

Types and Tokens for Logic with Diagrams

Frithjof Dau

Technische Universität Darmstadt, Fachbereich Mathematik
Schloßgartenstr. 7, D-64289 Darmstadt, dau@mathematik.tu-darmstadt.de

Abstract. It is well accepted that diagrams play a crucial role in human reasoning. But in mathematics, diagrams are most often only used for visualizations, but it is doubted that diagrams are rigor enough to play an essential role in a proof. This paper takes the opposite point of view: It is argued that rigor formal logic can be carried out with diagrams. In order to do that, it is first analyzed which problems can occur in diagrammatic systems, and how a diagrammatic system has to be designed in order to get a rigor logic system. Particularly, it will turn out that a separation between diagrams as representations of structures and these structures themselves is needed, and the structures should be defined mathematically. The argumentation for this point of view will be embedded into a case study, namely the existential graphs of Peirce. In the second part of this paper, the theoretical considerations are practically carried out by providing mathematical definitions for the semantics and the calculus of existential Alpha graphs, and by proving mathematically that the calculus is sound and complete.

1 Motivation and Introduction

The research field of *diagrammatic reasoning* investigates all forms of human reasoning and argumentation wherever diagrams are involved. This research area is constituted from multiple disciplines, including cognitive science and psychology as well as computer science, artificial intelligence, logic and mathematics. But it should not be overlooked that there has been until today a long-standing prejudice against non-symbolic representation in mathematics and logic. Without doubt diagrams are often used in mathematical reasoning, but usually only as illustrations or thought aids. Diagrams, many mathematicians say, are not rigorous enough to be used in a proof, or may even mislead us in a proof. This attitude is captured by the quotation below:

[The diagram] is only a heuristic to prompt certain trains of inference; ... it is dispensable as a proof-theoretic device; indeed ... it has no proper place in a proof as such. For the proof is a syntactic object consisting only of sentences arranged in a finite and inspectable area.

Neil Tennant 1991, quotation adopted from [Ba93]

Nonetheless, there exist some diagrammatic systems which were designed for mathematical reasoning. Well-known examples are Euler circles and Venn diagrams. More important to us, at the dawn of modern logic, two diagrammatic systems had been invented in order to formalize logic. The first system is Frege's Begriffsschrift, where Frege tried to provide a formal universal language. The other one is of more relevance for this conference, as Sowa's conceptual graphs are based on them: It is the systems of existential graphs (EGs) by Charles Sanders Peirce, which he used to study and describe logical argumentation. But none of these systems is used in contemporary mathematical logic. In contrast: For more than a century, linear *symbolic* representation systems (i.e., formal languages which are composed of signs which are a priori

meaningless, and which are therefore manipulated by means of purely formal rules) have been the exclusive subject for formal logic. There are only a few logicians who have done research on formal, but non-symbolic logic. The most important ones are without doubt Barwise and Etchemendy. They say that

there is no principle distinction between inference formalisms that use text and those that use diagrams. One can have rigorous, logically sound (and complete) formal systems based on diagrams.

Barwise and Etchemendy 1994, quotation adopted from [Sh01]

This paper advocates this view that rigor formal logic can be carried out by means of manipulating diagrams. The argumentation for this point of view will be embedded into a case study, where the theoretical considerations are practically carried out to formalize the Alpha graphs of Peirce. Although the argumentation is carried out on EGs, it can be transferred to other diagrammatic systems (e.g., for conceptual graphs) as well.

For those readers who are not familiar with EGs, Sec. 2 provides a short introduction into EGs. There are some authors who explored EGs, e.g. Zeman, Roberts or Shin. All these authors treated EGs as graphical entities. In Sec. 3, some of the problems which occur in this handling of EGs are analyzed. For this, the approach of Shin (see [Sh01]) will be used. It will turn out that informal definitions and a missing distinction between EGs and their graphical representations are the main problems. In fact, due to these problems, Shin's (and other authors as well) elaboration of EGs is from a mathematician's point of view insufficient and cannot serve as a diagrammatic approach to mathematical logic. I will argue that a separation between EGs as abstract structures and their graphical representations is appropriate, and that a *mathematical* definition for EGs is needed. A question which arises immediately is how the graphical representations should be defined and handled. In Secs. 4 and 6, two approaches to solve this question are presented. It will be shown that a mathematical formalization of the graphical representations may cause a new class of problems. In Sec. 5 we will discuss why mathematical logic does not have to cope with problems which arise in diagrammatic systems. It will turn out that the preciseness of mathematical logic is possible although the separation between formulas and their representation is usually not discussed. From this result we draw the conclusion that a mathematical formalization of the diagrammatic representations of EGs is *not* needed. In Sec. 6, the results of the preceding sections are brought together in order to describe my approach for a mathematical foundation of diagrams. In the remaining sections, the results of the theoretical discussion are applied to elaborate mathematically a complete description of the Alpha-part of EGs.

2 Existential Graphs

In this paper, we consider the Alpha- and Beta-part of existential graphs. Alpha is a system which corresponds to the propositional calculus of mathematical logic. Beta builds upon Alpha by introducing new symbols to Alpha, and it corresponds to first order predicate logic (FOPL), that is first order logic with predicate names, but without object names and without function names.

We start with the description of Alpha. The EGs of Alpha consist only of predicate names of arity 0, which Peirce called *medads*, and of closed, double-point-free curves which are called *cuts* and used to negate the enclosed subgraph. Medads can

be considered as (atomic) propositions, i.e., they correspond to propositional variables in propositional logic. Propositions can be written down on an area (Peirce used the term ‘*scribing*’ instead of ‘writing’), and writing down a proposition is to assert it. The area where the proposition is written on is what Peirce called the *sheet of assertion*. It may be a sheet of paper, a blackboard or any other surface. Writing several propositions next to each other (this operation is called a *juxtaposition*) asserts the truth of each proposition, i.e. the juxtaposition corresponds to the conjunction of the juxtaposed propositions. For example, writing the propositions ‘it rains’ and ‘it is cold’ next to each other yields the graph

it rains it is cold

which means ‘it rains and it is cold’.

Encircling a proposition is to negate it. The line which is used to encircle a proposition is called a *cut*, the space within a cut is called its *close* or *area*. For example,

(it rains it is cold)

has the meaning ‘it is not true that it rains and that it is cold’, i.e. ‘it does not rain or it is not cold’. Cuts may not overlap, but they may be nested. The next graph has two nested cuts.

(it rains (it is cold))

This graph has the meaning ‘it is not true that it rains and that it is not cold’, i.e. ‘if it rains, then it is cold’. The device of two nested cuts is called a *scroll*. From the last example we learn that a scroll can be read as an implication. A scroll with nothing on its first area is called *double cut* (it corresponds to a double negation). As mentioned before, the space within a cut is called the *area* of the cut. In the example above, we therefore have three distinct areas: All the space outside the outer cut, i.e. the sheet of assertion, the space between the outer and the inner cut, which is the area of the outer cut, and the space inside the inner cut, which is the area of the inner cut. An area is *oddly enclosed* if it is enclosed by an odd number of cuts, and it is *evenly enclosed* if it is enclosed by an even number of cuts.

We have the possibility to express conjunction and negation of propositions, thus Alpha has the expressiveness of propositional logic. Peirce also provided a set of five derivation rules for EGs. For Alpha, these rules are:

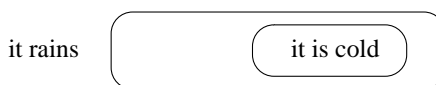
- Erasure: Any evenly enclosed subgraph may be erased.
- Insertion: Any graph may be scribed on any oddly enclosed area.
- Iteration: If a subgraph \mathfrak{G} occurs on the sheet of assertion or in a cut, then a copy of \mathfrak{G} may be scribed on the same or any nested area which does not belong to \mathfrak{G} .
- Deiteration: If a subgraph \mathfrak{G} could be the result of iteration, then it may be erased.
- Double Cut: Any double cut may be inserted around or removed from any graph of any area.

This set of rules is sound and complete. In the following, a simple example of a proof is provided (which will be an instantiation of modus ponens in EGs).

Let us start with the graph on the right. It has the meaning ‘it rains, and if it rains, then it is cold’.

it rains (it rains (it is cold))

The inner instance of ‘it rains’ may be considered a copy of the outer instance of ‘it rains’. Hence we can erase the inner instance of ‘it rains’ using the deiteration-rule.



This graph contains a double cut, which now may be removed.



Finally we erase the proposition ‘it rains’ with the erasure-rule.



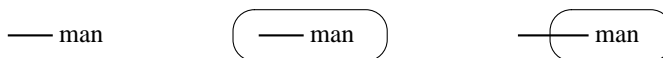
So the graph with the meaning ‘it rains, and if it rains, then it is cold’ implies the graph with the meaning ‘it is cold’.

If we go from the part Alpha of EGs to the part Beta, predicate names of arbitrary arity may be used, and a new syntactical item, the *line of identity (LI)*, is introduced. LIs are used to denote both the existence of objects and the identity between objects. They are attached to predicate names and drawn bold. Consider the following graph:



It contains two LIs, hence it denotes two (not necessarily different) objects. The first LI is attached to the unary predicate ‘man’, hence the first object denotes a man. Analogously the second LI denotes a woman. Both lines are attached to the dyadic predicate ‘loves’, i.e. the first object (the man) stands in the relation ‘loves’ to the second object (the woman). The meaning of the graph is therefore ‘there are a man and a woman such that the man loves the woman’, or in short: A man loves a woman.

LIs may cross cuts.¹ Consider the following graphs:



The meaning of the first graph is clear: it is ‘there is a man’. The second graph is built from the first graph by drawing a cut around it, i.e. the first graph is denied. Hence the meaning of the second graph is ‘it is not true that there is a man’, i.e. ‘there is no man’. In the third graph, the LI begins on the sheet of assertion. Hence the existence of the object is asserted and not denied. For this reason the meaning of the third graph is ‘there is something which is not a man’.

Now we have the possibility to express existential quantification, predicates of arbitrary arities, conjunction and negation. Hence we see that the part Beta of EGs corresponds to FOPL.

Essentially, the rules of the calculus for Beta are extensions of the five rules for Alpha such that they now encompass the properties of the lines of identity. For example, the erasure-rule and the insertion-rule are now formulated as follows:

- Erasure: Any evenly enclosed graph and any evenly enclosed portion of a LI may be erased.

¹ This is not fully correct: Peirce denies that a LI may cross a cut (a corollary in [Pe03] states ‘It follows that no line of identity can cross a cut.’), Roberts allows it, and it is not clear whether Shin allows it or not. In Peirce’s view, the third graph has *two* lines of identity which ‘meet’ at the cut. But the discussion of this needs a much deeper understanding of EGs and shall therefore be omitted here.

- Insertion: Any graph may be scribed on any oddly enclosed area, and two LIs (or portions of lines) oddly enclosed on the same area, may be joined.

For the formulation of the remaining three rules we refer to [Ro73] (we have taken the other formulations of the rules from this source).

3 Problems with Existential Graphs

To illustrate some of the problems which may occur in the handling of diagrams, we focus on the EGs as they are described in the book of Shin ([Sh01]), but we will refer to other authors like Peirce, Zeman or Roberts as well. For our discussion, it is sufficient to consider the definitions Shin provides for Beta graphs. It is labelled ‘Non-Math.(ematical) Definition’ to distinguish it from mathematical definitions as they will appear later in this paper.

Non-Math. Definition 1 (Beta Graphs)

The set of beta graphs, \mathcal{G}_β , is the smallest set satisfying the following:

1. An empty space is in \mathcal{G}_β .
2. A line of identity is in \mathcal{G}_β .
3. Juxtaposition closure If G_1 is in \mathcal{G}_β , ..., and G_n is in \mathcal{G}_β , then the juxtaposition of these n graphs, i.e. G_1, \dots, G_n , (we write ‘ $G_1 \dots G_n$ ’ for juxtaposition) is also in \mathcal{G}_β .
4. Predicate closure If G is in \mathcal{G}_β , then a graph with an n -ary predicate symbol written at the joint of n loose ends in G is also in \mathcal{G}_β .
5. Cut closure If G is in \mathcal{G}_β , then a graph in which a single cut is drawn in any subpart of G without crossing a predicate symbol is also in \mathcal{G}_β .
6. Branch closure If G is in \mathcal{G}_β , then a graph in which a line of identity in G branches is also in \mathcal{G}_β .



There are two points remarkable in this definition:

1. Although some mathematical terms are used, the definitions are formulated more or less in common spoken language and cannot be seen as *mathematical* definitions.
2. EGs are considered to be *graphical* entities (this can particularly seen in the cut closure rule for Beta graphs).

This approach, particularly the two points mentioned above, yields different kinds of problems which shall be elaborated in this section. We start with problems caused by the use of common language.

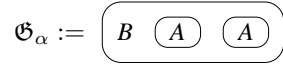
First of all, many important technical terms are defined either in an insufficient way or not at all. For example, the terms *sentence symbol*, *juxtaposition* or *single cut* (this holds already for Shin’s definition of Alpha graphs) as well as the terms *lines of identity*, *loose ends* or *branching of LIs* are not defined. Even if we have some pre-knowledge on terms like ‘single cut’ (e.g. we know that a single cut is a closed, double-point-free curve on the plane) or LIs, these definitions leave some issues open. E.g., is it not clear whether two cuts may touch, cross, intersect, or partly overlap, or whether a LI may terminate on a cut. For example, we might ask which of the following diagrams are well-defined diagrams of Beta graphs:





Examining the first three examples, in Shin's book we do not find any example of an Beta graph where a LI terminates on a cut. But, in contrast, this case is explicitly investigated in Peirce's manuscripts or in the book of Roberts. The fourth example is not explicitly excluded by Shin, but it is implicitly excluded based on the background a reader should have of EGs. Considering the fifth example, Peirce used to draw to nested cuts, i.e., scrolls, as follows: . So it seems that a touching of cuts is allowed. But we can raise the question whether scrolls should be handled as own syntactical devices, or whether Peirce's drawing is a sloppy version of two nested cuts which do not touch, i.e. of . So we have to make a *decision* whether cuts may touch or not.

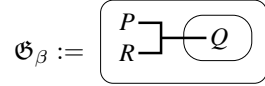
Answering these question is not only about deciding whether a diagram is an EG or not. It is even more important to have rigor definitions for all technical terms when transformation rules, e.g. the rules of a calculus, are applied to EGs. An important example for this is the in most publications undefined term *subgraph*. A rule in the calculus allows us to scribe a copy of a subgraph in the same cut (this is a special case of the *iteration-rule*), which occurs in the treatises of Peirce, Shin, Roberts, and later on in this paper.

To get an impression of the problems, we raise some simple questions on subgraphs. Consider the Alpha graph on the right:



In order to know how the iteration-rule can be applied, it must be possible to answer the following questions: Is  a subgraph of \mathfrak{G}_α ? Is $A \quad A$ a subgraph of \mathfrak{G}_α ? Do we have one or two subgraphs  of \mathfrak{G}_α (this is a question which corresponds to the distinction between subformulas and subformula instances in FOPL)?

For Beta, it is important to know how LIs are handled in subgraphs. Consider the following Beta graph:



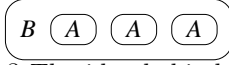


We might ask which of the following diagrams are subgraphs of \mathfrak{G}_β :



Shin (and other authors) does not offer a definition for subgraphs: The answer of the questions above is left to the intuition of the reader. From the discussion above, we draw a first conclusion:

Thesis 1: The definitions of Shin (and other authors) are insufficient for a precise understanding and handling of existential graphs.

If EGs are considered as *graphical* entities, a new class of difficulties has to be coped with. Consider again \mathfrak{G}_α . Remember that the iteration-rule should allow us to draw a copy of the subgraph  into the outer cut. But if we want to understand EGs and subgraphs as (graphical) diagrams, then this is obviously *not* possible, because in the outer cut, there is simply not enough space left for another copy of . But, of course, this is not intended by the rule, and everybody who is familiar with EGs will agree that  is a result of a valid application of the iteration-rule. Why that? The idea behind is that we may change the shape of LIs or cuts to a 'certain degree' without changing the meaning of an EG. For this reason it is evident that any attempt which tries to define EGs as purely graphical entities runs into problems.

For a discussion of the term 'certain degree', consider the three Alpha graphs in Figure 1. From the left to the right, we decrease the size of the outer cut. Thus

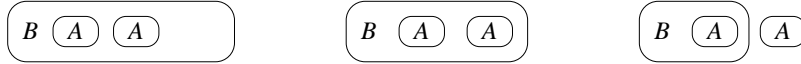



Fig. 1. Diagrams of Alpha graphs

there are obviously visual differences between these three diagrams. The question is whether the differences between the first two diagrams are comparable to the differences between the last two diagrams. We have already seen that the shape of a cut is – in some sense – of no relevance. The only thing we have to know is which other items of a graph are enclosed by the cut and which are not. Thus we see that the first two diagrams are (in some sense) the same graph, particularly they have the same meaning. In contrast to that the third diagram has a different meaning and has therefore to be treated differently.

If we – due to the visual differences – treat the first two diagrams to be syntactically different, we would get a syntax which is much too fine-grained. Any kind of equivalence between graphs would be postponed to the semantical level. Furthermore, we would need transformation rules which allow to transform the first graph into the second graph (and vice versa). This syntax would become very complicated and nearly unusable. Thus we see that any appropriate syntax should not distinguish between the first two diagrams.

Now the question arises which properties of the first two diagrams cause us to identify them. Should we syntactically identify graphs when they have the same meaning? This would inappropriately mix up syntax and semantics. For example, the empty sheet of assertion and the graph  have the same meaning, but they should obviously be syntactically distinguished.

In defining a reasonable syntax for the graphs, we see that we have to prescind from certain graphical properties of the diagrams (e.g. the *form* of a cut), while other properties are important (e.g. the number of cuts and other items, or whether an item of the diagram is enclosed by a cut or not). Particularly, EGs should not be understood as graphical entities at all. Instead of this, we have to distinguish between graphs and the diagrams which *represent* the graphs. This is according to Peirce's view. He says: 'A graph [...] is a symbol, and, as such, general, and is accordingly to be distinguished from a graph-replica.' Thus, Peirce distinguishes between *graphs*, which are so-to-speak abstract structures, and their *representations*. Due to this understanding, the first two diagrams in Figure 1 are not different graphs with the same meaning, but different representations, i.e., diagrams, of the same graph, and the third diagram is a representation of a different graph. Peirce explicitly said that arbitrary features of the diagrams may vary, as long as they represent the same diagram. At the beginning of [Pe03] he says:

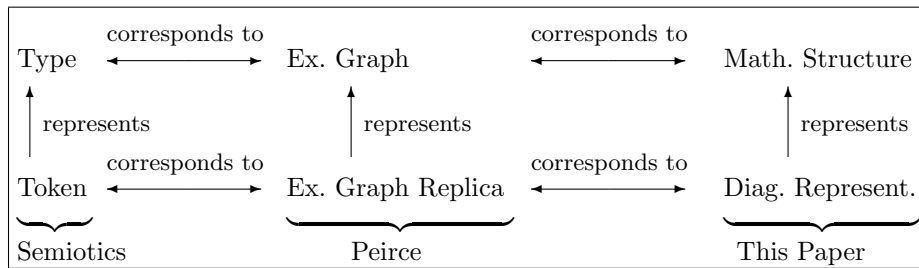
Convention No. Zero. Any feature of these diagrams that is not expressly or by previous conventions of languages required by the conventions to have a given character may be varied at will. This convention is numbered zero, because it is understood in all agreements.

For LIs, he says even more explicit in [Pe03] that 'its shape and length are matters of indifference.'

The distinction between graphs and graph-replicas obviously corresponds to the distinction between *types* (graphs) and *tokens* (graph replicas), as it is known from philosophy. The type-token issue is far from being settled; nonetheless, this important distinction helps us to draw our next conclusion:

Thesis 2: EGs should not be defined as graphical entities. Instead of that, we need a definition of EGs which copes exactly the crucial features of EGs, and the diagrams should be understood as (mere) representations of an underlying EG.

Roughly sketched, we now have the following situation:



4 The First Approach to Diagrams

One of the most important features of mathematics is its preciseness. The preciseness of mathematics is based on a very strict understanding of mathematical definitions and proofs. We have seen that the informal definitions of EGs lead to problems in their understanding and handling. I claim that mathematics is the best language to cope with problems of these kind, i.e.:

Thesis 3: Mathematics provides the highest level of precision available for definitions and proofs.

Thus, in my view, mathematics turns out to be the best instrument for coping the problems discussed in Sec. 3. Particularly, EGs should be defined as mathematical structures (and these structures prescind certain graphical features from diagrams), and the diagrams are representations for these structures. Nonetheless, the last thesis raises the question whether the graph replicas, i.e., the diagrams of EGs, should be defined mathematically as well. This approach shall be discussed in this section.

Let us assume we want to define the graphical representations of Alpha or Beta graphs mathematically. Then we would have two different kinds of objects: Mathematical structures which model EGs, and mathematical structures which model the representations of EGs, i.e., the diagrams. Let us call the first structures *type-structures* and the second structures *token-structures*. In finding a definition for the token-structures, we have two fundamental problems to cope with: First to find a definition for the token-structures which encodes the informally given diagrams as best as possible. Secondly, we have to show how the type-structures are represented by the token-structures. It should be possible to show that each token-structure represents uniquely a type-structure, and that each type-structure is represented by at least one token-structure. Let us call these two principal problems *representation problem*.

An obvious approach is to model the lines of an EG (i.e., the cuts and the LIs) as families of curves in the Euclidean plane \mathbb{R}^2 . For example, we can model each LI by a smooth, double-point-free curve and each cut by a smooth, double-point-free and closed curve.² Consider the first two graphs of Figure 1. They are two different tokens of the same type. Diagrams like these shall be called *type-equivalent* (this term is adopted from [HS02]).

If we define type-equivalence, we have to refer to the relationship between types and tokens. But the diagrams can be compared directly as well. If we consider again the first two graphs of Figure 1, we see that we have mappings from the cuts resp. the occurrences of propositional variables from the first graph to the second which fulfill certain conditions. For example, the mappings are bijective and they preserve some entailment-relations (e.g. if an occurrence of a propositional variable or a cut is enclosed by a cut, then this holds for the images as well). In some sense, we can say that the first graph can be topologically transformed into the second graph. Graphs like this shall be called *diagrammatically equivalent* (again, this term is adopted from [HS02]). If we have found adequate definitions for the type- and token-structures as well as for the relation ‘a token-structure represents a type-structure’, it should be mathematically *provable* that being type-equivalent and being diagrammatically equivalent means the same.

In any description of the diagrams, particularly if we provide a mathematical definition for them, we have to decide some of the bordercases we have discussed in Sec. 3. For example, we have to decide whether (and how often) LIs may touch cuts, or whether cuts may touch or even intersect each other. But in no (reasonable) mathematical definition, we can encode *all* graphical properties of a diagram directly (e.g. by curves). This is easiest to see for letters or words, i.e. the occurrences of propositional variables in Alpha graphs or for the relation names in Beta graphs. Of course the *location* of the occurrence of a propositional variable in an Alpha graph is important, but neither the size or the font of the propositional variable will be of relevance. Similar considerations hold for relation names in Beta graphs. As the shape of propositional variables or relation names should not be captured by the definition of the diagrams, it is reasonable to handle these items in a different way. The question arises how much of the properties of these items has to be captured by a definition of the diagrams. For example, the occurrences of propositional variables in an Alpha graph could be modelled by points or spots of the Euclidean plane to which we assign the variables. We see that even in the definition for the diagrams, we have to prescind certain graphical features from diagrams.

On the other hand: If we try to capture graphical properties of diagrams, some of the items of a diagram will probably be overspecified. For example, how should a simple line of identity – i.e., ————— – be modelled mathematically? We already said that LI could be modelled by a curve in the Euclidean plane. But when a diagram is given, it is usually drawn without providing a coordinate system. Thus, which length should this curve have? Where in the Euclidean plane is it located? Again we see that we cannot capture a diagram exactly by a mathematical definition.

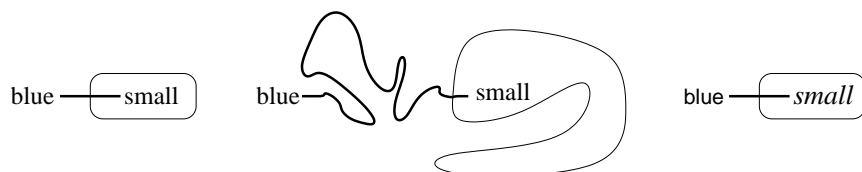
Finally, it is worth to note that we have a couple of (unconscious) heuristics in drawing diagrams for EGs. A simple example is that pending edges should be drawn ‘far away enough’ from any relation-signs. To see this, consider the following diagrams:

² It should be noted that Peirce’s understanding of EGs depends on his understanding of the continuum, and this understanding is very different from the set \mathbb{R} . Nevertheless a mathematization of the diagrams as a structure of lines and curves in \mathbb{R}^2 is convenient as \mathbb{R}^2 is the standard mathematization of the Euclidean plane in contemporary mathematics.



The leftmost diagram should be understood as ‘There is thing which is P which is in relation Q to another thing’, The rightmost diagram should be understood as ‘There is thing which is P and there is thing which is Q ’. But the meaning of the diagrams in the middle is not clear. So, one will avoid drawing diagrams like these.

Another heuristic is to draw a diagram as simple as possible. Furthermore, although we have seen that neither the size or the font of the propositional variable or a relation name will be of relevance, it is clear that the choice of a font and its size is not arbitrary if a diagram is drawn in a convenient way. This should become clear with the following three diagrams:



Although all diagrams are representations of the same graph (with the meaning ‘there exists something which is blue, but not small’), it is clear that the left-most diagram is the best representation of these three diagrams. Conventions like these cannot be captured by any mathematical definition.

Remember that the reason for finding mathematical definitions for diagrams was to solve the representation problem. We wanted to grasp the distinction between non well-formed and well-formed diagrams, as well as the relationship between graphs and their diagrams, as best as possible. We have already seen that we cannot capture all graphical features of a diagram by a mathematical definition (the more graphical properties are encompassed by a definition, the more technical overhead has to be expected). Now the final question is how a mathematically defined diagram is related to a concrete drawing of a diagram on a sheet of paper. This is a crucial last step from mathematical objects to objects of the real world. Thus, even if we provide mathematical definitions for the diagrams, we still have a representation problem. The initial representation problem between mathematically defined graphs and mathematically defined diagrams has shifted to a representation problem between mathematically defined diagrams and diagrams – i.e., drawings – in the real world. A mathematical definition for diagrams can clarify a lot of ambiguities, but it cannot solve the representation problem finally.

5 Linear Representations of First Order Predicate Logic

In the last section we have raised some questions concerning the representation problem, and we have seen that mathematics alone is not enough to solve these problems. It is likely the often unclear relationship between diagrams and represented structures which causes mathematicians to believe that diagrams cannot have a proper place in mathematical argumentation, esp. proofs. It is argued that only a symbolic system for logic can provide the preciseness which is needed in mathematical proofs (for a broader discussion of this see [Sh01]). It seems that the problems we have discussed in the last section simply do not occur in mathematics, esp. mathematical logic. In this section we will have a closer look on this. We start with a definition of the well-formed formulas of FOPL.

Definition 2. *The alphabet for first order logic consists of the following signs:*

- Variables: x_1, x_2, x_3, \dots (countably many)
- Relation symbols: R_1, R_2, R_3, \dots (countably many). To each predicate symbol R_i we assign an arity $ar(R_i) \in \mathbb{N}$.
- Constant symbols: c_1, c_2, c_3, \dots (countably many)
- Connectives: \wedge, \neg, \exists
- Auxiliary Symbols: $., ,, (,)$

Definition 3. *The formulas of FOPL are inductively defined as follows:*

1. Each variable and each constant name is a term.
2. If R is a predicate symbol with arity n and if t_1, \dots, t_n are terms, then $f := R(t_1, \dots, t_n)$ is a formula.
3. If f' is a formula, then $f := \neg f'$ is a formula.
4. If f_1 and f_2 are formulas, then $f := (f_1 \wedge f_2)$ is a formula.
5. If f' is a formula and α is a variable, then $f := \exists \alpha. f'$ is a formula.

It is easy to capture the idea behind this definitions: First, we fix a set of *signs*, and a formula is a *sequence* of these signs which has been composed according to certain *rules*.

Let us consider the following two strings (the relation name R_1 has arity 2):

$$\begin{array}{c} \exists x_1. \exists x_2. R_1(x_1, x_2) \\ \exists x_1. \exists x_2. R_1(x_1, x_2) \end{array}$$

Although these two strings are written in different places and although they look slightly different, they clearly represent the same formula. For our considerations, it is worth to raise the question which steps a reader has to pass from the perception of a string to the identification of a formula which is represented by the string. Roughly spoken, if a reader reads the two strings above, she passes the following steps:

1. The region on the paper (the blackboard, ...) must be identified where the representation of the formula is written on. In the example above, this is possible because we have written the two different strings into two different lines.
2. In this region, we must be able to identify representations of the *signs* which may occur in a formula (e.g. the sign ‘ \exists ’, which appears twice, or the sign ‘ R_1 ’, which appears only once). Nothing else may occur.
3. The representations of the signs must be assembled in a way such that we are able to identify their ordering on the region. That is: We must be able to identify the *sequence*. For our examples, we identify the ordering of the instances of signs by reading them from left to right.
4. Finally, after we have reconstructed the sequence of signs (internally), we can check whether this sequence is a well-defined formula, i.e., whether it is composed with the rules of Definition 3.

In the following, we will use the label (*) to refer to these four steps. The process (*) yields the same result for the two strings above: In both cases, the lines represent the same sequence of signs, which is in fact a well-formed formula. Thus every mathematician would (hopefully) agree that these two strings represent the same (well-defined) formula.

We want to stress that the process of perceiving a representation of a formula is not ‘deterministic’: It is not clear without ambiguity for each string whether it represents a formula or not. To see this, consider the following strings (the ‘type-setting problems’ are intended). The question is: Which of these strings represents a well-defined formula?

$$\exists \exists x_2 R(x_1, x_2) \tag{1}$$

$$\exists \begin{matrix} x_2 R_1(&) \\ \exists & \\ x_1 & \vdots & x_1, x_2 \end{matrix} \tag{2}$$

$$\exists x_1. \exists \heartsuit. R_1(x_1, x_2) \tag{3}$$

$$), \exists. R_1 x_1 \exists x_1(x_2 x_2) \tag{4}$$

$$\exists x_1. \exists x_2. R_1(x_1, x_2) \tag{5}$$

$$\exists x_1. \exists x_2. R_1(x_1, x_2) \tag{6}$$

$$\exists x_1. \exists x_2. R_1(x_1, x_2) \tag{7}$$

$$\exists x_1. \quad \exists x_2. R_1(\quad x_1, x_2) \tag{8}$$

$$\exists \quad x_1. \quad \exists x_2. R_1(\quad x_1, x_2) \tag{9}$$

In line 1, we are neither able to identify the signs, nor to identify their ordering. That is we cannot pass the steps (2) and (3) of (*), thus this line does not represent a formula. In line 2, we are able to identify the signs, but we are not able to identify their ordering. That is we cannot pass the step (3) of (*), thus this line does not represent a formula as well. Moreover, it may be doubted whether step 1 of (*) can be passed without problems. In line 3, we are able to identify the signs, but one of these signs does obviously not belong to our alphabet of first order logic, thus this line does not represent a formula. In line 4, we are able to identify the signs, all of them belong to our alphabet of first order logic, and we are able to identify their ordering, that is we reconstruct a sequence of signs of our alphabet. But we see that this sequence is not build according to our rules (we cannot pass the step (4) of (*)). Thus this line does not represent a formula. In the remaining lines, it is not uniquely determined whether the lines represent a formula or not. In line 5, the font for the variables has changed. In mathematical texts, different fonts are often used to denote mathematical entities of different kinds, but this is not a general rule. So it depends on the context whether lines 5 or 6 are accepted to represent formulas. Using different sizes of a font is usually driven by a specific purpose. The same holds for the use of significantly different distances between signs. It is hardly conceivable to find a purpose for using different font sizes or significantly different distances in formulas. Thus it is possible, but not sure, that the lines 7–9 are not accepted by a mathematician to represent a formula.

In standard books on logic, usually only the last step of (*) is discussed. The main reason for this is the following: The linear notion of formulas corresponds the way ordinary text is written: Text is assembled of letters which are written side by side and which are read from left to right. As we are used to read texts, we are trained as well to read strings which shall represent formulas. Thus the first three steps of (*) are unconsciously executed when we perceive a representation of a formula.

But as soon we perceive an unfamiliar representation (like in the strings 1-9), we become aware of the whole process described by (*). We realize that mathematical structures need representations, and in mathematics we have a clear separation between structures and their representations. The representations rely on conventions, either implicit or explicit, based on common cultural background as well as on mathematical socialization, and they are not fully explicated. Nonetheless, this usually poses no problems: Although these conventions can never be fully explicated,

as long as we provide representations of mathematical structures in accordance to these conventions, they are strong enough to provide a secure transformation from the external representation of a structure (e.g. on a sheet of paper or on a blackboard) into an internal representation of any mathematician, i.e., they refer to the represented structures in a clear and non-ambiguous way (as Barwise says in a broader discussion of representations: 'Every representation indicates a genuine possibility' [Ba93]).

The next thesis claims that this approach can be adopted for diagrammatic systems.

Thesis 4: In a mathematical theory, the mathematical structures need representations. A rigor mathematical theory can be developed without providing mathematical definitions for the representations. Instead of that, it is sufficient if we have conventions – either implicit or explicit – which describe the representations, as well as the relationship between structures and their representations, in a clear and non-ambiguous way.

6 The Second Approach to Diagrams

In Sec. 3, we have argued that in literature, EGs are described in an informal and insufficient way (Thesis 1). Furthermore, we should not mix up graphs and their representations. Particularly, EGs should be defined as formal structures and not as graphical entities (Thesis 2). Thesis 3 claims that mathematics is the best method to describe formal structures. From this we can conclude that EGs should be defined as *mathematical* structures. Nonetheless, we haven't already solved the question whether the representations should be defined mathematically as well.

In Sec. 4, we presented some difficulties when we try to define the diagrams mathematically. The arguments of Sec. 4 are not strong enough to claim that a mathematical definition of diagrams will always run into problems. In contrast: Finding a appropriate mathematical definition for the diagrams should clarify the points mentioned in Sec. 3. That is a mathematical definition would make clear without ambiguities which diagrams should be considered to be well-formed diagrams of EGs, and which not. Furthermore, the relation between graphs and their representations can be elaborated mathematically as well. But providing mathematical definitions for the diagrams may result in a technical overhead or overspecification of the formalization of EGs and their representations, and none mathematical definition can solve the representation problem finally.

On the other hand, as seen in Sec. 5, mathematical logic is a mathematical theory, although the representations of the types in logic, i.e., formulas, are not fully explicated, but they rely on different conventions.

In contrast to the attempt to capture the representations by mathematical definitions, we have seen that logic is a rigor mathematical theory, although representations are only captured not fully explicated conventions. This works because these conventions are mainly based on a common and solid cultural background, namely the form how text is presented. For diagrams, we have a lack of conventions.

Thus, in both approaches, we have to *provide* a set of conventions on how diagrams are written. Particularly for EGs, we have to make clear without ambiguities which diagrams shall be considered to be well-formed diagrams of EGs. In the first approach, these conventions are provided informally in common language. In the second approach, these conventions are described by a mathematical definition. Thus, the mathematical definition should capture as best as possible the informally

provided conventions of the first approach. The advantage of a mathematical definition is its preciseness, but we gain this preciseness for the cost of a technical overspecification of the diagrams. Furthermore, we have seen that a mathematical definition cannot capture exactly the diagrams.

As already said above, arguments like these are not strong enough to generally discard mathematical definitions for the token-structures. It has to be estimated whether a mathematical definition is really needed, or whether the conventions for drawing diagrams and for the relationship between representing diagrams and represented structures can be captured sufficiently by descriptions in common language. If the latter is possible, we are able to gain the rigorousness and preciseness of a mathematical theory without a technical overspecification of the diagrams. Thus, in this case, I claim that this approach should be preferred.

We have already mentioned that the Alpha system of EG is said to be equivalent to propositional logic, the Beta system of EG is said to be equivalent to first order predicate logic. In fact, we find good arguments for that in the books of Zeman, Roberts or Shin. But, as EGs are described informal and insufficiently, these argumentation cannot be seen as (mathematical) proofs. With our approach, we can solve these problems: If we define EGs as mathematical structures, we are now able to *prove mathematically* the equivalences between the Alpha system and the Beta system to propositional and first order predicate logic, respectively.

In the following section, we exemplify this approach for Alpha graphs. Of course this is a fairly simple example. Its advantage is that we can work out the definitions of Alpha and the proof of the equivalence to propositional logic on a couple of pages. Its disadvantage is that the problems we discussed in Sec. 3 appear much stronger in Beta, thus the benefit of a mathematical foundation of graphs cannot be seen that clearly for Alpha. An elaboration of this approach for the Beta system is in progress (a first step towards Beta can be found in [Da03]).

7 Syntax for Alpha Graphs

Alpha graphs are built up from two syntactical devices: sentence symbols and cuts. We first fix the sentence symbols, which we call *propositional variables*.

Definition 4 (Propositional Variables).

Let $\mathcal{P} := \{P_1, P_2, P_3, \dots\}$ be a countably infinite set of propositional variables.

Alpha graphs can be seen to built up from propositional variables and cuts by an appropriate inductive definition. We can try to transform this into a inductive mathematical definition. This is possible, but here Alpha graphs are defined in one step. To emphasize that these structures are abstract structures, not diagrams, they are termed *formal* Alpha graphs.

Let us briefly discuss what we have to capture in the mathematical definition. Consider first the two diagrams depicted in Figure 2.

According to the discussion in Secs. 3–6, these two diagrams represent the same Alpha graph: In fact, we have a one-to-one-correspondence between the cuts resp. the occurrences of propositional variables in both diagrams such that a cut or an occurrence of a propositional variable is enclosed by another cut in the first diagram if and only if this holds for their counterpart in the second diagram. For example, in both diagrams of Figure 2 the outermost cut encloses exactly all other cuts, one occurrence of the propositional variable P_2 and all occurrences of the propositional

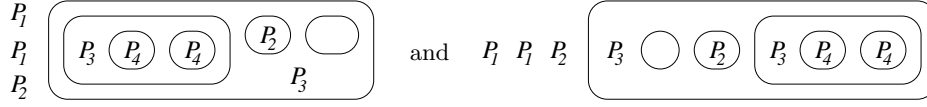


Fig. 2. Two Diagrams of the same Alpha graph

variables P_3 and P_4 . We will say more specifically that the occurrence of the propositional variable P_2 and the two cuts are enclosed *directly*, while all other enclosed items are enclosed *indirectly*, and we will say that the items which are directly enclosed by a cut are placed in the *area* of this cut. It is convenient to say that the outermost items (the outermost cut, the two occurrences of P_1 and one occurrence of P_2 are placed on the area of the sheet of assertion. This enclosing-relation is the main structural relation between cuts and the other items of a graph.

A mathematical definition of Alpha graphs must therefore capture the following:

1. A set of occurrences of propositional variables,
2. a set of cuts, and
3. the above-mentioned enclosing-relation.

To distinguish between propositional variables and occurrences of propositional variables, we will introduce *vertices* which are labelled with propositional variables. This yields the possibility that a propositional variable may occur several times in a graph, even in the same cut. The cuts are introduced as elements of a set Cut . It is reasonable to introduce the sheet of assertion as own syntactical device. The enclosing-relation will be captured by a mapping *area*, which assigns to each cut (and to the sheet of assertion) the set of all other elements of the graph which are directly enclosed by the cut. We know that we have some restrictions for cuts, e.g. cuts may not overlap. These restrictions are captured mathematically by conditions for the mapping *area*. A possible definition for formal Alpha graphs is the following:

Definition 5 (Formal Alpha Graph).

A formal Alpha graph is a 5-tuple $(V, \top, Cut, area, \kappa)$ with

- V and Cut are disjoint, finite sets whose elements are called vertices and cuts, respectively,
- \top is a single element with $\top \notin V \cup Cut$, called the sheet of assertion,
- $area : Cut \cup \{\top\} \rightarrow \mathfrak{P}(V \cup Cut)$ is a mapping such that
 - a) $c_1 \neq c_2 \Rightarrow area(c_1) \cap area(c_2) = \emptyset$,
 - b) $V \cup E \cup Cut = \bigcup_{d \in Cut \cup \{\top\}} area(d)$,
 - c) $c \notin area^n(c)$ for each $c \in Cut \cup \{\top\}$ and $n \in \mathbb{N}$ (with $area^0(c) := \{c\}$ and $area^{n+1}(c) := \bigcup \{area(d) \mid d \in area^n(c)\}$), and
- $\kappa : V \rightarrow \mathcal{P}$ is a mapping.

The elements of $Cut \cup \{\top\}$ are called contexts. As we have for every $x \in V \cup Cut$ exactly one context c with $x \in area(c)$, we can write $c = area^{-1}(x)$ for every $x \in area(c)$, or even more simple and suggestive: $c = ctx(x)$.

In this definition, we have already captured some important technical terms. Mainly we have defined the *sheet of assertion* and the *cuts* of formal Alpha graphs, and we have introduced the informal described mapping *area*. In the following, we will often speak more simply of ‘graphs’ instead of ‘formal Alpha graphs’.

There is a crucial difference between formal Alpha graphs and most other languages of logic: Usually, the well-formed formulas of a language are built up inductively. In contrast to that, formal Alpha graphs are defined in one step. The structure of a formula in an inductively defined language is given by its inductive construction. Of course we know that Alpha graphs bear a structure as well: A cut of the graph may contain other cuts, but cuts may not intersect. Thus, for two (different) cuts, we have three possibilities: The first cut encloses the second one, the second cut encloses the first one, or the two cuts are incomparable. If we incorporate the sheet of assertion into this consideration, it has to be expected that this idea induces an order \leq on the contexts which should be a tree, having the sheet of assertion \top as greatest element. As we now have a mathematical definition of Alpha graphs, we must be able to *prove* this proposition.³

In the next definition, we define an ordering on the vertices and edges which will capture the enclosing-relation.

Definition 6 (Ordering on the Contexts).

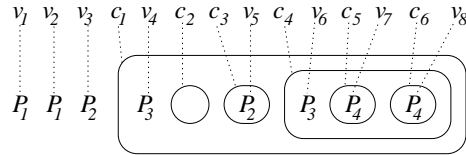
Let $\mathfrak{G} := (V, \top, Cut, area, \kappa)$ be a graph. We define a relation \leq on $Cut \cup \{\top\}$ as follows: $d \leq c := \iff d \in \bigcup_{n=0}^{\infty} area^n(c)$.

We set $x < y := \iff x \leq y \wedge y \not\leq x$ and $x \lesssim y := \iff x \leq y \wedge y \neq x$ to avoid misunderstandings. For $c \in Cut \cup \{\top\}$, we set $\leq[c] := \{x \in V \cup Cut \cup \{\top\} \mid x \leq c\}$ and $\lesssim[c] := \{x \in V \cup Cut \cup \{\top\} \mid x \lesssim c\}$. Every element x of $\lesssim[c]$ is said to be enclosed by c , and vice versa: c is said to enclose x . For every element of $area(c)$, we say more specifically that it is directly enclosed by c .

Analogously to [Da03], we get the following lemma:

Lemma 1. Let $\mathfrak{G} := (V, \top, Cut, area, \kappa)$ be a graph. Then \leq is an order on $Cut \cup \{\top\}$ which is a tree with the sheet of assertion \top as greatest element.

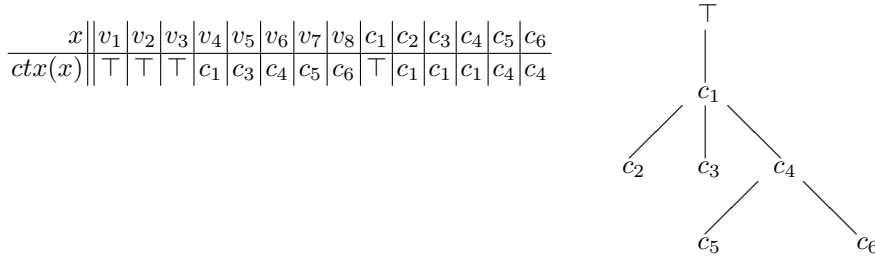
To provide an example of a formal Alpha graph, we consider the right diagram of Figure 2. Additionally, we have labelled the vertices and cuts with names for pairwise distinct elements. Below the diagram, the formal Alpha graph which is represented by the diagram is provided.



$$\mathfrak{G} := \left(\begin{array}{l} \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}, \top, \{c_1, c_2, c_3, c_4, c_5, c_6\}, \\ \{(\top, \{v_1, v_2, v_3, c_1\}), (c_1, \{v_4, c_2, c_3, c_4\}), (c_2, \emptyset), \\ (c_3, \{v_5\}), (c_4, \{v_6, c_5, c_6\}), (c_5, \{v_7\}), (c_6, \{v_8\})\}, \\ \{(v_1, P_1), (v_2, P_1), (v_3, P_2), (v_4, P_3), \\ (v_5, P_2), (v_6, P_3), (v_7, P_4), (v_8, P_4)\} \end{array} \right. \begin{array}{l} \left. \vphantom{\mathfrak{G}} \right\} V, \top, Cut \\ \left. \vphantom{\mathfrak{G}} \right\} area \\ \left. \vphantom{\mathfrak{G}} \right\} \kappa \end{array}$$

³ As we mathematize *informal* given entities (here: Alpha graphs), we cannot *prove* (mathematically) that the mathematization, e.g. the definition of formal Alpha graphs, is 'right'. So the attempt to prove that we have an induced tree on the context can be understood as a test on our mathematical 're-engineering' of Alpha graphs. If we cannot prove this proposition, our mathematization of Alpha graphs does not capture crucial features of Alpha graphs, thus we should rework the definition. If we can prove this proposition, this is a good argument that our definition is 'right'.

Next we show the mapping ctx , and the quasiorder \leq is presented by its Hasse-diagram.



Note that we did not further specify the objects v_1, \dots, v_8 , \top , and c_1, \dots, c_6 . It is quite obvious that a diagram of an EG cannot determine the mathematical objects for vertices or cuts, it only determines the *relationships* between these objects. In fact we can choose arbitrary sets for these mathematical objects (we only have to take into account that $\{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$, $\{\top\}$, $\{c_1, c_2, c_3, c_4, c_5, c_6\}$ must be pairwise disjoint). In other words: The diagram of a graph determines the graph only up to *isomorphism*. The isomorphism-relation is canonically defined as follows:

Definition 7 (Isomorphism).

Let $\mathfrak{G}_i := (V_i, \top_i, Cut_i, area_i, \kappa_i)$, $i = 1, 2$ be two formal Alpha graphs. Then we call $f = f_V \cup f_{Cut}$ isomorphism, if $f_V : V_1 \rightarrow V_2$ and $f_{Cut} : Cut_1 \cup \{\top_1\} \rightarrow Cut_2 \cup \{\top_2\}$ are bijective with $f_{Cut}(\top_1) = \top_2$ such that $f[area_1(c)] = area_2(f(c))$ for each $c \in Cut_1 \cup \{\top_1\}$ (we write $f[X]$ for $\{f(x) \mid x \in X\}$), and $\kappa_1(v) = \kappa_2(f_V(v))$ for all $v \in V_1$.

From now on, isomorphic graphs are implicitly identified.

We have seen that a *diagram of an Alpha graph* is a diagram which is built up from two different kinds of items, namely of closed, double-point-free and smooth curves which represent cuts (we will call them *cut-lines*), and signs which denote the propositional variables P_i , such that two different items of the diagram neither overlap nor intersect.

Of course each formal Alpha graph can be represented by an appropriate diagram. This diagram can be constructed iteratively over the tree of its context, using Lem. 1. Let on the other hand a diagram be given. It is easy to find (up to isomorphism) the corresponding formal Alpha graph $(V, \top, Cut, area, \kappa)$: We choose sets V and Cut of which its elements shall stand for the occurrences of propositional variables resp. the cut-lines in the diagram, and the mapping κ is defined accordingly. Then, the mapping *area* is now defined as follows: Let $c \in Cut$ be a cut. So we have a uniquely given cut-line cl in our diagram which corresponds to c . Now let $area(c)$ be the set of all $x \in V \cup Cut$ such that x corresponds to an item of the diagram (an occurrence of a PV or a cut-line) which is directly enclosed by the cut-line cl . Furthermore, let $area(\top)$ be the set of all $x \in V \cup Cut$ such that x corresponds to an item which is not enclosed by any cut-line. So we obtain a formal Alpha graph which is represented by the diagram. (The correspondence between formal Alpha graphs and diagrams can be much more explicated, but this is omitted due to space limitation).

Next we will define mathematically what a *subgraph* of a formal Alpha graph is.

Definition 8 (Subgraph).

Let $\mathfrak{G} := (V, \top, Cut, area, \kappa)$ be a graph. The graph $\mathfrak{G}' := (V', \top', Cut', area', \kappa')$ is called a subgraph of \mathfrak{G} in the context \top' if

- $V' \subseteq V$, $Cut' \subseteq Cut$ and $\top' \in Cut \cup \{\top\}$,
- $area'(\top') = area(\top') \cap (V' \cup Cut')$ and $area'(d) = area(d)$ for each $d \in Cut'$,
- $ctx(x) \in Cut' \cup \{\top'\}$ for each $x \in V' \cup Cut'$, and
- the mapping κ' is the restriction of κ to V' .

We write: $\mathfrak{G}' \subseteq \mathfrak{G}$ and $area^{-1}(\mathfrak{G}') = \top'$ resp. $ctx(\mathfrak{G}') = \top'$.

We provide some examples for this definition, based on the graph of Figure 2. We consider substructures of this graphs which are shaded in the diagram.



Fig. 3. Different subgraphs



Fig. 4. Two substructures which are no subgraphs

If we consider a graph and a context of this graph, then this context together with all cuts and vertices enclosed by this context form a subgraph. This is a special case for subgraphs which is captured by the following definition and lemma.

Definition 9 (Subgraph induced by a Cut).

Let $\mathfrak{G} := (V, \top, Cut, area, \kappa)$ be a formal Alpha graph and $c \in Cut \cup \{\top\}$ be a context. The graph $\mathfrak{G}[c] := (V', \top', Cut', area', \kappa')$ is defined as follows: $V' := \leq[c] \cap V$, $Cut' := \leq[c] \cap Cut$, $\top' := c$, $area' := area|_{Cut' \cup \{\top'\}}$, and $\kappa' := \kappa|_{V'}$.

Lemma 2. If $\mathfrak{G} := (V, \top, Cut, area, \kappa)$ is a formal graph and $c \in Cut \cup \{\top\}$ is a context, then $\mathfrak{G}[c]$ is a subgraph of \mathfrak{G} .

Proof: Trivial.

The next two definitions are necessary for defining and understanding the calculus we will present in Sec. 8. Most of the rules in this calculus modify only parts of a graph which are enclosed by a specific context. For some rules we have to distinguish whether this context is enclosed by an odd or even number of cuts. For this reason Definition 10 is needed. As we modify the graph only ‘inside’ the specific context, the part of the graph outside of this context remains unchanged. We will say that the starting graph and the resulting graph are isomorphic except for the context, which is captured by Definition 11.

Definition 10 (Even and Oddly Enclosed, Pos. and Neg. Contexts).

Let $\mathfrak{G} = (V, \top, Cut, area, \kappa)$ be a formal Alpha graph, let x be a subgraph or let x be an element of $V \cup Cut \cup \{\top\}$. We set $n := |\{c \in Cut \mid x \in \leq[c]\}|$. If n is even, x is said to be evenly enclosed, otherwise x is said to be oddly enclosed.

The sheet of assertion \top and each oddly enclosed cut is called a positive context, and each an evenly enclosed cut is called negative context.

Definition 11 (Partial Isomorphism).

For $i = 1, 2$, let $\mathfrak{G}_i := (V_i, \top_i, \text{Cut}_i, \text{area}_i, \kappa_i)$, be two formal Alpha graphs and let $c_i \in \text{Cut}_i \cup \{\top_i\}$ be given contexts. For $i = 1, 2$, we set $V'_i := \{v \in V_i \mid v \not\leq c_i\}$ and $\text{Cut}'_i := \{d \in \text{Cut}_i \cup \{\top_i\} \mid d \not\leq c_i\}$. Then $f = f_V \dot{\cup} f_{\text{Cut}}$ is called isomorphism except for $c_1 \in \text{Cut}_1 \cup \{\top_1\}$ and $c_2 \in \text{Cut}_2 \cup \{\top_2\}$ if $f_V : V'_1 \rightarrow V'_2$ and $f_{\text{Cut}} : \text{Cut}'_1 \rightarrow \text{Cut}'_2$ are bijective with $f_{\text{Cut}}(\top_1) = \top_2$, such that $f[\text{area}(c)] = \text{area}'(f(c))$ for each $c \in \text{Cut}_1 \cup \{\top_1\}'$, and $\kappa_1(v) = \kappa_2(f_V(v))$ for all $v \in V'_1$.

A common operation on diagrams is to *juxtapose* them, i.e. writing them side by side. On the side of the mathematical structure, this corresponds to the disjoint union of a set of graphs, which is captured by the next definition.

Definition 12 (Juxtaposition of Formal Alpha Graphs).

Let $\mathfrak{G}_i := (V_i, \top_i, \text{Cut}_i, \text{area}_i, \kappa_i)$ be graphs for $i = 1, \dots, n$ with $n \in \mathbb{N}_0$. The juxtaposition of the \mathfrak{G}_i is defined to be the following graph $\mathfrak{G} := (V, \top, \text{Cut}, \text{area}, \kappa)$:

- $V := \bigcup_{i=1, \dots, n} V_i \times \{i\}$,
- $\text{Cut} := \bigcup_{i=1, \dots, n} \text{Cut}_i \times \{i\}$,
- area is defined as follows: $\text{area}((c, i)) = \text{area}_i(c) \times \{i\}$ for $c \in \text{Cut}_i$, and $\text{area}(\top) = \bigcup_{i=1, \dots, n} \text{area}_i(\top_i) \times \{i\}$,
- $\kappa(v, i) := \kappa_i(v)$ for all $v \in VE$ and $i = 1, \dots, n$.

In the graphical notation, the juxtaposition of the \mathfrak{G}_i is simply noted by writing the graphs next to each other, i.e. we write: $\mathfrak{G}_1 \mathfrak{G}_2 \dots \mathfrak{G}_n$.

It should be noted that the juxtaposition of an empty set of graphs is allowed, too. It yields the empty graph, i.e. $(\emptyset, \top, \emptyset, \emptyset, \emptyset)$.

8 Semantics and Calculus for Formal Alpha Graphs

We start this section with providing a semantics for formal Alpha graphs. As in propositional logic, we assign truth values to the propositional variables by *valuations*. Propositional variables stand for propositions which are simply true or false.

Definition 13 (Valuation). A valuation is a mapping $\text{val} : \mathcal{P} \rightarrow \{\mathbf{ff}, \mathbf{tt}\}$.

Now we have to extend valuations to graphs by reflection of the meaning of cuts and juxtaposition. This is done close to the so-called *endoporeutic method* of Peirce.

Definition 14 (Evaluations).

Let val be a valuation and let $\mathfrak{G} := (V, \top, \text{Cut}, \text{area}, \kappa)$ be a graph. We evaluate \mathfrak{G} for val inductively over $c \in \text{Cut} \cup \{\top\}$. The evaluation of \mathfrak{G} in a context c is written $\text{val} \models \mathfrak{G}[c]$, and it is inductively defined as follows: $\text{val} \models \mathfrak{G}[c] : \Leftrightarrow$

- $\text{val}(\kappa(v)) = \mathbf{tt}$ for each $v \in V \cap \text{area}(c)$ (vertex condition), and
- $\text{val} \not\models \mathfrak{G}[c']$ for each $c' \in \text{Cut} \cap \text{area}(c)$ (cut condition: iteration over $\text{Cut} \cup \{\top\}$)

For $\text{val} \models \mathfrak{G}[\top]$ we write $\text{val} \models \mathfrak{G}$ and say that \mathfrak{G} is valid for val resp. val is a model for \mathfrak{G} . If we have graphs $\mathfrak{G}_1, \mathfrak{G}_2$ such that $\text{val} \models \mathfrak{G}_2$ for each valuation val , we write $\mathfrak{G}_1 \models \mathfrak{G}_2$.

Next, the calculus for formal Alpha graphs will be provided. The rules have already been informally presented in the introduction. We have now the possibility to describe the rules in a mathematically precise manner. Here are the appropriate mathematical definitions:

– **erasure and insertion**

We first provide a general definition for inserting and erasing a subgraph.

Let $\mathfrak{G} := (V, \top, \text{Cut}, \text{area}, \kappa)$ be a graph which contains the subgraph $\mathfrak{G}_0 := (V_0, \top_0, \text{Cut}_0, \text{area}_0, \kappa_0)$. Let $\mathfrak{G}' := (V', \top', \text{Cut}', \text{area}', \kappa')$ be defined as follows:

- $V' := V \setminus V_0$, $\top' := \top$ and $\text{Cut}' := \text{Cut} \setminus \text{Cut}_0$,
- $\text{area}'(d) := \begin{cases} \text{area}(d) & d \neq \top_0 \\ \text{area}(d) \setminus (V_0 \cup \text{Cut}_0) & d = \top_0 \end{cases}$
- $\kappa' := \kappa|_{V'}$

Then we say that \mathfrak{G}' is derived from \mathfrak{G} by *erasing the subgraph \mathfrak{G}_0 from the context \top_0* , and \mathfrak{G} is derived from \mathfrak{G}' by *inserting the graph \mathfrak{G}_0 into the context \top_0* . The rules ‘erasure’ and ‘insertion’ are restrictions of the definition above:

Let \mathfrak{G} be a graph and let \mathfrak{G}_0 be a subgraph of \mathfrak{G} with $c := \text{ctx}(\mathfrak{G}_0)$, and let \mathfrak{G}' be obtained from \mathfrak{G} by erasing \mathfrak{G}_0 from the context c . If c is positive, then \mathfrak{G}' is derived from \mathfrak{G} by *erasing \mathfrak{G}_0 from a positive context*, and if c is negative, then \mathfrak{G}' is derived from \mathfrak{G} by *inserting \mathfrak{G}_0 into a negative context*.

– **iteration and deiteration**

Let $\mathfrak{G} := (V, \top, \text{Cut}, \text{area}, \kappa)$ be a graph which contains the subgraph $\mathfrak{G}_0 := (V_0, \top_0, \text{Cut}_0, \text{area}_0, \kappa_0)$, and let c be a context with $c \notin \text{Cut}_0$.

Let $\mathfrak{G}' := (V', \top', \text{Cut}', \text{area}', \kappa')$ be the following graph:

- $V' := V \times \{1\} \cup V_0 \times \{2\}$, $\top' := \top$ and $\text{Cut}' := \text{Cut} \times \{1\} \cup \text{Cut}_0 \times \{2\}$.
- area' is defined as follows:
for $(d, i) \in \text{Cut}'$ and $d \neq c$ let $\text{area}'((d, i)) := \text{area}(d) \times \{i\}$, and let
 $\text{area}'((c, 1)) := \text{area}(c) \times \{1\} \cup \text{area}_0(\top_0) \times \{2\}$.
- $\kappa'((k, i)) := \kappa(k)$ for all $(k, i) \in V'$

Then we say that \mathfrak{G}' is derived from \mathfrak{G} by *iterating the subgraph \mathfrak{G}_0 into the context c* and \mathfrak{G} is derived from \mathfrak{G}' by *deiterating the subgraph \mathfrak{G}_0 from the context c* .

– **double cuts**

Let $\mathfrak{G} := (V, \top, \text{Cut}, \text{area}, \kappa)$ be a graph and $c_1, c_2 \in \text{Cut}$ with $\text{area}(c_1) = \{c_2\}$. Let $c_0 := \text{ctx}(c_1)$ (i.e., $c_1 \in \text{area}(c_0)$) and set $\mathfrak{G}' := (V, \top, \text{Cut}', \text{area}', \kappa)$ with

- $\text{Cut}' := \text{Cut} \setminus \{c_1, c_2\}$
- $\text{area}'(d) := \begin{cases} \text{area}(d) & \text{for } d \neq c_0 \\ \text{area}(c_0) \cup \text{area}(c_2) & \text{for } d = c_0 \end{cases}$

Then we say that \mathfrak{G}' is derived from \mathfrak{G} by *erasing the double cuts c_1, c_2* and \mathfrak{G} is derived from \mathfrak{G}' by *inserting the double cuts c_1, c_2* .

Based on the calculus, we can now define the syntactical entailment relation.

Definition 15. Let $\mathfrak{G}_a, \mathfrak{G}_b$ be two graphs. Then \mathfrak{G}_b can be derived from \mathfrak{G}_a (which is written $\mathfrak{G}_a \vdash \mathfrak{G}_b$), if there is a finite sequence $(\mathfrak{G}_1, \mathfrak{G}_2, \dots, \mathfrak{G}_n)$ with $\mathfrak{G}_a = \mathfrak{G}_1$ and $\mathfrak{G}_b = \mathfrak{G}_n$ such that each \mathfrak{G}_{i+1} is derived from \mathfrak{G}_i by applying one of the rules of the calculus. The sequence is called a proof for $\mathfrak{G}_a \vdash \mathfrak{G}_b$. Two graphs $\mathfrak{G}_1, \mathfrak{G}_2$ with $\mathfrak{G}_1 \vdash \mathfrak{G}_2$ and $\mathfrak{G}_2 \vdash \mathfrak{G}_1$ are said to be provably equivalent.

If $\mathfrak{H} := \{\mathfrak{G}_i \mid i \in I\}$ is a (possibly empty) set of graphs, then a graph \mathfrak{G} can be derived from \mathfrak{H} if there is a finite subset $\{\mathfrak{G}_1, \dots, \mathfrak{G}_n\} \subseteq \mathfrak{H}$ with $\mathfrak{G}_1 \dots \mathfrak{G}_n \vdash \mathfrak{G}$ (remember that $\mathfrak{G}_1 \dots \mathfrak{G}_n$ is the juxtaposition of $\mathfrak{G}_1, \dots, \mathfrak{G}_n$).

Before we start with the proof that the calculus is sound and complete, we finally show in this section three simple metalemmata in the sense that they show some *schemata* for proofs with EGs, i.e., they are derived ‘macro’-rules.

All rules in the calculus which are applied in a context only depend on whether the context is positive or negative. In particular if a proof for $\mathfrak{G}_a \vdash \mathfrak{G}_b$ is given, this proof can be carried out in arbitrary positive contexts. This yields immediately the following lemma, which can be found in [So97].

Lemma 3 (Cut-And-Paste-Theorem).

Let $\mathfrak{G}_a \vdash \mathfrak{G}_b$ for two graphs $\mathfrak{G}_a, \mathfrak{G}_b$. It follows:

- If \mathfrak{G}_a is a subgraph of \mathfrak{G} in a pos. context, then \mathfrak{G}_a may be replaced by \mathfrak{G}_b .
- If \mathfrak{G}_b is a subgraph of \mathfrak{G} in a neg. context, then \mathfrak{G}_b may be replaced by \mathfrak{G}_a .

In particular we have that derivable graphs \mathfrak{G}_0 (i.e., graphs with $\vdash \mathfrak{G}_0$) can be inserted into arbitrary contexts of arbitrary graphs.

From this lemma we obtain the graph version of the well known deduction theorem.

Lemma 4 (Deduction Theorem).

Let $\mathfrak{G}_a, \mathfrak{G}_b$ be graphs. Then $\mathfrak{G}_a \vdash \mathfrak{G}_b \iff \vdash \mathfrak{G}_a \mathfrak{G}_b$

Proof: We show both directions separately.

$$\text{‘}\implies\text{’}: \quad \text{dc} \quad \text{ins} \quad \text{it} \quad \text{L.3}$$

$$\text{‘}\impliedby\text{’}: \quad \text{L.3} \quad \text{deit} \quad \text{dc} \quad \text{era} \quad \square$$

The following lemma is quite obvious:

Lemma 5. Let $\mathfrak{G}, \mathfrak{G}_a, \mathfrak{G}_b$ be graphs with $\mathfrak{G} \vdash \mathfrak{G}_a$ and $\mathfrak{G} \vdash \mathfrak{G}_b$. Then $\mathfrak{G} \vdash \mathfrak{G}_a \mathfrak{G}_b$.

Proof: $\mathfrak{G} \vdash \mathfrak{G} \mathfrak{G} \xrightarrow{\text{it}} \mathfrak{G} \vdash \mathfrak{G}_a \mathfrak{G} \xrightarrow{\text{L.3}} \mathfrak{G} \vdash \mathfrak{G}_a \mathfrak{G}_b \quad \square$

9 Soundness and Completeness

In this chapter we will show that the rules we presented in Sec. 8 are sound and complete with respect to the given semantics. In the first section, we will prove the soundness of the calculus, in the next section we will prove its completeness.

9.1 Soundness

Most of the rules modify only the area of one specific context c (for example, the rule ‘erasure’ removes a subgraph from the area of a positive context). If a graph \mathfrak{G}' is derived from a graph \mathfrak{G} by applying one of these rules (i.e., by modifying a context c in the graph \mathfrak{G}), \mathfrak{G} and \mathfrak{G}' are isomorphic except for c . As it has to be shown that no rule can transform a valid graph into a nonvalid one, the following theorem is the basis for proving the soundness of most rules.

Theorem 1 (Main Lemma for Soundness).

Let $\mathfrak{G} := (V, \top, \text{Cut}, \text{area}, \kappa)$ and $\mathfrak{G}' := (V', \top', \text{Cut}', \text{area}', \kappa')$ be graphs and let $f = f_V \cup f_{\text{Cut}}$ be an isomorphism from \mathfrak{G} to \mathfrak{G}' except for $c \in \text{Cut} \cup \{\top\}$ and $c' \in \text{Cut}' \cup \{\top'\}$. Let $\text{val} : \mathcal{P} \mapsto \{\mathbf{ff}, \mathbf{tt}\}$ be a valuation. Let $P(d)$ be the following property for Cuts $d \in \text{Cut} \cup \{\top\}$:

- If d is positive and $\text{val} \models \mathfrak{G}[d]$, then $\text{val} \models \mathfrak{G}'[f(d)]$, and
- If d is negative and $\text{val} \not\models \mathfrak{G}[d]$, then $\text{val} \not\models \mathfrak{G}'[f(d)]$.

If P holds for c , then P holds for each $d \in \text{Cut} \cup \{\top\}$ with $d \not\prec c$. In particular $v \models \mathfrak{G}'$ follows from $v \models \mathfrak{G}$.

Proof: We set $D := \{d \in \text{Cut} \cup \{\top\} \mid d \not\prec c\}$. D is a tree such that for each $d \in D$ with $d \neq c$ and each $e \in \text{Cut} \cup \{\top\}$ with $e < d$ we have $e \in D$. For this reason we can carry out the proof by induction over D . As c satisfies P , it is sufficient to carry out the induction step for $d \neq c$. So let $d \in D$, $d \neq c$ be a context such that $P(e)$ holds for all cuts $e \in \text{area}(d) \cap \text{Cut}$.

First Case: d is positive and $\text{val} \models \mathfrak{G}[d]$.

We have to check the vertex- and cut-conditions for $f(d)$. We start with the vertex conditions for $f(d)$, i.e., for vertices $v' \in V'$ with $\text{ctx}'(v') = f(d)$.

For each $v \in V$ with $\text{ctx}(v) = d$, it holds $\kappa(v) = \kappa'(f(v))$, hence

$$\text{val}(\kappa(v)) = \mathbf{tt} \iff \text{val}(\kappa'(f(v))) = \mathbf{tt}.$$

As f_V is a bijection from $\text{area}(d) \cap V$ to $\text{area}'(f(d)) \cap V'$, we gain the following: All vertex conditions in d hold iff all vertex conditions in $f(d)$ hold.

As we have $\text{val} \models \mathfrak{G}[d]$, we get that $\text{val} \not\models \mathfrak{G}[e]$ for all cuts $e \in \text{area}(d)$. These cuts are negative and are mapped bijectively to the cuts $e' \in \text{area}(f(d))$. As they are negative, we conclude from the induction hypothesis or the presupposition (for $e = c$) that $\text{val} \not\models \mathfrak{G}'[f(e)]$ for all cuts $e \in \text{area}(d)$, i.e., $\text{val} \not\models \mathfrak{G}'[e']$ for all cuts $e' \in \text{area}'(f(d))$.

As we have checked all vertex- and cut-conditions for $f(d)$, we get $\text{val} \models \mathfrak{G}'[f(d)]$.

Second Case: d is negative and $\text{val} \not\models \mathfrak{G}[d]$.

This is shown analogously to the first case. □

With this lemma, we can prove the correctness of the rules. Due to space limitations, the prove will only be carried out for the rules ‘iteration’ and ‘deiteration’. The other rules can be handled similarly.

Lemma 6 (Iteration and Deiteration are Sound).

If \mathfrak{G} and \mathfrak{G}' are graphs, val is a valuation with $\text{val} \models \mathfrak{G}$ and \mathfrak{G}' is derived from \mathfrak{G} by applying one of the rules ‘iteration’ or ‘deiteration’, then $\text{val} \models \mathfrak{G}'$.

Proof: Let $\mathfrak{G}_0 := (V_0, \top_0, \text{Cut}_0, \text{area}_0, \kappa_0)$ be the subgraph of \mathfrak{G} which is iterated into the context $c \leq \text{ctx}(\mathfrak{G}_0)$, $c \notin \text{Cut}_0$. We use the mathematical notation which was given in Sec. 8. In particular, $(c, 1)$ is the context in \mathfrak{G}' which corresponds to the context c in \mathfrak{G} . There are two cases to consider:

First Case: $\text{val} \models \mathfrak{G}_0$. From this we conclude $\text{val} \models \mathfrak{G}[c] \iff \text{val} \models \mathfrak{G}'[(c, 1)]$. As \mathfrak{G} and \mathfrak{G}' are isomorphic except for $c \in \text{Cut} \cup \{\top\}$ and $(c, 1) \in \text{Cut}' \cup \{\top'\}$, Lem. 1 can be applied now. This yields

$$\text{val} \models \mathfrak{G} \iff \text{val} \models \mathfrak{G}' . \tag{*}$$

Second Case: $val \not\models \mathfrak{G}_0$. This yields $val \not\models \mathfrak{G}[\top_0]$ and $val \not\models \mathfrak{G}'[(\top_0, 1)]$. As \mathfrak{G} and \mathfrak{G}' are isomorphic except for $\top_0 \in Cut \cup \{\top\}$ and $(\top_0, 1) \in Cut' \cup \{\top'\}$, Lemma 1 can be applied now. This yields again (*).

The direction ‘ \implies ’ of (*) yields the correctness of the iteration-rule. The opposite direction ‘ \impliedby ’ of (*) yields the correctness of the deiteration-rule. \square

Lemma 7 (Erasure and Insertion are Sound).

If \mathfrak{G} and \mathfrak{G}' are graphs, v is a valuation with $val \models \mathfrak{G}$ and \mathfrak{G}' is derived from \mathfrak{G} by applying one of the rules ‘erasure’ or ‘insertion’, then $val \models \mathfrak{G}'$.

Lemma 8 (Double Cut is Sound).

If \mathfrak{G} and \mathfrak{G}' are graphs, val is a valuation with $val \models \mathfrak{G}$ and \mathfrak{G}' is derived from \mathfrak{G} by applying the rule ‘double cut’, then $val \models \mathfrak{G}'$.

From the preceding lemmata the soundness of the calculus follows immediately:

Theorem 2 (Soundness of the Alpha-Calculus).

Two formal Alpha graphs \mathfrak{G} , \mathfrak{G}' satisfy $\mathfrak{G} \vdash \mathfrak{G}' \implies \mathfrak{G} \models \mathfrak{G}'$.

9.2 Completeness

As the empty sheet of assertion is always true, the graph \square is always false. This leads to the following definition and lemmata.

Definition 16. *A set \mathfrak{H} of graphs is called consistent if $\mathfrak{H} \vdash \square$ does not hold. A graph \mathfrak{G} is called consistent if $\{\mathfrak{G}\}$ is consistent.*

Lemma 9. *A set \mathfrak{H} of graphs is not consistent if and only if $\mathfrak{H} \vdash \mathfrak{G}'$ for each graph \mathfrak{G}' .*

Proof: Only the direction ‘ \implies ’ has to be proven. So let $\mathfrak{G}_1, \dots, \mathfrak{G}_n \in \mathfrak{H}$ with $\mathfrak{G}_1 \dots \mathfrak{G}_n \vdash \square$. Let \mathfrak{G} be the juxtaposition of $\mathfrak{G}_1, \dots, \mathfrak{G}_n$. We conclude:

$$\mathfrak{G} \vdash \square \xLeftrightarrow{\text{T.4}} \vdash \left(\mathfrak{G} \begin{array}{c} \circ \\ \circ \end{array} \right) \xLeftrightarrow{\text{dc}} \vdash \left(\mathfrak{G} \right) \xRightarrow{\text{ins}} \vdash \left(\mathfrak{G} \begin{array}{c} \mathfrak{G}' \\ \mathfrak{G}' \end{array} \right) \xLeftrightarrow{\text{T.4}} \mathfrak{G} \vdash \mathfrak{G}' \xLeftrightarrow{\text{Def.}\vdash} \mathfrak{H} \vdash \mathfrak{G}' \quad \square$$

Lemma 10. *Let \mathfrak{G} be a graph and let \mathfrak{H} be a set of graphs. Then we have*

$$\mathfrak{H} \vdash \mathfrak{G} \iff \mathfrak{H} \cup \{ \left(\mathfrak{G} \right) \} \vdash \square \quad \text{and} \quad \mathfrak{H} \vdash \left(\mathfrak{G} \right) \iff \mathfrak{H} \cup \{ \mathfrak{G} \} \vdash \square \quad .$$

Proof of the first equivalence (the second is shown analogously):

$$\begin{aligned} \mathfrak{H} \vdash \mathfrak{G} &\xLeftrightarrow{\text{Def.}\vdash} \text{there are } \mathfrak{G}_1, \dots, \mathfrak{G}_n \in \mathfrak{H} \text{ with } \mathfrak{G}_1 \dots \mathfrak{G}_n \vdash \mathfrak{G} \\ &\xLeftrightarrow{\text{T.4}} \text{there are } \mathfrak{G}_1, \dots, \mathfrak{G}_n \in \mathfrak{H} \text{ with } \vdash \left(\mathfrak{G}_1 \dots \mathfrak{G}_n \begin{array}{c} \mathfrak{G} \\ \mathfrak{G} \end{array} \right) \\ &\xLeftrightarrow{\text{dc}} \text{there are } \mathfrak{G}_1, \dots, \mathfrak{G}_n \in \mathfrak{H} \text{ with } \vdash \left(\mathfrak{G}_1 \dots \mathfrak{G}_n \begin{array}{c} \mathfrak{G} \\ \square \end{array} \right) \\ &\xLeftrightarrow{\text{T.4}} \text{there are } \mathfrak{G}_1, \dots, \mathfrak{G}_n \in \mathfrak{H} \text{ with } \mathfrak{G}_1 \dots \mathfrak{G}_n \left(\mathfrak{G} \right) \vdash \square \\ &\xLeftrightarrow{\text{Def.}\vdash} \mathfrak{H} \cup \{ \left(\mathfrak{G} \right) \} \vdash \square \quad \square \end{aligned}$$

Consistent sets of graphs can be extended to maximal (with respect to \subseteq) consistent sets graphs, which have canonically given valuations satisfying them. This will be elaborated with the next lemmata.

Lemma 11. *Let \mathfrak{H} be a maximal consistent set of graphs. Then:*

1. *Either $\mathfrak{H} \vdash \mathfrak{G}$ or $\mathfrak{H} \vdash \overline{\mathfrak{G}}$ for each graph \mathfrak{G} .*
2. *$\mathfrak{H} \vdash \mathfrak{G} \iff \mathfrak{G} \in \mathfrak{H}$ for each graph \mathfrak{G} .*
3. *$\mathfrak{G}_1 \mathfrak{G}_2 \in \mathfrak{H} \iff \mathfrak{G}_1 \in \mathfrak{H}$ and $\mathfrak{G}_2 \in \mathfrak{H}$ for all graphs $\mathfrak{G}_1, \mathfrak{G}_2$.*

Proof: It is easy to see that $\mathfrak{G} \overline{\mathfrak{G}} \vdash \square$. Hence if \mathfrak{H} is consistent, $\mathfrak{H} \vdash \mathfrak{G}$ and $\mathfrak{H} \vdash \overline{\mathfrak{G}}$ cannot hold both. Now we can prove all propositions of this lemma:

1. Assume $\mathfrak{H} \not\vdash \mathfrak{G}$ for a graph \mathfrak{G} . Lem. 10 yields that $\mathfrak{H} \cup \{\overline{\mathfrak{G}}\}$ is consistent. As \mathfrak{H} is maximal, we conclude that $\overline{\mathfrak{G}} \in \mathfrak{H}$, hence we have $\mathfrak{H} \vdash \overline{\mathfrak{G}}$.
2. Let $\mathfrak{H} \vdash \mathfrak{G}$. As \mathfrak{H} is consistent, we get that $\mathfrak{H} \not\vdash \overline{\mathfrak{G}}$. So Lem. 10 yields that $\mathfrak{H} \cup \{\mathfrak{G}\}$ is consistent. As \mathfrak{H} is maximal, we conclude $\mathfrak{G} \in \mathfrak{H}$.
3. Follows immediately from 1. and 2. □

Lemma 12. *Let \mathfrak{H} be a consistent set of graphs. Then there is a maximal set \mathfrak{H}' of graphs with $\mathfrak{H}' \supseteq \mathfrak{H}$.*

Proof: Let $\mathfrak{G}_1, \mathfrak{G}_2, \mathfrak{G}_3, \dots$ be an enumeration of all graphs. We define inductively a set of graphs \mathfrak{H}_i for each $i \in \mathbb{N}$. We start with setting $\mathfrak{H}_1 := \mathfrak{H}$. Assume now that $\mathfrak{H}_i \supseteq \mathfrak{H}$ is defined and consistent.

If $\mathfrak{H}_i \vdash \overline{\mathfrak{G}_i}$ does not hold, then $\mathfrak{H}_{i+1} := \mathfrak{H}_i \cup \{\mathfrak{G}_i\}$ is consistent due to Lem. 10.

Otherwise, if $\mathfrak{H}_i \vdash \mathfrak{G}_i$ holds, then $\mathfrak{H}_{i+1} := \mathfrak{H}_i \cup \{\overline{\mathfrak{G}_i}\}$ is consistent.

Now $\mathfrak{H}' := \bigcup_{n \in \mathbb{N}} \mathfrak{H}'_n$ is obviously a consistent maximal graph set with $\mathfrak{H}' \supseteq \mathfrak{H}$. □

Theorem 3. *Let \mathfrak{H} be a maximal consistent set of graphs. Then there exists a canonically given valuation val such that $val \models \mathfrak{G}$ for each graph $\mathfrak{G} \in \mathfrak{H}$.*

Proof: Let us first define a graph which states the propositional variable P_i . We set $\mathfrak{G}(P_i) := (\{v\}, \top, \emptyset, \emptyset, \{(v, P_i)\})$ with an arbitrary vertex v . Let $val : \mathcal{P} \mapsto \{\mathbf{ff}, \mathbf{tt}\}$ be defined as follows: $val(P_i) := \mathbf{tt} : \iff \mathfrak{H} \vdash \mathfrak{G}(P_i)$.

Now let $\mathfrak{G}' := (V, \top, Cut, area, \kappa)$ be a formal Alpha graph. We show

$$val \models \mathfrak{G}'[c] \iff \mathfrak{H} \vdash \mathfrak{G}'[c] \quad (10)$$

for each $c \in Cut \cup \{\top\}$. The proof is done by induction over $Cut \cup \{\top\}$. So let $c \in Cut \cup \{\top\}$ be a cut such that (10) holds for each $d < c$. We have:

$$\begin{aligned} val \models \mathfrak{G}'[c] &\stackrel{\text{Def. evaluation}}{\iff} val(\kappa(v)) = \mathbf{tt} \text{ for each } v \in V \cap area(c) \\ &\quad \text{and } val \not\models \mathfrak{G}[d] \text{ for each } d \in Cut \cap area(d) \\ &\stackrel{\text{Def. val and Ind.Hyp.}}{\iff} \mathfrak{H} \vdash \mathfrak{G}(\kappa(v)) \text{ for each } v \in V \cap area(c) \\ &\quad \text{and } \mathfrak{H} \not\vdash \mathfrak{G}'[d] \text{ for each } d \in Cut \cap area(d) \\ &\stackrel{\text{L. 11}}{\iff} \mathfrak{G}(\kappa(v)) \in \mathfrak{H} \text{ for each } v \in V \cap area(c) \\ &\quad \text{and } \overline{\mathfrak{G}'[d]} \in \mathfrak{H} \text{ for each } d \in Cut \cap area(d) \\ &\stackrel{\text{L. 11}}{\iff} \mathfrak{G}[c] \in \mathfrak{H} \end{aligned}$$

As we have $\mathfrak{G} = \mathfrak{G}[\top]$, applying (10) to $c := \top$ yields $val \models \mathfrak{G}' \iff \mathfrak{H} \vdash \mathfrak{G}'$. □

Now we are prepared to prove the completeness of the calculus.

Theorem 4 (Completeness of the Calculus).

Two formal Alpha graphs $\mathfrak{G}_1, \mathfrak{G}_2$ satisfy $\mathfrak{G}_1 \models \mathfrak{G}_2 \implies \mathfrak{G}_1 \vdash \mathfrak{G}_2$.

Proof: Assume that $\mathfrak{G}_1 \vdash \mathfrak{G}_2$ does not hold. Then Cor. 10 yields that $\mathfrak{G}_1 \circledast \mathfrak{G}_2$ is consistent. According to the last lemma, let be \mathfrak{H} be a maximal consistent set of graphs which includes this graph. Due to Thm. 3, \mathfrak{H} has a canonically given valuation *val*. This valuation is in particular a model for $\mathfrak{G}_1 \circledast \mathfrak{G}_2$. So *v* is a model for \mathfrak{G}_1 , but not for \mathfrak{G}_2 , which is a contradiction to the assumption. \square

References

- [Ba93] John Barwise: Heterogenous reasoning, in G. W. Mineau, B. Moulin, J. F. Sowa (Eds.): *Conceptual Graphs for Knowledge Representation*. LNAI 699, Springer Verlag, Berlin–New York 2000, 64–74.
- [Bu91] R. W. Burch: *A Peircean Reduction Theses: The Foundations of Topological Logic*. Texas Tech University Press, 1991.
- [Da00] F. Dau: Negations in Simple Concept Graphs, in: B. Ganter, G. W. Mineau (Eds.): *Conceptual Structures: Logical, Linguistic, and Computational Issues*. LNAI 1867, Springer Verlag, Berlin–New York 2000, 263–276.
- [Da01] F. Dau: Concept Graphs and Predicate Logic, in: H. S. Delugach, G. Stumme (Eds.): *Conceptual Structures: Broadening the Base*. LNAI 2120, Springer Verlag, Berlin–New York 2001, 72–86.
- [Da02] F. Dau: *An Embedding of Existential Graphs into Concept Graphs with Negations*. In: D. Corbett, U. Priss (Eds.): *Conceptual Structures: Integration and Interfaces*, LNAI, Springer Verlag, Berlin–Heidelberg 2002.
- [Da03] F. Dau: *The Logic System of Concept Graphs with Negation (And Its Relationship to Predicate Logic)*. LNAI, Vol. 2892, Springer, Berlin–Heidelberg–New York, 2003.
- [Ha95] E. M. Hammer, *Logic and Visual Information*. CSLI Publications, Stanford, California, 1995.
- [Ha98] E. M. Hammer, *Semantics for Existential Graphs*. *Journal Philosophical Logic*, Vol. 27, 1998, 489–503.
- [HS02] J. Howse, F. Molina, S. Shin, J. Taylor: *On Diagram Tokens and Types*
- [Pe98] C. S. Peirce: *Reasoning and the Logic of Things*. The Cambridge Conferences Lectures of 1898. Ed. by K. L. Kremer, Harvard Univ. Press, Cambridge 1992.
- [PS09] C. S. Peirce, J. F. Sowa: *Existential Graphs: MS 514 by Charles Sanders Peirce with commentary by John F. Sowa*
<http://users.bestweb.net/~sowa/peirce/ms514.htm>
- [Pe03] C. S. Peirce: *Existential graphs*. 1903. Partly published in the collected papers of Peirce. Complete german translation in:
Helmut Pape: *Charles Sanders Peirce: Phänomen und Logik der Zeichen*. Suhrkamp Taschenbuch Wissenschaft, 1983.
- [Ro73] D. D. Roberts: *The Existential Graphs of Charles S. Peirce*. Mouton, The Hague, Paris, 1973.
- [Ro92] D. D. Roberts: *The Existential Graphs*. *Computers Math. Applic.*, Vol. 23, No. 6–9, 1992, 639–63.
- [Sh94] S. Shin: *The Logical Status of Diagrams*. Cambridge University Press, 1994.
- [Sh99] S. Shin: *Reconstituting Beta Graphs into an Efficacious System*. *Journal of Logic, Language and Information*, Vol. 8, No. 3, July 1999.
- [Sh01] S. Shin: *The Iconic Logic of Peirce’s Graphs*. Bradford Book, Massachusetts, 2002.
- [So84] J. F. Sowa: *Conceptual Structures: Information Processing in Mind and Machine*. The System Programming Series. Adison-Wesley, Reading 1984.
- [So92] J. F. Sowa: *Conceptual Graphs Summary*, in: T. E. Nagle, J. A. Nagle, L. L. Gerholz, P. W. Eklund (Eds.): *Conceptual Structures: current research and practice*, Ellis Horwood, 1992, 3–51.
- [So97] J. F. Sowa: *Logic: Graphical and Algebraic*, Manuskript, Croton-on-Hudson 1997.