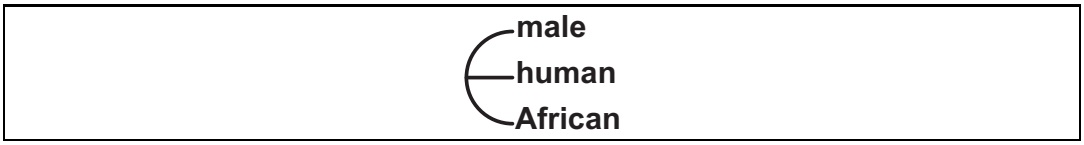

Figure 6: A teridentity is a ligature that connects three lines

	EGIF	Predicate Calculus
Teridentity	`(male *x)(human *y)` `(African *z) [x y z]`	$\exists x \exists y \exists z (\text{male}(x) \wedge \text{human}(y) \wedge \text{african}(z) \wedge x = y \wedge y = z)$
One label	`(male *x) (human x)` `(African x)`	$\exists x (\text{male}(x) \wedge \text{human}(x) \wedge \text{african}(x))$

Table 3: EGIF and Predicate Calculus for Figure 6

calculus. In fact, EGs are isomorphic to the *discourse representation structures*, which Kamp and Reyle [24] developed for representing natural language semantics.

> A graph may be complex or indivisible. Thus [Figure 6 shows] a graph instance composed of instances of three indivisible graphs which assert 'there is a male', 'there is something human', and 'there is something African'. The syntactic junction or point of *teridentity* asserts the identity of something denoted by all three.

In EGIF, Figure 6 may be represented by three indivisible nodes followed by a coreference node `[x y z]`, which shows that that three lines of identity refer to the same individual. Peirce called such a junction a ligature. Since all three lines refer to the same thing, the defining labels `*y` and `*z` may be replaced by bound labels for `*x`. Then the coreference node is no longer needed, and it may be deleted, as shown in the following table:

In modern terminology, Peirce's indivisible graphs are called *atoms*. In predicate calculus, each atom consists of a single relation followed by its list of *arguments*, which Peirce called logical subjects. In EGs, an N-adic relation has N *pegs*, each of which is attached to a line of identity for its logical subject. In EGIF, each atom is represented by a pair of parentheses that enclose a relation name followed by a list of defining labels or bound labels for its logical subjects..

Peirce represented a *proposition* as a relation with zero pegs. He called it a *medad*; the prefix me- comes from the Greek μη for not. In EGIF, a proposition p is represented as a relation with no logical subjects: `(p)`. In early versions of EGs, Peirce distinguished two subsets: Alpha for propositional logic and Beta for

first-order logic. By treating medads as relations, he avoided the need to distinguish Alpha from Beta. The same rules of inference apply to both.

Peirce continued, "Every indivisible graph instance must be wholly contained in a single area. The line of identity may be regarded as a graph composed of any number of dyads '—is—' or as a single dyad." To illustrate that option, consider the graph man – African, which may be read "There is an African man." Replacing the line with two copies of —is— would break the line into three segments: man—is—is—African. This EG may be read, "There is a man that is something that is something African." Following is the EGIF and predicate calculus:

```
(man *x) (is x*y) (is y*z) (African z)
```
$$\exists x \exists y \exists z (\text{man}(x) \wedge x = y \wedge y = z \wedge \text{African}(z))$$

The two copies of the dyad is are equivalent to two coreference nodes [x *y] [y *z], which may be replaced by a single coreference node [x *y *z]. Since all three labels are coreferent, the bound label x may replace all occurrences of the other defining labels and bound labels. The result is (man *x) (African x).

With these examples, Peirce presented EG syntax in less than three printed pages. In another four pages, he presented the rules of inference, a brief summary of endoporeutic, and an example that shows how an Aristotelian syllogism can be translated to an EG and proved by the EG rules of inference. As he showed, EGs have only two explicit operators: a line to represent the existential quantifier and an oval to represent negation. Conjunction is an implicit operator, expressed by drawing any number of graphs in the same area. Equality is expressed by joining lines.

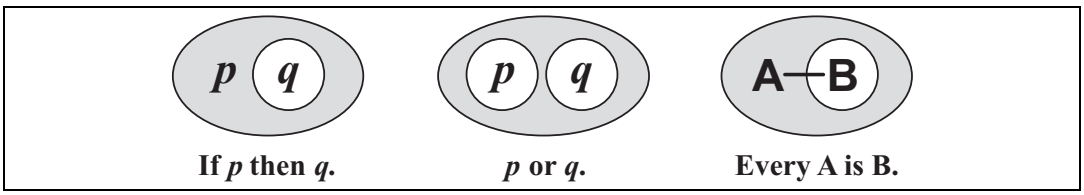

Figure 7: Three logical operators expressed as EGs

All other operators of first-order logic are represented by combining these primitives. Figure 7 shows three composite operators defined in terms of nested negations. Although these three operators are composite, their graphic patterns are just as readable as the algebraic formulas with the special symbols $\supset, \vee$, and $\forall$. In fact, the explicit nesting of EG ovals directly shows the scope of quantifiers. But the usual Boolean operators $\wedge, \vee$, and $\supset$ look so similar that students find it hard to

	EGIF	Predicate Calculus
`If` p, `Then` q.	`[If (p) [Then (q)]]`	$p \supset q$
p `or` q.	`~[~[(p)]~[(q)]]`	$p \vee q$
`Every` A `is` B	`[If (A *x) [Then (B x)]]`	$\forall x(A(x) \supset B(x)$

Table 4: EGIF and Predicate Calculus for Figure 7

	EGIF	Predicate Calculus
Left	`[*x][*y] ~[[x,y]]`	$\exists x \exists y \sim (x = y)$
Middle	`[*x][*y]~[(is x y)]`	$\exists x \exists y \sim \texttt{is}(x, y)$
Right	`(P *x) (P*y)~[(is x y)]`	$\exists x \exists y (P(x) \wedge P(y) \wedge \sim \texttt{is}\ (x, y))$

Table 5: EGIF and Predicate Calculus for Figure 8

remember that each one has a different effect on the scope of quantifiers. In EGIF, the propositions or medads `(p)` and `(q)` are enclosed in parentheses.

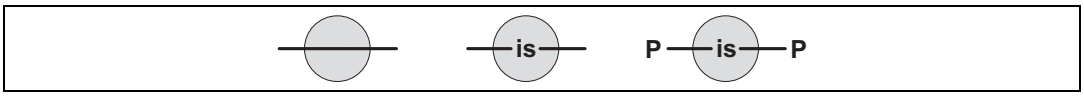

Figure 8: Stating that two things are not identical

Sometimes, the dyad —`is`— can clarify the translation of an EG to a sentence or formula. For example, Figure 8 shows three ways of saying that there exist two things. In the EG on the left, the shaded area negates the junction of the lines of identity on either side. To emphasize what is being negated, the EG in the middle replaces part of the line with the dyad —`is`—. Therefore, that EG may be read "There is something `x` that is not something `y`." The graph on the right says that there exist two different things with the property P or simply "There are two Ps."

As these examples show, EGs express identity by joining two lines and deny identity with an oval on a line. The linear notations require named labels and symbols such as `x=y` or `x` $\neq$ `y`. Without labels on lines, graphic EGs do not need special rules for replacing one label with another. But Peirce said that a complex graph with criss-crossing lines may be clearer if some connections are shown by labels, which he called *selectives* [45, 4.460]. When all the lines are replaced by labels, the EG becomes a nest of nodes, isomorphic to EGIF.

In summary, EGIF has the same primitives and conventions as EGs, but with the adaptations necessary to linearize the graphs. An oval for negation is represented by a tilde ~ followed by a pair of square brackets to enclose the EGIF for the subgraph inside the oval. A line of identity is represented by a defining label and zero or one bound label. (See Figure 4 for examples of defining labels with no bound labels.) The graphic options for connecting and extending lines are shown by coreference nodes. A ligature is represented by a coreference node with two or more labels (defining or bound). Many coreference nodes may be eliminated by replacing several defining and bound labels with a single defining label and multiple bound labels. (See Figure 6.) The only coreference nodes that cannot be eliminated are those that are enclosed in a negation and show a junction of two or more lines whose defining labels are outside the negation. (See Figure 8.)

3 EG Rules of Inference

All proofs in Peirce's system are based on "permissions" or "formal rules... by which one graph may be transformed into another without danger of passing from truth to falsity and without referring to any interpretation of the graphs" (CP 4.423). Peirce presented the permissions as three pairs of rules, one of which states conditions for inserting a graph, and the other states the inverse conditions for erasing a graph. Unlike Frege's rules, Peirce's rules are symmetric: any change by one rule can be undone by its inverse. This property enables some remarkable metalevel proofs (shown in Section 6) that are not possible with the common proof procedures. As before, all quotations by Peirce, unless otherwise cited, are from [48, 3:162-169]:

> There are three simple rules for modifying premises when they have once been scribed in order to get any sound necessary conclusion from them.... I will now state what modifications are permissible in any graph we may have scribed.

Peirce's rules are a generalization and simplification of the rules for *natural deduction*, which Gentzen [11] independently discovered many years later. For both Peirce and Gentzen, the rules are grouped in pairs, one of which inserts an operator, which the other erases. For both of them, the only axiom is a blank sheet of paper: anything that can be proved without any prior assumptions is a theorem. Section 6 presents a more detailed comparison with Gentzen's method.

> **1st Permission**. Any graph-instance on an unshaded area may be erased; and on a shaded area that already exists, any graph-instance may be inserted. This includes the right to cut any line of identity on

> an unshaded area, and to prolong one or join two on a shaded area. (The shading itself must not be erased of course, because it is not a graph-instance.)

The proof of soundness depends on the fact that erasing a graph reduces the number of options that might be false, and inserting a graph increases the number of options that might be false. Rule 1e, which permits erasures in an unshaded (positive) area, cannot make a true statement false; therefore, that area must be at least as true as it was before. Conversely, Rule 1i, which permits insertions in a shaded (negative), area cannot make a false statement true; therefore, the negation of that false area must be at least as true as it was before. A formal proof of soundness requires a version of model theory. Section 4 uses Peirce's model-theoretic semantics, which he called *endoporeutic* for "outside-in evaluation."

These rules apply equally well to propositional logic and predicate logic. Since EGs have no named variables, the algebraic rules for dealing with variables are replaced by rules for cutting or joining lines of identity (which correspond to erasing or inserting an equality or the graph —`is`—). Cutting a line in a positive area has the effect of *existential generalization*, because the newly separated ends of the line may represent different existentially quantified variables. Joining two lines in a negative area has the effect of *universal instantiation*, because it replaces a universally quantified variable with an arbitrary term.

> **2nd Permission**. Any graph-instance may be **iterated** (i.e. duplicated) in the same area or in any area enclosed within that, provided the new lines of identity so introduced have identically the same connexions they had before the iteration. And if any graph-instance is already duplicated in the same area or in two areas one of which is included (whether immediately or not) within the other, their connexions being identical, then the inner of the instances (or either of them if they are in the same area) may be erased. This is called the Rule of Iteration and Deiteration.

In other writings, Peirce stated more detail about the effect of these these rules on lines of identity. Iteration (2i) prolongs a line from the outside inward: any line of identity may be prolonged in the same area or into any enclosed area. Deiteration (2e) retracts a line from the inside outward: any line of identity that is not attached to anything may be erased, starting from the innermost area in which it occurs. Peirce also said that no graph may be copied into any area within itself; it is permissible, however, to copy a graph and then make a copy of the new graph in some area of the original graph.

The proof of soundness of iteration (2i) and deiteration (2e) shows that they are equivalence relations: they can never change the truth value of a graph. First, note that a copy of a graph p in the same area is equivalent to the conjunction $p \wedge p$; inserting a copy of p by Rule 2i or erasing it by 2e cannot change the truth value. For a copy of a subgraph into a nested area, the method of endoporeutic shows that the subgraph makes its contribution to the truth value at its first (outermost) occurrence. The presence or absence of a more deeply nested copy is irrelevant.

> **3rd Permission**. Any ring-shaped area which is entirely vacant may be suppressed by extending the areas within and without it so that they form one. And a vacant ring-shaped area may be created in any area by shading or by obliterating shading so as to separate two parts of any area by the new ring shaped area.

A vacant ring-shaped area corresponds to a *double negation*: two negations with nothing between them. The third permission says that a double negation may be erased (3e) or inserted (3i) around any graph on any area, shaded or unshaded. Note that Peirce considered the *empty graph*, represented by a blank area on a sheet of paper, as a valid existential graph; therefore, a double negation may be drawn or erased around a blank. An important qualification, which Peirce discussed elsewhere, is that such a ring is considered vacant even if it contains lines of identity, provided that the lines begin outside the ring and continue to the area enclosed by the ring without having any connections to one another in the area of the ring. Both rules 3e and 3i are equivalence rules, as endoporeutic shows.

In discussing these rules, Peirce said that their purpose is "to dissect the reasoning into the greatest possible number of distinct steps and so to force attention to every requisite of the reasoning." In that comment, Peirce was not comparing the EG rules to the rules for predicate calculus, because EG proofs are often shorter than other proof procedures for FOL. Instead, he was comparing the EG rules to Aristotle's syllogisms. To illustrate the issues, he proved the syllogism named Barbara:

> I will now, by way of an example of the way of working with this syntax, show how by successive steps of inference to pass from the premises of a simple syllogism to its conclusion. [Figure 9] shows the two premises "Any M is P" and "Any S is M." The first step consists in passing to [Figure 10] by the 2nd Permission.

The rule 2i allows the graph on the left of Figure 4 to be iterated (copied) into the unshaded area of the graph on the right. In translating EGs to EGIF, any labels

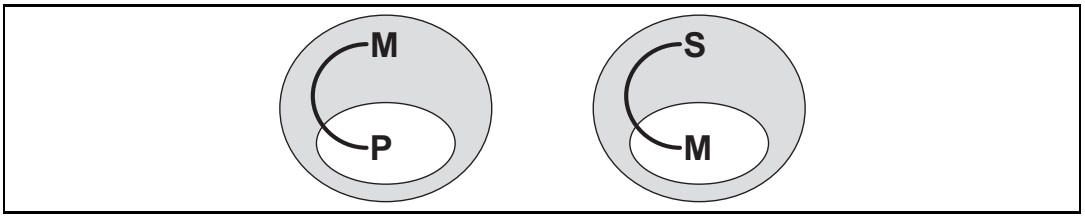

Figure 9: Two premises of the syllogism named Barbara

may be chosen for the lines of identity. To minimize the name clashes in EGIF proofs, each line should be assigned a unique label. For example, $*x$ may be used as the defining label for the graph on the left of Figure 9, and `*y` for the graph on the right. Following are the EGIF statements for Figure 9 and for Figure 10. Since these graphs have the form of implications, the keywords `If` and `Then` are used to represent the negations:

```
[If (M *x) [Then (P x)]] [If (S *y) [Then (M y)]]
[If (M *x) [Then (P x)]] [If (S *y) [Then [If (M *x)
  [Then (P x)]] (M y)]]
```

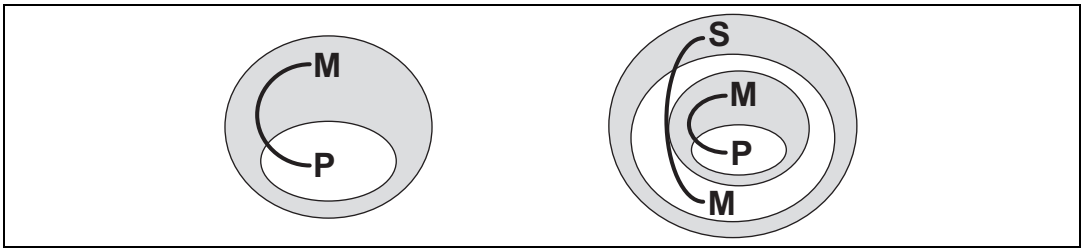

Figure 10: After iterating the left EG into the right EG

> The second step is simply to erase "Any M is P" by the 1st Permission. The third step is to join the two ligatures by the 1st Permission as shown in [Figure 11]. It will be observed that in iterating the major premise, I had a right to put the new graph instance at any part of the area into which I put it; and I took care to have the ligature of the minor premise **touch** the shaded area of the iterated graph instance. Now by the 1st Permission I have a right to insert what I please into a shaded area, and without making the new line of junction leave the shaded area, I make it touch the unshaded line of identity of the minor premise.

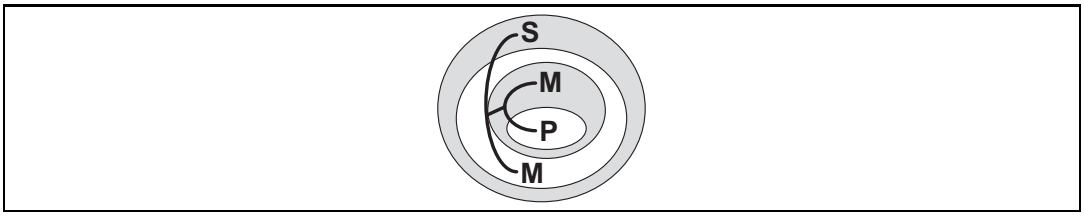

Figure 11: After joining the two lines

Peirce's explanations of lines that touch a boundary depend on two-dimensional features of the drawing. If he had not made the boundary of the iterated graph instance touch the line of identity, the join would take two steps: prolong the outer line into the shaded area by Rule 2i, and join the two lines by Rule 1i. Since EGIF cannot represent lines touching an oval, both steps are required. Therefore, the EGIF proof from Figure 10 to Figure 11 takes three steps: by 1e, erase the copy of the outer EG that had been iterated; by 2i, prolong the line that connects S to M into the shaded area by inserting the coreference node `[y]`; and by 1i, show the join of that line to the one that connects M to P by replacing the labels `*x` and `x` with `y`:

```
[If (S *y) [Then [If (M *x) [Then (P x)]] (M y)]]
[If (S *y) [Then [If [y] (M *x) [Then (P x)]] (M y)]]
[If (S *y) [Then [If (M y) [Then (P y)]] (M y)]]
```

Before the two lines were joined, the inner copy of M could not be erased by deiteration because the two copies of M were attached to different ligatures. But after the join, both copies of M are attached to the same ligature, and the inner copy of M can be erased by Rule 2e.

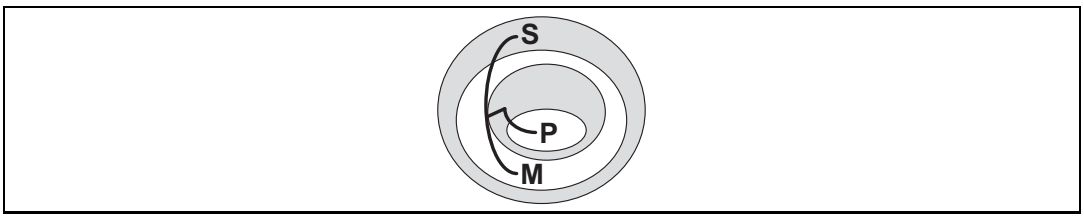

Figure 12: After deiterating (erasing) the inner copy of M

> This gives me a right in the fourth step to deiterate M so as to give [Figure 12] by the second permission. The fifth step is to delete the M on an unshaded field giving [Figure 13].

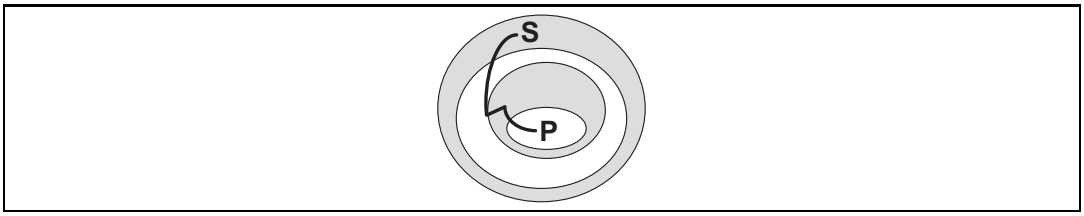

Figure 13: After erasing M in a shaded area

> The Sixth step authorized by permission the third consists in getting rid of the empty ring shaped shaded area round the P, giving [Figure 14].

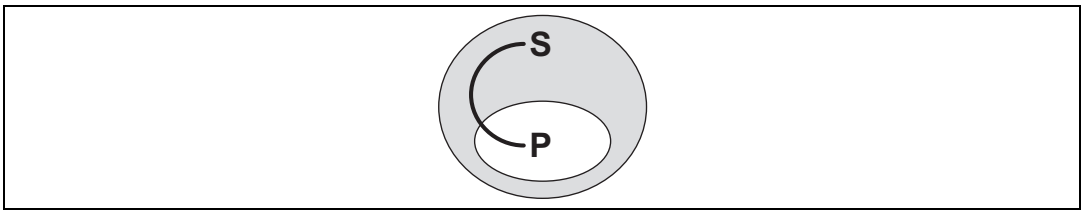

Figure 14: Conclusion of the syllogism

Peirce's rules are fundamentally semantic: each one inserts or erases one or more meaningful units. The differences between an EG proof and an EGIF proof result from syntactic details. In the proof of Barbara, the major differences result from using labels instead of lines of identity. Labels simplify the issues about lines crossing or touching borders, but they may require additional steps when lines of identity are joined.

For propositional logic, which Peirce called Alpha graphs, the meaningful units are relations and negations. For first-order logic with equality, Beta graphs add lines of identity and ligatures of lines. For EGIF, all meaningful units are expressed by *nodes*: relations, negations, and coreference nodes. Peirce treated functions as a special case of relations, but the EGIF grammar presented in the appendix adds nodes called *functions* to support the full semantics of Common Logic.

Since EGIF does not use shading, a positive area is defined by an even number of negations, and a negative area by an odd number of negations. The conditions for first-order logic with equality include those conditions with further constraints and operations on the labels for lines of identity. Although Peirce did not suggest a linear notation for EGs, he recognized that labels on the lines would be useful for complex graphs crossing lines of identity. He called those labels *selectives* and used capital letters as identifiers:

> in any case in which the lines of identity become too intricate to be perspicuous, it is advantageous to replace some of them by signs of a sort that in this system are called *selectives*. [45, 4.460]
>
> Any ligature may be replaced by replicas of one selective placed at every hook and also in the outermost area that it enters. In the interpretation, it is necessary to refer to the outermost replica of each selective first, and generally to proceed in the interpretation from the outside to the inside of all [ovals]. [45, 4.408]

Although Peirce did not suggest a linear notation for EGs, these two quotations summarize the crucial first step. But they omit one important step: If two or more ligatures are joined in an area that is nested inside the areas of their outermost points (defining nodes), their juncture must be represented by a coreference node. After that step, the lines are gone, and the EG becomes a nest of nodes. Ovals become nodes that contain other nodes. The only arbitrary choices are the symbols. Inside each oval, the order of listing the nodes is logically irrelevant. But the convention of putting defining labels before their bound labels is a convenience for anyone (human or computer) searching the lists.

The method for translating a well-formed EG to EGIF guarantees that the EGIF is also well formed: all syntactic and semantic constraints on the EGIF are satisfied. If all rules of inference are applied to the graphic EGs before the translation to EGIF, no other constraints need to be stated. But the EG rules combined with the method of translating EG to EGIF imply additional constraints on proofs that use EGIF directly.

These constraints show why human insight is important: the EG rules are easier to see than to describe. But ordinary computer programs cannot "see" the drawings. Therefore, some human who understands EGs and EGIF must design software to enforce the constraints. Before stating them, some definitions are necessary: the *scope* of a defining label; the EGIF equivalents for Peirce's verbs *prolong*, *join*, and *cut*; and another verb *simplify* for reducing the number of labels. Note to the reader: If you always derive EGIF from a well-formed EG, you can skip the next page and a half. Go to Figure 15 and its explanation.

- If a defining label d occurs in an area a, the *scope* of d is the area a and any area directly or indirectly enclosed by any negation in a. Any bound label in the scope of d that has the same identifier as d is said to be *bound* to d. The node that contains d must precede (occur to the left of) all nodes that directly or indirectly contain labels bound to d.

- No area may contain two or more defining labels with the same identifier. If the result of an insertion (by rule 1i or 2i) would violate that constraint, the identifier of the newly inserted defining label and all its bound labels shall be replaced by some identifier that does not violate the constraint.
- To *prolong* a line of identity with a defining label d into any area a in the scope of d is to insert a coreference node in a that has a single bound label that is bound to d. If the area a contains a defining label e with the same identifier as d, the identifier of e and all its bound labels shall be replaced by an identifier that is distinct from all other labels in the same EG.
- To *join* two lines of identity with defining labels d and e in any area a that is in the scope of both d and e is to insert a coreference node in a that contains bound labels for d and e and no others.
- To *cut* a line of identity with defining node d in an area a in the scope of d is to insert a defining node e whose defining label is distinct from all other labels whose scope includes a and to replace some or all of the labels bound to d that are in the scope of e with labels bound to e.
- To *simplify* the coreference nodes in an area a is to perform the following operations as often as they are applicable:

  1. If a coreference node in a contains two or more bound labels with the same identifier, erase all but one of them.
  2. If two coreference nodes in a each contain a bound label with the same identifier, they are *merged*: erase one of the coreference nodes, move all its bound labels to the other, and remove any duplicates according to operation 1.
  3. If a defining node d in a has a bound label in a coreference node c in a and c also contains a bound label b that is not bound to d, then the defining node d is erased, the label in c that was bound to d is erased, and every other label that was bound to d is replaced with a copy of b.
  4. If a coreference node c in area a contains exactly one bound label b, and area a also contains another node with a defining label or a bound label with the same identifier, then node c may be erased.

The basic constraint on erasing or inserting nodes is that no operation may leave or insert a bound label outside the scope of its defining label. Following are further conditions for Peirce's first and second permissions; no more conditions are needed for the third permission:

1. (a) In a negative area, no node that contains a bound label may be inserted in an area that is not in the scope of its defining label. No defining node may be inserted in an area that is in the scope of another defining label that has the same identifier.

   (e) In a positive area, no defining node that has one or more bound labels may be erased, unless all the nodes that contain those bound labels are erased in the same operation.

2. (a) If a defining node is iterated, the copy must be converted to a coreference node that contains a single bound label with the same identifier. A defining node that is enclosed in a negation, however, may remain unchanged when the negation is copied; but to avoid possible conflicts with future operations, the identifier of that defining label and all its bound labels should be replaced with an identifier that is not otherwise used.

   (e) Any nodes that could have been derived by rule 2i may be erased. (Whether or not a node had previously been derived by 2i is irrelevant.)

Peirce's meticulous attention to the smallest steps of reasoning enabled him to prove the soundness of every rule. Frege [9] assumed nine unprovable and non-obvious axioms. Whitehead and Russell [61] assumed five axiom schemata, of which one was redundant, but nobody discovered the redundancy for another 16 years. For EGs, only one axiom is necessary: a blank sheet of assertion, from which all the axioms and rules of inference by Frege, Whitehead, and Russell can be proved by Peirce's rules.

As an example, Figure 15 shows a proof of Frege's first axiom $a \supset (b \supset a)$. With the key words `If` and `Then`, that axiom in EGIF would be `[If (a) [Then [If (b) [Then (a)]]]]`, but the proof is easier to see when the negations are shown explicitly.

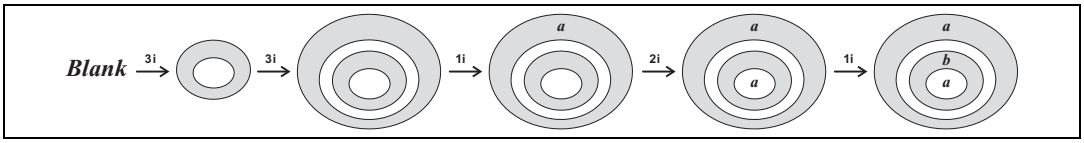

Figure 15: Proof of Frege's first axiom

The axiom in predicate calculus has five symbols, and each step of the EG proof inserts one symbol in its proper place. In EGIF, propositions are represented as relations with zero arguments: (a) and (b). Since there are no lines of identity, the EGIF proof takes the same five steps:

1. By rule 3i, Insert a double negation around a blank: ∼[∼[ ]]
2. By 3i, insert a double negation around the previous one:
   ∼ [∼ [∼ [∼[ ]]]]
3. By 1i, insert (a): ∼[(a) ∼[∼[∼[ ]]]
4. By 2i, iterate (a): ∼[(a)∼[∼[∼[(a)]]]]
5. By 1i, insert (b): ∼[(a)∼[∼[(b)∼[(a)]]]]

Frege's axiom contains five symbols, and each step of the proof inserts one symbol into its proper place in the final result. Frege's two rules of inference were *modus ponens* and *universal instantiation*. Figure 16 shows a proof of *modus ponens*, which derives q from a statement p and an implication $p \supset q$:

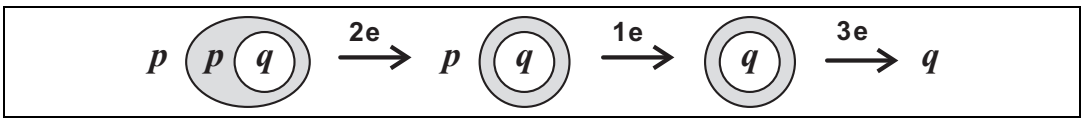

Figure 16: Proof of modus ponens

Proof in EGIF:

0. Starting graphs: (p) ∼[(p)∼ [(q)]]
1. By 2e, erase the nested copy of (p): (p) ∼[∼[(q)]]
2. By 1e, erase (p): ∼[∼[(q)]]
3. By 3e, erase the double negation: (q)

The rule of *universal instantiation* allows any term t to be substituted for a universally quantified variable in a statement of the form $(\forall x)P(x)$ to derive $P(t)$. In EGs, the term t would be represented by a graph of the form —t, in which t represents some monadic relation defined by a graph attached to the line of identity. The formal definition of EGIF in the appendix also supports functions; in EGs, a function could be represented with an arrowhead at the end of the line: ←t.

The universal quantifier $\forall$ corresponds to $\sim\exists\sim$, which is represented by a line whose outermost part occurs in a negative area. Since a pure graph has no named variables, there is no notion of substitution. Instead, the proof in Figure 17 performs the equivalent operation by joining two lines.

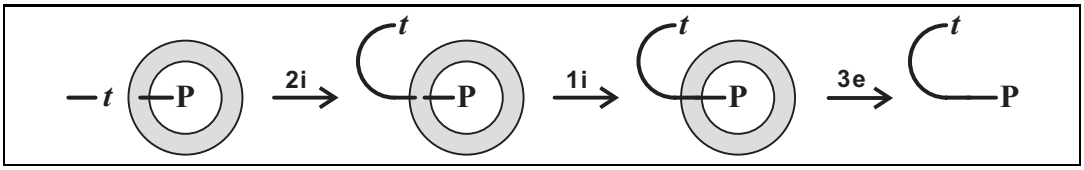

Figure 17: Proof of universal instantiation

In predicate calculus, the proof from `(∃x)t(x)` and `(∀y)P(y)` to `(∃x)(t(x) ∧ P(x))` would take one step. But the three EG steps in Figure 17 are easy to see and explain. In EGIF, the step of joining two lies by 2i requires two steps to relabel identifiers and simplify the result: In predicate calculus, the process of relabeling is just as complex, but it's not called a separate step.

0. Starting graphs: `(t *x) ∼[[*y] ∼[ (P y) ]]`
1. By 2i, iterate `*x` (prolong the line) by inserting `[x]`:
   `(t *x) ∼[ [x] [*y] ∼[ (P y) ]]`
2. By 1i, join `x` and `*y` by replacing `*y` and every occurrence of `y` with `x`:
   `(t *x) ∼[ [x] [x] ∼[ (P x) ]]`
3. By 2e, delete the two unnecessary copies of `[x]`:
   `(t *x) ∼[ ∼[ (P x) ]]`
4. By 3e, erase the double negation: `(t *x) (P x)`

In the *Principia Mathematica*, Whitehead and Russell proved a theorem that Leibniz called the *Praeclarum Theorema* (Splendid Theorem): $((p \supset x) \wedge (q \supset s)) \supset ((p \wedge q) \supset (r \wedge s))$. Starting with five axioms, their proof took a total of 43 steps. With Peirce's rules, this theorem can be proved in just seven steps, starting with a blank sheet of paper (Figure 18).

The first three steps of Figure 18 illustrate a proof procedure that can be used to prove every theorem of mathematics: start with a blank sheet of paper, draw a double negation around a blank area, insert the hypothesis in the shaded area, and copy the hypothesis, or subgraphs of it, into the unshaded area. Section 6 shows that these steps are a simplification and generalization of Gentzen's system of natural deduction.

All five proofs in this section suggest a useful heuristic for guiding the proof: Every proof proceeds in a straight line from premises to conclusion, and most steps tend to make the current EG look more like the desired conclusion. Therefore, begin a proof by drawing an EG for the conclusion and select options that can transform

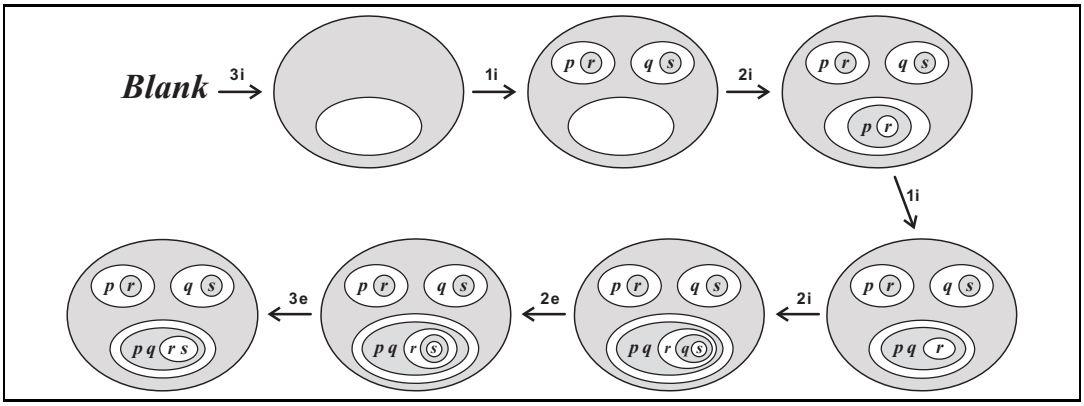

Figure 18: Proof of Leibniz's Praeclarum Theorema

the current EG to one that looks more like the conclusion. After only four steps, the graph in Figure 18 looks almost like the desired conclusion, except for a missing copy of s in the innermost area. Since that area is unshaded, the only way to get s in there is to iterate some graph or subgraph that contains s and erase the parts that are not needed. Step five iterates (by rule 2i) the only subgraph that contains s, and the last two steps erase parts that are not needed. The EGIF proof takes the same seven steps, but they're harder to visualize:

1. By 3i, draw a double negation around the blank: `~[ ~[  ] ]`
2. By 1i, insert the hypothesis in the negative area:
   `~[ ~[(p) ~[(r)]] ~[(q) ~[(s)]] ~[  ]]`
3. By 2i, iterate the left part of the hypothesis into the positive area:
   `~[ ~[(p) ~[(r)]] ~[(q) ~[(s)]] ~[ ~[(p) ~[(r)]]]]`
4. By 1i, insert `(q)`:
   `~[ ~[(p) ~[(r)]] ~[(q) ~[(s)]] ~[ ~[(p) (q) ~[(r)]]]]`
5. By 2i, iterate the right part of the hypothesis into the innermost area:
   `~[ ~[(p) ~[(r)]] ~[(q) ~[(s)]]`
   `  ~[ ~[(p) (q) ~[(r)] ~[(q) ~[(s)]]]]]`
6. By 2e, deiterate `(q)`: `~[ ~[(p) ~[(r)]] ~[(q) ~[(s)]] ~[ ~[(p) (q) ~[(r)] ~[ ~[(s)]]]]`
7. By 3e, erase the double negation to generate the conclusion:
   `~[ ~[(p) ~[(r)]] ~[(q) ~[(s)]] ~[ ~[(p) (q) ~[(r)] (s)]]]`

The visual heuristics that help teachers explain an EG proof can also help students discover proofs by themselves. These heuristics are especially evident in Section 5, which shows how Euclid's diagrams can be inserted or erased in the areas of an existential graph.

4 EG Semantics: Endoporeutic

In his first publication on model theory, Tarski (1933) quoted Aristotle as a precedent. The medieval Scholastics developed Aristotle's insights further, and Ockham (1323) presented a model-theoretic analysis of Latin semantics. Although Ockham wasn't as formal as Tarski, he covered the Latin equivalents of the Boolean connectives, the existential quantifier (*aliquis*), the universal quantifier (omnis), and even modal, temporal, and causal terms. Peirce lectured on Ockham's logic at Harvard, and he defined logic as "the formal science of the conditions of the truth of representations" [45, 2.220]. To see the similarity, compare quotations by Peirce and Tarski:

- Peirce [42]: "All that the formal logician has to say is, that if facts capable of expression in such and such forms of words are true, another fact whose expression is related in a certain way to the expression of these others is also true.... The proposition 'If A, then B' may conveniently be regarded as equivalent to 'Every case of the truth of A is a case of the truth of B'."
- Tarski [59]: "In terms of these concepts [of model], we can define the concept of logical consequence as follows: *The sentence X follows logically from the sentences of the class K if and only if every model of the class K is also a model of the class X.*"

Peirce's most important contribution to model theory was the method of *endoporeutic*, which he defined as an "outside-in" method for evaluating the truth of an existential graph. Before presenting his rules of inference, Peirce briefly summarized endoporeutic [48, 3:165], which he used to justify each rule:

> The rule of interpretation which necessarily follows from the diagrammatization is that the interpretation is "endoporeutic" (or proceeds inwardly) that is to say a ligature denotes "something" or "anything not" according as its **outermost part** lies on an unshaded or a shaded area respectively.

For every model M and proposition p, endoporeutic determines the same truth value as Tarski's definition. But Peirce's method is based on a two-person game between

Graphist, who proposes an EG, and Grapheus, a skeptic who demands a proof. In fact, endoporeutic is a version of *game theoretical* semantics (GTS), which was developed by Hintikka [19]. The similarity between endoporeutic and Hintikka's GTS was first discovered by Hilpinen [16]. For their introduction to model theory, Barwise and Etchemendy [1] chose the title *Tarski's World*, but what they presented was GTS.

In modern terminology, endoporeutic can be defined as a *two-person zero-sum perfect-information game*, of the same genre as board games like chess, checkers, and tic-tac-toe. Unlike those games, which frequently end in a draw, every finite EG determines a game that must end in a win for one of the players. But Henkin [15], the first modern logician to rediscover the GTS methods, showed that it could even evaluate the denotation of some infinitely long formulas in a finite number of steps. Peirce also considered the possibility of infinite EGs: "A graph with an endless nest of [ovals] is essentially of doubtful meaning, except in special cases" [45, 4.494]. As an example, he showed an infinite graph (one in which a certain pattern is repeated endlessly) for which endoporeutic would stop in just 2 or 3 steps. The version of endoporeutic presented here is based on Peirce's writings supplemented with ideas adapted from Hintikka, Hilpinen, Pietarinen [50], and the game-playing methods of artificial intelligence.

In the game of endoporeutic, Graphist proposes an existential graph g and claims that g is true. But Grapheus is a skeptic or devil's advocate who tries to show that g is false. The game begins with some state of affairs M, which corresponds to a Tarski-style model $M = (D, R)$: the domain D is a set of individuals, and R is a set of relations defined over D. The model M, like any Tarski-style model, can be represented as an EG with with no negations: each individual in D is represented by a line of identity; each n-tuple of each relation r in R is represented by a copy of the string that names r with n pegs attached to n lines of identity. The game proceeds according to a recursive procedure with Graphist as the initial proposer and Grapheus as the initial skeptic. During the game, they switch sides as they peel off each negation:

1. If g contains no negations, then the game is over, and the winner is determined by one of three possible cases:
   - If g is an empty graph, it is true because it says nothing false. The current proposer wins.
   - Else if there is a mapping (graph homomorphism) of g to M, the current proposer wins. The mapping need not be an isomorphism, since multiple lines of identity in g may map to the same line in M. Each relation node in g must map to a relation node in M with the same name.

- Else the current skeptic wins.

2. Else if g consists of just a single negation, the graph inside the oval becomes the new g, and the shading of each area in g is reversed. Then the two players switch sides: the proposer becomes the new skeptic; the skeptic becomes the new proposer; and the game continues with the new g and the new roles for both players.

3. Else if g consists of two or more negations, the skeptic chooses any one of the negations as the new g, and the game continues.

4. Else g consists of two or more parts: a subgraph g_0, which has no negations, and one or more negations: $g_1, ..., g_n$. The graph g_0 may contain some lines of identity that are joined to lines inside one or more of the negations. There are now two possibilities:

   - If there is no mapping of g_0 to M, the game is over, and the skeptic wins.
   - Else one or more mappings are possible, and the proposer may choose any one. That choice maps every line of identity x in g_0 to some line y in M. If x is joined to any line z in any negation g_i, then z must remain mapped to y for the duration of the game. Then the subgraph g_0 is erased, leaving g as a conjunction of one or more negations, and the game continues.

Since each pass through this procedure reduces the size of g, any game that starts with a finite graph must terminate in a finite number of moves. Since none of the ending conditions results in a draw, either Graphist, the original proposer, or Grapheus, the original skeptic, must have a winning strategy for any g and model M. If Graphist has a winning strategy, g is true of M. If Grapheus has a winning strategy, g is false of M.

These rules can evaluate Alpha graphs (propositional logic) in exactly the same way as Beta graphs. A model M for propositional logic would be a set of medads, such as `{(p), (q), (t)}`. A graph g with no negations is true of M if the set of medads in g is a subset of the medads in M. As an exercise, evaluate the EG for the Praeclarum Theorema in terms of some set of medads. Since it is a theorem, it should be true of every model, including the empty set. To illustrate these rules for Beta graphs (FOL), let the EG in Figure 19 be the model M. The three subgraphs state "There is lightning at 1 pm, there is thunder and lightning at 2 pm, and there is thunder at 3pm."

As the first example, let the graph g be `lightning—at—2pm`. Since g has no negations, Graphist, the original proposer, wins because there is a mapping from g

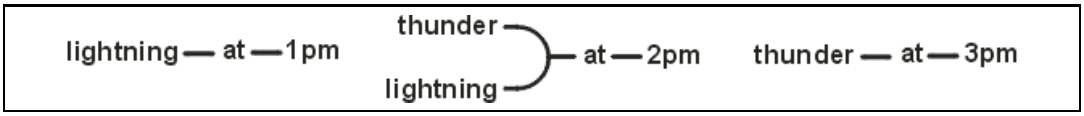

Figure 19: An EG used as the model M

to the middle part of M. Therefore, that graph is true. But if g happened to be `lightning—at—3pm`, then Grapheus, the original skeptic, would win because this g cannot be mapped to M. Therefore, it is false.

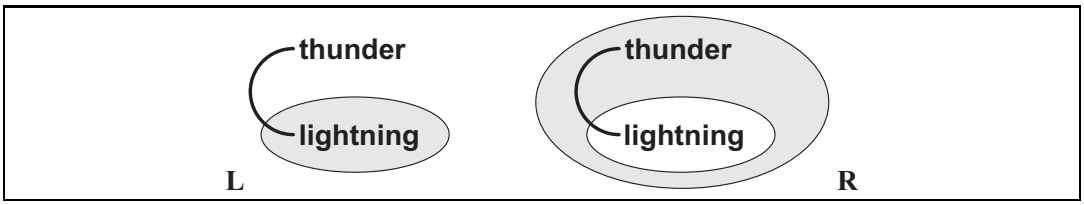

Figure 20: Two EGs, L and R, to be evaluated in terms of M

As an example that requires more than one pass through the procedure, consider the graph L on the left of Figure 20. L represents the sentence "There is thunder, and not lightning," and it meets the conditions for Step 4 of endoporeutic: the subgraph g_0 for the part outside the negation is `-thunder`. Graphist, the proposer, can choose any of two possible mappings of g_0 to M: one for thunder at 2 pm and the other for thunder at 3 pm; this choice also determines the mapping of the line inside the negation. If Graphist chooses 3 pm, the subgraph g_0 is erased, and the game continues. At Step 2, the negation is erased, causing the new g to become `-lightning`, but with the line forced to be mapped to the subgraph of M at 3 pm. Then the two players change sides, making Graphist the new skeptic. The game continues at Step 1, where g cannot be mapped to M because lightning occurred at 2 pm, not 3 pm. Therefore, the skeptic wins because this subgraph with this mapping is false. But the current skeptic is Graphist, who had been the original proposer for the graph L, which is therefore true. Note that if Graphist had made a mistake in the original choice of mapping, then Graphist would have lost. The truth of a graph, however, is not determined by mistakes; it depends only on the existence of a winning strategy if both players make the best choice at each option. As a final example, the graph R on the right adds one more negation. Therefore, R meets the condition for Step 2 of the game, which removes the negation and causes the two players to change sides. That makes Grapheus the proposer for a graph that is now identical to L, which has a winning strategy. Therefore, Grapheus wins the game that started with R. Since a win by Grapheus implies a loss by Graphist, the graph

R is false.

To prove that Peirce's rules of inference are sound, it is necessary to show that each rule preserves truth, i.e., if Graphist has a winning strategy with a given EG, no rule of inference can transform it to a graph that allows Grapheus to win. Therefore, the rules must monotonically increase the winning options for Graphist and monotonically decrease the winning options for Grapheus. Since the two players switch sides when a negation is removed (Step 2), Graphist is the proposer in every positive area and the skeptic in every negative area. The two steps of endoporeutic that depend on the number of options are Step 3, where the skeptic chooses one of the negations, and Step 4, where the proposer chooses one of the possible mappings of g_0 to M.

1. Rule 1e erases an arbitrary subgraph in a positive area. Erasing a negation decreases the number of options for Grapheus, who is the skeptic in a positive area. Erasing all or part of a relational graph reduces the constraints on the mapping to the model M and thereby increases the options for Graphist. Rule 1i, which inserts an arbitrary subgraph in a negative area, has the opposite effects: If negations are added, Graphist, who is the skeptic, has more options to choose. Adding a subgraph to a relational graph adds more constraints on the mapping to M and thereby reduces the options for Grapheus.

2. Rules 2e (deiteration) and 2i (iteration) have no effect on the truth value of any graph because any subgraph g that could be iterated or deiterated has its effect on the evaluation at its first occurrence. There are two possibilities to consider: g is true, or g is false. If g is true, a copy of g inserted or erased from any area has no effect on the truth value of that area. If g is false, the original area in which g occurs is false; the truth value of any nested area is irrelevant.

3. The only effect of evaluating a double negation is to cause the proposer and skeptic to reverse their roles twice. Therefore, Rules 3e (erasing a double negation) and 3i (inserting a double negation) can have no effect on the truth value of any graph.

In summary, rules 1e and 1i monotonically increase the options for Graphist and monotonically decrease the options for Grapheus. Rules 2e, 2i, 3e, and 3i have no effect on the winning strategies for either side. Therefore, if an EG is true, the EG that results from applying these rules will also be true. Therefore, all the rules are *sound*, because they preserve truth. Peirce's rules are also complete for FOL because the rules for other complete systems can be proved in terms of them See Figures 15, 16 and 17.

What distinguishes the game-theoretical method from Tarski's approach is its procedural nature. Unlike Tarski's definition, which maps all variables in a formula to individuals in a model M, endoporeutic is a "lazy" method that avoids mapping a line to M until the proposer chooses it in Step 4. Any subgraph that is not chosen is never evaluated; some very large or even infinite subgraphs can often be ignored.

One reason why Peirce had such difficulty in explaining it is that he and his readers lacked the terminology of the game-playing algorithms of artificial intelligence. No one clearly deciphered Peirce's writings until Hilpinen noticed the similarity to GTS. The discussion in this section is a reconstruction and clarification in modern terminology.

5 Formalizing Euclid

To formalize Euclid's geometry in existential graphs, map the diagrams to EGs, represent the EGs in EGIF, and justify every step of Euclid's proofs by EG rules of inference supplemented with two special rules: observation and imagination. In his writings on diagrammatic reasoning, Peirce described diagrams as "mainly" icons, but he noted that they may also contain symbols (symbolide features) and "features approaching the nature of indices" such as names or labels:

> A Diagram is mainly an Icon, and an Icon of intelligible relations... Now since a diagram, though it will ordinarily have Symbolide Features, as well as features approaching the nature of Indices, is nevertheless in the main an Icon of the forms of relations in the constitution of its Object, the appropriateness of it for the representation of necessary inference is easily seen. [45, 4.531]

For mathematics, Peirce distinguished two kinds of diagrams: geometrical and algebraic. He also discussed physical models, stereoscopic images, and motion pictures as aids to reasoning. Modern discussions of theorem proving ignore diagrams and never talk about observation or imagination. But every computer proof by any algorithm makes observations (searches for patterns) and performs experiments in the imagination (tentative constructions that may succeed or fail in later steps of the proof). In fact, observation and imagination are fundamental for any method of reasoning, formal or informal: (1) observe the current diagram (physical or mental); (2) imagine or construct the next one. As Peirce said,

> all deductive reasoning, even simple syllogism, involves an element of observation; namely deduction consists in constructing an icon or diagram the relation of whose parts shall present a complete analogy with

> those of the parts of the object of reasoning, of experimenting upon this image in the imagination, and of observing the result so as to discover unnoticed and hidden relations among the parts. [45, 3.363]

While rereading Euclid's *Elements*, Peirce discovered a distinction between a theorem and a corollary. For every major theorem, Euclid drew a new diagram. But he used the same diagram to observe the patterns stated as corollaries. Mathematicians have used those terms for centuries, but Peirce was the first to state a clear distinction between them: the proof of a theorem requires "an experiment in the imagination"; the proof of a corollary just makes observations about an already given or imagined diagram. Yet logicians today are still searching for syntactic criteria to distinguish theorems and corollaries. But Peirce's distinction is semantic, not syntactic:

> Corollarial deduction is where it is only necessary to imagine any case in which the premisses are true in order to perceive immediately that the conclusion holds in that case. Ordinary syllogisms and some deductions in the logic of relatives belong to this class. Theorematic deduction is deduction in which it is necessary to experiment in the imagination upon the image of the premiss in order from the result of such experiment to make corollarial deductions to the truth of the conclusion. [46, 35–39]

As an example, Figure 21 is Euclid's diagram for Proposition 1: "On a given finite straight line to construct an equilateral triangle." That statement mentions a straight line and a triangle that consists of three straight lines. But it does not mention circles. The insight to imagine the circles requires creativity. To make the proofs intelligible by students, Euclid added the results of imagination to the diagram for each theorem. For Figure 21, an experiment in the imagination added two circles.

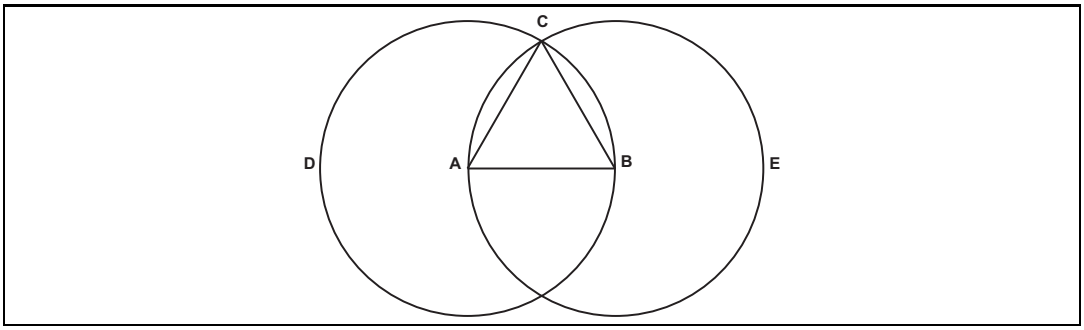

Figure 21: Diagram for Euclid's Proposition 1

Since Figure 21 already includes the result of imagination, the remainder of the proof uses only observations and applications of axioms (postulates and common notions): Apply Postulate 3 to draw two circles. Observe point C at the intersection of the two circles. Draw two more lines by Postulate 1. Observe that the two new lines are radii of one circle or the other. Finally, use Common Notion 1 to show that the three lines are congruent. Euclid's proof adds letters that relate each step to points on the diagram. In Heath's [6] translation,

> Let AB be the given finite straight line. Thus it is required to construct an equilateral triangle on the straight line AB. With center A and distance AB, let the circle BCD be described [Postulate 3]. Again with center B and distance BA, let the circle ACE be described [Post. 3]. And from the point C, in which the circles cut one another, to the points A, B let the straight lines CA, CB be joined [Post. 1]. Now, since the point A is the center of the circle CDB, AC is equal to AB [Definition 15]. Again, since the point B is the center of the circle CAE, BC is equal to BA [Def. 15]. But CA was also proved equal to AB. Therefore, each of the straight lines CA, CB is equal to AB. And things which are equal to the same thing are also equal to one another [Common Notion 1]. Therefore, CA is also equal to CB. Therefore, the three straight lines CA, AB, BC are equal to one another. Therefore, the triangle ABC is equilateral, and it has been constructed on the given finite straight line. QED [6]

Peirce maintained that continuity is the basis for generality. For Figure 21, there are no constraints on the diagram: nothing in the statement of Proposition 1 or its proof depends on the size, position, or orientation of the line AB. On any plane, there is an uncountable infinity of possible line segments, and Euclid's method could be used to construct an equilateral triangle on any of them.

Another criticism of Euclid's proofs is that they are not as precisely specified as modern algebraic versions. To address that criticism, every proof by Euclid can be translated, line by line, to operations on existential graphs, and every step of the graphic proof may be stated in the formally defined EGIF notation. Since there is a one-for-one correspondence, both notations can be used. Graphic EGs are more readable and teachable. They conform to the way people think, and they are language independent. For algebraic rigor, every step of an EGIF proof is sufficiently precise to be implemented in a computer program.

Finally, Euclid's proofs are stated as procedures that construct a figure, not as logical assertions that can be proved. Then the proof mixes imperative statements about what to draw with declarative statements about the result. That issue can

be addressed by translating Euclid's imperatives to declaratives. For example, his Proposition 1, "On a given finite straight line to construct an equilateral triangle," may be expressed as a conditional: "If there is a finite straight line AB, then there is an equilateral triangle with AB as one of its sides." In the EG or EGIF proof, every imperative sentence beginning with *Let* may be used as a directive for the next step.

Since icons in an EG cannot occur in EGIF, icons shall be replaced by names for the figure types: Point, Line, Circle, Triangle. To name the instances of a figure type, Euclid concatenated the letters that name the points in the order they were used to construct that instance. For example, a line drawn from point A to point B would be named AB. But the same line, if drawn from B to A would be named BA. To guarantee unique names in EGIF, the EGIF names AB and BA shall be written in alphabetical order as AB. In case of name conflicts, the name of the figure type shall be concatenated at the end: `XYZ_circle` or `XYZ_triangle`.

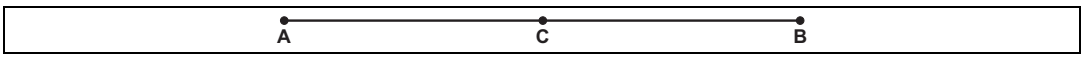

Figure 22: The line AB has a midpoint C

Any translation of English to logic requires some ontology of the relevant entities and relations. To define an ontology that describes Figure 22, begin with an English description of the details of interest: "There is a line AB from point A to point B. The point C is on AB. The lines AC and BC are congruent." The following list of relations constitutes a small ontology that may be used to express the English description in EGs or EGIF:

- `(Point X)`: X is a point.
- `(Line XY X Y)`: XY is a line with endpoints X and Y.
- `(On Z XY)`: point Z is on line XY.
- `("≅" XZ YZ)`: XZ and YZ are congruent.

With this ontology, the English description of Figure 22 may be translated to EGIF:

```
(Point *A) (Point *B) (Line *AB A B) (Point *C) (On C AB)
(Line *AC A C) (Line *BC B C) ("≅" AC BC)
```

Names that begin with an asterisk, such as `*A` or `*AB`, are defining labels with an implicit existential quantifier. Names with the same spelling, but without an

asterisk are bound labels that refer to the same entity as their defining label. The character ≅ represents the congruence relation. Since it is not an alphanumeric character, its EGIF name is enclosed in double quotes as "≅". The details of EGIF are specified in the appendix of this article.

Figure 22, by the way, raises an issue that has been debated since antiquity. Zeno and Cantor assumed that a line consists of an infinity of points. But Aristotle and Peirce maintained that points are markers on a line, not parts of it [26]. With Cantor's ontology, it's impossible to break the line AB in two congruent halves, because the midpoint C could not be a part of both halves. To avoid this issue, assume the Aristotle-Peirce ontology: the parts of a line are shorter line segments. Therefore, the midpoint is just a marker, and the two halves are congruent.

Given these preliminaries, write the EGIF proof by following Euclid's proof, line by line. The first sentence states the if-part of a conditional, and the second states the implication: "Let AB be the given finite straight line. Thus it is required to construct an equilateral triangle on the straight line AB." In English, if there is a finite straight line AB, then there is an equilateral triangle with AB as one of its sides. In EGIF, this statement would be written

```
[If (Point *A)(Point *B)(Line *AB A B)]
   [Then (Point *C) (Line *AC A C) (Line *BC B C)
   (Triangle *ABC AB AC BC) ("≅" AB AC BC)]]
```

Literally, this says that if there exist a point A, a point B, and a line AB from A to B, then there exist a point C, a line AC from A to C, a line BC from B to C, and a triangle ABC with congruent sides AB, AC, and BC. Common Logic allows relations to be *polyadic* (having any number of arguments). Therefore, the relation named "≅" may be used to assert that any number of figures are congruent.

All that detail is shown by the EG in Figure 23. To draw it, insert line AB in the shaded area (the hypothesis); insert, triangle ABC in the unshaded area (the conclusion); draw lines of identity (shown in red) from A and B in the shaded area to the unshaded area; insert the congruent symbol ≅ inside the triangle and draw lines of identity to each of the three sides.

Note that Figure 23 is language independent: all the details are shown, not said. If necessary, any detail may be observed and stated in any version of language or logic. As Peirce said, an icon plus an indexical can state a proposition. The letters A, B, and C are indexicals that correspond to pronouns in natural languages. In EGs, they serve the same purpose as lines of identity. Therefore, the two red lines from A to A and B to B are redundant. Either the lines or the letters could be deleted. Since the letters are the same as Euclid's, they are kept in Figure 24.

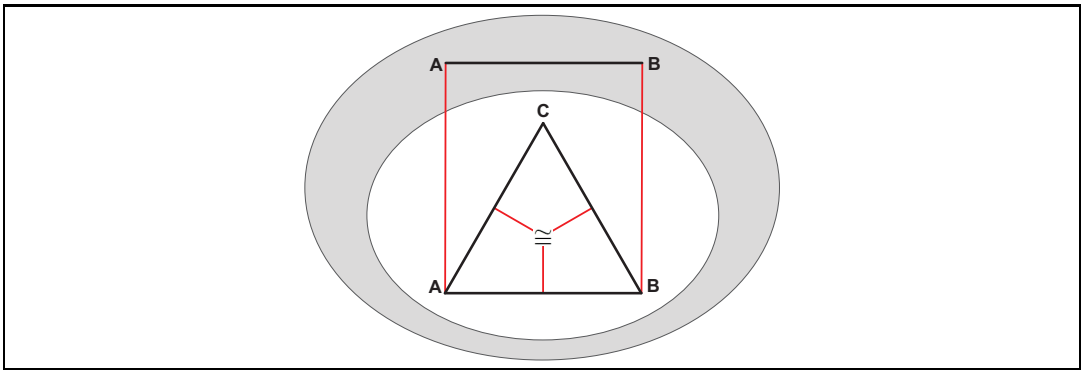

Figure 23: EG for Euclid's Proposition 1

But the three red lines attached to the congruent symbol are retained. The EGs in Figure 23 or 24 represent the conclusion to be proved.

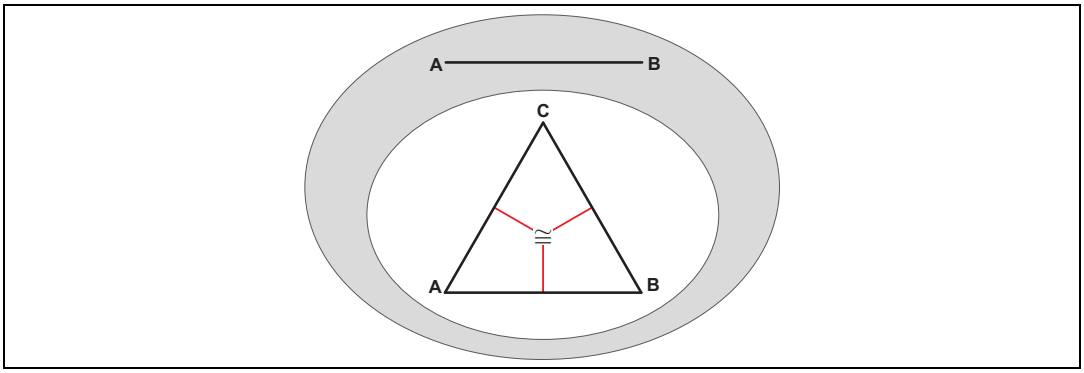

Figure 24: Keep copies of the letters A and B, but delete the connecting lines

Before starting the proof, begin with a blank sheet of assertion (SA), and assert EG diagrams or EGIF sentences for the postulates and common notions to be used in the proof. The first is Postulate 1. As Euclid stated it, "To draw a straight line from any point to any point." As a conditional, "If there are two points, then there exists a line from one point to the other point." Figure 25 shows the EG. The corresponding EGIF is `[If (Point *X) (Point *Y) [Then (Line *XY X Y) ]]`.

The next is Postulate 3: "To describe a circle with any center and distance." As a conditional, "If there are two points, then there exists a circle with one point at its center and a radius from the center to the other point." Figure 26 shows the EG. The EGIF is `[If (Point *X) (Point *Y) [Then (Circle *XYZ) (Center XYZ X) (Radius XYZ XY) ]]`.

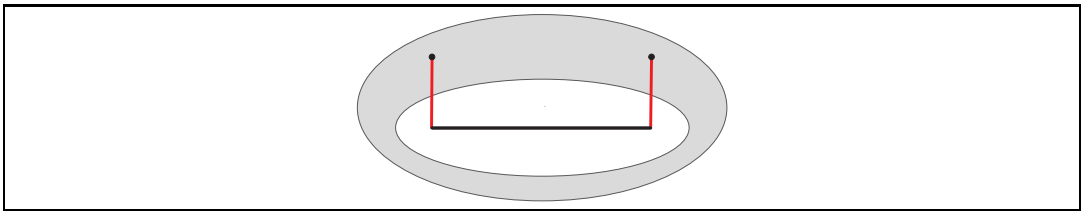

Figure 25: EG for Postulate 1

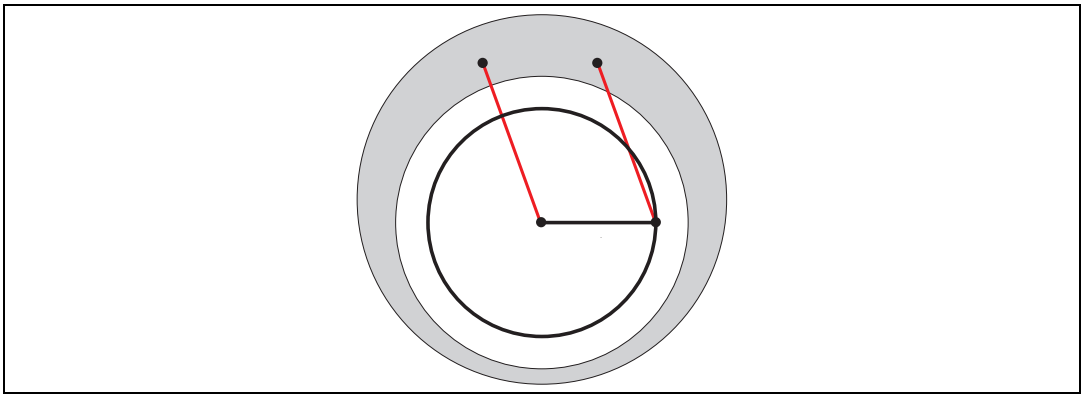

Figure 26: EG for Postulate 3

Finally, Common Notion 1 as Euclid stated it: "Things which are equal to the same thing are also equal to one another." As a conditional, "If X is congruent to Z and Y is congruent to Z, then X is congruent to Y." The EGIF for Figure 27: `[If ("≅" *X *Z) ("≅" *Y Z) [Then ("≅" X Y) ]]`.

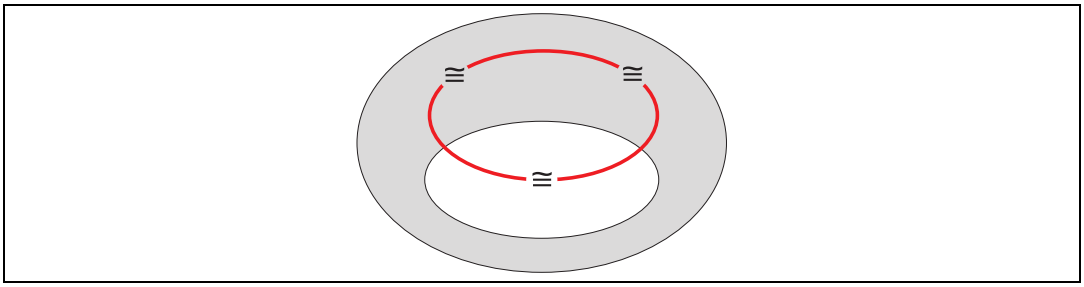

Figure 27: EG for Common Notion 1

Note that Aristotle's definition uses the plural word *things*. That implies two or more, and the three red lines in Figure 24 show the congruence relation as triadic. Since Common Logic allows polyadic relations with any number of arguments, that

option may be specified by a recursive definition. For this proof, the triadic option is sufficient, and it may be specified with one addition to the EGIF:

```
[If ("≅" *X *Z) ("≅" *Y Z) [Then ("≅" X Y) ("≅" X Y Z) ]].
```

But the word *things* implies an unordered set. More axioms would be needed to state that all possible permutations of the arguments of ≅ are equivalent. To avoid those details, just assume that ≅ is a dyadic or triadic relation and that the order of the arguments is irrelevant.

Given the two postulates and the common notion on the Sheet of Assertion, the first step of the proof uses rule 3i to draw a double negation around a blank area anywhere on SA. To make it more readable in EGIF, use the keywords If and Then:

1. `[If [Then ] ]`

By 1i, insert the hypothesis into the negative area:

2. `[If (Point *A) (Point *B) (Line *AB A B) [Then ]]`

By 2i, copy the EG from the shaded area to the unshaded area; also copy the labels A and B instead of drawing lines of identity. The result is the EG in Figure 28. For EGIF, indicate the copies by inserting coreference nodes for A, B, and AB. The bound labels in the coreference nodes indicate the presence of the corresponding entities in that area.

3. `[If (Point *A) (Point *B) (Line *AB A B) [Then [A] [B] [AB] ]]`

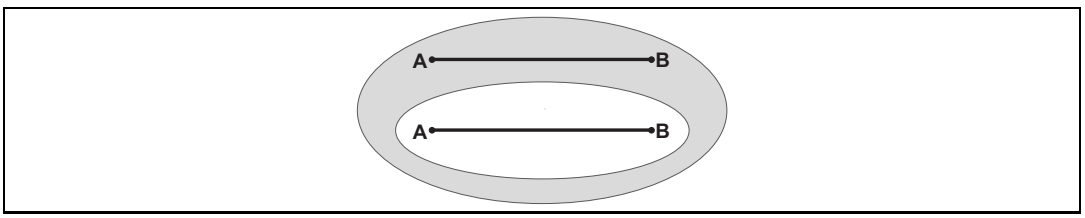

Figure 28: EG after step 3 of the proof

The next two sentences express Euclid's insight that circles are required: "With center A and distance AB, let the circle BCD be described [Postulate 3]. Again with center B and distance BA, let the circle ACE be described [Post. 3]." But his choice of names for the two circles indicate that has already made the observation that the two circles will intersect at point C. That observation is obvious from a construction of the diagram in an EG, but it is not obvious in EGIF. By 2i, insert Postulate 3 into the positive area:

4. `[If (Point *A) (Point *B) (Line *AB A B)`
   `[Then [A] [B] [AB]`
   `[If (Point *X) (Point *Y)`
   `[Then (Circle *XYZ) (Center XYZ X) (Radius XYZ XY)]]]]`

Since modus ponens is a derived rule of inference for EGs, it can be used to shorten EG proofs. By two applications of modus ponens, infer the circles BCD and ACE in Figure 29:

5. `[If (Point *A) (Point *B) (Line *AB A B)`
   `[Then [A] [B] [AB]`
   `[If (Point *X) (Point *Y)`
   `[Then (Circle *XYZ) (Center XYZ X) (Radius XYZ X Y) ]]`
   `(Circle *BCD) (Center BCD A) (Radius BCD AB)`
   `(Circle *ACE) (Center ACE B) (Radius ACE AB) ]]`

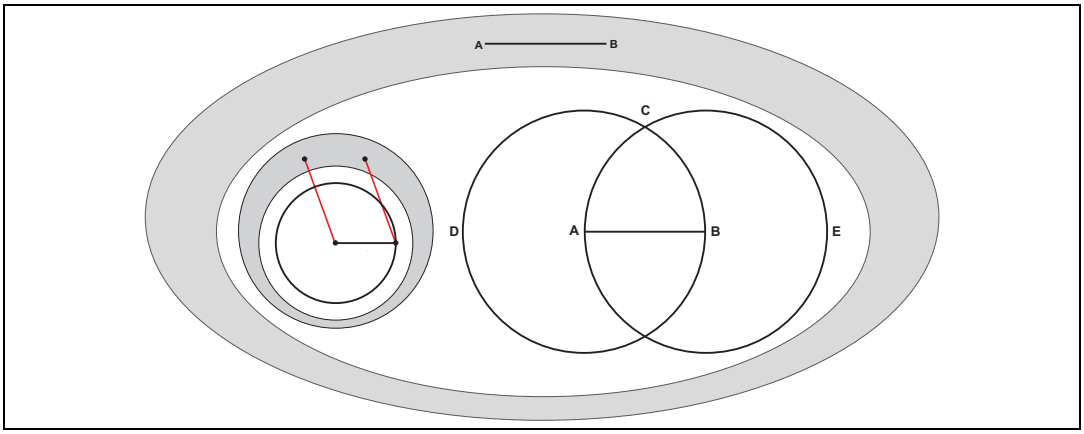

Figure 29: EG after step 5 of the proof

Since Postulate 3 is no longer needed, it may be erased by 1e:

6. `[If (Point *A) (Point *B) (Line *AB A B)`
   `[Then [A] [B] [AB]`
   `(Circle *BCD) (Center BCD A) (Radius BCD AB)`
   `(Circle *ACE) (Center ACE B) (Radius ACE AB) ]]`

Euclid: "And from the point C, in which the circles cut one another, to the points A, B let the straight lines CA, CB be joined [Post. 1]." This sentence makes an observation that there exists a point C where the circles intersect. By 2i, insert Postulate 1 in the positive area:

7. `[If (Point *A) (Point *B) (Line *AB A B)`
   `[Then [A] [B] [AB]`
   `(Circle *BCD] (Center BCD A) (Radius BCD AB)]`
   `(Circle *ACE] (Center ACE B) (Radius ACE AB)`
   `(Point *C) (On C BCD) (On C ACE) ]`
   `[If (Point *X ) (Point *Y) [Then (Line *XY) (Line XY X Y) ]]]]`

By two applications of modus ponens, infer lines AC and BC. These are the same lines that Euclid named CA and CB, but with the letters written in alphabetical order.

8. `[If (Point *A) (Point *B) (Line *AB A B)`
   `[Then [A] [B] [AB]`
   `(Circle *BCD] (Center BCD A) (Radius BCD AB)`
   `(Circle *ACE] (Center ACE B) (Radius ACE AB)`
   `(Point *C) (On C BCD) (On C ACE)`
   `[If (Point *X) (Point *Y) [Then (Line *XY) (Line XY X Y)]]`
   `(Line *AC) (Line *BC)]]`

By 1e, erase Postulate 1 in the positive area:

9. `[If (Point *A) (Point *B) (Line *AB A B)`
   `[Then [A] [B] [AB]`
   `(Circle *BCD] (Center BCD A) (Radius BCD AB)`
   `(Circle *ACE] (Center ACE B) (Radius ACE AB)`
   `(Point *C) (On C BCD) (On C ACE)`
   `(Line *AC) (Line *BC) ]]`

"Now, since the point A is the center of the circle CDB, AC is equal to AB [Definition 15]. Again, since the point B is the center of the circle CAE, BC is equal to BA [Def. 15]." These two sentences depend on the observations that line AC is a radius of circle BCD and that BC is a radius of ACE. Therefore, AC is congruent to AB, and BC is congruent to AB:

10. `[If (Point *A) (Point *B) (Line *AB A B)`
    `[Then [A] [B] [AB]`
    `(Circle *BCD] (Center BCD A) (Radius BCD AB)`
    `(Circle *ACE] (Center ACE B) (Radius ACE AB)`
    `(Point *C) (On C BCD) (On C ACE)`
    `(Line *AC) (Radius BCD AC) ("≅" AC AB)`
    `(Line *BC) (Radius ACE BC) ("≅" BC AB) ]]`

"But CA was also proved equal to AB. Therefore, each of the straight lines CA, CB is equal to AB. And things which are equal to the same thing are also equal to one another [Common Notion 1]. Therefore, CA is also equal to CB. Therefore, the three straight lines CA, AB, BC are equal to one another." By 2i, insert Common Notion 1 in the positive area and apply it to infer the dyadic and triadic congruences in the last line:

11. `[If (Point *A) (Point *B) (Line *AB A B)`
    `[Then [A] [B] [AB]`
    `(Circle *BCD] (Center BCD A) (Radius BCD AB)`
    `(Circle *ACE] (Center ACE B) (Radius ACE AB)`
    `(Point *C) (On C BCD) (On C ACE)`
    `(Line *AC) (Radius BCD AC) ("≅" AC AB)`
    `(Line *BC) (Radius ACE BC) ("≅" BC AB)`
    `[If ("≅" *X *Z) ("≅" *Y Z)`
    `[Then ("≅" X Y) ("≅" X Y Z) ]]`
    `("≅" AB AC) ("≅" AB AC BC) ]]`

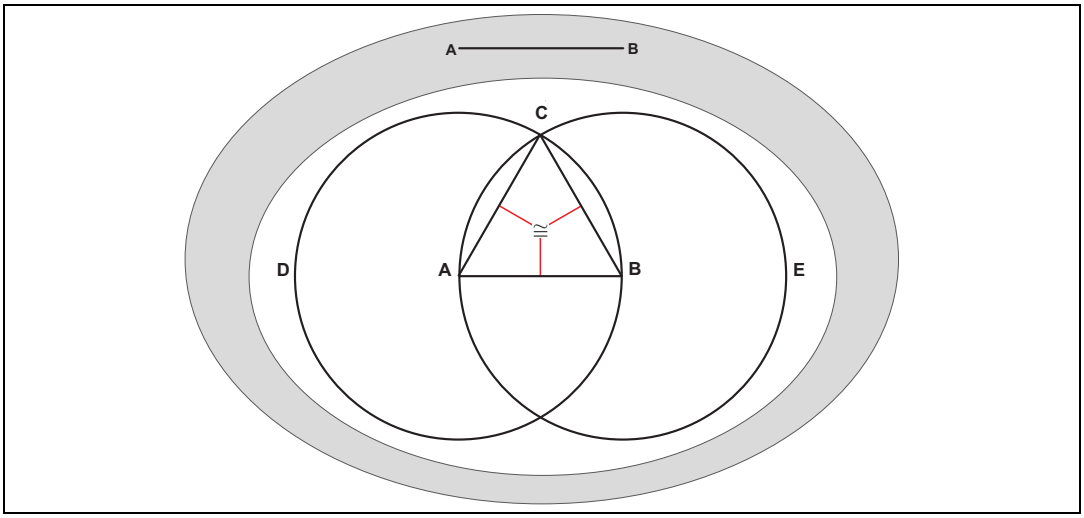

Figure 30: EG after step 1

"Therefore, the triangle ABC is equilateral, and it has been constructed on the given finite straight line. QED" By observation, there exists a triangle ABC with congruent sides AB, AC, BC. By rule 1e, erase all the nodes in the positive area of step 11 that are not required to represent the EG in Figure 22 or23. The result is the theorem to be proved:

```
12. [If (Point *A) (Point *B) (Line *AB A B)
    [Then (Point *C) (Line *AC A C) (Line *BC B C)
      (Triangle *ABC AB AC BC) ("≅" AB AC BC)]]
```

This example shows that a Euclidean proof can be translated, line by line, to a proof by EG rules of inference supplemented with the rules of observation and imagination. But it has implications that go far beyond techniques for analyzing and formalizing Euclid's proofs. Although the formalists were correct in discovering the lack of rigor in any step that requires observation or imagination, they were wrong in claiming that those steps were weaknesses or even flaws. For over two thousand years, Euclid's proofs were considered the ultimate standard of rigor. Many mathematicians found explanations and proofs that could be improved, but nobody found theorems that were false. Newton, for example, used Descartes' analytic geometry to discover his famous principles. But when he published them, he used a Euclidean style of definition and proof because that was the accepted standard of rigor in his day. Most readers did not know or trust the "new fangled" methods. They had more confidence in the methods they had learned and used in the universities.

For logic and mathematics, they show that diagrams have a legitimate role to play in formal reasoning. For artificial intelligence, they introduce computational techniques for combining and relating symbols and images. For psychology and neuroscience, mental models and image processing are intimately connected with all aspects of logic, language, and reasoning. Finally, they develop and extend Peirce's insights, hints, and conjectures about the continuum of semiotic processes in every aspect of cognition and life.

6 Research Issues

Peirce's voluminous manuscripts about logic are a gold mine of insights and innovations. Their relative neglect during the 20th century makes them a largely unexplored treasure for the 21st. His existential graphs and their rules of inference are still at the forefront of research in logic and cognitive science. Since each rule inserts or erases one semantic unit, the rules can be adapted to any notation just by defining its negative areas, positive areas, and semantic units. The option of inserting arbitrary icons in EGs opens new avenues of research in theory and applications.

The symmetry of the EG structure and rules of inference simplifies proof theory. For example, the EG rules imply the *reversibility theorem* and the *cut-and-paste theorem*. The proofs are simplest for the EG and EGIF notations, but they can be adapted to many other notations, including Peirce-Peano algebra.

- **Reversibility Theorem.** Any proof of $p \vdash q$ by Peirce's rules of inference can be converted to a proof of $\sim q \vdash \sim p$ by negating each step and reversing the order.

- **Proof.** Let $s_0, ..., s_n$ be the steps of the proof with s_0 as the EG that represents p, s_n as the EG that represents q, and each r_i from 1 to n as the rule that converts s_{i-1} to s_i. For the reverse sequence of negated EGs, note that $\sim[s_n]$ represents $\sim q$, and $\sim[s_0]$ represents $\sim p$. Furthermore, each $\sim[s_i]$ is converted to $\sim[s_{i-1}]$ by the inverse of rule r_i. The only question is whether the inverse conversions are permissible. Since the rules 2e, 2i, 3e, and 3i are equivalence operations, their inverses are permissible in any area, positive or negative. But 1e can only be performed in a negative area, and its inverse 1i can only be performed in a positive area. Since each step of the reverse proof is negated, the polarity (negative or positive) of every nested area in each $\sim[s_i]$ is reversed. Therefore, any application of 1i or 1e in the forward direction can be undone by its inverse (1e or 1i) in the reverse direction.

With traditional rules of inference, a proof of $p \vdash q$ can usually be converted to a proof of $\sim q \vdash \sim p$. But the conversions are more complex, because the traditional rules don't have exact inverses. For some kinds of proofs, Peirce's method is much shorter because of a property that is not shared by other common proof procedures: his rules can perform surgical operations (insertions or erasures) in an area of a graph or formula that is nested arbitrarily deep. Furthermore, the rules depend only on whether the area is positive or negative. Those properties imply the *cut-and-paste theorem* [56], which can be adapted to any notation for first-order logic:

- **Cut-and-Paste Theorem.** If a proof $p \vdash q$ is possible on a blank sheet sheet of assertion, then in any positive area of an EG where p occurs, q may be substituted for p.

- **Proof.** Since the nested area in which p occurs is positive, every step of a proof on a blank sheet can be carried out in that area. Therefore, it is permissible to "cut out" the steps of a proof from p to q in the outer area and "paste" them into the nested area. After q has been derived, Rule 1e can be used to erase the original p and any remaining steps of the proof other than q.

From the cut-and-paste theorem (CAPT), other important theorems can be proved as corollaries. One example is the deduction theorem:

- **Deduction Theorem.** If a proof $p \vdash q$ is possible, then $p \supset q$ is provable.

- **Proof.** To show that $p \supset q$ is provable, use CAPT to derive the corresponding EGIF from a blank:

  1. By 3i, draw a double negation around a blank: ∼[∼[ ]]
  2. By 1i, insert (p) in the negative area: ∼[(p) ∼[ ]]
  3. By 2i, iterate (p) to derive the equivalent of $p \supset p$: ∼[(p) ∼[(p)]]
  4. By CAPT, replace the inner (p) with (q): ∼[(p) ∼[(q)]]

The *constructive dilemma* is another another corollary of CAPT that is difficult to prove in many systems:

- **Constructive Dilemma.** If $p_1 \vdash q$ and $p_2 \vdash q$, then $p_1 \vee p_2 \vdash q$.
- **Proof:** Use two applications of CAPT.

  0. Starting EGIF: ∼[∼[(p1)] ∼[(p2)]]
  1. By CAPT, replace (p1) with q: ∼[∼[(q)] ∼[(p2)]]
  2. By CAPT, replace (p2) with q: ∼[∼[(q)] ∼[(q)]]
  3. By 2e, deiterate one copy of ∼[(q)]: ∼[∼[(q)]]
  4. By 3e, erase the double negation: (q)

In a positive area, erasing a graph or part of a graph makes the result more general because it is true in a broader range of cases. For example, (animal x) is more general than (cat x) because a cat is defined as an animal with certain feline characteristics. Replacing (cat x) with (animal x) is permissible because it has the effect of erasing the feline characteristics. Conversely, in a negative area, inserting a graph makes the result more specialized because it is true in a narrower range of cases. Therefore, replacing (animal x) with (cat x) in a negative area is permissible because it has the effect of inserting those feline characteristics.

The rules of generalization and specialization may be used as derived rules of inference. The proof of generalization uses CAPT in a one-step proof. But the proof of specialization takes six steps because it's harder to justify the erasure of graphs in a negative area. Such proofs usually require more steps with any proof procedure.

- **Generalization and Specialization.** For any propositions p and q stated in EG, EGIF, or semantically equivalent notations, if $p \vdash q$, then p is said to be more specialized than q, and q is said to be more generalized than p.
- **Rule of generalization.** In any positive area, p may be replaced by a generalization q.

- **Proof:** Assume that the area that contains p also contains some arbitrary EG (A), which may be blank.

  0. Starting EGIF in some positive area: `(p) (A)`
  1. By CAPT, replace `(p)` with `(q)`: `(q) (A)`

- **Rule of specialization.** In any negative area, q may be replaced by a specialization p.

- **Proof:** Assume that the area that contains q also contains some arbitrary EG (A), which may be blank.

  0. Starting EGIF in some positive area: ∼`[(q) (A)]`
  1. Since $p \supset q$, insert the corresponding EGIF in the same positive area: ∼`[(q) (A)]` ∼`[(p) ∼[(q)]]`
  2. By rule 2i, iterate ∼`[(q) (A)]` into the innermost area with `(q)`: ∼`[(q) (A)]` ∼`[(p) ∼[(q) ∼[(q) (A)]]]`
  3. By 1e, erase the original copy of ∼`[(q) (A)]`: ∼`[(p) ∼[(q) ∼[(q) (A)]]]`
  4. By 2e, erase the innermost copy of `(q)`: ∼`[(p) ∼[(q) ∼[(A)]]]`
  5. By 1e, erase `(q)`: ∼`[(p) ∼[∼[(A)]]]`
  6. By 3e, erase the double negation: ∼`[(p) (A)]`

Converting various proof procedures to EG form provides fundamental insights into the nature of the proofs and their interrelationships. Gentzen [11] developed two proof procedures called *sequent calculus* and *natural deduction*. A proof by either one can be systematically converted to a proof by Peirce's rules. The converse, however, does not hold because Gentzen's rules cannot operate on deeply nested expressions. For some proofs, many steps are needed to bring an expression to the surface of a formula before those rules can be applied. An example is the *cut-free* version of the sequent calculus, in which proofs can sometimes be exponentially longer than proofs in the usual version. Dau [3] showed that with Peirce's rules, the corresponding cut-free proofs are longer by just a polynomial factor.

Like Peirce's rules, Gentzen's rules for natural deduction also come in pairs, one of which inserts an operator, which the other eliminates. Gentzen, however, required six pairs of rules for each of his six operators: $\wedge, \vee, \supset, \sim, \forall$, and $\exists$. Peirce had only three operators and three pairs of rules. Even when limited to the same three operators, Gentzen's rules are highly irregular. They lack the symmetry of paired rules that are exact inverses of one another. Four of the twelve rules even

require a provability test, expressed by the operator $\vdash$. Yet Gentzen's rules can all be proved as derived rules of inference in terms of Peirce's rules. In fact, two of them were just proved above: the rule for $\supset$-introduction is the deduction theorem, and the rule for $\vee$-elimination is the constructive dilemma. Figure 31 shows Gentzen's six pairs of rules.

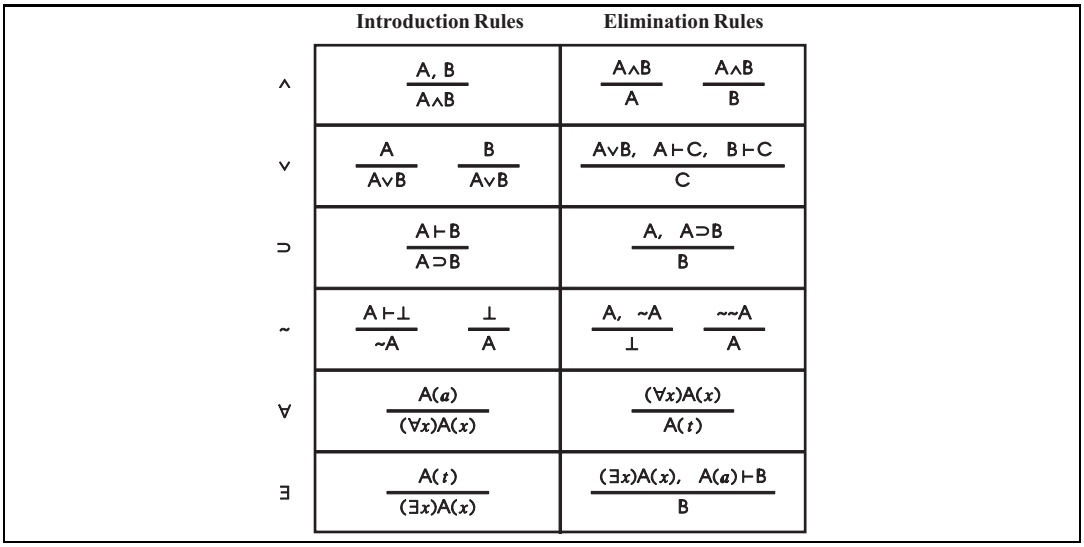

Figure 31: Rules of inference for Gentzen's natural deduction

Gentzen's rules can be proved as derived rules of inference in terms of Peirce's rules, but some of them require further explanation. The symbol $\perp$ represents a proposition that is always false. In EGs, $\perp$ is represented by an oval with nothing inside; in EGIF, it is `~[ ]`. Peirce called `~[ ]` the *pseudograph* because it is always false. In the resolution method for theorem proving, it is called the *empty clause*. From the pseudograph, any proposition `(A)` can be derived. Start with `~[ ]`; by 1e, insert `~[(A)]` to derive `~[~[(A)]]`; by 3e, erase the double negation to derive `(A)`. By the way, *intuitionistic logic* is a version of FOL that blocks the derivation of an arbitrary proposition p from a contradiction; that proof can be blocked by prohibiting rule 3e for erasing a double negation.

Two other special symbols are the arbitrary term t in the rules for $\exists$-introduction and $\forall$-elimination and the arbitrary value or free variable a in the rules for $\forall$-introduction and $\exists$-elimination. The arbitrary term t is an ordinary expression with no free variables, and it can be represented as an EG —`t` or EGIF `(t *x)`, where `t` is any relation or graph with one peg designated to represent the value of the term. The rule for $\forall$-elimination, also called universal instantiation, was proved

in Figure 17. The proof for ∃-introduction takes one step:

0. Starting EGIF: `(t *x) (A x)`

1. By 1e, erase part of the line that connects `*x` to `x` and introduce a new defining label for `(A *y)`.

Since EGs don't have variables, they don't have free variables. But proof procedures that use arbitrary values or free variables have been a source of controversy for years. Kit Fine [8] interpreted Gentzen's rule of ∀-introduction as an assumption that some implicit proposition φ implies that for any individual a, the statement $A(a)$ is true.

0. Starting assumption: `[*x]` ⊢ `(A x)`

1. By the deduction theorem, conclude `If [*x] [Then (A x)]]`. This EGIF is equivalent to $(\forall x)A(x)$.

For Gentzen's rule of ∃-elimination, $A(a)$ is true of some arbitrary values, but not all. The conclusion B is provable from $A(a)$ only for values of a for which $A(a)$ is true.

0. Starting assumptions: `[*x] (A x); (A *a)` ⊢ `(B)`

1. Let `*a` be some `*x` for which `(A x)` true. Therefore: `(B)`

Although each of Gentzen's rules can be derived from Peirce's rules, proofs in the two systems take different paths. EG proofs proceed in a straight line from a blank sheet to the conclusion: each step inserts or erases one subgraph in the immediately preceding graph. No bookkeeping is required to keep track of information from any earlier step. But Gentzen's proofs interrupt the flow of a proof at every rule that contains the pattern $p \vdash q$. When deriving q from p, those rules use a method of bookkeeping called *discharging an assumption*:

1. Whenever the symbol ⊢ appears in one of Gentzen's rules, some proposition p is assumed.

2. The current proof is suspended, and the state of the current rule is recorded.

3. A side proof of q from p is initiated.

4. When the conclusion q is derived, the assumption p is discharged by resuming the previous proof and using q in the rule of inference that had been interrupted.

Such side proofs might be invoked recursively to an arbitrary depth. EG proofs avoid that bookkeeping by an option that most notations for logic can't represent: drawing a double negation around a blank. For example, in the proof of the Praeclarum Theorema (Figure 18), the first step is to draw a double negation on a blank sheet, and the second step is to insert the hypothesis into the shaded area. The result is a well-formed EG, and no bookkeeping is necessary to keep track of how that EG had been derived. In Gentzen's method, the first step is to assume the hypothesis, and a record of that assumption must be retained until the very end of the proof, when it is finally discharged to form the conclusion. This observation is the basis for converting a proof by Gentzen's method of natural deduction to a proof by Peirce's method:

- Replace each of Gentzen's rules that does not contain the symbol $\vdash$ with one or more of Peirce's rules that produce an equivalent result.
- Whenever one of Gentzen's rules containing $\vdash$ is invoked, apply rule 3i to insert a double negation around the blank, copy the hypothesis into the negative area by rule 1i, and continue.

An EG proof generated by this procedure will be longer than a direct proof that starts with Peirce's rules. As an exercise, use Gentzen's rules and his method of discharging assumptions to prove the Praeclarum Theorema. To avoid the bookkeeping, introduce a nest of two ovals when the symbol $\vdash$ appears in a rule. Insert the hypothesis into the shaded area, but continue to use the algebraic notation for the formulas. Finally, translate each formula to an EG, and add any intermediate steps needed to complete the proof according to Peirce's rules. Proofs by Gentzen's other method, the sequent calculus, can be converted to EG proofs more directly because they don't require extra steps to discharge assumptions.

Although Peirce understood the importance of functions in mathematics, he did not distinguish functions and relations in his versions of logic. In effect, he treated a function, such as $f(x) = y$, as a special case of a relation $f(x, y)$. What distinguishes a function from other relations is that for every value of x, there is exactly one value for y. Since EGIF has the semantics of Common Logic, which does distinguish functions from relations, EGIF uses the notation `(f x | y)` for $f(x) = y$. In general, a function may have any number of arguments on the left of the vertical bar, and exactly one value on the right: `(g x y z | w)`. EGIF also allows functions with zero arguments, such as `(h | x)`. For details, see the EGIF grammar in the appendix.

In computational systems, the widely used method of resolution applies a single rule of inference to a set of *clauses* [55]. Each clause is a disjunction of positive or

negative *literals*, which are single atoms or their negations. In the following example, the two clauses on the left of $\vdash$ are combined by resolution to derive the clause on the right:

$$\sim p \vee \sim q \vee r,\ \sim r \vee u \vee v\ \vdash \sim p \vee \sim q \vee u \vee v.$$

The leftmost clause has a positive literal r, and the middle clause has a negated copy $\sim r$. Resolution is a cancellation method that erases both r and $\sim r$ and merges the remaining literals to form the clause on the right. Figure 32 shows the effect of resolution as expressed in EGs:

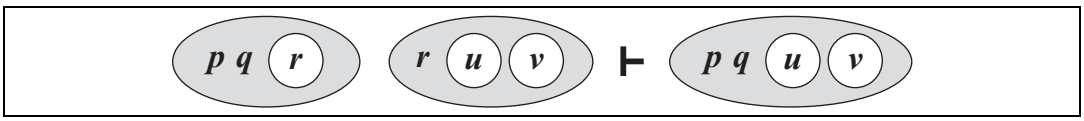

Figure 32: Resolution expressed in EGs

With EGs, resolution takes four steps: by 2i, iterate the middle EG into the doubly nested (unshaded) area of the EG on the left; by 2e, erase the innermost copy of r; by 1e, erase the remaining copy of r; by 3e, erase the double negation. The result is the EG on the right. Resolution is commonly used in a refutation procedure, which attempts to prove a statement s by deriving a contradiction from $\sim s$:

1. Negate the statement to be proved: $\sim s$.
2. Convert $\sim s$ to clause form.
3. Use repeated steps of resolution to derive the empty clause or pseudograph, `[ ]`.
4. Since the empty clause is false, the original s must be true.

Many logicians have observed that proofs by resolution are "almost" the inverse of proofs by natural deduction. They aren't exact inverses, however, because the rule of resolution is simple and regular, but Gentzen's six pairs of insertion and elimination rules are highly irregular. Showing how one method can be derived from the other is a challenge that Larry Wos [65], a pioneer in automated reasoning methods, stated as problem 24 in his list of 33 unsolved problems:

> Is there a mapping between clause representation and natural deduction representation (and corresponding inference rules and strategies) that

causes reasoning programs based respectively on the two approaches or paradigms to attack a given assignment in an essentially identical fashion?

Problem 24 remained unsolved for years. But the solution in terms of Peirce's version of natural deduction follows from the reversibility theorem. For Gentzen's version, translate each step in Peirce's version to one or more steps in terms of Gentzen's rules:

1. Start by deriving an EG or EGIF proof by resolution.
2. Draw a negation around each step of the proof, and reverse the order.
3. Since the last step was the pseudograph, ∼`[ ]`, the first step of the reversed proof will be a double negation around a blank, ∼`[`∼`[ ]]`. By rule 3e, erase the double negation to derive the blank as the new starting point.
4. The final step of the reversed proof will be a double negation of the theorem to be proved. By 3e, erase the double negation.

For propositional logic, there are no quantifiers. Each application of the resolution rule corresponds to four EG rules, which can be reversed inside a negation. With quantifiers, two additional procedures must also be checked for reversibility: *skolemization* in step 2 and *unification* in step 3. In clause form, all variables are governed by universal quantifiers: $\forall x_1, ..., \forall x_n$. Skolemization replaces each existential quantifier with a term t, which is a constant or a function applied to one or more of the universally quantified variables. Unification involves universal instantiations followed by joining the pegs of functions or relations that are being unified. After the pegs are joined, rule 2e allows redundant copies of relation or function symbols to be erased.

To show that skolemization and unification are reversible with EG rules, note that the universal quantifiers of an algebraic formula correspond to lines of identity or defining nodes `[*x1]`,...,`[*xn]` just inside the shaded area of the corresponding EG. To show that skolemization is reversible inside a negation, consider the following example, in which the skolem function $s(x)$ replaces the variable y:

$$\forall x \exists y R(x, y) \Rightarrow \exists f \forall x R(x, f(x)) \Rightarrow \forall x R(x, s(x))$$

If the left formula is true, the middle one must be true for at least one function f, and the right formula names such a function s. When negated, these implications are reversible: if the right formula is false, no function f exists for the middle one, and the left one is also false. This procedure can be generalized to any number of

universal quantifiers, including zero for a skolem constant. In Common Logic and EGIF, a skolem constant may be represented by a function with zero arguments.

Unification is an application of ∀-elimination followed by merging identical functions or relations applied to identical values. In EGs, it is performed by joining a subgraph that represents a constant or functional term to a line of identity followed by joining pegs according to the rule of inference discussed in Appendix A.3. Its inverse inside a negation corresponds to Gentzen's ∃-introduction. In EGs, it is performed by cutting the line and erasing the subgraph. The operation of joining lines whose values are known to be identical is an equivalence that can be reversed in any area.

For issues of efficiency and computability, the humanly readable notation is almost irrelevant, since most theorem provers translate any input formats to their internal format before beginning a proof. The translation time is usually trivial compared to the time required to find a proof. Stewart [57] showed that a theorem prover based on EGs with Peirce's rules of inference was comparable in performance to a resolution theorem prover. But the world's fastest theorem provers use multiple methods that are more specialized for the kinds of problem than for the input notations. They also depend on efficient methods for storing intermediate results and recognizing which are the most relevant at each step.

7 Natural Logic

Since the 1970s, psychologists, philosophers, linguists, and logicians have been searching for some innate *natural logic or language of thought*. The goal was a theory and notation that could support the full range of human reasoning from the babbling of an infant to the most advanced science [30, 10, 60]. But Richard Montague [38]) went one step further. He claimed that language and logic have the same foundation:

> There is in my opinion no important theoretical difference between natural languages and the artificial languages of logicians; indeed, I consider it possible to comprehend the syntax and semantics of both kinds of languages within a single natural and mathematically precise theory.

Montague's claim stimulated 40 years of research on formal semantics for natural language (NL). But MacCartney and Manning [33]) noted that "truly natural language is fiendishly complex. The formal approach faces countless thorny problems: idioms, ellipsis, paraphrase, ambiguity, vagueness, lexical semantics, the impact of pragmatics, and so on." Instead of a complete translation to a logical form, they

applied the term "natural logic" to a system of local features "which characterizes valid patterns of inference in terms of syntactic forms which are as close as possible to surface forms."

Montague did not approve of Chomsky's emphasis on syntax for NLs, and he would not accept a syntactic foundation for logic. Nor would Peirce, Polya, or Euclid. Peirce would treat Montague's claim as an empirical issue: What are the criteria for being humanly *natural*? Is there a unique logic for all languages? Or would the logic vary for different languages and cultures? How would variations in subject matter, levels of education, or businesses and professions affect the logical forms?

Wittgenstein [63] claimed that every natural language supports an open-ended variety of ways of talking and thinking, which he called *Sprachspiele* (language games). In 1939, he taught a course on the foundations of mathematics and logic [4]. Alan Turing attended all the lectures and debated the issues with Wittgenstein. Some logicians dismissed Wittgenstein's claims as eccentric, but the fact that Turing took them seriously is significant. Nobody knows exactly how Peirce would have replied to Montague or Wittgenstein, but the variety of logics that Peirce himself developed suggests that he might have had more sympathy with Wittgenstein [39].

Since all humans have similar bodies, sensory organs, and requirements for food, shelter, and social relations, they share a common core of concepts and ways of thinking. But cultures that have been isolated for centuries may have patterns of thought that don't have direct translations to one another. The Pirahã language of Brazil, for example, appears to be unusually difficult for outsiders to learn, even for linguists who know other languages of the region [7]. Cognitive linguists have explored these issues in depth [62, 29, 64, 31, 58].

In contrast with the linear, symbolic languages and logics, the psychologist Allan Paivio [40, 41] proposed a dual coding approach for relating language and imagery. The psycholinguist David McNeill [35] went one step further. He claimed that speech itself is multimodal: "Gestures orchestrate speech":

> Images and speech are equal and simultaneously present in the mind... Gestures look upward, into the discourse structure, as well as downward, into the thought structure. A gesture will occur only if one's current thought contrasts with the background discourse. If there is a contrast, how the thought is related to the discourse determines what kind of gesture it will be, how large it will be, how internally complex it will be, and so forth. [34, 2]

> Forced suppressions only shift the gesture to some other part of the body — the feet or an overactive torso. We once taped an individual who had

> been highly recommended to us as an elaborate and vigorous gesturer. Somewhat maliciously, when asked to recount our cartoon stimulus, he sat on his hands yet unwittingly began to perform the gestures typical of the experiment with his feet, insofar as anatomically possible. [35, 3]

When Peirce's rules of inference are applied to icons in two, three, or even four dimensions (3D plus time), they can accommodate images, diagrams, and gestures as an integral part of the formalism. Instead of being heuristic aids, the diagrams become formal components of the propositions, axioms, arguments, and proofs. In Peirce's own words, they provide "a moving picture of the action of the mind in thought" [46, p. 298]. In the same letter in which he wrote the tutorial on EGs (quoted in Section 2), Peirce explained what he meant by moving pictures:

> Boole plainly thought in algebraic symbols; and so did I, until, at great pains, I learned to think in diagrams, which is a much superior method. I am convinced there is a far better one, capable of wonders; but the great cost of the apparatus forbids my learning it. It consists in thinking in stereoscopic moving pictures. Of course one might substitute the real objects moving in solid space; and that might not be so very unreasonably costly. [48, 3: 191]

As Peirce said, thinking in moving pictures is far better than thinking in diagrams, which is superior to thinking in algebraic symbols. All three ways of thinking use icons, and the methods of observation and imagination may be used for all of them. With the modern software for virtual reality (VR), Peirce's dream can become a reality: an integrated combination of EG and VR.

Although semantics is the foundation for any logic, a convenient syntax can simplify the mapping to and from natural languages (NLs). The *discourse representation structure* (DRS) by Hans Kamp [23] is widely used as a logical notation for NL semantics. Coincidentally, the DRS notation, as developed by Kamp and Reyle [24] happens to be isomorphic to EGs. In fact, EGIF has a more direct mapping to DRS than to EG. As an example, Figure 33 shows an EG and DRS for the sentence *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 ¬ symbol, and implication is represented by two boxes connected by an arrow. As Figure 33 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

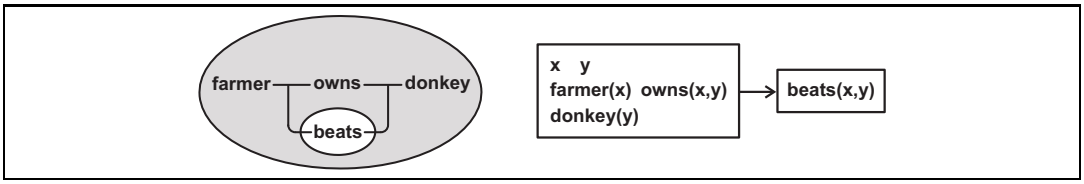

Figure 33: EG (left) and DRS (right) for *If a farmer owns a donkey, then he beats it.*

rules, the EG and DRS in Figure 33 can be translated to the same EGIF and to equivalent formulas in Peirce-Peano algebra:

```
[If (farmer *x) (donkey *y) (owns x y) [Then (beats x y)]]
```
$\sim (\exists x \exists y \text{ farmer}(x) \wedge \text{donkey}(y) \wedge \text{owns } (x, y) \wedge \sim \text{beats}(x, y))$
$\forall x \forall y \text{ farmer}(x) \wedge \text{donkey}(y) \wedge \text{owns } (x, y) \supset \text{beats}(x, y)$

As these translations show, the unlabeled EG lines are assigned labels `x` and `y` in both DRS and EGIF. The nested negations in the EG may be represented by the keywords `If` and `Then` in the EGIF, and those same keywords may be used to represent the DRS boxes. The first Peirce-Peano formula uses only existential quantifiers and negations. The second formula uses $\supset$ for implication, but the scoping rules require the $\exists$ quantifiers to be moved to the front and be replaced by $\forall$ quantifiers.

As another example of the similarity between EG and DRS, consider the following pair of sentences: *Pedro is a farmer. He owns a donkey.* Kamp and Reyle [24] 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. On the left of Figure 34 are the EGs for the two sentences; on the right are the DRSs. Following are the EGIF and the algebraic formulas. (Periods are added to separate the two formulas, but no periods are needed for EGIF.)

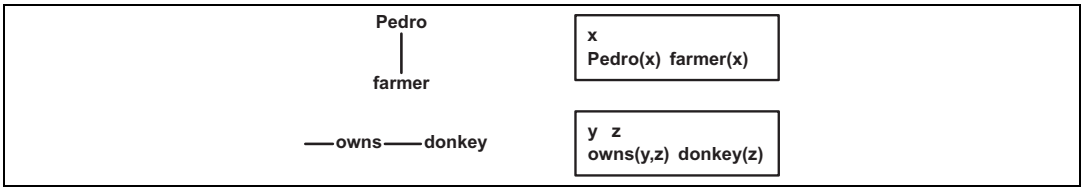

Figure 34: EGs and DRSs for *Pedro is a farmer. He owns a donkey.*

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

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. For the EG, DRS, and EGIF, the linkage is much simpler. To combine the two EGs, connect the line of identity for *Pedro* to the line for *he* in Figure 34. To combine the two DRSs, transfer the contents of one DRS box to the other, move the list of variables to the top, and insert the equality $x = y$. Following is the EGIF for Figure 35 and its simplified form after deleting the coreferences node `[x y]`:

```
(Pedro *x) (farmer x) (owns *y *z) (donkey z) [x y]
(Pedro *x) (farmer x) (owns x *z) (donkey z)
```

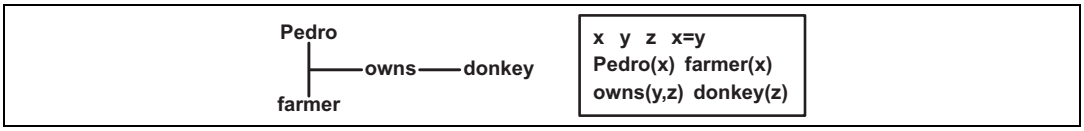

Figure 35: Combining the EGs and DRSs in Figure 34

Since the EG and DRS notations are isomorphic and they're represented by the same EGIF, Peirce's rules of inference can be applied to them in exactly equivalent steps. The proof in Figure 36 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.* The proof in Figure 36 takes four steps, but the first arrow combines two steps: extending lines by rule 2i and connecting lines by rule 1i. The following EGIF proof shows all four steps. For the DRS proof, translate each EGIF step to DRS.

0. Starting graphs: To avoid a name clash, the label `x` was replaced by `w`.
   ```
   (Pedro *w) (farmer w) (owns w *z) (donkey z)
   [If (farmer *x) (donkey *y) (owns x y) [Then (beats x y)]]
   ```

1. By rule 2i, extend the lines of identity by inserting nodes `[w]` and `[z]` in the area of the if-clause:
   ```
   (Pedro *w) (farmer w) (owns w *z) (donkey z)
   [If (farmer *x) (donkey *y) (owns x y) [w] [z]
   [Then (beats x y)]]
   ```

2. By 1i, connect the line for `w` to `x` and `z` to `y` and relabel:
   ```
   (Pedro *w) (farmer w) (owns w *z) (donkey z)
   [If (farmer w) (donkey z) (owns w z) [Then (beats w z)]]
   ```

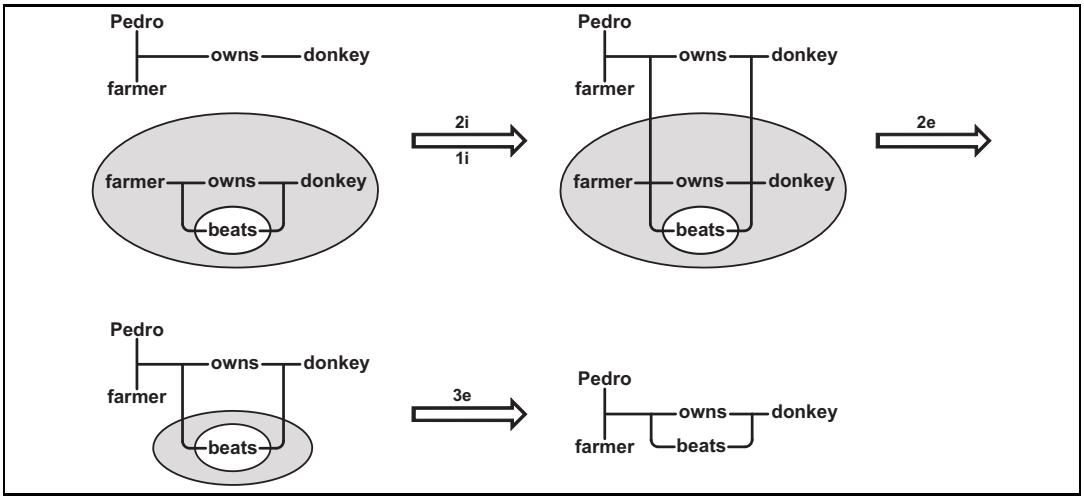

Figure 36: A proof according to Peirce's rules

3. By 2e, erase the copy of `(farmer w) (donkey z) (owns w z)` in the if-clause:
   `(Pedro *w) (farmer w) (owns w *z) (donkey z)`
   `[If [Then (beats w z)]]`

4. By 3e, erase the double negation (the if-then pair with an empty if-clause):
   `(Pedro *w) (farmer w) (owns w *z) (donkey z) (beats w z)`

For disjunctions, Figure 37 shows the EG and DRS for an example by Kamp and Reyle [24, p. 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 37 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. Both Peirce and Kamp allowed spaces inside relation names, but CGIF requires names with spaces or other special characters to be enclosed in quotes:

```
(Jones *x) (Smith *y) (Cooper *z)
~[ ~[ ("book on semantics" *u) (owns x u)]
   ~[ ("book on logic" *v) (owns y v)]
   ~[ ("book on unicorns" *w) (owns z w)] ]
```

For better readability, the Conceptual Graph Interchange Format allows the keywords Either and Or as synonyms for the negation symbol. That option could be added to EGIF:

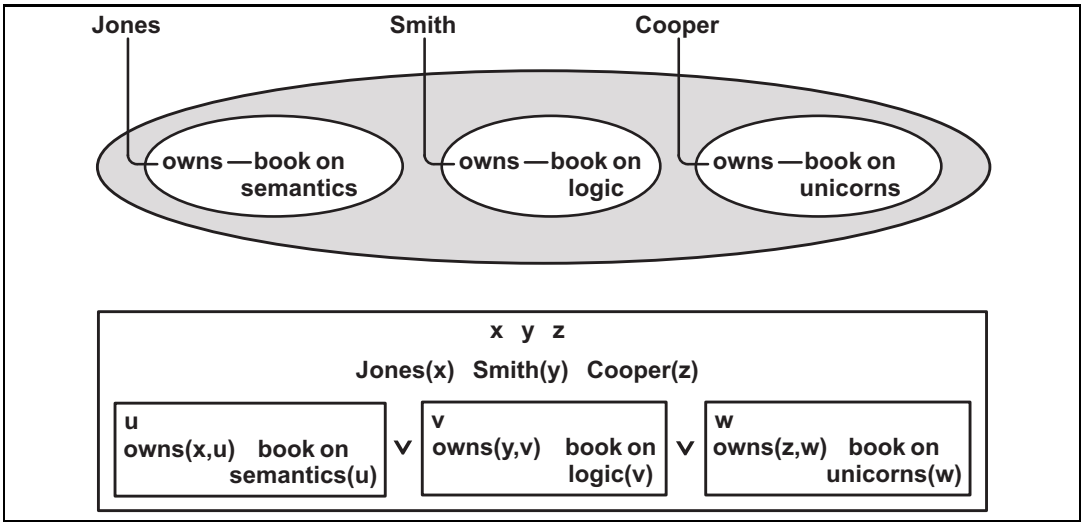

Figure 37: EG and DRS for a disjunction with three alternatives

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

The simplicity and generality of the EG structure, rules of inference, and methods for evaluating truth or falsity makes existential graphs a good candidate for a mental logic. But Figure 38 shows that Peirce's rules, as he stated them, aren't always applicable to the words and phrases of natural languages. Peirce's rules 1i and 1e, however, may be extended with the rules of specialization and generalization, which were derived in Section 6:

- **Specialization (Rule 1i):** In any negative area, any proposition q expressed in any notation may be replaced by a specialization p that implies q. In particular, p may be identical to $q \wedge A$ where A is an arbitrary proposition.
- **Generalization (Rule 1e):** In any positive area, any proposition p expressed in any notation may be replaced by a generalization q that is implied by p. In particular, p may be identical to $q \wedge A$ where A is an arbitrary proposition.

In Figure 38, the word *every*, which is on the boundary of an oval, creates the same pattern of nested negations as *if-then*. The word *cat*, by itself, expresses the proposition *There is a cat*. The phrase *cat in the house* expresses the conjunction *There is a cat and it is in the house*. By the rule of specialization (1i), the word cat

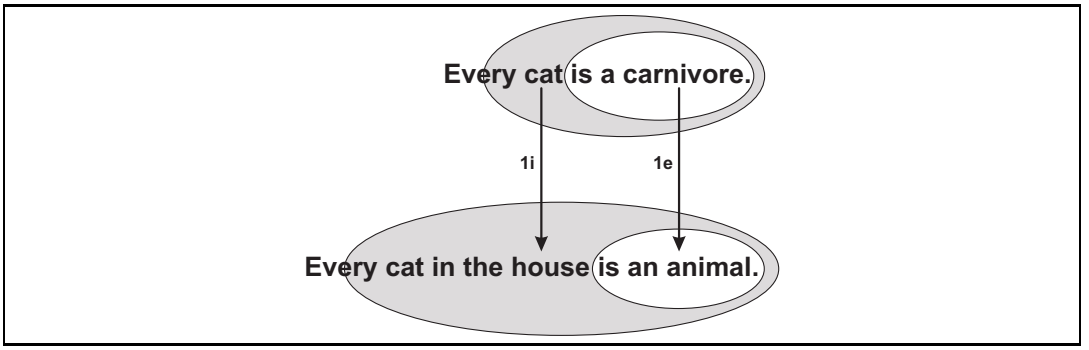

Figure 38: Specializing and generalizing English phrases

may be specialized to the phrase *cat in the house*. By the rule of generalization (1e), the word carnivore may be generalized to *animal*. In this example, the propositions are stated by printed words, but the mental models could represent them by icons with indexicals that link them to a real or imaginary world.

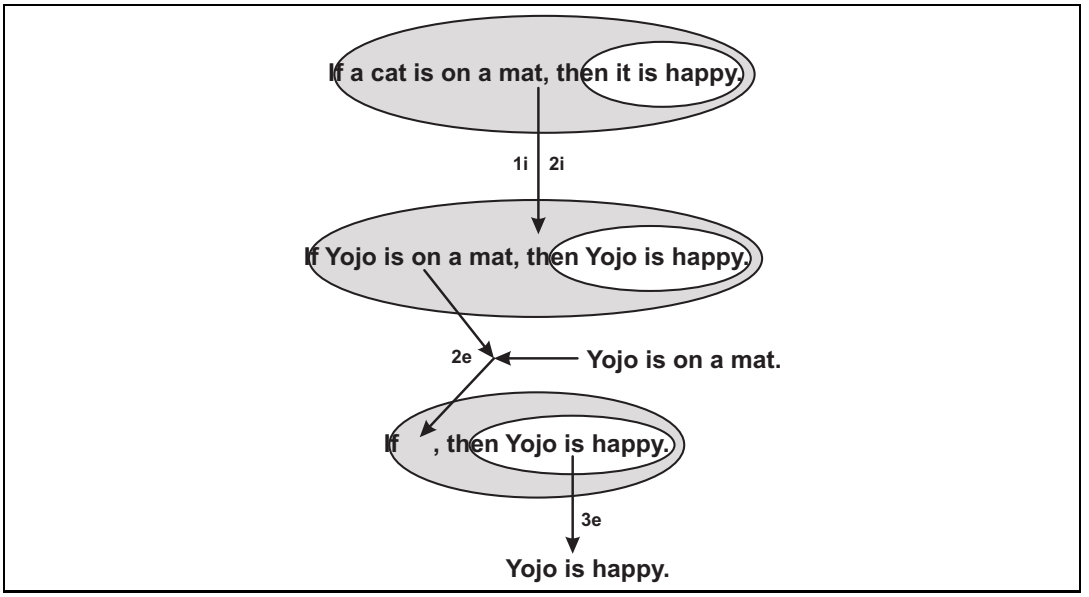

Figure 39: A four-step proof by Peirce's rules

Peirce's rules of iteration/deiteration and double negation may be used as he stated them, but some prior changes by the rules of generalization/specialization may be required. Figure 39 shows a four step proof. By specialization (1i), the clause *a cat is on a mat* is specialized to *Yojo is on a mat*. The pronoun *it* represents

an EG line of identity or an EGIF coreference node. By iteration (2i), a copy of the name Yojo replaces the coreferent pronoun *it*. 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 words *if* and *then* on the boundaries of the ovals are erased by the rule of double negation (3e).

Since Peirce's rules are sound and complete for EG, EGIF, and DRS, they are also sound and complete for that subset of any NL that can be translated to and from DRS. They are also sufficient to support the inferences discussed by MacCartney and Manning [33], but not the "fiendishly complex" aspects of natural languages. Majumdar and Sowa [37], however, have shown that a full translation of NL documents to conceptual graphs is practical. Furthermore, many of those complex aspects can be handled by using background knowledge derived from simpler passages in the same documents or from related documents and textbooks. Finally, if all else fails, a computer system can do what any human would do: ask a question. People rarely understand everything they read or hear. A large part of any discourse. is devoted to questions and metalevel discussions aspects that may be unclear or debatable.

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 40 shows one measure in the usual notation. Below it is a note-by-note translation to the boxes and circles of a conceptual graph (CG). 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. A translation of the CG to EG would have more nodes and take even longer to read. Traditional music notation is more iconic than an EG, but the most iconic is the music in a musician's head. An EG supplemented with arbitrary icons may include the traditional notation, a recording of the music, and links that connect both of them to EG lines of identity.

Although music notations may be translated to a logic that states the equivalent information, the music as played or heard is intimately connected with a wide range of feelings and previous experience. The connection of those feelings to any kind of language, logic, or diagrams can only be expressed by informal metaphors. Different musicians and listeners, or even the same person on different occasions, may use use vastly different metaphors.

Unlike Frege and Russell, who focused on the foundations of mathematics, Peirce integrated logic with semiotics and applied it to every aspect of science, philosophy, language, and life. To represent modal logics in existential graphs, Peirce experimented with colors and dotted lines. Frege insisted on a single domain of quantification called *everything*, but Peirce used *tinctures* to distinguish different *universes*:

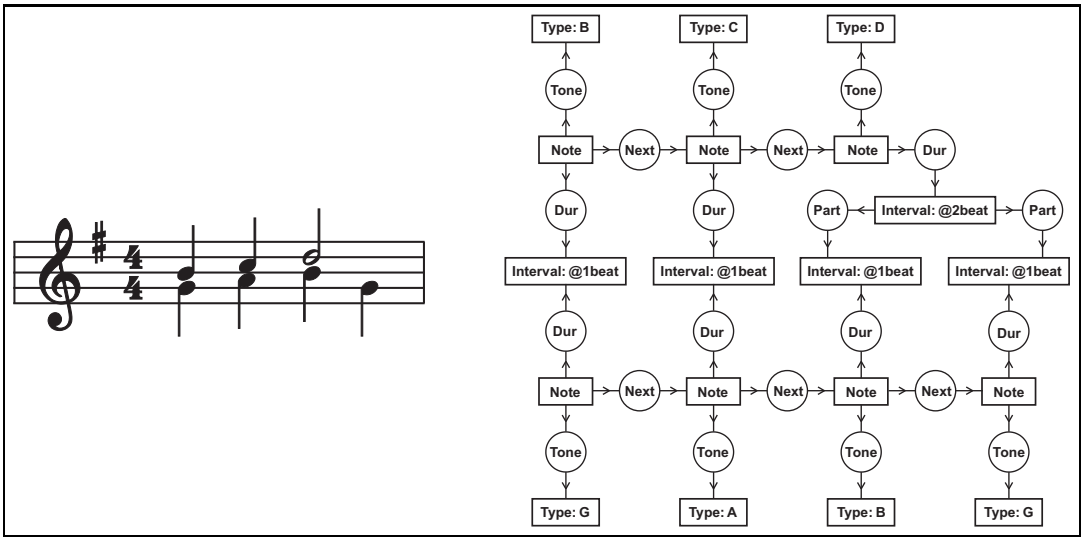

Figure 40: Two diagrams for representing the logic and ontology of music

> The nature of the universe or universes of discourse (for several may be referred to in a single assertion in the rather unusual cases in which such precision is required) is denoted either by using modifications of the heraldic tinctures, marked in something like the usual manner in pale ink upon the surface, or by scribing the graphs in colored inks. [46, 670:18]

Peirce considered three universes: actualities, possibilities, and the necessitated. He subdivided each universe in four ways to define 12 modes. In the universe of possibilities, for example, he distinguished objective possibility (an alethic mode), subjective possibility (epistemic), social possibility (deontic), and an interrogative mode, which corresponds to scientific inquiry by hypothesis and experiment. For the necessitated, he called the four subdivisions the rationally necessitated, the compelled, the commanded, and the determined. Most of his writings on these topics were unpublished, and he changed his terminology from one manuscript to the next. Peirce admitted that a complete analysis and classification would be "a labor for generations of analysts, not for one" [46, 478:165].

For the semantics of modal logic, Hintikka [18] proposed an infinite family of possible worlds, each described by the set of propositions that are true in that world. Kripke [28] showed how an *accessibility relation* among those worlds would imply the various axioms for different versions of modal logic. Dunn [5] combined Hintikka's method with Kripke's. He defined each world w by a set of laws and facts (L, F): the facts F of w are identical to Hintikka's propositions, and the laws

L are the subset of F that are necessarily true. Together, L and F imply Kripke's accessibility relation. Dunn's semantics has a direct mapping to Peirce's system. For each modality, the laws correspond to the necessitated; the facts correspond to the actualities; and the possibilities consist of all propositions consistent with the laws. For each world w, the possibilities include all the facts of w and of every world accessible from w.

Peirce's icons and his goals for 3D moving pictures correspond to the mental models by Johnson-Laird and his colleagues. But those models are rarely as well behaved as the theories that logicians specify. In fact, people typically accept statements that are false according to all versions of modal logic, including Peirce's: "Possibly it's raining and possibly it isn't" and "It isn't raining, but it might have been" [53]. The modal statements that people typically make also diverge from probability theory, and they're more consistent with the hypothesis that people are thinking in terms of some sort of mental model [17]. However, people who have some training in modal logic or in professions that require precision, such as science and the law, tend stay closer to the constraints of modal logics.

Peirce would not be surprised by the discrepancies between ordinary thinking and formal logics. In his classification of the sciences [45, 1.180-202], he placed formal logic under mathematics as pure theory, prior to any application. But he placed normative logic under philosophy as a standard for evaluating the way people actually think. Peirce realized that vagueness is inevitable, and that a vague statement, such as "Maybe it will rain, maybe not," can be preferable to an irrelevant attempt at precision:

> The vague might be defined as that to which the principle of contradiction does not apply... But wherever degree or any other possibility of continuous variation subsists, absolute precision is impossible. Much else must be vague, because no man's interpretation of words is based on exactly the same experience as any other man's. Even in our most intellectual conceptions, the more we strive to be precise, the more unattainable precision seems. It should never be forgotten that our own thinking is carried on as a dialogue, and though mostly in a lesser degree, is subject to almost every imperfection of language. [45, 5:505-506]

In summary, Peirce's writings, Wittgenstein's language games, and research in anthropology suggest that no single language, logic, or system of diagrams can include everything that might be called natural. As guidelines for analyzing all the possibilities, Peirce stated two fundamental principles:

> Do not block the way of inquiry. [45, 1.135]

> The elements of every concept enter into logical thought at the gate of perception and make their exit at the gate of purposive action; and whatever cannot show its passports at both those two gates is to be arrested as unauthorized by reason. [47, 2.241]

The first principle implies that the search must be open ended, not limited to any particular language, culture, or profession. The second requires a grounding in perception and action. Since anything perceptible can be represented by an icon, the option of including arbitrary icons in EGs suggests a way to support everything: use EGs as a metalanguage that may include, describe, and relate icons of any representation, linear or diagrammatic.

References

[1] Barwise, Jon, and John Etchemendy (1993). *Tarski's World*. Stanford, CA: CSLI Publications.

[2] Damasio, Antonio R. (2010) *Self Comes to Mind: Constructing the Conscious Brain*, New York: Pantheon Books.

[3] Dau, Frithjof (2006). Some notes on proofs with Alpha graphs. In *Conceptual Structures: Inspiration and Application*, (LNAI 4068), H. Schärfe, P. Hitzler, and P. Øhrstrom (eds.), 172-188. Berlin: Springer.

[4] Diamond, Cora (ed) (1975) *Wittgenstein's Lectures on the Foundations of Mathematics*, Cambridge 1939, Chicago: University of Chicago Press.

[5] Dunn, J. Michael (1973) A truth value semantics for modal logic, in H. Leblanc, ed., *Truth, Syntax and Modality*, Amsterdam: North-Holland, pp. 87-100.

[6] Euclid, *The Thirteen Books of the Elements*, translated by Thomas L. Heath, Second Edition, Dover: New York.

[7] Everett, Daniel L. (2008) *Don't Sleep, There Are Snakes: Life and Language in the Amazon Jungle*, London: Profile Books.

[8] Fine, Kit (1985). *Reasoning with Arbitrary Objects*. Oxford: Basil Blackwood.

[9] Frege, Gottlob (1879). Begriffsschrift. In *From Frege to Gödel*, J. van Heijenoort (ed.) (1967), 1-82. Cambridge, MA: Harvard University Press.

[10] Fodor, Jerry A. (1975) *The Language of Thought*, Cambridge, MA: Harvard University Press.

[11] Gentzen, Gerhard (1934). Untersuchungen über das logische Schließen [Investigations into logical deduction]. In *The Collected Papers of Gerhard Gentzen*, M. E. Szabo, editor and translator (1969), pp. 68-131. Amsterdam: North-Holland Publishing Co.

[12] Hadamard, Jacques (1945) *The Psychology of Invention in the Mathematical Field*, Princeton University Press, Princeton.

[13] Halmos, Paul R. (1968) Mathematics as a creative art, American Scientist 56, 375-389.

[14] Hayes, Patrick, and Chris Menzel (2006) IKL Specification Document, http://www.ihmc.us/users/phayes/IKL/SPEC/SPEC.html

[15] Henkin, Leon (1961). Some remarks on infinitely long formulas. In *Infinitistic Methods, (Proceedings of symposium on foundations of mathematics)*, 176-183. London: Pergamon Press.

[16] Hilpinen, Risto (1982). On C. S. Peirce's theory of the proposition: Peirce as a precursor of game-theoretical semantics. *The Monist* 65: 182-188.

[17] Hinterecker, Thomas, Markos Knauff, & Philip Johnson-Laird (2016) Modality, probability, and mental models, *Journal of Experimental Psychology: Learning, Memory, and Cognition* 42, 1606-1620.

[18] Hintikka, Jaakko (1961) Modality and quantification, *Theoria* 27, 110-128.

[19] Hintikka, Jaakko (1973). *Logic, Language Games, and Information*. Oxford: Clarendon Press.

[20] ISO/IEC (2007). Common Logic (CL) — A Framework for a family of Logic-Based Languages, (IS 24707). Geneva: International Organisation for Standardisation.

[21] ISO/IEC (1996) Extended BNF, IS 14977, International Organisation for Standardisation, Geneva.

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

[23] Kamp, Hans (1981) Events, discourse representations, and temporal references, *Langages* 64, 39-64.

[24] Kamp, Hans, & Uwe Reyle (1993) *From Discourse to Logic*, Dordrecht: Kluwer.

[25] Kamp, Hans, Josef van Genabith, & Uwe Reyle (2011) *Discourse Representation Theory: An Updated Survey*, in D. Gabbay (ed.), Handbook of Philosophical Logic, 2nd ed., Vol XV, pp. 125-394. http://www.ims.uni-stuttgart.de/institut/mitarbeiter/uwe/Papers/DRT.pdf

[26] Ketner, Kenneth Laine, & Hilary Putnam (1992) *Introduction to Peirce* (RLT).

[27] Knight, Will (2016) Can this man make AI more human? Interview with Gary Marcus, *MIT Technology Review*.

[28] Kripke, Saul A. (1963) Semantical considerations on modal logic, Acta Philosophica Fennica, Modal and Many-valued Logics, pp. 83-94.

[29] Lakoff, George (1987) *Women, Fire, and Dangerous Things*, University of Chicago Press, Chicago.

[30] Lakoff, George (1970) Linguistics and natural logic, *Synthese* 22, 151-271.

[31] Langacker, Ronald W. (1999) A view from cognitive linguistics, *Behavioral and Brain Sciences* 22, 625.

[32] Limber, John (1973) The genesis of complex sentences. In , T. Moore, ed., Cognitive *Development and the Acquisition of Language*, Academic Press, New York, 169-186.

[33] MacCartney, Bill, & Christopher D. Manning (2009) An extended model of natural

logic, *Proc. 8th International Conference on Computational Semantics*, Tilburg, pp 140–156.

[34] McNeill, David (1992) *Hand and Mind: What Gestures Reveal about Thought*, Chicago: University of Chicago Press.

[35] McNeill, David (2016) Why We Gesture: The surprising role of hand movements in communication, Cambridge: University Press.

[36] Mossakowski, Till, Mihai Codescu, Oliver Kutz, Christophe Lange, & Michael Grüninger (2014) Proof support for Common Logic, ARONL IJCAR, `http://www.iltp.de/ARQNL-2014/download/arqnl2014_paper5.pdf`

[37] Majumdar, Arun K., and John F. Sowa (2009). Two paradigms are better than one, and multiple paradigms are even better. In *Proceedings of ICCS 2009, (LNAI 5662)*, S. Rudolph, F. Dau, and S.O. Kuznetsov (eds.), 32-47. Berlin: Springer.

[38] Montague, Richard (1970) English as a formal language, reprinted in Montague (1974), pp. 188-221.

[39] Nubiola, Jaime (1996) Scholarship on the relations between Ludwig Wittgenstein and Charles S. Peirce, in I. Angelelli & M. Cerezo, eds., *Proceedings of the III Symposium on the History of Logic*, Gruyter, Berlin.

[40] Paivio, Allan (1971) *Imagery and Verbal Processes*, New York: Holt, Rinehart and Winston.

[41] Paivio, Allan (2007) *Mind and Its Evolution: A Dual Coding Approach*, New York: Psychology Press.

[42] Peirce, Charles Sanders (1869). Grounds of validity of the laws of logic. *Journal of Speculative Philosophy* 2: 193-208. `http://www.peirce.org/writings/p41.html` (accessed 15 November 2009).

[43] Peirce, Charles Sanders (1885). On the algebra of logic. American Journal of Mathematics 7:180-202.

[44] 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.

[45] Peirce, Charles Sanders (1931-1958 CP). Collected Papers of C. S. Peirce, C. Hartshorne, P. Weiss, and A. Burks (eds.), 8 vols., 1931-1958. Cambridge, MA: Harvard University Press.

[46] Peirce, Charles Sanders (1902) *Logic, Considered as Semeiotic*, MS L75, edited by Joseph Ransdell, `http://www.iupui.edu/~arisbe/menu/library/bycsp/L75/l75.htm`

[47] Peirce, Charles Sanders (EP) The Essential Peirce, ed. by N. Houser, C. Kloesel, and members of the Peirce Edition Project, 2 vols., Indiana University Press, Bloomington, 1991-1998.

[48] Peirce, Charles Sanders (NEM) The New Elements of Mathematics, ed. by Carolyn Eisele, 4 vols., The Hague: Mouton, 1976. Peirce, Charles Sanders (RLT). Reasoning and the Logic of Things, (The Cambridge Conferences Lectures of 1898), K. L. Ketner (ed) (1992). Cambridge, MA: Harvard University Press.

[49] Petitto, Laura-Ann (2005) How the brain begets language: On the neural tissue underlying human language acquisition. In J. McGilvray, ed., *The Cambridge Companion to Chomsky*, Cambridge: University Press, pp 84-101.

[50] Pietarinen, Ahti-Veikko (2006) *Signs of Logic: Peircean Themes on the Philosophy of Language, Games, and Communication*, Synthese Library, vol. 329, Berlin: Springer.

[51] Pólya, George (1954) *Mathematics and Plausible Reasoning*, Volume I: *Induction and Analogy in Mathematics*, Volume II: *Patterns of Plausible Inference*, Princeton: University Press.

[52] Putnam, Hilary (1982) Peirce the Logician, *Historia Mathematica* **9**:290-301, reprinted in Putnam (1990) pp. 252-260.

[53] Ragni, Marco, & P. N. Johnson-Laird (2018) Reasoning about possibilities: human reasoning violates all normal modal logics, PNAS, `http://mentalmodels.princeton.edu/papers/2018humans-vs-modal.pdf`

[54] Roberts, Don D. (1973). *The Existential Graphs of Charles S. Peirce*. The Hague: Mouton.

[55] Robinson, J. Alan (1965). A machine oriented logic based on the resolution principle. *Journal of the ACM* 12: 23-41.

[56] Sowa, John F. (2000) *Knowledge Representation: Logical, Philosophical, and Computational Foundations*, Brooks/Cole Publishing Co., Pacific Grove, CA.

[57] Stewart, John (1996). *Theorem Proving Using Existential Graphs*, MS Thesis, Computer and Information Science. Santa Cruz: University of California.

[58] Talmy, Leonard (2000) *Toward a Cognitive Semantics, Volume I: Concept Structuring Systems, Volume II: Typology and Process in Concept Structure*, Cambridge, MA: MIT Press.

[59] Tarski, Alfred (1936). Über den Begriff der logischen Folgerung [On the concept of logical consequence]. In *Logic, Semantics, Metamathematics*, A. Tarski (1982), 2nd ed, 409-420. Indianapolis: Hackett.

[60] van Benthem, Johan (2008) A brief history of natural logic, Technical Report PP-2008-05, Institute for Logic, Language, and Information. `http://www.illc.uva.nl/Publications/ResearchReports/PP-2008-05.text.pdf`

[61] Whitehead, Alfred North, & Bertrand Russell (1910) *Principia Mathematica*, 2nd ed. 1925, Cambridge: Cambridge University Press.

[62] Whorf, Benjamin Lee (1956) *Language, Thought, and Reality*, Cambridge, MA: MIT Press.

[63] Wittgenstein, Ludwig (1953) *Philosophische Untersuchungen*, translated as *Philosophical Investigations* by G. E. M. Anscombe, P. M. S. Hacker, & Joachim Schulte, revised 4th edition, Oxford: Wiley-Blackwell, 2009.

[64] Wierzbicka, Anna (1996) *Semantics: Primes and Universals*, Oxford: Oxford University Press.

[65] Wos, Larry (1988) *Automated Reasoning: 33 Basic Research Problems*, Englewood Cliffs, NJ: Prentice Hall.

A Appendix: EGIF Grammar

Existential Graph Interchange Format (EGIF) is a linear notation that serves as a bridge between EGs and other notations for logic. Over the years, Peirce had written manuscripts with variant notations, terminology, and explanations [54]. Those variations have raised some still unresolved questions about possible differences in their semantics. With its formally defined semantics, EGIF provides one precise interpretation of each graph translated to it. Whether that interpretation is the one Peirce had intended is not always clear. But the EGIF interpretation serves as a fixed reference point against which other interpretations may be compared and analyzed.

Every EGIF statement has a formally defined mapping to CGIF (Conceptual Graph Interchange Format), whose semantics is specified in ISO/IEC standard 24707 for Common Logic (CL). The semantics of the corresponding CGIF statement shall be called the *default semantics* of EGIF. To express the full CL semantics, the grammar rules in Section 2.3 specify two extensions to EGIF: (1) functions and (2) bound labels as names of relations or functions. EG relations, by themselves, can represent functions. But treating functions as a special case can simplify the inferences and shorten the proofs.

The option of bound labels as names of relations and functions goes beyond first-order logic. It supports some features of Peirce's Gamma graphs, which add extensions for higher-order logic, modal logic, and metalanguage. Section A.4 adds grammar rules for metalanguage in a way that is consistent with the proposed IKL extensions to Common Logic. Whether IKL is compatible with Peirce's intentions is another question for further research.

Section 2.1 presents lexical rules for the symbols and character strings of EGIF. These rules support the features that Peirce used and add new features to support the data types of modern programming languages. Section 2.2 presents the phrase-structure rules. Section 2.3 states context-sensitive constraints. Section 2.4 discusses extensions beyond Common Logic. But Peirce also speculated about variations that are not supported by these rules. Anyone who wishes to extend EGIF to represent them should state where they go beyond the version of EGIF specified here.

A.1 Lexical Rules

All EGIF grammar rules are stated as Extended Backus-Naur Form (EBNF) rules, as defined by [21]. The lexical rules specify names and identifiers, which exclude white space except as noted. The phrase-structure rules in Section 2.2 specify larger combinations that may have zero or more characters of white space between con-

stituents. Each EBNF rule is preceded by an English sentence that serves as an informative description of the syntactic category. If any question arises, the EBNF rule shall be normative.

The following four lexical categories are defined formally in Section A.2 of the ISO standard for Common Logic [20]. The brief definitions here are informative summaries. White space, which is any sequence of one or more white characters, is permitted only in quoted strings and enclosed names.

- A *digit* is any of the ten decimal digits: `0, 1, 2, 3, 4, 5, 6, 7, 8, 9`.
- An *integer* is an optional sign (+ or −) followed by a sequence of one or more digits.
- An *enclosed name* is any sequence of Unicode characters, except control characters, preceded and followed by a double quote `"`. Any double quote internal to an enclosed name shall be represented by the string \". Any backslash internal to an enclosed name shall be represented by the string \\.
- A *letter* is any of the 26 upper case letters from `A` to `Z` or the 26 lower case letters from `a` to `z`.
- A *white character is* a space, a tab, a new line, a page feed, or a carriage return.

In addition to the above lexical categories, EGIF uses the following lexical category, which is specified in Section B.2 of ISO/IEC 24707:

- An *identifier* is a letter followed by zero or more letters, digits, or underscores. `identifier = letter, {letter | digit | '_'};`

Identifiers and enclosed names are case sensitive: the identifier `Apple` is distinct from `apple`. But an identifier is considered identical to the enclosed name with the same case and spelling: Apple is identical to `"Apple"`, and apple is identical to `"apple"`. An integer is also considered identical to the enclosed name that contains the same sign and string of digits.

A.2 Phrase-Structure Rules

Unlike the lexical rules, the phrase-structure rules for EGIF permit an arbitrary amount of white space between constituents. White space is only necessary to separate adjacent identifiers. For example, the following two EGIF statements are semantically identical:

```
~[(mother*x)~[(woman x)]]
~ [ ( mother * x ) ~ [( woman x ) ] ]
```

In English, either statement may be read "If some x is a mother, then x is a woman." For better readability, a nest of two negations, which Peirce called a *scroll* may be written with the first negation `~[` replaced by `[If` and the second negation replaced by `[Then`:

```
[If (mother *x) [Then (woman x)]]
```

Each phrase-structure rule is described by an informal English comment and by a formal EBNF rule. For any question, the EBNF rule shall be normative. For any derivation, the rule that defines EG shall be the starting rule. The next paragraph defines informal terms and conventions that relate EGIF to the graphic EGs.

An *area* is a space where existential graphs may be drawn or asserted. The outermost area on which EGs may be asserted is called the *sheet of assertion* (SA). The SA may be cut or fenced off by oval enclosures, which are areas that contain *nested* EGs. For propositional and first-order logic (Peirce's Alpha and Beta graphs), the only oval enclosures are used to represent negations. In EGIF, the area of a negation is represented as `~[ EG ]`, where `EG` represents an existential graph as a set of *nodes* (possibly empty). Some of the nodes in the EG may be negations whose *nested areas* may contain more deeply nested EGs. There is no limit to the depth of nesting. In an informal definition, a term is printed in italics, as in *bound label*. But the same term in EBNF is printed, as in `Bound Label`. EBNF permits spaces in an identifier, but EGIF does not.

1. A *bound label* is an identifier. The bound label shall be *bound* to a defining label with the same identifier as the bound label.
   `Bound Label = identifier;`

2. A *coreference node* consists of a left bracket `[`, one or more names, and a right bracket `]`. A *defining node* is a coreference node that contains exactly one defining label. An *extension node* is a coreference node that contains exactly one name that is not a defining label.
   `Coreference Node = '[', Name, {Name}, ']';`

3. A *defining label* consists of an asterisk `*` and an identifier.
   `Defining Label = '*', identifier;`

4. A *double negation* is either a negation whose enclosed EG is a negation or a scroll whose first EG is a blank.

```
Double Negation = ' ', ' [', Negation, ']' | '[', 'If',
'[' 'Then', EG, ']' ']';
```

5. An *existential graph* (EG) is a set of nodes. An existential graph with zero nodes is called a blank or an empty existential graph. The order of nodes in the set is semantically irrelevant; but any node that contains a bound label shall follow (occur to the right of) the node that contains its defining label.
   `EG = {Node};`

6. A *function* consists of a left parenthesis `(`, a type label, zero or more names, a vertical bar `|`, a name, and a right parenthesis `)`. The names to the left of the bar may be called the inputs, and the one to the right may be called the output.
   `Function = '(', Type Label, {Name}, '|', Name, ')';`

7. A *name* is one of a defining label, a bound label, an identifier, an enclosed name, or an integer.
   `Name = Defining Label | Bound Label | identifier | enclosed name | integer;`

8. A *negation* consists of a tilde ∼, a left bracket `[`, an existential graph, and a right bracket `]`.
   `Negation = ' ', '[', EG, ']';`

9. A *node* is a coreference node, a relation, a function, a negation, or a scroll.
   `Node = Coreference Node | Relation | Function | Negation | Scroll;`

10. A *relation* consists of a left parenthesis (, a type label, zero or more names, and a right parenthesis).
    `Relation = '(', Type Label, {Name}, ')';`

11. A *scroll* consists of a left bracket `[`, the letters `If`, an EG, a left bracket `[`, the letters `Then`, an EG, and two right brackets `] ]`. Syntactically, a scroll is an optional notation for replacing the tilde in two negations with the keywords `If` and `Then`. Both notations have identical semantics.
    `Scroll = '[', 'If', EG, '[', 'Then', EG, ']', ']';`

12. A *type label* is any name except a defining label. If the name is a bound label, the value associated with its defining label determines the type of some relation or function.
    `Type Label = Name - Defining Label;`

A.3 Constraints and Examples

As the grammar rules show, an EG is represented by zero or more EGIF nodes. The only constraints on the nodes or their ordering are determined by the location of defining labels and their bound labels. The following six rules are equivalent to the rules for scope of quantifiers in predicate calculus. These rules are not needed for the graphic EGs, because the lines of identity show the scope by direct connections, not by labels.

1. No area may contain two or more defining labels with the same identifier.
2. The *scope* of a defining label shall include the area in which it occurs and any area nested directly or indirectly in this area, unless it is blocked by rule 5 below.
3. Every bound label shall be in the scope of exactly one defining label, which shall have exactly the same identifier. It is said to be *bound* to that defining label.
4. Every defining label shall precede (occur to the left of) every one of its bound labels.
5. If a defining label with some identifier x occurs in an area nested within the scope of a defining label in an outer area with the same identifier x, then the scope of the outer defining label shall be blocked from that area: every bound label with the identifier x that occurs in this inner area shall be bound to the defining label in this area.
6. In any area, all permutations of the nodes that preserve the above constraints shall be semantically equivalent.

A name enclosed in double quotes, such as `"John Q. Public"`, may contain spaces and punctuation. To avoid quotes, other stylistic options may be used, such as `John_Q_Public` or `JohnQPublic`, but these three variants are distinct. To specify multiple names as synonyms, put them in a coreference node:

```
["John Q. Public" John_Q_Public JohnQPublic JQPublic JQP]
```

Peirce treated functions as special special cases of relations. He didn't have a notation that distinguished them from ordinary relations. In EGIF, a function may be considered a kind of relation for which the value represented by the argument (or peg) after the vertical bar is uniquely determined by the values of the pegs that precede the bar. The pegs to the left of the vertical bar may be called the *inputs*, and the one to the right of the bar may be called the *output*.

- If f is a function with n inputs and if for every i from 1 to n, the ith input of one instance of f is coreferent with the ith input of another instance of f in the same area, then a line of identity may be drawn to connect the output pegs of both instances.

The act of drawing a line of identity between the output pegs in the graphic notation corresponds to inserting a coreference node in EGIF or an equality in other notations for logic. This rule of inference is the basis for the method of unification in theorem proving systems. The EGIF grammar allows functions with zero inputs; every instance of such a function would have exactly the same output value. Therefore, the output pegs of all instances of that function may be joined. These rules imply that a function with zero input pegs may be used to represent a *Skolem constant*.

Peirce did not introduce an EG notation for proper names or constants. Instead, he used monadic relations, such as `-Alexander` or `-is Alexander`. This relation would be true of anyone named Alexander. In EGIF, a name that is true of exactly one individual may be used as the name of a function with zero inputs. In the following function, the defining label `*p` would represent a line of identity for the unique person with that name:

```
("Philippus Aureolus Theophrastus Bombastus von Hohenheim"
| *p)
```

A name such as Alexander, which may be unique in a specific context, could be written as a function with an input peg for the context and an output peg for the person of that name: `(Alexander c | *p)`. Although Peirce did not define a notation for functions in EGs, the output peg of a function could be distinguished by an arrowhead in the graphic form.

The EBNF rules for relation and function allow a bound label to be the type label. That option supports the feature of Common Logic that allows quantified variables to refer to functions and relations. As an example, consider the sentence "There is family relation between any two members of the same family." To translate that sentence to any version logic, restate it with explicit variables F, R, x, and y: "For any family F and any two members x and y of F, there is a family relation R that is true of x and y." That sentence may be translated to the following EGIF:

```
[If (family *F) (memberOf *x F) (memberOf *y F)
   [Then (familyRelation *R) (R x y) ] ]
```

The expression `(familyRelation *R)` asserts that there exists a family relation R, and the next expression uses R as a type label. For his Gamma graphs, which are discussed in the next section, Peirce experimented with graphical ways of representing such options.

A.4 Extensions for Gamma Graphs

Peirce developed various notations for logic, algebraic and graphic. He also experimented with semantic extensions that go beyond first-order logic. In 1885, he introduced the algebraic notation that became predicate calculus [52]. He used the terms *first-intentional logic* for quantifiers that range over simple individuals and *second-intentional logic* for quantifiers that range over relations. In that article, he used second intentional logic to define equality $x = y$ by a statement that for every relation $R, R(x)$ if and only if $R(y)$. Ernst Schröder translated Peirce's terms as *erste Ordnung* and *zweite Ordnung*, which Bertrand Russell translated back to English as *first order* and *second order*. Peirce also introduced notations for three-valued logic, modal logic, and metalanguage about logic. Roberts [54] summarized the various graphical and algebraic notations and cited the publications and manuscripts in which Peirce discussed them.

Peirce used the term *Gamma graphs* for the versions of EGs that went beyond first-order (or first-intentional) logic. As early as 1898, he used the following example of a metalevel statement in EGs:

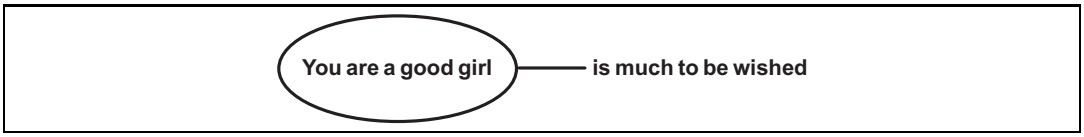

Figure 41: A metalevel existential graph by Peirce [44])

In describing that graph, Peirce wrote "When we wish to assert something about a proposition without asserting the proposition itself, we will enclose it in a lightly drawn oval, which is supposed to fence it off from the field of assertions." When translated to EGIF, the enclosed proposition would be a medad, a relation with no pegs: `("You are a good girl")`. The line of identity, which refers to that proposition could be represented by a defining label `*p`. To show that `p` refers to the proposition expressed by the EG, another syntactic option is necessary:

> A proposition node is a coreference node that contains a name and an EG that expresses the proposition that the name refers to.
> `Proposition Node = '[', Name, EG, ']';`

With this option, the EGIF for Figure 41 would be

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

For the semantics of that option, the IKL extension to Common Logic could be used. Hayes and Menzel [14] introduced an operator named *that* as a kind of function that

maps an IKL sentence to a name that refers to the proposition stated by that sentence. The IKL semantics for the that-operator could be adopted for proposition nodes in EGIF. These issues are discussed further in Section 6.

In some writings, Peirce used ovals with colors or dotted boundaries to represent modality. EGIF can use identifiers such as `Possible` or `Necessary` for relations applied to propositions. The following EGIF says "That you are a good girl is possible but not necessary."

```
[*p ("You are a good girl")] (possible p) ~[(necessary p)]
```

Examples like this may be generalized to any kind of metalevel propositions and methods for reasoning about them: modal, multivalued, intuitionistic, fuzzy, or statistical. In his published works and much more voluminous unpublished manuscripts, Peirce analyzed and proposed many semantic features. For Alpha and Beta graphs, the semantics of Common Logic or its FOL subset is consistent with Peirce's examples. But determining exactly what he had in mind for his many variations of Gamma graphs is still a research project. In some late manuscripts, he also mentioned a version of Delta graphs. No one knows what Peirce intended, but EGIF can be a useful tool for exploring the options.

A.5 Computer tools for processing EGIF

EGIF was originally defined as a subset of the Conceptual Graph Interchange Format (CGIF), one of the dialects of Common Logic (CL). But after some simplifications in the notation, the version of EGIF defined in this appendix still has a direct mapping to CGIF, and its semantics is still defined by its mapping to CGIF. But its syntax is simpler. As an example, "A cat is on a mat" would be translated to the following EGIF:

```
(Cat *x) (On x *y) (Mat y)
```

To translate any EGIF sentence to an equivalent CGIF sentence, put every defining label in a defining node at the beginning of the area in which it occurs. Then replace each defining label inside a relation node with a bound label with the same identifier. The result is an EGIF sentence that is logically equivalent to the original:

```
[*x] [*y] (Cat x) (On x y) (Mat y)
```

To derive a logically equivalent CGIF sentence, place a question mark `?` in front of every bound label:

```
[*x] [*y] (Cat ?x) (On ?x ?y) (Mat ?y)
```

The equivalent sentence in the Common Logic Interchange Format (CLIF) requires the keyword exists for the quantifier and an explicit Boolean operator for the implicit conjunction:

```
(exists (x y) (and (Cat x) (On x y) (Mat y)))
```

The Heterogeneous Tool Set (HeTS) provides theorem provers for Common Logic and translators to and from CL and other logics, including the logics for the Semantic Web [36]. Any EGIF sentence may be translated to CGIF or CLIF by the above steps and be processed by any of the HeTS tools. For natural languages, discourse representation theory [25] addresses the linguistic analysis that is prior to any representation in logic. But the base logic (DRS) is isomorphic to EGs and CGIF. For examples, see Figures 33 to 37.

Received 28 February 2018