Native diagrammatic soundness and completeness proofs for Peirce's Existential Graphs (Alpha)

Gianluca Caterina† Rocco Gangle† Fernando Tohmé‡
Center for Diagrammatic and Computational Philosophy
†Endicott College, Beverly MA, USA
‡Universidad Nacional del Sur, Bahía Blanca, Argentina

September 20, 2022

Abstract

Peirce's diagrammatic system of Existential Graphs (EG_α) is a logical proof system corresponding to the Propositional Calculus (PL). Most known proofs of soundness and completeness for EG_α depend upon a translation of Peirce's diagrammatic syntax into that of a suitable Frege-style system. In this paper, drawing upon standard results but using the native diagrammatic notational framework of the graphs, we present a purely syntactic proof of soundness, and hence consistency, for EG_α, along with two separate completeness proofs that are constructive in the sense that we provide an algorithm in each case to construct an EG_α formal proof starting from the empty Sheet of Assertion, given any expression that is in fact a tautology according to the standard semantics of the system.

1 Introduction

The Existential Graphs (EG) are a family of diagrammatic proof systems developed by C.S. Peirce over the course of roughly two decades at the beginning of the 20th century. The Alpha (EG_α), Beta (EG_β) and Gamma (EG_γ) versions of the graphs correspond, respectively, to the Propositional Calculus (PL), First Order Logic (FOL) and various systems of Modal and Higher-Order Logics.

A systematic analysis of EG only began in the 1960s, with the work of D. Roberts [10] and J. Zeman [13], who gave the first formal proofs of soundness and completeness for EG_α and EG_β [13]. More recently, scholars have begun to investigate the formal structure of the graphs from a variety of different perspectives, including proof theory [1, 2], the logic of assertions [5], computational complexity [6], and category theory [7, 4]. A common critique of EG, which one may speculate is the main reason the graphs have often been considered by the mainstream community of logicians as little more than a bizarre curiosity, is that

they are simply a translation of standard logic into a (seemingly unwieldy) diagrammatic notation. To support that view, it is often pointed out that proofs of the main formal results in EG, in particular soundness and completeness, follow, in fact, methods that are a straightforward reproduction of those commonly used in standard logic textbooks, such as translations into Frege-style systems or, alternatively, Henkin-style methods to prove completeness.

In this paper we pursue the strategy of proving the soundness and completeness of EG_α in a "native" diagrammatic mode. That is to say, we formulate the respective formal proofs according to Peirce's own notation in such a way that the diagrammatic nature of the EG_α syntax functions both rigorously and perspicuously in helping the reader to follow and to understand the reasoning those proofs involve.

From a philosophical point of view, these natively diagrammatic soundness and completeness proofs shed some light on how formal notation can both spur intuition and guide inquiry. Peirce's EG possesses a number of notational virtues in this regard, perhaps most obviously in the way that the commutativity and associativity of conjunction are captured in a straightforward and natural way by treating any continuous (that is, uncut) area on the sheet of assertion as a unified space the components of which may be permuted at will without any change of logical (semantic) value. Such topological transformations are themselves susceptible to mathematical formalization, of course, but they are also easily grasped by the intuition that subgraphs may arbitrarily "float about" on any given area so long as they do not cross any cuts. Ordinary and familiar habits of imagination (moving figures smoothly through two-dimensional space) here become tools facilitating rigorous logical reasoning. In general, the characterization of all of the EG transformation rules in terms of the writing (including iteration) and erasure (including de-iteration) of subgraphs on or from selected areas of whatever given graph provides a translation of the typical features of visual Gestalts into a basis for logical syntactic manipulations.

In what follows, we show that we can obtain a purely syntactical version of soundness, and henceforth consistency, for EG_α along with two constructive proofs of the system's completeness. The completeness proofs are constructive in the sense that we provide an algorithm in each case such that, given any logical graph T that is a tautology according to the standard semantics for the system, the algorithm will construct a formal proof beginning from the blank Sheet of Assertion and resulting in T after finitely many applications of Peirce's formal diagrammatic rules. The first of these proofs makes use of the reduction of expressions to Conjunctive Normal Form, whereas the second adapts Kalmar's [8] well-known strategy for proving completeness, constructively, for PL within a Hilbert-style framework.

The main idea behind our result, and what provides the basis for our original contribution, is that some unique features of Peirce's diagrammatic syntax and deduction rules lend themselves to exploitation for streamlining certain proof-theoretical methods as compared to other proof systems. In particular, the fact that, syntactically speaking, the only logical building block of the language of EG_α – a closed curve in the plane – can be interpreted in

a twofold manner as both a generalized negation operator (NAND) as well as the constant symbol standing for the semantic truth value *false* makes the relations between syntax and semantics especially perspicuous in some contexts of reasoning about (and constructing) proofs. Equally important is the fact that the logic of EG_α includes a distinctive *iteration* rule, which allows for *deep inferences* to be generated and reasoned about directly. These two features are brought together below in section 3, where the distinctive point of view of "cuts-only" EG_α graphs is developed for which semantic evaluation and proof construction are nicely correlated. This approach is then extended to EG_α as a whole in the subsequent proofs for soundness and completeness.

2 Introducing Alpha Graphs

We first present a schematic introduction to the system EG_α for those who might be unfamiliar with the graphs. Reasons of space dictate a cursory and informal presentation. For a more rigorous mathematical presentation, the reader is referred to [4]; detailed but more accessible treatments may be found in [10] and [11].

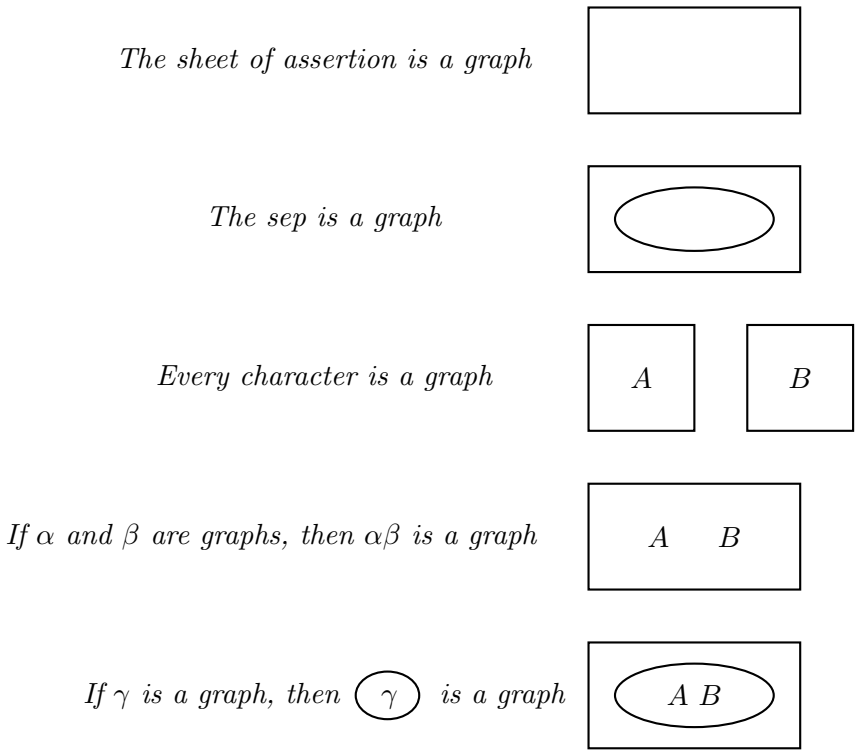
Every EG_α graph is understood by Peirce to represent a propositional assertion according to a regimented diagrammatic syntax. The syntax consists of three elements: the Sheet of Assertion, characters and seps (or cuts). An EG_α graph is composed of characters and seps "scribed" on the Sheet of Assertion.

1. The blank sheet, called the Sheet of Assertion (SA) is both the site on which graphs are *scribed* and is itself a graph.

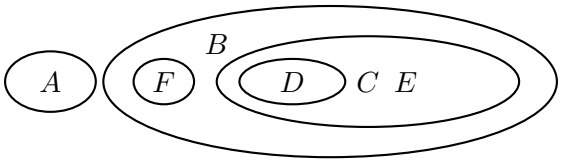
2. A *character* is any easily reproducible symbol, typically letters of the Roman alphabet, scribed on some part of the SA.

3. Characters may be enclosed, along with a local area surrounding them (which may or may not include other characters and cuts), by a closed curve called a *sep* (or *cut*). It is convenient to construct these generally as ovals or circles. These seps may not intersect characters, nor may they intersect one another. They may, however, be nested with any number of characters and seps scribed in the areas or enclosures they distinguish.

With these three constituents, the class of EG_α graphs may be characterized by way of the following recursive definition:

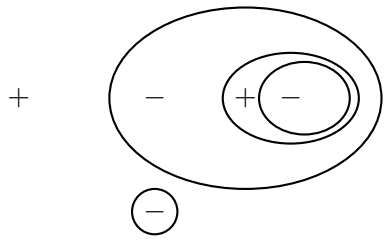


Informally speaking, an element of EG_α is simply a collection of non-intersecting, possibly nested, ellipses, and symbols scattered on the plane. Also, when it is not necessary, we omit to draw rectangles around the graph to signify the sheet of assertion. The example below should clarify this:



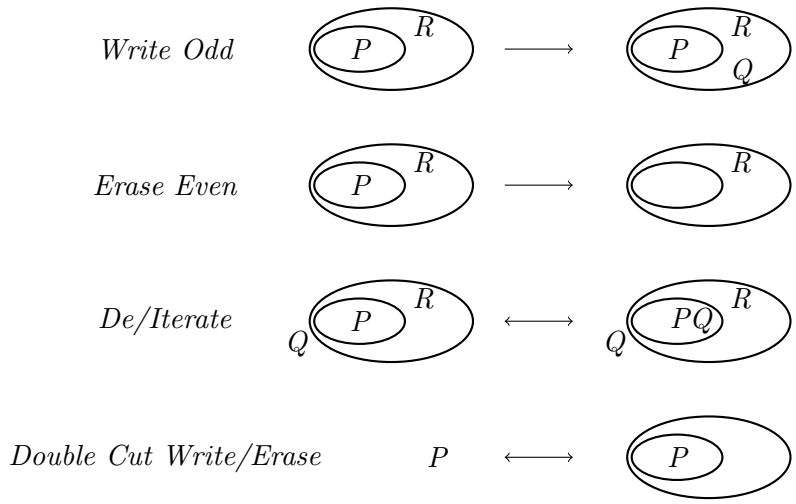
Let us first notice that the EG_α graphs constructed in this way separate regions in the sheet of assertion into evenly (+) and oddly (−) enclosed areas, as the following picture shows.

Figure 1



Peirce's transformation rules for EG_α are summarized below. A full treatment of these rules may be found in [10]. As a helpful reminder to the reader, instances of all of the rules are first provided in an informal, iconic way in Figure 2.

Figure 2



The rules pictured in Figure 2 should be understood as *schemas*, where variable letters can be substituted by arbitrary (sub)graphs. In other words, according to these rules, we can transform a graph G into another graph G' by:

1. (WO) Inscribing any graph on an odd area;
2. (EE) Erasing any graph from an even area;
3. (IT^+) Copying and pasting a subgraph[1] H on any area that is inside the area that contains H and is not part of H itself;
4. (IT^-) Deleting any subgraph that can be obtained using IT^+;
5. Inserting (DC^+) or erasing (DC^-) double cuts around any subgraph of G.

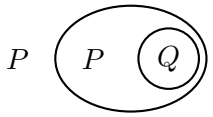
[1]Informally speaking, a *subgraph* of G is any collection of parts of G that can be inscribed within a sep which does not intersect any other sep.

3 Cuts-only graphs

In this section we focus on the fragment of EG_α consisting of the collection of all *cuts-only* graphs, that is, those graphs with no occurrence of variables (characters). We call this fragment EG^*_α. Below, we show that this fragment is sound and complete with respect to Peirce's deduction rules. The idea is then to bootstrap these relatively straightforward proofs of soundness and completeness for the fragment to the entire system EG_α. In fact the dynamics of the fragment EG^*_α under the deduction rules can be shown in a certain sense to underlie the system as a whole.

In order to clearly understand this strong connection between EG_α and the fragment EG^*_α, it is necessary to notice that, given a graph G in EG_α containing some variable letters (*types*), perhaps repeated at various locations of the graph (*tokens*), it is reasonable to conceive of the possible replacements of the variables with either the empty sheet or the empty cut as corresponding to semantic valuations (*interpretations*) of those variables. The empty sheet is the "inert" element, interpreted as the truth value **true**, whereas the empty cut is interpreted as the truth value **false**. We then can associate to any $G \in EG_\alpha$ its *blow-up* $\hat{G} \subset EG^*_\alpha$ which is the collection of cuts-only graphs obtained by all possible combinations of the substitutions just outlined that respect the variable types.

For instance, the blow-up of the following graph G

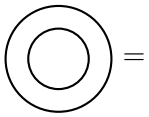


is the collection of the four following graphs:

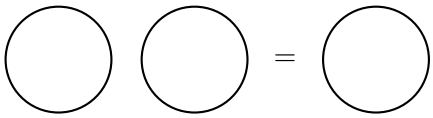
Each of the above graphs correspond to one of the possible interpretations, where we assume that **true** corresponds to the empty sheet and **false** corresponds to the empty cut, as highlighted in each case by the thicker cuts.

Given a graph G and an interpretation, the truth value of G can be obtained via Spencer-Brown's primary algebra reduction procedure, which shows how any cuts-only graph G^* can be reduced to either the empty sheet or the empty cut by a finite number of applications of the following rules. See [12] for more details.

1. *Law of crossing*



2. *Law of calling*



More precisely, the following result holds:

Lemma 1 *Every cuts-only graph G^* reduces to either the empty cut or the empty sheet, and the result does not depend on the order in which the steps of the reduction procedure are performed.*

Proof: See [9].

At this point the main feature of EG^*_α starts to emerge: we have a single syntactic element (the empty cut), which combines the role of parenthesis and negation (NAND) while also coordinating semantically with the truth value **false**, which we may denote in what follows with the more conventional symbol $\bot$. This observation simply unfolds the fact that interior of the empty cut can be interpreted as the empty sheet, and therefore the empty cut, along with its interior, can be interpreted as the truth value $\bot$. For the same reason, being allowed to write on the empty sheet, and hence being allowed to write inside the empty cut, can be interpreted as the negation of the syntactical content being written. We believe that this property, which employs the duality operator/operand naturally present in the topological structure of the diagrammatic notation, is both desirable and worth of further study, especially in relation to frameworks based on type theory and, more broadly, on categorical logic.

In the next section we will see that, when working in EG^*_α, a proof of the empty cut from the blank Sheet of Assertion cannot be generated, a sort of syntactical justification that the empty cut is in fact, naturally interpreted as $\bot$. The reader is referred to [3] for a thorough, enlightening discussion, along with a detailed historical account, about the process that lead Peirce to his unifying notation for logical constants.

There is another important fact that we would like to highlight before moving further. Consider a cuts-only graph G^* and let us denote by $V(G^*)$ the result of the reduction outlined above. Notice that both the law of calling and the law of crossing are two instances of the transformation rules for EG^*_α: the law of crossing is just the double cut rule, whereas the law of calling is an instance of the iteration/deiteration rule. Both laws are reversible, and therefore, if $V(G^*)$ reduces to the empty sheet, by inverting the steps of the procedure we actually recover, constructively, a *proof* of G^*, which is simply a finite sequence of transformations from the empty sheet to G^*. In this sense, the concept of truth for elements of EG_α resolves naturally into that of *provability* for collection of elements of EG^*_α:

$$V(G^*) = \top \Rightarrow \ \top \vdash G^*$$

This intuition will be formalized in the next section.

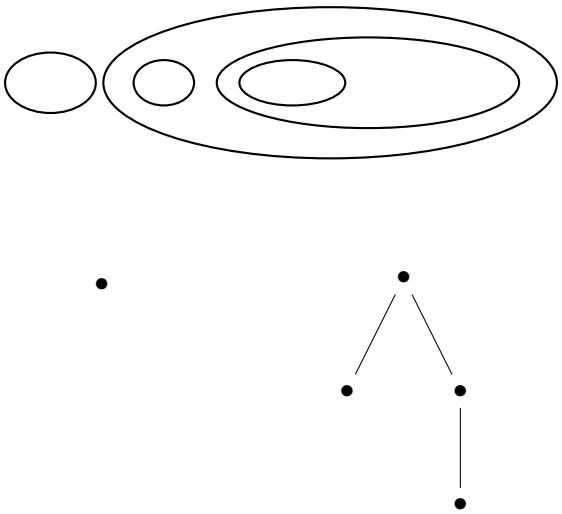
3.1 Cuts-only graphs as forests

In the paper [7] a categorical framework is presented for EG_α and EG^*_α as presheaves over suitable base categories. A useful representation for EG^*_α is given by the notion of *forests*:

Lemma 2 *Any cuts-only graph can be represented as a forest, i.e. as a finite union of trees.*

Proof: See [7].

As an example, here below the cuts-only graph on the top is represented by a forest on the bottom.



In analogy with Spencer-Brown's reduction procedure described above, [7] presents a procedure to reduce any forest to either a single root or the empty set, via a finite sequence of prunings. The procedure is based on an algorithm that labels all the nodes of the forest correspondent to a cuts-only graph G^* with 0s and 1s. Details are given in the next subsection.

3.2 Reduction algorithm for forests

Given a forest representation of a cuts-only graph $G^* \in EG^*_\alpha$, we define a *child* of a node n in such a forest as a node that is immediately below n. A *terminal node* is a node that has no children. A *pre-terminal node* is a node with at least one child that is terminal.

Given a cuts-only graph G^*, we reduce it to either the empty sheet or the empty cut by applying the following iterative pruning procedure:

Procedure 1

(*i*). *Consider the forest representation F of the graph G^*.*

(*ii*). *Take any pre-terminal node n in F.*

(*iii*). *Eliminate the node n and all nodes below it, yielding a new forest F'.*

(*iv*). *Replace F with F' and repeat (ii) and (ii) until the forest F' is either empty or consists of single nodes only (nodes of at most depth 1).*

(*v*). *If the forest F' is empty, replace it with the empty Sheet of Assertion. If it consists of single nodes only, replace it with a single empty cut.*

Since each step in the sequence represents the "pruning" of at least two nodes from the previous step and every element of G^* is composed of a finite number of nodes, successive applications of the pruning procedure will eventually reduce a forest to either the empty sheet or a collection of empty cuts at level one.

Perhaps less obvious is the fact that any such sequence for a given graph G^*, necessarily concludes in the same final step. More precisely, notice that every time we apply steps (ii) and (iii), we need to make a choice among all the pre-terminal nodes that exist at that stage, and we need to show that any variant sequence of such choices leads nonetheless invariably to the same final result.

Theorem 1 *Given a finite forest F, the algorithm outlined in Procedure 1 will output either the empty sheet or the empty cut, independently of the choices made in step (ii).*

Proof: We first label all the nodes of F either 0 or 1 according to the following rule:

- All nodes are labeled 1 if they have at least one child labeled 0 and they are labeled 0 otherwise.

It should be clear that this labeling is well-defined and uniquely determined for any finite forest. We note first that prior to applying the pruning procedure every terminal node of the forest is necessarily labeled 0 since these, by definition, have no children labeled 0. We then note that repeated applications of steps (ii) and (iii) in the pruning procedure above will always select and delete a node labeled 1. It is impossible for a node labeled 0 to have any children also labeled 0 according to the labeling rule. Since only a pre-terminal node can be chosen and deleted in steps (ii) and (iii), no such node can ever be labeled 0. Thus every node selected in step (ii) must be labeled 1. Eventually, *all* nodes labeled 1 must be deleted. This will result in a final forest that has at most depth 1. □

Suppose G^* has a single tree in its forest representation. Let us notice that if the root of the tree ends up labeled with 1, that means that the tree reduces to the empty tree,

and therefore the correspondent graph reduces to the empty sheet, whereas if the root is labeled with 0, then the tree reduces to the single node and therefore the correspondent graph reduces to the empty cut. We can then define again a *valuation* V of a cuts-only graph G^* such that $V(G^*) = 1$ if G^* its root is labeled with 1 and $V(G^*) = 0$ if its root is labeled with 0.

In the more general case when $G*$, in its forest representation, is composed by a finite number of trees greater than one, we then assume that $V(G^*) = 1$ if all the roots are labeled 1 and 0 otherwise, in accordance with our interpretation of 1 as $\top$ and 0 as $\bot$.

3.3 Soundness and Completeness

For the cuts-only fragment EG^*_α we can define syntactical and semantical entailment in a standard way. Notice that, since every cuts-only graph $G^* \in EG^*_\alpha$ can be thought as an interpreted alpha-graph $G \in EG_\alpha$, where true tokens are replaced by empty sheets and false ones by empty cuts, defining semantic entailment within EG^*_α does not involve quantifying over possible interpretations.

Definition 1 *Let us consider two cuts-only graphs G^* and H^*. We say that G^** entails syntactically *H^** in *n* steps *if and only if there is a sequence of graphs*

$$G^* = G^*_0, G^*_1, \ldots G^*_n = H^*$$

such that G^*_{i+1} is derivable from G^*_i using one of the transformation rules, for all $i = 0, \ldots, n-1$. We use the notation

$$G^* \vdash^n H^*$$

to indicate that G^* entails syntactically *graph H^* in n steps (in what follows, we may drop the subscript n when not needed).*

Definition 2 *Let us consider two cuts-only graphs G^* and H^*. We say that G^** entails semantically *H^* if and only if*

$$V(G^*) = 1 \Rightarrow V(H^*) = 1$$

We use the notation

$$G^* \vDash H^*$$

to indicate that G^* entails semantically *graph H^*.*

Not surprisingly, given the lack of quantification over the interpretations, completeness trivially holds:

Theorem 2 *For any two cuts-only graphs G^* and H^*, semantical entailment implies syntactical entailment. That is,*

$$G^* \vDash H^* \Rightarrow G^* \vdash H^*$$

Proof: Since we consider only cuts-only graphs, only Spencer-Brown's rules, the laws of *calling* and *crossing* (see the Appendix for their definition), are used. These rules are reversible. By definition of semantical entailment we have that our hypothesis entails that

$$G^* \vdash$$

and

$$\vdash H^*$$

Therefore

$$G^* \vdash H^*.$$

□

We now establish soundness for cuts-only graphs.

Theorem 3

For any two cuts-only graphs G^* and H^*, syntactical entailment implies semantical entailment, that is

$$G^* \vdash H^* \Rightarrow G^* \vDash H^*$$

Before providing the proof, we note that as a corollary of Theorem 3 we will immediately receive as a result the consistency of EG^*_α:

Corollary 1 *There is no finite n such that*

$$\vdash^n \bigcirc$$

where $\bigcirc$ denotes the empty cut.

Proof: According to Theorem 3, if the empty sheet entails syntactically the empty cut, then the empty cut would evaluate to $\top$, which is a contradiction. □

3.4 Proof of Theorem 3

We now show that each of Peirce's logical rules is truth-preserving, and we will do so using use the forest model for cuts-only graphs. Without losing generality, we can restrict to the case when G^*, as a forest, is formed by a single tree. That amounts to showing that

$$\text{if} \quad G^* \vdash_r H^* \quad \text{then} \quad V(G^*) = 1 \ \Rightarrow V(H^*) = 1$$

where $G^* \vdash_r H^*$ means that H^* is the graph obtained by a single application of a rule r to G^*, where r ranges over the set of Peirce's logical rules: DC^+, DC^-, WO, EE, IT^+, IT^-.

Let us start with some preliminary definitions. Modeling G^* as a tree, we first define a total "vertical order" amongst the nodes, which intuitively we can think of their depths in a tree using the natural numbers, so that the root has depth 1, its children have depth 2 and so on. Likewise we can also define a total "horizontal" order amongst nodes of a given depth, which we agree goes from the left to the right.

These two orders can be combined to identify uniquely each node in a tree by denoting with $n_{i,j}$ the j^th node at depth i. Finally, we define $l(n_{i,j})$ to be the value (0 or 1) assigned to node $n_{i,j}$ by the labeling algorithm.

At this point we can establish a result that highlights a combinatorial property common to all cuts-only graphs that evaluate to true. That is, in the forest representation, to all the forests that evaluate to 1.

Lemma 3 *Let G^* be a cuts-only graph such that $V(G^*) = 1$. Then, there exists a canonical alternating sequence of nodes of minimal length from the root to a terminal node n_{2N,j_N}, $Alt(G^*) = n_{2N,j_N}, n_{2N-1,j_{N-1}}, \dots n_{1,j_1}$ such that*

$$l(n_{2N,j_N}) = 0, l(n_{2N-1,j_{N-1}}) = 1, \dots l(n_{2,j_2}) = 0, l(n_{1,j_1}) = 1.$$

Proof: Let us first notice that, by hypothesis, the root must be labeled with 1, that is, $l(n_{1,1}) = 1$. Hence, at least one of the children of the root has to be labeled with 0. Let us define $\mathbf{N}_2 = \{n_{2,i} : l(n_{2,i}) = 0\}$. Then let us denote by $\mathbf{N}_2^t \subseteq \mathbf{N}_2$ the subset of $\mathbf{N}_2$ which consists of only termial nodes. If $\mathbf{N}_2^t \neq \emptyset$, we pick $n_{2,\bar{i}} \in \mathbf{N}_2^t$ such that $\bar{i} = \min\{i : n_{2,i} \in \mathbf{N}_2^t\}$ and we are done.

If $\mathbf{N}_2^t = \emptyset$, let us consider the first node in $\mathbf{N}_2$. Then we have a natural number, say j_2 such that $l(n_{2,j_2}) = 0$ and n_{2,j_2} is not terminal. Hence, all the children of n_{2,j_2} must be labeled with 1. Let us consider the first child of n_{2,j_2} in the order of the nodes of depth 3,n_{3,j_3}. Then, $l(n_{3,j_3}) = 1$. By definition, n_{3,j_3} cannot be terminal and thus we can repeat the argument until we reach an even depth $2N$ such that there exists a node labeled 0 that is terminal.

Let us now define $Alt_2(G^*)$ the alternating sequences

$$n_{2k,j_{2k}}, n_{2k-1,j_{k-1}}, \dots n_{1,j_1}$$

such that

$$l(n_{2k,j_{2k}}) = 0, l(n_{2k-1,j_{k-1}}) = 1, \dots l(n_{2,j_2}) = 0, l(n_{1,j_1}) = 1$$

for some $k \in \mathbb{N}$.
The previous argument that every forest takes a definite numbering scheme of its nodes shows that we can find a "canonical" such sequence $s \in Alt_2(G^*)$ of minimal length that is furthest leftmost, which we denote by, abusing a bit the notation, $Alt(G^*)$ □

The previous lemma tells us that every cuts-only graph that evaluates to 1 is such that there is at least one of its even-depth nodes labeled 0:

Corollary 2 *Let G^* be a cuts-only graph such that $V(G^*) = 1$. Then, there exists $i \in \{1, 2, \ldots N\}$ and $j_{2i} \in \mathbb{N}$ such that $l(n_{2i,j_{2i}}) = 0$.*

The core of the soundness proof consists in showing that, for every cuts-only graph G^* such that $V(G^*) = 1$, if $G^* \vdash_r H^*$ there exists an $Alt(H^*)$ for each of Peirce's logical rules r. Since the existence of such an $Alt(H^*)$ guarantees that $V(H^*) = 1$, the logical soundness of the rule for which this relation is established is thereby demonstrated in each case. We give a proof of this claim first for the *non reversible* rules EE and WO. The remaining rules, namely $\{DC^+, DC^-, IT^+, IT^-\}$ are *reversible* in the sense that if $G^* \vdash_s H^*$, where $s \in S$ is a reversible rule, then $G^* \vdash_{s^-} H^*$, where s^- is the rule inverse to s. For these rules, as should be expected, soundness derives from the stronger claim, which we will prove, that, roughly speaking, the labeling of the *significant* part of the tree stays unchanged under those rules, not just the topmost value. We will clarify this in the next subsections.

3.4.1 Erase Even

In order to show that truth is invariant under the rule Erase Even (EE), it is enough to prove that the canonical sequence either remains invariant under EE or is simply truncated from below. More precisely, we have the following:

Theorem 4 *Let G^* be a cuts-only graph such that $V(G^*) = 0$, let $Alt(G^*)$ be its associated canonical sequence and assume that*

$$G^* \vdash_{EE} H^*$$

We then have that either
(A) $Alt(H^*) = Alt(G^*)$ or
(B) $Alt(H^*)$ is an initial segment of $Alt(G^*)$.

Let us first define, for any node n in the tree representation of any cuts-only graph G^*, the *full subtree* rooted at n, say T_n as a subset of all possible paths from the node n to some terminal nodes. By a *path* from n of length N here we mean a sequence of nodes $n = n_0, n_2, \ldots, n_{N-1}$ such that, for any $i = 1, N-1$ we have that n_i is the child of n_{i-1}.. A *subtree* rooted at n is just a subset of T_n, therefore, in general, not a full subtree.

Erasing Even corresponds to the pruning of one or more subtrees rooted at odd depths (clearly, the stick above the pruned roots also gets pruned). Here is an example:

The key of the proof lies upon understanding how the pruning of even-depth subtrees affects the labeling of the pruned tree globally. Interestingly, a sort of "locality" principle holds. We have indeed the following:

Lemma 4 *Let G^* be a cuts-only graph such that $V(G^*) = 1$, and let us assume that*

$$G^* \vdash_{EE} H^*$$

The only possible differences in the labeling of H^* with respect to the labeling of G^* is that nodes labeled with 1 at even depth in G^* can switch to 0 in the labeling of H^*, and nodes labeled with 0 at odd depth in G^* can switch to 1 in the labeling of H^*.

Proof: This can be seen by looking at the possible changes in the labeling determined by Erasing Even and at the propagation of such changes. Indeed, suppose we prune a full subtree $T_{n_{2N+1,j}}$ rooted at $n_{2N+1,j}$ (without losing generality, we can restrict to the pruning of one single full subtree). Suppose that the father of such a node, say $n_{2N,i}$, is such that $l(n_{2N,i}) = 0$. In that case, all of its children are labeled 1, and therefore erasing some of those children will not change its value. On the other hand, if $l(n_{2N,i}) = 1$, it could be the case that the pruning makes all of its children labeled with 1 (if we prune all the 0s) and therefore its label will switch to 0.

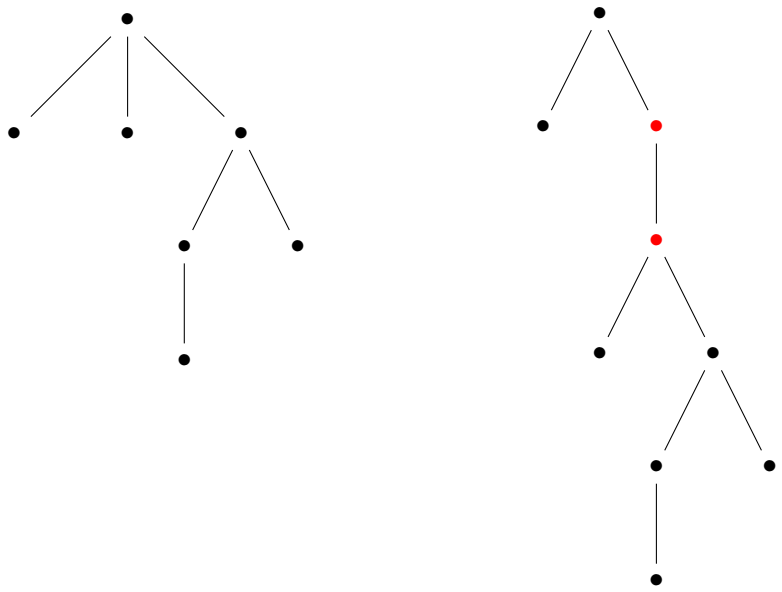
What is crucial is to understand how this (possible) change can (possibly) propagate upward in the labeling. In fact, such propagation can only have the effect of possibly switching some 0s to 1s at odd depths and some 1s to 0s at even depth as we move upward towards the root. In particular, if we prune-even a full subtree that does not intersect with the canonical sequence $Alt(G^*)$, such sequence will still appear unchanged in H^*, that is, $Alt(G^*) = Alt(H^*)$. If we prune-even a full subtree that does intersect with $Alt(G^*)$, then an initial segment of $Alt(G^*)$ will survive in H^*, that is, $Alt(H^*)$ is an initial segment of $Alt(G^*)$. □

3.4.2 Write Odd

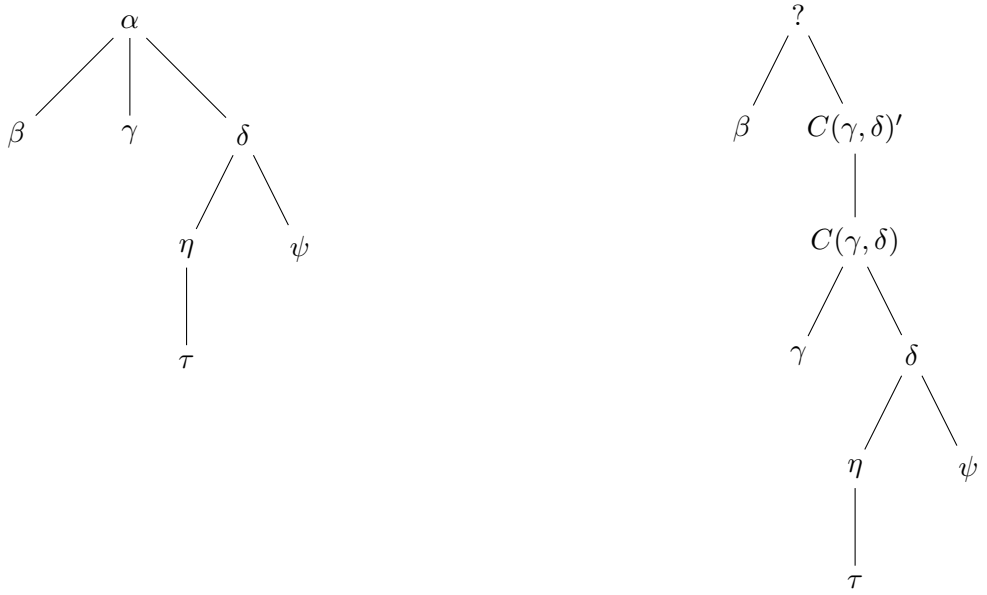
The proof for Writing Odd is completely dual to that for Erase Even.

3.4.3 Insert and Erase Double Cut

In our tree model, the insertion of a Double Cut is, informally speaking, equivalent to the gluing a single edge with two nodes, which we will refer as a *stick* inside the tree. The most general case is the insertion of double cut around a subgraph, which in the tree model, corresponds to the insertion of a stick between the children and their parent, as illustrated by the following example:



Let us now understand how the insertion (removal) of such a stick affects the labeling of the tree into which it is glued (from which it is removed).



Here the Greek letters represent labels (either 0 or 1) at each node. The resulting tree generated by the insertion of the stick around the subgraphs rooted at γ and δ is represented on the right. A priori, we do not know what the value of the root is, hence the question mark at the top of the diagram.

Let us notice that the labeling rule we established in the previous section has a neat algebraic counterpart. It can indeed be expressed in the following way:

- Label all the terminal nodes with 1.
- The label of a node n, say $l(n)$ is equal to

$$l(n) = 1 - \Pi_c l(n_c)$$

where by $\Pi_c l(n_c)$ we denote the product of the labels of all the children of n.

In the example above, if we label the father of γ and δ is denoted by $C(\gamma, \delta)$ we have that $C'(\gamma, \delta) = |1 - C(\gamma, \delta)|$, hence $C(\gamma, \delta) = 1 - \gamma\delta$, so that

$$C'(\gamma, \delta) = |1 - 1 - C(\gamma, \delta)| = \gamma\delta$$

showing that the root must be labeled with α.

The argument just outlined generalizes easily. Let us consider a labeled cuts-only graph G^* and denote by G^*_{DC} the labeled cuts-only graph obtained by G^* by inserting a stick. Then the following result holds.

Theorem 5 *The labeled tree obtained from G^*_{DC} by deleting the stick that had previously inserted in G^* coincides with the labeled tree G^*.*

Corollary 3 *Double Cut is a sound rule.*

3.4.4 Iteration and Deiteration

In order to discuss Iteration, let us first define a notion of *dependence* between nodes.

Definition 3 *Given a tree T, we say that one of its nodes, say n, depends on node m if n belongs to the full subtree T_m rooted at m.*

In our tree model, the rule of Iteration corresponds to copying and pasting any full subtree T_n of the tree corresponding to a cuts-only graph G^* immediately below any arbitrary node k which depends on f, where f is the father of n (we allow f to be the empty node, which takes care of the case n is the root of the tree). We also impose the constraint that we are not allowed to paste T_n at any node which depends on n itself.

To show the soundness of this rule, let us first notice that the value of the node n, say $l(n)$, is completely determined once for all (remember that T_n is a full subtree). Suppose now we glue T_n immediately below a node k which in turns depends on the father of n, which we called f, and such that k does not depend on n.

The key to understanding how the labeling of the tree after the iteration changes is simply to recognize that, since we are operating within a context of nodes that depend on f (the father), all the possible changes in the labeling will eventually converge into the father node f, and we can therefore restrict the analysis of cases to possible changes in the labeling of f. The only two possibilities for $l(n)$ are $l(n) = 0$ or $l(n) = 1$.

If $l(n) = 0$, then $l(m) = 1$ since m is the father of n, and therefore gluing T_n below any node that depends on f will have no effect on $l(f)$, since one of its children, namely n, will still be labeled with 0 after the iteration, hence the labeling above the node f will not change as well.

If $l(n) = 1$, gluing T_n and any node whatsoever, and in particular immediately below node k, does not have any effect on the value of the father of k, leaving therefore the labeling of the tree, at least above node k, unchanged.

The proof of the soundness of Deiteration is similar to the one just described.

Remark 1 *A few general principles emerge from the brute force analysis above. The value 1 is akin to a neutral element, in the sense that introducing a node labeled 1 at any point in the tree does not affect the label of its father. This agrees with the idea that 1 is the indicator of truth in our system.*

4 $\mathbf{EG}_\alpha$: Soundness and Completeness

We begin this section by showing that there is a natural way to lift the soundness of EG^*_α (the cuts-only fragment) to that of EG_α (Peirce's full system including variables). As for completeness, we present two constructive proofs for the completeness of EG_α. The first one uses a reduction to Conjuctive Normal Form, while the second one is based on a classic method developed originally by Kalmar in 1935 [8]. Both proofs exploit the diagrammatic

features of Peirce's notation so as perspicuously to capture the core reasoning involved in the proofs and to streamline their presentation and readability.

4.1 Soundness

We first extend the definition of syntactical entailment to EG_α.

Definition 4 *Let us consider two EG_α graphs G and H. We say that G* entails syntactically *H* in *n* steps *if and only if there is a sequence of graphs*

$$G = G_0, G_1, \ldots G_n = H$$

such that G_{i+1} is derivable from G_i using one of the transformation rules, for all $i = 0, \ldots, n-1$. We use the notation

$$G \vdash^n H$$

to indicate that G entails syntactically *graph H in n steps (in what follows, we may drop the subscript n when not needed).*

Let us now indicate by Var the alphabet of characters on which alpha graphs are built, and define an *interpretation* to be a map from the collection of alpha graphs to the 2-element set whose elements are the empty sheet and the empty cut (here the "space" before the comma is meant to represent the blank Sheet of Assertion).

$$h : Var \to \{\ , \bigcirc\}$$

For any alpha graph G let us also define G^{*h} to be the cuts-only graph obtained by replacing each variable X in G with $h(X)$.

We can now define semantic entailment for alpha graphs:

Definition 5 *Let us consider two EG_α graphs G and H. We say that G* entails semantically *H if and only if*

$$V(G^{*h}) = 1 \Rightarrow V(H^{*h}) = 1$$

for all interpretations h.

We use the notation $G \vDash H$ to indicate that G *entails semantically* the graph H.

Before proving completeness, let us remember that, for every EG_α graph G, the only difference between itself and its correspondent interpreted graph G^{*h} via h is that the variables in G are replaced by either the empty cut or the empty sheet in G^{*h}. This is important since transformation rules act, in general, on subgraphs of G (in particular variables), and can therefore be lifted onto transformations acting on subgraphs of G.

Theorem 6 *Let G and H be two α-graphs. We have that*

$$G \vdash H \Rightarrow G \vDash H$$

Proof: The main idea here is to notice that the same sequence of applications of the rules that transforms G into H can be applied to the cuts-only graph G^{*h} yielding the cuts-only graph H^{*h}. Since soundness has been proven for cuts-only graphs, the result follows. More formally, we can proceed by checking the soundness of each rule.

For Double Cut, let G_{DC} the graph obtained by G via an application of the Double Cut Rule and h such that $V(G^{*h}) = 1$. Then G^{*h}_{DC} is obtained by $V(G^{*h}) = 1$ via an application of DC in EG^*_α, which we have already proved to be sound in that system.

For the remaining rules, the same strategy applies. □

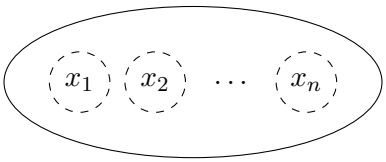
4.2 Completeness I

In order to prove the completeness of EG_α, we will show how to construct a proof of an arbitrary tautology. Let us first define tautologies:

Definition 6 *An EG_α graph G is a tautology if, for every possible interpretation h, $V(G^{*h}) = 1$.*

Here we switch back to the model of nested circles, instead of using tree-like structures or purely syntactical expressions. It is more suitable for our discussion to work within the original Peircean representations.

First of all, since EG_α has been shown to correspond to Propositional Calculus ([10]), it is not surprising that every graph G can be put in a logically equivalent Conjunctive Normal Form (CNF), say CNF(G). This means that G can be written as a conjunction of disjunctions of subgraphs that can be interpreted as *literals*.[2] More precisely, a graph G can be written as a juxtaposition of a finite number of graphs of the following form:



Here the dashed circle represents a literal, that is, the variable contained in it may or may not be enclosed by a single cut (we also admit the case of the *empty variable*. A proof of the above reduction can be found in [6], in which an algorithm – invoking only rules from the reversible fragment of Peirce's logical system – performing such task is presented.

[2] A literal is either an atomic variable or a negated atomic variable.

The proof is therefore constructive, and, as expected, iteration and its inverse (deiteration) play a crucial role in the algorithm.

Without loss of generality, we can restrict our attention to the case where CNF(G) is formed by just *one* disjunction of literals, that is, one *main sep* containing literals, as in the picture above.

Lemma 5 *If G is a tautology such that CNF(G) is formed by just one disjunction of literals, then at least one of the variables inside the main sep must occur both with and without a sep around it.*

Proof: It is easy to check that, if the conclusion of the theorem does not hold, that is, if it is not the case that at least one of the variables inside the main sep does occur both with and without a sep around it, then the interpretation h such that:

$$x_i = \begin{cases} & if \quad (x_i) = x_i \\ \bigcirc & if \quad (x_i) = (x_i) \end{cases}$$

would yield $V(G^{*h}) = 0$, contradicting the hypothesis that G is a tautology. Therefore at least one of the variables must appear both with and without a sep around it. □

Theorem 7 *For every tautology G there exists a proof*

$$\vdash \text{CNF}(G)$$

Proof: We can can partition the variables appearing in CNF(G) into the following classes:

1. Λ_1-variables: they only appear without a cut around them, of the form x_i;
2. Λ_2-variables: they only appear with a cut around them, of the form (x_i);
3. Λ_3-variables: they appear both in the form x_i and (x_i).

We can then immediately build a proof for G:

1. Start with the empty sheet.
2. Draw a double cut.
3. Iterate the innermost cut n times, where $n-1$ times is the cardinality of Λ_3.
4. Write on the depth 1 area all the Λ_1-variables, all the Λ_2-variables with a single cut around each of them and all the Λ_3-variables.

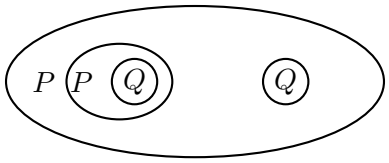
5. Iterate the Λ_3-variables inside the cuts opened up at step 3.
6. Perform DC^- as needed.

The resulting graph is CNF(G). $\square$

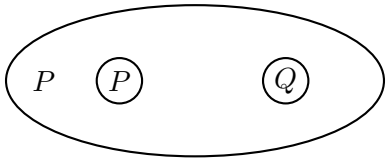
In order to illustrate the strategy outlined above, let us look at the tautology

$$(P \wedge (P \Rightarrow Q)) \Rightarrow Q$$

which is represented in EG_α by



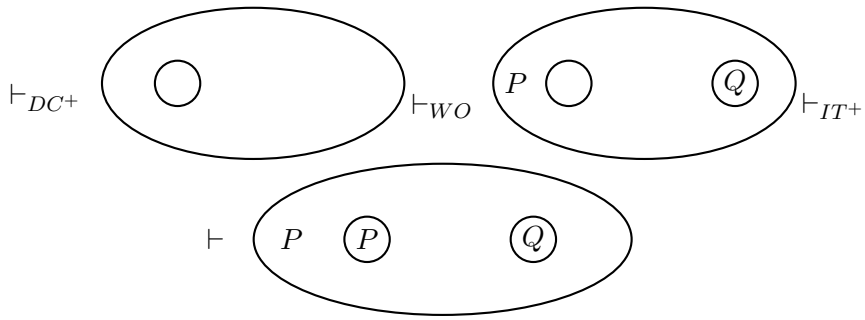
Then, CNF(G) becomes:



In this case we have

1. Λ_1-variables: $\emptyset$
2. Λ_2-variables: Q
3. Λ_3-variables: P

The proof is then:



4.3 Completeness II

We now present a constructive proof for completeness which does not invoke the reduction to CNF. The method that we use is not new, having been first developed by Kalmar [8] in 1935. What is interesting, however, is the fact that in the framework of EG_α, which uses the *deep* inference rule of iteration and treats variable substitution essentially as *endomaps* on EG_α, such a proof becomes much more streamlined and, in our opinion, elegant.

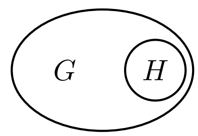
Let us first recall the Deduction Theorem for EG_α:

Theorem 8 *Suppose that $G_1, G_2, \ldots, G_n, G$ are graphs such that*

$$G_1, G_2, \ldots, G_n, G \vdash H$$

then

$$G_1, G_2, \ldots, G_n \quad \vdash$$



For a proof, see [6].

Let us now consider a graph G along with all the variable letters occurring in G, say $x_1, x_2, \ldots, x_n$, and let us fix an interpretation h. Let us denote by e the empty sheet and by c the empty cut, and consider the set $S = \{e, c\}$. There is a natural action of S on EG_α defined by

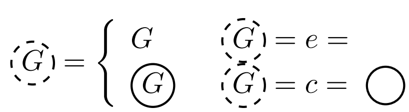
$$S \times EG_\alpha \longrightarrow EG_\alpha$$

$$(s, G) \mapsto s[G]$$

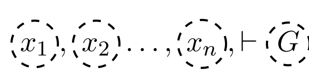
defined by

$$s[G] = \begin{cases} G & if\ s = e \\ \textcircled{G} & if\ s = c \end{cases}$$

In order to simplify our discussion, and give the notion a diagrammatic flavor, we can resort once again to the dashed cut, by setting, for a fixed interpretation h,



Lemma 6 *For any graph G with variable letters $x_1, x_2, \ldots, x_n$ and any interpretation h, we have that*



Proof: First notice that, by construction, the cuts-only graph $(G)^{*h}$ evaluates to the empty sheet, and therefore it is provable: that is, it is derivable from the empty sheet in a finite number of steps. But then from the graphs

$$(x_1)\ (x_2)\ldots\ (x_n)$$

we can derive

$$(x_1)\ (x_2)\ldots\ (x_n)\ (G)^{*h}$$

The structure of variables defined by $(G)^{*h}$ is the same as that of (G) except that all the variables that evaluate to true are replaced by an empty sheet placeholder, and those that evaluate to false are replaced by an empty cut placeholder. But now we can draw from our stash of variables that are written on the empty sheet and just iterate inside $(G)^{*h}$. □

In fact, using the neat fact that

$$(x_i) = x_i$$

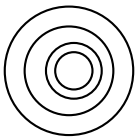
it follows that iterating (x_i) down at any of its placeholders will always return x_i (after possibly a double cut move).

As a concrete example, consider the implication $P \Rightarrow Q$ along with the interpretation $h(P) = \top$ and $h(Q) = \bot$. The claim of the lemma is then that

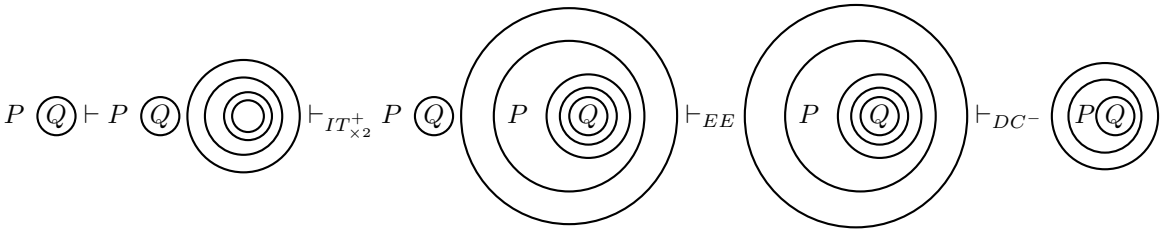
$$P,\ (Q) \vdash (G)$$

where G is the graph

We know that, by construction, $(G)^{*h}$ is provable. Since the latter is equal to $(G)^{*h}$, that means that the following graph is provable:



But then [3]



At this point, the completeness theorem follows using a standard argument of variables elimination. In EG_α however, using iteration and the dual role of the sep as an operator and operand, the proof becomes very short and streamlined, and we thought it would be useful for the reader to include the details of it below.

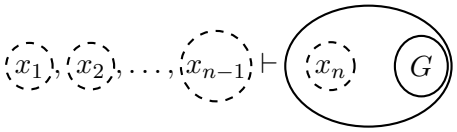
Theorem 9 *For any $G \in EG_\alpha$ we have that*

$$\models G \quad \Rightarrow \quad \vdash G$$

Proof: Let us assume that $var(G) = x_1, x_2, \ldots, x_n$. Since G is a tautology, then $(G) = G$ and by the previous lemma we have that

$$(x_1), (x_2), \ldots, (x_n) \vdash G$$

By the deduction theorem we have that



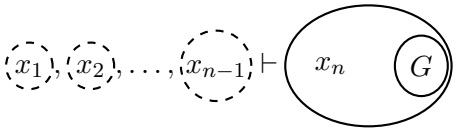
Let us pick two interpretations h_1 and h_2 such that they differ only on x_n, that is, such that

$$h_1(x_i) = h_2(x_i),\ i = 1, \ldots, n-1$$

and

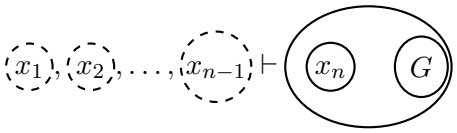
$$h_1(x_n) = \ , h_2(x_n) = \bigcirc$$

By the previous lemma we have that

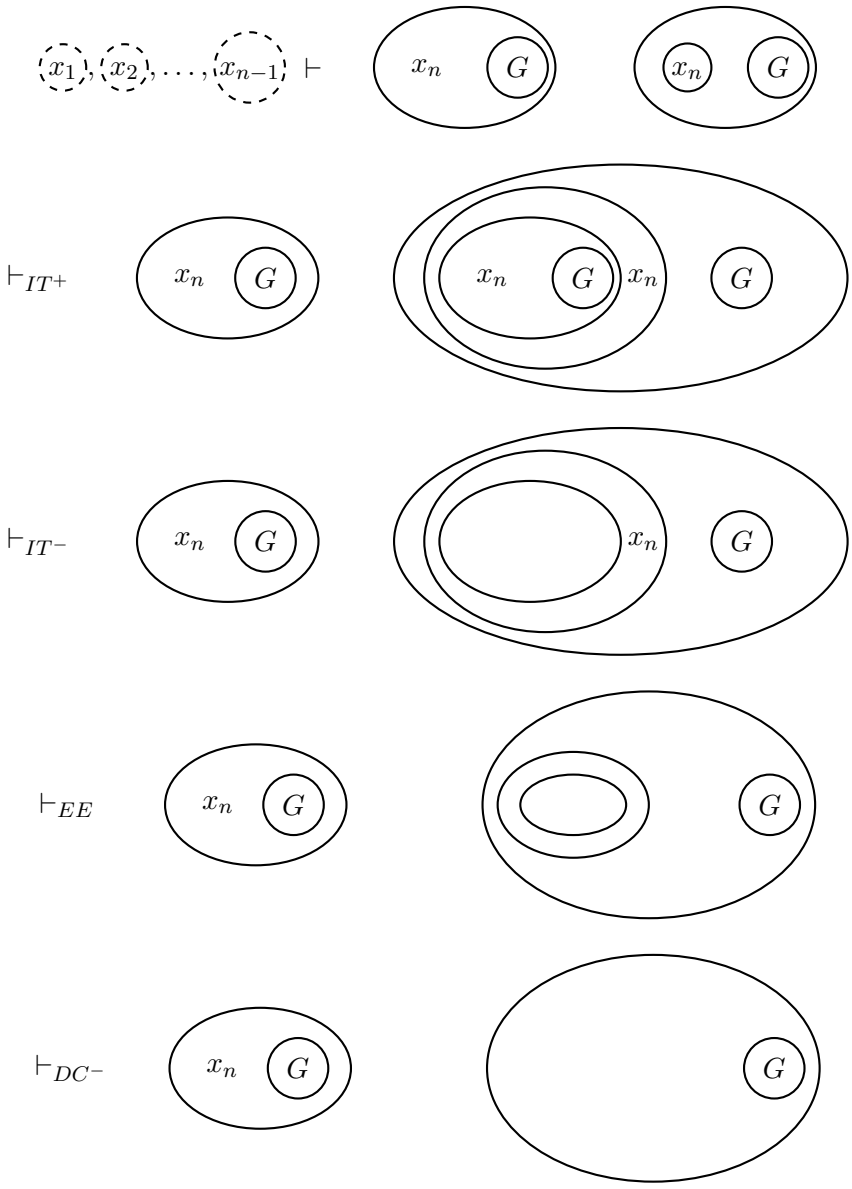


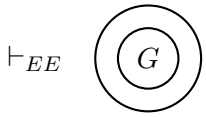
[3] In the sequence below, for the sake of brevity the two iterations for P and Q have been collapsed to a single one.

and



From the above we infer that





$$\vdash_{DC-} \qquad G$$

This shows that we have eliminated x_n. We can then proceed to iterate this process as many times as necessary until all the variables are eliminated.

5 Conclusion

We have presented proofs of the soundness and completeness of Peirce's system of Existential Graphs (Alpha) that, unlike previous proofs, do not require the translation of Peirce's notation into the more standard linear notation of Propositional Logic. Within this "native" diagrammatic presentation, both soundness and completeness proofs rely on the relationship between the full system EG_α and the variable-free (cuts-only) fragment EG^*_α. The tight connection between the evaluation of truth-values and the constructibility of proofs in EG^*_α provides an interesting point of view on an otherwise somewhat obscure aspect of Propositional Logic. Lifting this relationship, which to some degree blurs the hard and fast distinction between syntax and semantics, to the more general level of EG_α offers a fresh perspective on the latter. Making proofs available that draw directly upon the features of Peirce's diagrammatic notation and syntactic transformation rules without requiring the mediation of any translation into standard linear notation helps to foster direct study and investigation of the properties of the EG system. As logicians become more comfortable with such diagrammatic methods, new perspectives and insights are to be expected. It is hoped that these results help to motivate further research on Peirce's system of Existential Graphs, in particular to encourage constructive investigations into their rich logical and diagrammatic structures that pursue similar "native" methods of representation and proof.

References

[1] M. Ma, A. Pietarinen. Peirce Calculi for Classical Propositional Logic. *The Review of Symbolic Logic* **13** 3, 1-32 (2020).

[2] M. Ma, A. Pietarinen. Proof Analysis of Peirce's Alpha System of Graphs. *Studia Logica* **305** 625-647 (2017).

[3] F. Bellucci, A. Pietarinen. Existential Graphs as an instrument of logical analysis: Part I. Alpha.

[4] G. Brady, T.H. Trimble. A Categorical Interpretation of C.S. Peirce's Propositional Logic Alpha. *Journal of Pure and Applied Algebra* **49** 213–230 (2000).

[5] D. Chiffi, A. Pietarinen. On the Logical Philosophy of Assertive Graphs *Journal of Logic, Language and Information* **29** 375-397 (2020).

[6] F. Dau. Some Notes on Proofs with Alpha Graphs. *Conceptual Structures: Inspiration and Application*. ICCS 2006. Lecture Notes in Computer Science **4068** (2006).

[7] R. Gangle, G. Caterina, F. Tohmé. A Generic Figures Reconstruction of Peirce's Existential Graphs (Alpha). *Erkenntnis*, Springer (2020).

[8] L. Kalmár. Über die Axiomatisierbarkeit des Aussagenkalküls. *Acta Scientiarum Mathematicarum* **7** 222-243 (1935).

[9] L. Kauffman. Cybernetics & Human Knowing. Imprint Academic (2001).

[10] D.D. Roberts. *The Existential Graphs of C.S. Peirce*. Mouton, The Hague (1973).

[11] S.J. Shin. *The Iconic Logic of Peirce's Graphs*. MIT Press, Cambridge MA.

[12] G. Spencer-Brown. *Laws of Form*. Allen & Unwin, London (1969).

[13] J. Zeman *Peirce's Graphs. Conceptual Structures: Fulfilling Peirce's Dream*. ICCS 1997. Lecture Notes in Artificial Intelligence (1997).