Diagrammatic Reasoning With EGs and EGIF

John F. Sowa

Abstract. Diagrammatic reasoning dominated mathematics until algebraic methods became popular in the 17th century. Although Peirce developed the algebraic notation for predicate calculus in 1885, he kept searching for a more iconic graph-based logic. In 1897, he developed existential graphs (EGs), which have the full expressive power of the ISO standard for Common Logic (CL). In the last two decades of his life, Peirce wrote extensively about diagrammatic reasoning as a general theory and EGs as the formal representation. They are two aspects of the same theory. Peirce's rules of inference for EGs can be generalized to any notation for logic, linear or graphic. Two additional rules, which he described informally, are *observation* and *imagination*. For precisely defined diagrams, as in Euclid's geometry, those operations can be defined formally. To bridge the gap between algebra and diagrams, the Existential Graph Interchange Format (EGIF) has a formal mapping to EGs and to Common Logic. When the rules of observation and imagination are defined in terms of EGIF, they enable any software developed for Common Logic to be used in diagrammatic reasoning.

This is a slightly revised preprint of an article that was published in *Cuadernos de sistemática peirceana* **8**, Bogotá: Centro de Sistemática Peirceana, 2016, pp. 44-76..

1. Syntax of Existential Graphs

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 (Putnam 1982). 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 propositional EGs (Alpha) and first-order EGs (Beta), Peirce used the same structure and rules of inference in every manuscript from 1897 to 1911. For Gamma graphs, which represent metalanguage, higher-order logic, and modal logic, he developed syntactic and semantic variations (Roberts 1973; Pietarinen 2006). For diagrammatic reasoning, see *Diagrammatology* by Stjernfelt (2007).

The examples in Figures 1 to 3 and the explanations by Peirce are taken from a tutorial he wrote in 1911 (NEM 3:159). The EG on the left of Figure 1 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." For the other two EGs, Peirce explained, "To deny that there is any phoenix, we shade that assertion which we deny as a whole [EG in the middle]. Thus what I have just scribed 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'."

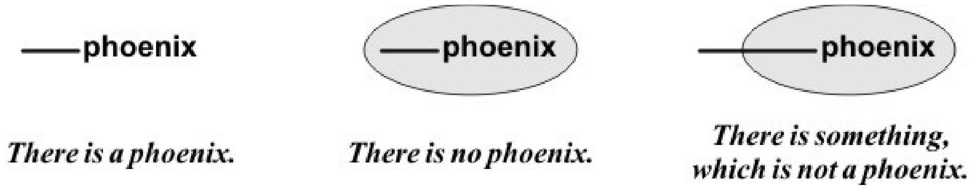

Figure 1. Three existential graphs

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. The following table shows the Existential Graph Interchange Format (EGIF) for each of the graphs in Figure 1 and the corresponding formula in predicate calculus (Peirce-Peano algebra).

Figure 1	EGIF	Predicate Calculus
Left	**(phoenix *x)**	**∃x phoenix(x)**
Middle	**~[(phoenix *x)]**	**~∃x phoenix(x)**
Right	**[*x] ~[(phoenix x)]**	**∃x ~phoenix(x)**

In the EGIF for the graph on the left of Figure 1, 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 ∃**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 square brackets to enclose the negated subgraph: **~[ ]**. 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 2. 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 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’.” Since a nest of two negations occurs frequently in EGs, Peirce called it a *scroll* and drew it without raising his pen.

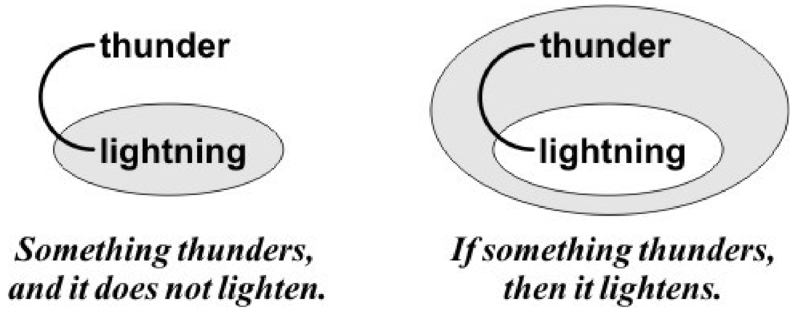
thunder
lightning
Something thunders, and it does not lighten.
thunder
lightning
If something thunders, then it lightens.

Figure 2. An EG with one negation and an EG with two negations

In EGs and EGIF, conjunction is implicit; no symbol is required. When EGIF is translated to predicate calculus with just the operators ∧,, **~** and ∃, the scope of the ∃**x** quantifier is identical to the scope of the defining label ***x**. EGIF also uses the option of replacing the two negations with the keywords **If** and **Then**. That option has no effect on the scope of defining labels. But when predicate calculus uses the implication operator ⊃, the ∃**x** quantifier must be moved to the front and be converted to the universal quantifier ∀**x**.. That conversion is not necessary with EGs or EGIF.

Figure 2	EGIF	Predicate Calculus
Left	**(thunder *x) ~[(lightning x)]**	**∃x (thunder(x) ∧~lightning(x))**
Right	**~[(thunder *x) ~[(lightning x)]]**	**~∃x (thunder(x) ∧~lightning(x))**
Optional	**[If (thunder *x) [Then (lightning x)]]**	**∀x (thunder(x) ⊃lightning(x))**

Peirce continued, “A graph may be complex or indivisible. Thus [Figure 3 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.”

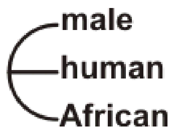

Figure 3. There is a male African human

In EGIF, Figure 3 may be represented by three indivisible nodes followed by a *coreference node* **[x y z]**, which shows 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 next table.

Figure 3	**EGIF**	**Predicate Calculus**
Teridentity	**(male *x) (human *y) (African *z) [x y z]**	**∃x ∃y ∃z (male(x) ∧human(y) ∧African(z) ∧x=y ∧y=z)**
One label	**(male *x) (human x) (African x)**	**∃x (male(x) ∧human(x) ∧African(x))**

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. When an EG is translated to EGIF, the labels that correspond to each peg are listed in the order of the pegs, as viewed from left to right. When Peirce drew EGs, he was consistent in preserving that order.

Peirce represented a *proposition* as a relation with zero pegs. He called it a *medad*; the prefix *me-* comes from the Greek *mêden* for *nothing*. 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 semantics and rules of inference apply to both.

Peirce: “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)**
>
> **∃x ∃y ∃z (man(x) ∧x=y ∧y=z ∧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 (Sowa 2011). 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.

All other operators of first-order logic are represented by combining these primitives. Figure 4 shows three composite operators defined in terms of nested negations.

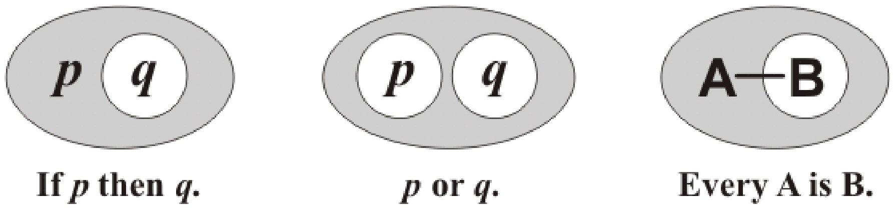

Figure 4. Three composite operators

Figure 4	**EGIF**	**Predicate Calculus**
If p, then q.	**[If (p) [Then (q)]]**	**p ⊃q**
p or q.	**~[~[(p)] ~[(q)]]**	**p ∨q**
Every A is B.	**[If (A *x) [Then (B x)]]**	**∀x (A(x) ⊃B(x))**

Although these three operators are composite, their graphic patterns are just as readable as the algebraic formulas with Peano's symbols ⊃, ∨, and ∀. In fact, the explicit nesting of EG ovals directly shows the scope of quantifiers. But the operators ∧, ∨, and ⊃ look so similar that students find it hard to 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 because they are represented as relations with zero logical subjects (arguments).

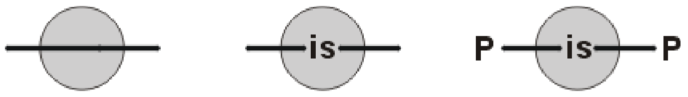

Figure 5. Stating that two things are not identical

Sometimes, the dyad **—is—** can clarify the translation of an EG to a sentence or formula. For example, Figure 5 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."

Figure 5	**EGIF**	**Predicate Calculus**
Left	**[*x] [*y] ~[[x y]]**	**∃x ∃y ~(x=y)**
Middle	**[*x] [*y] ~[(is x y)]**	**∃x ∃y ~is(x,y)**
Right	**(P *x) (P *y) ~[(is x y)]**	**∃x ∃y (P(x) ∧P(y) ∧~is(x,y))**

As these examples show, EGs state 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≠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 crisscrossing lines may be clearer if some connections are shown by labels. He would erase part of a line or ligature and place a letter, such as **x**, at each endpoint where the line had been erased.

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 1 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 3.) 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 5.) For the formal EGIF syntax, see Section 5. For more examples, see "An introduction to EGs" (Sowa 2017).

2. Common Logic

Common Logic (CL) evolved from two projects to develop parallel standards for conceptual graphs (CGs) and the Knowledge Interchange Format (Genesereth & Fikes 1992). Eventually, the CG and KIF projects were merged in Common Logic, for which Hayes and Menzel (2001) defined the model theory. The CL standard specifies an abstract syntax with three dialects that express the full semantics: the Common Logic Interchange Format (CLIF), the Conceptual Graph Interchange Format (CGIF), and the XML-based notation XCL (ISO/IEC 24707).

Conceptual graphs are a typed version of logic, but Peirce's untyped EGs are sufficiently general to express the full CL semantics. To express both, the standard defines two versions of CGIF:

- **Core CGIF or EGIF.** A typeless notation that expresses the full semantics. EGIF is semantically identical to core CGIF, but it also has some syntactic options that are not available in ISO standard CGIF. A revised, but not standardized version of CGIF includes those options.
- **Extended CGIF.** An extension of the core, which adds a universal quantifier; type labels for restricting the range of quantifiers; Boolean contexts with type labels **If**, **Then**, **Either**, **Or**, **Equivalence**, and **Iff**; and the option of importing external text into any CGIF text.

Although extended CGIF is a typed language, it is not *strongly typed*, because type labels are used only to restrict the range of quantifiers. Instead of causing a syntax error, a type mismatch in CGIF just causes the subexpression in which the mismatch occurs to be false.

As an example, Figure 6 shows an EG and a CG for the sentence *If a cat is on a mat, then it is a happy pet.* In the shaded area of the EG, the two lines of identity represent a cat and a mat, and the ligature joins the line for the cat to the line for the pet. In the CG, the dotted line, called a *coreference link*, connects the concept **[Cat]** to the concept **[Pet]**. It indicates that both concept nodes refer to the same entity. The **Attr** relation indicates that the cat, also called a pet, has an attribute, which is an instance of happiness.

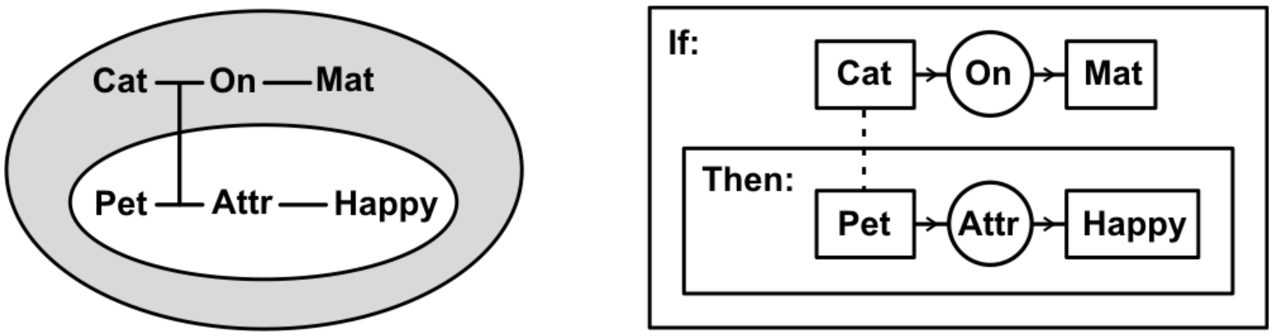

Figure 6: EG and CG for *If a cat is on a mat, then it is a happy pet.*

In both EGIF and CGIF, defining labels are shown with an asterisk, as in ***x** for the cat. For bound labels, ISO standard CGIF adopted the KIF convention with a question mark as prefix, as in **?x**. But EGIF and revised CGIF omit the question mark. Following are the translations of Figure 6 to EGIF, revised CGIF, and CLIF:

EGIF: **[If (Cat *x) (Mat *y) (On x y)
[Then (Pet x) (Happy *z) (Attr x z)]]**

CGIF: **[If [Cat *x] [Mat *y] (On x y)
[Then [Pet x] [Happy *z] (Attr x z)]]**

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

The three notations have similar and sometimes identical syntax for relations. For longer statements, there are three syntactic differences: the conjunction **and** is explicit in CLIF, but implicit in the others; quantifiers are explicit in CLIF, but just an asterisk in the others; and the scope of quantifiers is different for if-then statements. As a result, CLIF usually requires more parentheses.

As a mathematician, Peirce recognized the importance of functions. But in his logic, he did not distinguish functions and relations: a function of N arguments may be represented as a relation with N+1 arguments, in which the value of argument N+1 is determined by the values of the first N. Common Logic does distinguish functions from relations, since the *unification algorithm* for functional expressions is important for theorem provers. In EG diagrams, a function may be distinguished from a relation by an arrowhead on the last line of identity attached to the function name. Figure 7 shows an EG that represents a functional expression that combines four functions named **+1**, **+**, **sqrt**, and **÷**.

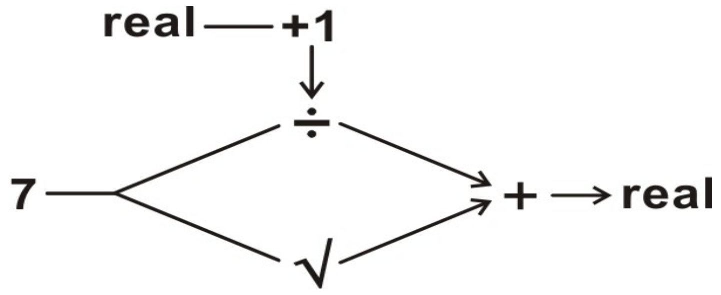

Figure 7: An EG that represents a functional expression

Following are three linear notations for defining a function *f* by the expression in Figure 7. The first line is a common algebraic notation for defining a function *f*(*x*) in terms of an expression that contains *x*. The second is an EGIF definition of **(f x | y)**. The third is the CLIF equivalent of the first two.

***f*(*x*) = (7 ÷ (*x* + 1)) + sqrt(7)**

```
[If   (number *x)   (number *y)   ("+1" x | *u)   ("÷" 7 u | *v)   (sqrt 7 | *w)   ("+" v w | y)
    [Then   (f x | y) ] ]
```

```
(forall   ((x number)  (y number))   (if   (= y   ("+"   ("÷" 7 ("+1" x))   (sqrt 7)) ))   (= y (f x)) ))
```

This example illustrates several points: CLIF notation is closer to ordinary algebra than EGIF notation, which requires the additional labels **u**, **v**, and **w**. Even though CLIF requires more parentheses, the extra labels make the EGIF statement somewhat longer. Both CLIF and EGIF use the same names for functions and relations; names that contain special characters, such as **"+1"**, **"+"**, and **"÷"**, must be enclosed in quotes. When an EG diagram is mapped to CLIF or EGIF, any line with an outgoing arrowhead is the last argument of a function. All other lines attached to a function or relation, including those with incoming arrowheads, are listed in order from left to right.

Common Logic allows quantifiers to range over functions and relations, but CL retains a first-order style of model theory and proof theory. To support a version of higher-order logic without the computational complexity of an infinite hierarchy of relations, the CL model theory uses a single domain D that includes individuals, functions, and relations. The option of single domain was suggested by Quine (1954) and used in various theorem provers that allow quantifiers to range over functions and relations (Chen et al., 1993). In his introduction of higher-order logic in 1885 and his later model theory for EGs (endoporeutic), Peirce assumed a single domain that contained all individuals and relations. In some manuscripts on modal logic, he discussed multiple “universes of discourse”, but each universe contained both individuals and relations. For these reasons, the single domain of CL semantics seems appropriate for Peirce’s version of HOL.

Common Logic is more expressive than most logic-based languages and notations. 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 (Mossakowski et al. 2014). EGIF may be translated CLIF and be processed by the HETS tools. Delugach (2014) designed a CGIF parser and tools for generating the graphic form of conceptual graphs. Those tools, which are available from Source Forge, may be adapted to EGs and EGIF.

For Peirce’s own rules of inference, the model-theoretic semantics he called *endoporeutic*, and comparisons with other proof procedures, see the commentary by Sowa (2011). Stewart (1996) showed that a theorem prover based on Peirce’s rules was comparable in speed to a theorem prover based on resolution. Dau (2006) showed that for certain kinds of proofs, Peirce’s rules could be orders of magnitude faster than the more common methods.

3. Propositions, Situations, and Metalanguage

Peirce’s first use for the oval was to negate the graphs nested inside. But Peirce (1898) generalized the ovals to context enclosures, which allow relations other than negation to be linked to an oval. Figure 8 shows his example of an EG for the sentence “That you are a good girl is much to be wished.” This oval is not shaded because it’s not negated.

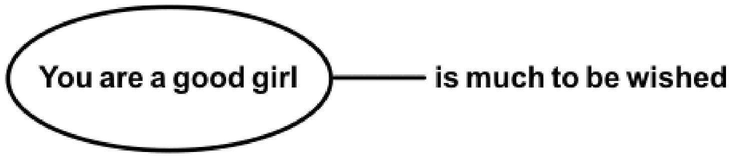

Figure 8: An EG with a metalevel comment about an EG

In describing Figure 8, 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.” Since Peirce allowed blanks to occur in the names of relations, he used the English sentence **You are a good girl** as the name of a proposition.

For Peirce and for Common Logic, a proposition is a relation with zero pegs (or arguments). In the three notations EGIF, CGIF, and CLIF, names that contain blanks or special characters must be enclosed in double quotes. Therefore, **"You are a good girl"** may be used as the name of a proposition and **"is much to be wished"** as the name of a monadic relation. To express the EG in Figure 8, the monadic relation is asserted of the proposition: **("is much to be wished" "You are a good girl")**.

If a proposition does not have a name, some operator is necessary to give it a name. For that purpose, the IKL extension to Common Logic has an operator named **that**, which serves as a metalevel function from propositions to names (Hayes & Menzel 2006). For example, the sentence “Bill thinks that a happy cat is on a mat” makes a statement about a proposition that does not have a name. That proposition may be represented by the following EGIF:

```
(Cat *x) (Happy *y) (Attr x y) (Mat *z) (On x z)
```

Literally, this EGIF says “x is a Cat; y is an instance of happiness; x has attribute y; z is a mat; and x is on z.” Following are the EGIF and CLIF notations for mapping this statement to a name of the proposition:

```
(Think *u) (Agnt u Bill) (Thme u *v)
    (that (Cat *x) (Happy *y) (Attr x y) (Mat *z) (On x z) | v)
(exists (u) (and (Think u) (Agnt u Bill)
    (Thme u (that (exists (x y z) (and (Cat x) (Happy y) (Attr x y) (Mat z) (On x z))))) ))
```

The first line of the EGIF says “There is an instance of thinking u, the agent of u is Bill, and the theme of u is v." The second line assigns the name **v** to the proposition. With its functional notation, CLIF does not require the name **v**.

To support the IKL extension in EGIF, the following grammar rule may be added to Section 5.2:

```
Metafunction  =  '(' 'that' EG '|' Name ')' ;
```

Language about language is necessary for talking about the beliefs, desires, and intentions of the speaker and anyone else who may be mentioned. For example, the sentence *Tom believes that Mary wants to marry a sailor*, contains three clauses, whose nesting may be marked by brackets:

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

The outer clause asserts that Tom has a belief, which is the proposition that Mary wants a situation, which is described by the infinitive *to marry a sailor*. Each clause makes a comment about the clause nested in it. References to the individuals mentioned in those clauses may cross context boundaries in various ways. The sailor, for example, may exist in any of those three contexts:

1. Tom believes that Mary wants to meet someone who is sailor and marry him.
2. Tom believes that there is a sailor and Mary wants to marry him.
3. There is a sailor, and Tom believes that Mary to marry wants him.

The two conceptual graphs in Figure 9 represent the first and third interpretations. In the CG on the left, the existential quantifier for the concept **[Sailor]** is nested inside the situation that Mary wants. Whether such a sailor actually exists and whether Tom or Mary knows his identity are undetermined. The CG on the right explicitly states that such a sailor exists; the connections of contexts and relations imply that Tom knows him and that Tom believes that Mary also knows him. The second option (not shown in Figure 9) would place the concept **[Sailor]** inside the context of type **Proposition**; it would leave the sailor's existence undetermined, but it would imply that Tom believes he exists and that Mary knows him.

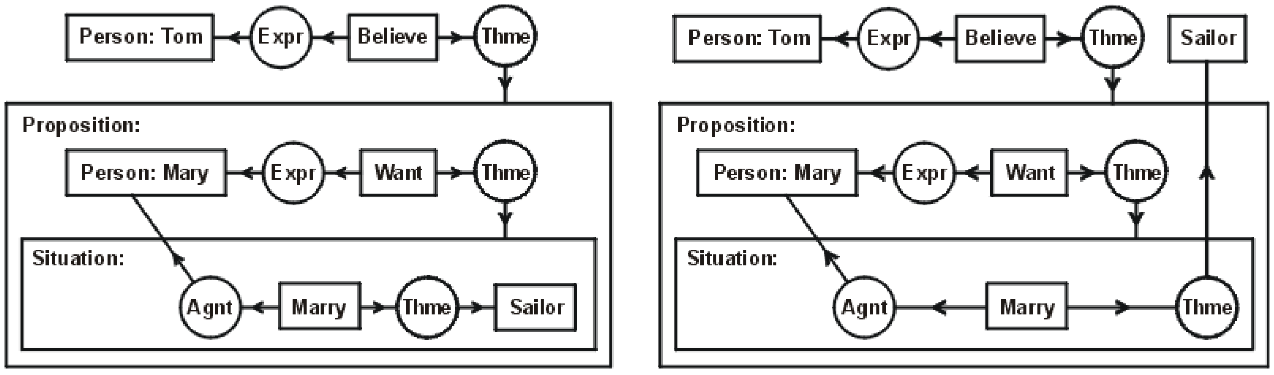

Figure 9: Two interpretations of *Tom believes that Mary wants to marry a sailor*

The oval in Figure 9 represents a proposition that already has a name. Common Logic in any notation can represent it. In Figure 9, the propositions do not have names, but the IKL operator **that** can create a name for them. But the situations in Figure 9 are regions of space-time. Neither CL nor IKL has any special provision for situations, but any version of logic that can represent propositions can specify a situation as as a region of space-time described by a proposition.

The first step toward translating the CGs in Figure 9 to IKL is to write them in an extended version of CGIF, which allows CGs to be nested inside concept nodes of type **Proposition** or **Situation**. Following is the CGIF for the CG on the left:

```
[Person: Tom] [Believe: *x1] (Expr x1 Tom) (Thme x1 *p1)
    [Proposition: p1 [Person: Mary] [Want: *x2] (Expr x2 Mary) (Thme x2 *s)
        [Situation: s [Marry: *x3] [Sailor: *x4] (Agnt x3 Mary) (Thme x3 x4) ] ]
```

In this CGIF, **Expr** is an abbreviation for *Experiencer*, **Thme** for *Theme*, and **Agnt** for *Agent*. The CG in Figure 9 and the corresponding CGIF use some syntactic options to make the formal statement more concise and closer to English word order. The relation **Dscr** (*Description*) relates a situation to a proposition that describes it. These abbreviations and definitions may be expanded to Core CGIF, which is identical to EGIF:

```
(Person Tom) (Believe *x1) (Expr x1 Tom) (Thme x1 *p1) (Proposition p1)
   (that (Person Mary) (Want *x2) (Expr x2 Mary) (Thme x2 *s) (Situation s)
      (Dscr s *p2) (that (Marry *x3) (Sailor *x4) (Agnt x3 Mary) (Thme x3 x4) | p2 ) | p1 )
```

This EGIF can then be translated to CLIF notation:

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

As these examples illustrate, diagrams and "syntactic sugar" can make logic more readable. Figure 9 directly shows the nested English clauses. Although the corresponding CGIF is not quite as readable, it still has a direct mapping to English. EGIF adds more detail, and the deep nesting of CLIF creates a tail of eleven closing parentheses.

4. Diagrammatic Reasoning

Peirce's writings on logic, semiotic, and diagrammatic reasoning, which had been neglected during the 20th century, are now at the forefront of research in the 21st. Frege's rejection of psychologism and "mental pictures" reinforced the behaviorism of the early 20th century. But the latest work in neuroscience uses "folk psychology" and introspection to interpret the data from brain scans (Dehaene 2014). The neuroscientist Damasio (2010) explicitly said that brains create "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 psychologist Johnson-Laird (2002), who had written extensively about mental models, said that Peirce's existential graphs and rules of inference are a good candidate for a neural theory of reasoning:

> 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 theories.

Diagrammatic reasoning is one of Peirce's most brilliant insights. His observations, quoted below, would be obvious to many mathematicians (Polya 1954). But they undermine the theories of Frege and the mainstream of analytic philosophy:

> 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. (EP 2:212)

> 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. (NEM 4:219)

> We form in the imagination some sort of diagrammatic, that is, iconic, representation of the facts, as skeletonized as possible. The impression of the present writer is that with ordinary persons this is always a visual image, or mixed visual and muscular... This diagram, which has been constructed to represent intuitively or semi-intuitively the same relations which are abstractly expressed in the premisses, is then observed, and a hypothesis suggests itself that there is a certain relation between some of its parts — or perhaps this hypothesis had already been suggested. In order to test this, various experiments are made upon the diagram, which is changed in various ways. (CP 2.778)

> Diagrammatic reasoning is the only really fertile reasoning. If logicians would only embrace this method, we should no longer see attempts to base their science on the fragile foundations of metaphysics or a psychology not based on logical theory. (CP 4.571)

Note the last quotation. It reverses Frege's notion of psychologism: logic is not based on psychology; instead, psychology is based on logic, which is based on mathematics, which is based on diagrammatic reasoning, which is based on the same mechanisms of perception and action as everyday life and language. Mental pictures are memories, reconstructions, or transformations of pictures from the outside world.

For board games like chess, diagrammatic reasoning is the essence of the game. Most chess experts can play a good blindfold game. For them, the board and pieces are the equivalent of Peirce's "drawing or model", which is a helpful, but optional aid to the imagination. To study and compare the thought processes of mathematicians, Hadamard (1945) asked some of the most creative to answer a few questions. Their responses support the observations by Peirce and Polya. Einstein even used Peirce's words *visual* and *muscular*:

> 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.

By relating logic and imagery, diagrammatic reasoning can bridge the gap between abstract symbols and physical reality. Einstein (1944) criticized the "fear of metaphysics" (Angst vor der Metaphysik) as a "malady (Krankheit) of 20th-century empirical philosophy." In reviewing Quine's *Word and Object*, Rescher (1962) was struck by the absence of any discussion of events, processes, actions, and change. C. I. Lewis (1960), whose modal logic and epistemology were strongly influenced by Peirce, criticized the sterility of purely formal definitions:

> It is so easy... to get impressive "results" by replacing the vaguer concepts which convey real meaning by virtue of common usage by pseudo precise concepts which are manipulable by "exact" methods — the trouble being that nobody any longer knows whether anything actual or of practical import is being discussed.

There is a continuum from counting sticks and drawings in the sand to the most elaborate mathematics and computer science. The methods of observation and imagination in mathematics apply to every branch of science, engineering, and commonsense. While he was developing his ideas on existential graphs, Peirce also produced a detailed classification of the sciences. Figure 10 is based on his outline from 1903 (CP 1.180 to 1.202) and his more detailed analyses over the next several years (CP 1.203 to 1.677 and manuscript R602).

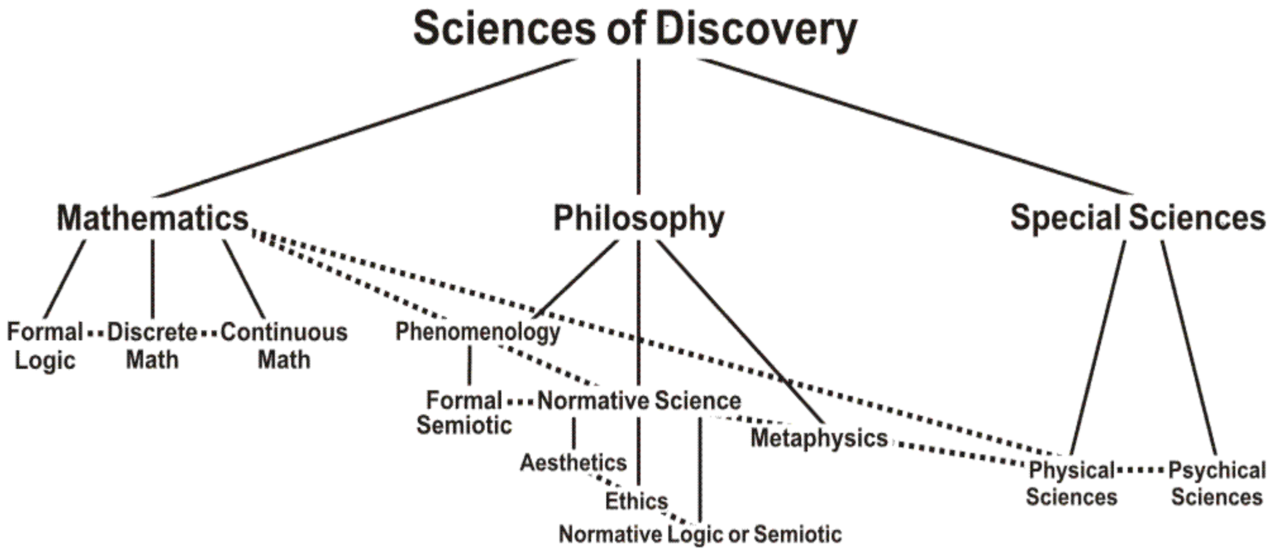

Figure 10: Peirce's classification of the sciences

In Figure 10, the dotted lines show dependencies: sciences on the lower right apply theories from sciences on the upper left. Since dependence is transitive, all sciences apply structures and theories from mathematics to their subject matter. Note that logic appears in two places: formal logic is a branch of pure mathematics; normative logic (how people should reason) is based on mathematics, phenomenology, aesthetics, and ethics. Empirical science depends on metaphysics, but it also has a direct dependence on mathematics. Observation and imagination are the source of new ideas, and any version of mathematics may be used to analyze and develop them.

Peirce's classification is an ontology of all possible theories and their applications to anything that exists or may exist. In his logic, Peirce considered three universes of discourse: the possible, the actual, and the necessitated. The universe of possibilities is the domain of pure mathematics. Every mathematical theory begins with some hypothesis expressed in a diagram or its algebraic representation. The special sciences study the universe of actuality. The hypotheses (diagrams) of mathematics are applied to aspects of actuality in order to make predictions. The hypotheses that make reliable predictions are the laws of science. They are the best known approximations to the laws of nature. The totality of laws of nature is the universe of the necessitated.

Peirce's sequence of phenomenology, normative science, and metaphysics is compatible with Aristotle's topics in *Tôn meta ta physica* (That which is after the physics). Book Alpha begins "All people (*pantes anthropoi*) by nature (*physei*) reach for (*oregontai*) knowledge (*tou eidenai*)." It continues with an analysis of what people and other animals experience. That corresponds to Peirce's definition of phenomenology: the study of "all that is in any way or in any sense present to the mind, quite regardless of whether it corresponds to any real thing or not" (CP 1.284 to 1.310). Reaching for knowledge implies desire (a value judgment) for methods that determine what is true. That corresponds to Peirce's methodeutic, which is a branch of normative logic. Aristotle also discussed the normative values of beauty (*kalos*) and the good (*agathon*) before analyzing the nature of being and the categories of entities (ontology).

Plato and Aristotle disagreed about the role of mathematics. Plato claimed that mathematical forms (Peirce's diagrams) are prior to any physical embodiment, but Aristotle claimed that mathematical entities are not separable from sensible things. Peirce's three universes resolve this conflict: the infinity of all mathematical entities (diagrams) are in the universe of possibilities; those diagrams are potential signs (icons) that may be used to describe anything in the universe of actualities; and theorems about those diagrams form the universe of necessities. Although Aristotle did not discuss signs in his metaphysics, his earlier writings (the *Organon*) covered logic and semiotic, his analysis of *sêmeion*, *symbolon*, and *logos*. For Peirce, mathematical phenomenology leads to the *phenomenological categories* of Firstness, Secondness, and Thirdness, which classify all the signs of perception, language, and the sciences. The dotted lines of Figure 10 show the flow of diagrams and theorems from mathematics to the other sciences:

- **Possibility.** Every mathematical theory develops the implications of some possible pattern (diagram). There is no reason for excluding any possibility or deprecating it as a fantasy.
- **Actuality.** The special sciences observe patterns in the actual universe, find and apply mathematical theories about those patterns, use those theories to make predictions about what may happen, make new observations to test those predictions, revise the theories, and repeat.
- **Necessity.** The propositions entailed by any pattern in any diagrammatic reasoning are necessarily true of any occurrence of that pattern. All theories of science are fallible, but the best are reliable on those domains for which they have been thoroughly tested.

All mathematical theories must be available for applications to the special sciences. All semiotic patterns are necessary for representing natural and artificial languages. In fact, every artificial language in mathematics and computer science is a disciplined application of the syntactic and semantic mechanisms of natural languages. Value judgments are necessary for reasoning about the beliefs, desires, and intentions in any social activity or organization — and the organizations must include colonies of any species from bacteria to humans or even aliens from another galaxy.

When Peirce discussed diagrammatic reasoning, he did not limit his diagrams to existential graphs. He said that a portrait or drawing, by itself, could represent a possibility. But with one or more indexes (names or labels), a diagram could assert a proposition. Any area of an EG may contain a name that represents a proposition, and that name may be replaced by an EG that expresses the same proposition. A generalization of that principle would allow any labeled diagram or image that expresses the same proposition to replace the EG. As an example, Figure 11 shows two if-then scrolls, in which nested EGs have been replaced by EGIF (on the left) or by Euclidean diagrams (on the right). Euclid's naming conventions are used to relate the EGIF labels in the scroll on the left to the letters A, B, and C in the scroll on the right.

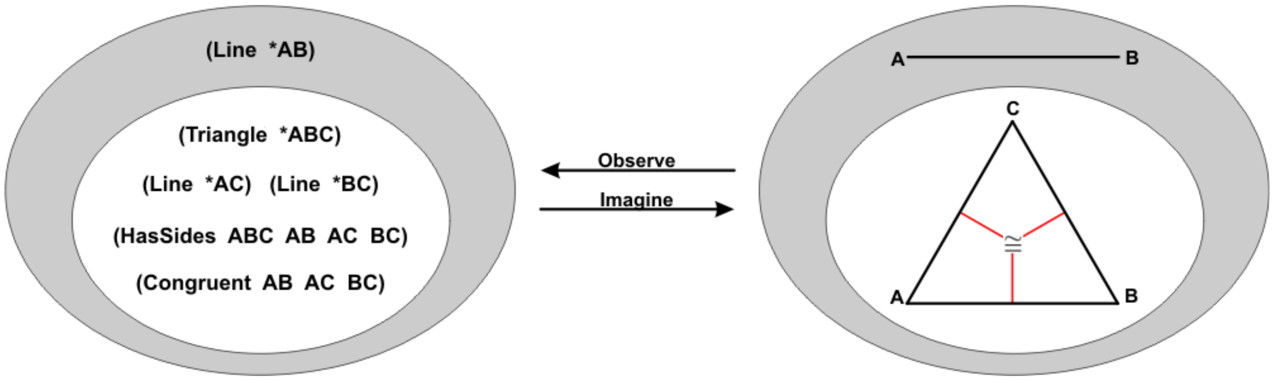

Figure 11: Replacing EGs with EGIF or with Euclidean diagrams

Both scrolls in Figure 11 state Proposition 1 in Euclid's *Elements*, which Heath translated as "On a given finite straight line, to draw an equilateral triangle." The EGIF on the left states that proposition as a conditional: "If there is a line AB, then there is a triangle ABC, which has sides AB, BC, and AC, and the lines AB, BC, and AC are congruent." The Euclidean drawings on the right express the equivalent of the EGIF. The scroll on the right is precise and readable, and it isn't biased toward any natural language. The arrows in the middle of Figure 11 represent the operations that characterize diagrammatic reasoning: *observe* and *imagine*.

Peirce discussed observation and imagination and used them in many examples, but he did not call them rules of inference. For diagrams as precisely drawn as Euclid's, the rules can be stated precisely and added to the rules for EGs. For existential graphs, they define a sound and complete proof procedure for first-order logic and for the version of higher-order logic specified by the model theory for Common Logic (which seems to be equivalent to the version that Peirce had intended). To apply the rules to any notation, the word *diagram* will be used for a statement of a proposition in any form, linear (symbolic) or graphic (mixed imagery and symbols). Before stating the rules, three definitions are required:

- **Generalization.** A diagram G (stated in any form) is a *generalization* of a diagram S if every case (model) for which G is true, S is also true. In particular, the *blank* (an empty diagram or blank area) is always true. Therefore, the blank is a generalization of every diagram, including itself.
- **Specialization.** A diagram S (stated in any form) is a *specialization* of a diagram G if G is a generalization of S.
- **Equivalence.** Two diagrams D1 and D2 are *equivalent* if each one is a generalization and a specialization of the other. In particular, every diagram is equivalent to any copy of itself.

Over the years, Peirce stated his rules of inference in slightly different, but equivalent ways. The version stated below is based on his manuscripts from 1909 to 1911. The rules are grouped in three pairs: one rule (i) inserts a diagram, and the other (e) erases a diagram. The only axiom is the blank, which is always true. The negation of a blank is always false. The rules of inference depend on whether an area is positive (unshaded) or negative (shaded). They do not depend on how deeply nested any area may be.

1. **(i) Insert.** In a negative area, any diagram D (including the blank) may be replaced by any specialization of D.

   (e) Erase. In a positive area, any diagram D may be replaced by any generalization of D (including the blank).
2. **(i) Iterate.** Any diagram D in any area *c* may be iterated (copied) in the same area *c* or into any area nested in *c*. No diagram may be copied directly into itself. But it is permissible to copy a diagram D in the same area *c* and then copy the copy of D into some area nested inside the original D.

   (e) Deiterate. Any diagram D that may have been derived by rule 2i may be erased. Whether or not D had been derived by 2i is irrelevant.
3. **(i) Double negation.** A double negation (nest of two negations with nothing between the inner and outer) may be drawn around any diagram, subdiagram, or set of diagrams in any area. In any area, lines of identity that originate outside the area and pass through it without a connection to anything in the area are not considered part of that area. In EGIF, those lines would not have any bound labels in the area through which they pass.

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

For proofs that involve a mixture of notations, either linear or graphic, rule 1i for insertions in a negative context may be allowed to insert diagrams in any notation, symbolic or graphic. Similarly, rule 1e for erasures in a positive context may be allowed to erase diagrams in any notation. The only options that require special treatment are the rules of iteration (2i) and deiteration (2e) in the same area:

- **(2i) Observation.** If a graphic diagram D occurs in any area *c*, any symbolic generalization or equivalence of D may be inserted in *c*.
- **(2i) Imagination.** If a symbolic diagram D occurs in any area *c*, any graphic generalization or equivalence of D may be inserted in *c*.
- **(2e) Erasure.** If a diagram D occurs in any area *c*, any generalization or equivalence of D that also occurs in *c* may be erased.

If the diagramming conventions are precisely defined, these rules are sound: observation and imagination would add duplicate information in area A; and erasure would delete duplicates. For scenes in nature, photographs, and informal drawings, these rules may be useful, but fallible approximations. For more discussion and examples, see “Peirce, Polya, and Euclid: Integrating logic, heuristics, and geometry” (Sowa 2015) and “Reasoning with diagrams and images” (Sowa 2018).

In conclusion, mental models, diagrammatic reasoning, and existential graphs are three aspects of the same theory of cognition. They cover a continuum in methods of reasoning from a vague guess or impression to the most precise and elaborate theories of science. Although EGs are precise, two-dimensional diagrams are easier to show than to describe. Therefore, EGIF is convenient as a linearization that represents only the semantically significant features. When embedded diagrams are as precisely defined as Euclid's, diagrammatic reasoning with Peirce's rules can be as precise as any method designed for linear notations. But the option of including continuous, informal, or even blurred images, also enables EGs to represent a wide range of heuristic or commonsense methods.

For an overview of the ways Peirce anticipated and often surpassed innovations by his successors, see "Peirce's contributions to the 21st century" Sowa (2006a). For more about combining metalanguage and diagrammatic reasoning to represent modal logic, see "Worlds, models, and descriptions" (Sowa 2006b). For the semantics of natural languages, see "From existential graphs to conceptual graphs" (Sowa 2013) and "Two paradigms are better than one, and multiple paradigms are even better" (Majumdar & Sowa 2009).

5. EGIF Grammar

The Existential Graph Interchange Format 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. Those variations have raised still unresolved questions about possible semantic differences. 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. When analyzing the variations, Peirce scholars may add extensions to EGIF and define alternative semantics.

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 5.3 specify two extensions to first-order 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 make a major improvement in the speed of computation. The option of bound labels as names of relations and functions goes beyond first-order logic, and it supports some features of Peirce's Gamma graphs.

But EGIF can also be used to represent and analyze Peirce's graphs in his various manuscripts and publications. His original Alpha and Beta graphs, which represent propositional and first-order logic, are consistent with the default semantics. But with his Gamma graphs, he experimented with various extensions for higher-order logic, modal logic, and metalanguage.

Section 5.1 presents the lexical rules for the kinds of character strings used in EGIF. These rules include some useful CL features such as integers and character strings. Section 5.2 presents the phrase-structure rules. Section 5.3 states context-sensitive constraints and gives some examples.

5.1 Lexical Rules

All EGIF grammar rules are stated as Extended Backus-Naur Form (EBNF) rules, as defined by the ISO/IEC standard 14977. The lexical rules specify names and identifiers, which exclude white space except as noted. The phrase-structure rules in Section 5.2 specify larger combinations that may have zero or more characters of white space between constituents. 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 lexical categories are defined formally in Section A.2 of ISO/IEC 24707, which is the normative specification for Common Logic and its standard dialects. 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**, **5**, **7**, **8**, **9**.
- An *integer* is an optional sign (**+** or **−**) followed by a sequence of one or more digits.
- A *string* is any sequence of Unicode characters preceded and followed by a single quote **'**. Any single

quote internal to a string shall be represented by the string **\'**. Any backslash internal to an enclosed name shall be represented by the string \\.

- 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.

5.2 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, 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 about syntax, the EBNF rule shall be normative. For any derivation, the rule that defines EG shall be the starting rule. The next paragraph defines 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 separated 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 in boldface, 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. A coreference node that contains two or more names states that all its names have the same denotation. For example, **[x 39]** is equivalent to the equality **x=39**, which in CLIF is **(= x 39)**.

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* 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, an integer, or a string. Note: A defining label, bound label, identifier or enclosed name may denote anything in the universe of discourse. But an integer shall denote the number determined by its string of digits, and a string shall denote the character string enclosed within its outer quotes.

   Name = Defining Label | Bound Label | identifier | enclosed name | integer | string;

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;

To support the IKL extension to Common Logic, Section 3 suggested a grammar rule for the metalevel operator **that.** The IKL syntax may be added as an option to Rule 6:

Function = '(', Type Label, {Name}, '|', Name, ')' | '(' 'that' EG '|' Name ')';

5.3 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 scope is shown by connected lines, 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*.

- In EG diagrams, a function of N input pegs may be represented as a relation with N+1 pegs, but a line of identity connected to its output peg shall have an arrowhead that points away from the name of the function.
- If a line of identity connected to the output peg of one function *f* is connected to a peg of a function or relation *g*, the output peg of *f* shall have an arrowhead pointing to *g*. Note that the EGIF **(f *x | x)** would represent a function whose output peg would be connected to a line on its left that loops back and points to its right side.
- If a line of identity with an arrowhead pointing away from a function forms a ligature with another line, the point of contact shall be covered with a heavy dot. A single heavy dot may have more than one arrowhead pointing to it.
- **Rule of unification.**. If *f* is a function with N inputs and if for every *i* from 1 to N, the *i*-th input of one instance of *f* is coreferent with the *i*-th input of another instance of *f,* then the lines for both output pegs would form a ligature with both pointing to the same heavy dot.

A ligature of two or more output lines in an EG corresponds to inserting a coreference node in EGIF or an equality in other notations for logic. The unification rule, which can be proved as a derived rule of inference from Peirce's rules, is essential for speed 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,* which is important for many theorem provers.

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. The following function, for example, would represent a unique person who called himself Paracelsus:

("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)**.

In Section 5.2, EBNF Rule 12 allows any name except a defining label to be used as a type label. Therefore, a bound label may be used as the name of a function or relation. As an example, consider the sentence "There is family relation between any two members of the same family." To translate that sentence to EGIF, restate it as an if-then statement with explicit variables F, R, *x*, and *y*: "If there is a family F with members *x* and *y*, then there is a family relation R that is true of *x* and *y*." In EGIF, the defining label ***R** asserts existence, and the bound label **R** is used in **(R x y)**:

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

References

Chen, Weidong, Michael Kifer, & David S. Warren (1993) Hilog: A Foundation for Higher-Order Logic Programming, *Journal of Logic Programming* **15:3**, pp. 187-230.

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

Dau, Frithjof (2006) Some notes on proofs with Alpha graphs, in H. Schärfe, P. Hitzler, & P. Øhrstrøm, eds., *Conceptual Structures: Inspiration and Application*, LNAI 4068, Springer, Berlin, pp. 172-188.

Dehaene, Stanislas (2014) *Consciousness and the Brain*, New York: Viking.

Delugach, Harry S. (2014) Implementation and Visualization of Conceptual Graphs in Charger, *International J. of Conceptual Structures and Smart Applications* **2:2**, 1-19.

Einstein, Albert (1944) Remarks on Bertrand Russell's Theory of Knowledge, in P. A. Schilpp, ed., *The Philosophy of Bertrand Russell*, Library of Living Philosophers.

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

Genesereth, Michael R., & Richard Fikes, eds. (1992) *Knowledge Interchange Format, Version 3.0 Reference Manual*, TR Logic-92-1, Computer Science Department, Stanford University.

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

Hayes, Patrick (2005) Translating semantic web languages into Common Logic, http://www.ihmc.us/users/phayes/CL/SW2SCL.html

Hayes, Patrick, & Chris Menzel (2001) A semantics for the Knowledge Interchange Format, *Proc. IJCAI 2001 Workshop on the IEEE Standard Upper Ontology*, Seattle.

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

ISO/IEC (1996). *Extended BNF*, (IS 14977). Geneva: International Organisation for Standardisation. http://standards.iso.org/ittf/PubliclyAvailableStandards/s026153_ISO_IEC_14977_1996(E).zip

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

Johnson-Laird, Philip N. (2002) Peirce, logic diagrams, and the elementary operations of reasoning, *Thinking and Reasoning* **8:2**, 69-95.

Lewis, Clarence Irving (1960) Letter to Hao Wang, quoted on page 116 of H. Wang (1986) *Beyond Analytic Philosophy: Doing Justice to What We Know*, MIT Press, Cambridge, MA.

Majumdar, Arun K., & John F. Sowa (2009) Two paradigms are better than one and multiple paradigms are even better, in S. Rudolph, F. Dau, and S.O. Kuznetsov, eds., *Proceedings of ICCS'09*, LNAI 5662, Springer, pp. 32-47. http://jfsowa.com/pubs/paradigm.pdf

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

Peirce, Charles Sanders (1880) On the algebra of logic, *American Journal of Mathematics* **3**, 15-57.

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

Peirce, Charles Sanders (1898). *Reasoning and the Logic of Things*, (The Cambridge Conferences Lectures of 1898), K. L. Ketner (ed) (1992). Cambridge, MA: Harvard University Press.

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. For Peirce's classification of the sciences (CP 1.176 to 1.677), see https://www.textlog.de/4258.html

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.

Peirce, Charles Sanders (NEM) *The New Elements of Mathematics*, ed. by Carolyn Eisele, 4 vols., The Hague: Mouton, 1976.

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

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

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

Quine, Willard Van Orman (1954) Reduction to a dyadic predicate, *J. Symbolic Logic* **19**, reprinted in W. V. Quine, *Selected Logic Papers*, Enlarged Edition, Harvard University Press, Cambridge, MA, 1995, pp. 224-226.

Rescher, Nicholas (1962) The revolt against process, *Journal of Philosophy*, vol. 59, pp. 410-417.

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

Sowa, John F. (2006a) Peirce's contributions to the 21st Century, in H. Schärfe, P. Hitzler, & P. Øhrstrøm, eds., Conceptual Structures: Inspiration and Application, LNAI 4068, Berlin: Springer, pp. 54-69. http://jfsowa.com/pubs/csp21st.pdf

Sowa, John F. (2006b) Worlds, models, and descriptions, *Studia Logica*, Special Issue *Ways of Worlds II*, **84:2**, 323-360. http://jfsowa.com/pubs/worlds.pdf

Sowa, John F. (2008) Conceptual graphs, in F. van Harmelen, V. Lifschitz, and B. Porter, eds., *Handbook of Knowledge Representation*, Amsterdam: Elsevier, pp. 213-237. http://jfsowa.com/cg/cg_hbook.pdf

Sowa, John F. (2011) Peirce's tutorial on existential graphs, *Semiotica* **186:1-4**, 345-394. http://jfsowa.com/pubs/egtut.pdf

Sowa, John F. (2013) From existential graphs to conceptual graphs, *International Journal of Conceptual Structures* **1:1**, 39-72. http://jfsowa.com/pubs/eg2cg.pdf

Sowa, John F. (2015) Peirce, Polya, and Euclid: Integrating logic, heuristics, and geometry, talk presented at the conference of the American Philosophical Association in Vancouver, Canada. Revised and extended slides (2018), http://jfsowa.com/talks/ppe.pdf

Sowa, John F. (2017) An introduction to existential graphs, a collection of slides from various lectures. http://jfsowa.com/talks/egintro.pdf

Sowa, John F. (2018) Reasoning with diagrams and images, *Journal of Applied Logics* **5:5**, 987-1059. http://www.collegepublications.co.uk/downloads/ifcolog00025.pdf

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

Stjernfelt, Frederik (2007) *Diagrammatology: An Investigation on the Borderlines of Phenomenology, Ontology, and Semiotics*, Berlin: Springer.