The Display Problem Revisited

Tyke Nunez[*]

Abstract

In this essay I give a complete join semi-lattice of possible display-equivalence schemes for display logic, using the standard connectives, and leaving fixed only the schemes governing the star. In addition to proving the completeness of this list, I offer a discussion of the basic properties of these schemes.

1 Introduction

In this essay I will build on the work begun in (Belnap, 1996). In his essay Belnap presented various options for how to set up the display equivalences of display logic, a refinement of Gentzen's sequent calculus.[1] Belnap describes most of the schemes I will deal with, although I will complete the list he presents, using his connectives, and leaving fixed only the schemes governing the star, as he does.[2]

Even so, I hope that having a complete lattice of display equivalence schemes will allow a more systematic understanding of the properties these give to the structural-connectives, whose meaning seems not to be very well understood. These properties will be very much like the properties introduced by structural rules, the other kind of rule in display logic governing the structural-connectives. The chief

[*]Thanks to audiences at Logica 2010 and the University of Pittsburgh, especially Nuel Belnap, Shawn Standefer, Kohei Kishida and Kathryn Lindeman, for comments on the presentation associated with this essay as well as this essay itself.

[1](Gentzen, 1969). Regrettably, I am unaware of an 'easy introduction' to display logic. Belnap's original paper (Belnap, 1982), and its subsequent slightly updated version in §62 of (Anderson, Jr, & Dunn, 1992) are, I think, the best places to start.

[2]Where it isn't a detriment to clarity I will take up Belnap's names for the schemes and skip over points he has already made.

difference between these rules and the equivalence schemes is that the latter must secure the display property, which is, roughly, that any structure can be 'displayed' alone as either the entire antecedent or consequent of a consecution display equivalent to the original one.[3] This property, the defining feature of display logic, sets a restriction on the possible schemes of display logic; securing this property is what Belnap dubs 'the display problem'.

2 Structural-connectives & star scheme

Display logic replaces Gentzen's polyvalent comma with a bivalent circle, $X \circ Y$. Like the comma, the $\circ$ means 'something like' conjunction on the left and 'something like' disjunction on the right of the turnstile. Display logic also has a single place star connective, $*X$, that allows one to flip structures from one side of the turnstile to the other and back again. Its meaning is often thought of as 'something like' negation. Although display logic has other structural connectives, this essay will focus on these two.

Strictly speaking, the generic un-indexed structural-connectives (like the un-indexed formula-connectives), are functions that map each family index (S4, r, e, h, b, etc.) into a specific structural connective of the family associated with that index. For the most part, however, I will suppress these indices and give generic un-indexed formulations of schemes of display equivalences. I hope the reader will not lose sight of them entirely, however, because much of the motivation for canvassing the possible schemes and their properties lies in the greater control this will afford logicians working in display logic over the basic properties they build into the connectives of their languages.

Every scheme I will deal with treats negative structuring, $*$, when it appears alone, in the same way. Star in each scheme has full contraposition and double star elimination. Consecutions on the same

[3]A more precise formulation of this property is that "each antecedent part X of a consecution S can be displayed as the antecedent (itself) of a display equivalent consecution $X \vdash W$; and the consequent W is determined only by the position of X in S, not by what X looks like. Similarly for consequent parts of S." (Anderson et al., 1992, p. 301)

line below are display equivalent (i.e. interderivable):

$$\begin{array}{cccc} X \vdash Y & *Y \vdash *X & X \vdash **Y & **X \vdash Y \\ X \vdash *Y & Y \vdash *X & & \\ *X \vdash Y & *Y \vdash X & & \\ \ldots X \ldots & \ldots **X \ldots & & \end{array}$$

The last line is intended to convey general intersubstitutability, which follows from the first line only, in the presence of the other schemes granting the display property. We can think of the scheme as a set of display equivalence classes. Since schemes are sets of classes, in order to make things easier on the eye I will use square brackets to mark equivalence classes and reserve curly brackets for other sets. For example, leaving out the last line, the scheme in the above table is:

$$\{\, [X \vdash Y,\ *Y \vdash *X,\ X \vdash **Y,\ **X \vdash Y], \\ [X \vdash *Y,\ Y \vdash *X], [*X \vdash Y,\ *Y \vdash X] \,\}^{4}$$

Consecutions of the form of one member of a class are interderivable with corresponding consecutions of the form of the other members of the class.

Although there is nothing essential about this treatment of negative structuring, having a connective that allows structure to be moved from one side of the turn-style to the other is of obvious use in securing the display property. In part this is because in every scheme (as well as every structural rule) antecedent [consequent] structures remain antecedent [consequent] parts, which guarantees that the same structure cannot be displayed both on the left and on the right.

3 The display problem

Because structural variables are schematic, restricting the components of the display equivalence classes only to antecedent parts does not result in a loss of generality, although it does prevent redundantly

[4] In what follows I will present the equivalence classes in set notation, but in order to make this notation more legible I will adapt Belnap's graphic method of presentation, in which consecutions on the same line are members of the same equivalence class. Although at the moment this may sound awkward, when it comes up below I think it will feel natural.

treating the same scheme in two different forms. To see this consider a consecution, $X \circ Y \vdash *Z$, composed entirely of antecedent parts and another version of the same consecution with consequent parts, e.g. $*X \circ Y \vdash *Z$. It is obvious that because each variable ranges over both X and $*X$, treating both consecutions in our schemes would lead to redundancy. As such, for simplicity and readability in the rest of the essay I will use only antecedent parts.[5]

What will differ between the schemes is how the star interacts with the binary circle. Since the largest arity connective in the formulation of display logic we are dealing with is binary, the schemes will at most contain three structural variables. Now given that I have proposed to deal only with star and circle and that the schemes for star alone, which involve only two variables, are presented above, what is left are the schemes involving three variables.

Formulating the consecutions only with antecedent parts, there are twelve that will be involved in the rest of our schemas. These are grouped by which variable is displayed:[6]

$Z(XY)$ group:	$X(YZ)$ group:	$Y(ZX)$ group:
(1) $Z \vdash *X \circ *Y$	(5) $X \vdash *Y \circ *Z$	(9) $Y \vdash *Z \circ *X$
(2) $Z \vdash *Y \circ *X$	(6) $X \vdash *Z \circ *Y$	(10) $Y \vdash *X \circ *Z$
(3) $Z \vdash *(X \circ Y)$	(7) $X \vdash *(Y \circ Z)$	(11) $Y \vdash *(Z \circ X)$
(4) $Z \vdash *(Y \circ X)$	(8) $X \vdash *(Z \circ Y)$	(12) $Y \vdash *(X \circ Z)$

We can think of these groups as sets.

With the exhaustion of the relevant consecutions in this list, the display problem becomes specified: each of the display equivalence classes in a scheme must have at least one member of each of the three groups.

An important related corollary of this condition that Belnap doesn't explicitly mention is that every complete scheme of display equivalences must include all twelve consecutions, although portions of the schemes can be largely independent of one another. This follows

[5]Belnap's reasons for treating only antecedent parts are different. (Belnap, 1996, p. 85)

[6]I have altered Belnap's numbering slightly because a more systematic numbering makes the relations between the schemes easier to notice. Belnap dubs these groups the $Z(XY)$, $X(YZ)$, $Y(ZX)$ 'families' (not to be confused with language families). To avoid the ambiguity, I'll instead use 'group'.

from the fact that if one of the twelve did not belong to an equivalence class, the display property would fail.

A re-lettering of a set of consecutions is obtained by exchanging every instance of a variable or variables with the same variable. Re-lettering will be an oft used strategy for figuring out how the schemes work and why the semi-lattice of schemes I am about to present is complete within the bounds set, so having a precise idea of it is important. Let $\alpha, \beta \in \{ X, Y, Z \}$ and $\alpha \neq \beta$. Then an α, β re-lettering of a set of consecutions will involve exchanging all of the instances of α with instances of β and all instances of β with instances of α.

For example, the $Z(XY)$ group is an X, Z re-lettering of the $X(YZ)$ group. We can see this because exchanging X with Z in (1) yields (6), in (2) yields (5), in (3) yields (8), and in (4) yields (7). Similarly the $Z(XY)$ group is an Y, Z re-lettering of the $Y(ZX)$ group and the $X(YZ)$ group is an X, Y re-lettering of the $Y(ZX)$ group. When letters are not specified, any of the possible re-letterings will do.

Not including one of the consecutions in any display equivalence class would mean not including any of its re-letterings in an equivalence class. I explain this in detail in §6.

4 The schemes

In this section I will present all of the possible schemes. In the next I will discuss their properties. In §6 I will argue that this list is complete and that it forms a join semi-lattice. As above, all consecutions on the same line are in the same equivalence class.

There are three basic types of schemes. The first (the **GA, A, A′, B, C$_a$** and **C$_b$** schemes) have three or six consecutions in each equivalence class. The second (the **P, P′, Q** and **Q′** schemes) have four consecutions in each class. And the third type (which includes only the **easy** scheme) has all twelve consecutions in the same class.

All of the schemes of the first type are built out of the equivalence classes of the **GA** Scheme $= \{\, GA_{1a}, GA_{1b}, GA_{2a}, GA_{2b} \,\}$

$$\begin{aligned}
GA_{1a} &= [(3) \quad Z \vdash *(X \circ Y), (7) \quad X \vdash *(Y \circ Z), (11) \quad Y \vdash *(Z \circ X)]\\
GA_{1b} &= [(2) \quad Z \vdash *Y \circ *X, (6) \quad X \vdash *Z \circ *Y, (10) \quad Y \vdash *X \circ *Z]\\
GA_{2a} &= [(4) \quad Z \vdash *(Y \circ X), (8) \quad X \vdash *(Z \circ Y), (12) \quad Y \vdash *(X \circ Z)]\\
GA_{2b} &= [(1) \quad Z \vdash *X \circ *Y, (5) \quad X \vdash *Y \circ *Z, \ (9) \quad Y \vdash *Z \circ *X]
\end{aligned}$$

The equivalence classes of the **GA** scheme are the bases for three six-six schemes:

$$\begin{aligned}
&\mathbf{A}\text{ scheme} = \{\, A_1, A_2 \,\} && \mathbf{A'}\text{ scheme} = \{\, A'_1, A'_2 \,\}\\
&A_1 = [GA_{1a} \cup GA_{1b}] && A'_1 = [GA_{1a} \cup GA_{2b}]\\
&A_2 = [GA_{2a} \cup GA_{2b}] && A'_2 = [GA_{1b} \cup GA_{2a}]
\end{aligned}$$

$$\begin{aligned}
&\mathbf{B}\text{ scheme} = \{\, B_a, B_b \,\}\\
&B_a = [GA_{1a} \cup GA_{2a}]\\
&B_b = [GA_{1b} \cup GA_{2b}]
\end{aligned}$$

Two six-three-three schemes based on the equivalence classes of the **B** and **GA** schemes are also possible:

$$\mathbf{C}_{\boldsymbol{a}}\text{ scheme} = \{\, B_a, GA_{1b}, GA_{2b} \,\} \quad \mathbf{C}_{\boldsymbol{b}}\text{ scheme} = \{\, B_b, GA_{1a}, GA_{2a} \,\}$$

There are only four schemes of the second type:

$$\begin{aligned}
&\mathbf{P}\text{ scheme} = \{\, P_1, P_2, P_3 \,\} && \mathbf{P'}\text{ scheme} = \{\, P'_1, P'_2, P'_3 \,\}\\
&P_1 = [(3), (5), (6), (12)] && P'_1 = [(4), (5), (6), (11)]\\
&P_2 = [(4), (7), (9), (10)] && P'_2 = [(3), (8), (9), (10)]\\
&P_3 = [(1), (2), (8), (11)] && P'_3 = [(1), (2), (7), (12)]\\
&\mathbf{Q}\text{ scheme} = \{\, Q_1, Q_2, Q_3 \,\} && \mathbf{Q'}\text{ scheme} = \{\, Q'_1, Q'_2, Q'_3 \,\}\\
&Q_1 = [2), (7), (8), (9)] && Q'_1 = [(1), (7), (8), (10)]\\
&Q_2 = [(1), (6), (11), (12)] && Q'_2 = [(2), (5), (11), (12)]\\
&Q_3 = [(3), (4), (5), (10)] && Q'_3 = [(3), (4), (6), (9)]
\end{aligned}$$

The final scheme is just the **easy** scheme. It has one equivalence class that contains (1)–(12) so:

$$\mathbf{easy}\text{ scheme} = \{[(1), (2), (3), (4), (5), (6), (7), (8), (9), (10), (11), (12)]\}$$

5 Properties of the schemes

While I doubt all these schemes will be equally useful, I will prescind from passing judgment here on whether a scheme will likely find a use or not. Rather, my aim will be to point out what seem to be their interesting properties, which will help in my argument that the list of schemes is complete.

A basic feature that shapes the properties of the schemes of the twelve component consecutions is that they divide into those with an antecedent circle, and those with a consequent circle. If the circle occurs within an even [odd] number of stared parentheses on the left, then it is an antecedent [consequent] circle because when it is not contained within a star it will be on the left [right] side of the turn-style. As Belnap points out, the circle in the antecedent and circle in the consequent are independently specifiable connectives.[7] With the technique of re-lettering, we can explain this by noting that when we re-letter consecutions, an antecedent circle never becomes a consequent circle [and vice versa].

We can cut back the number of schemes which deserve attention by pointing out that all of the prime schemes are only notational variants on the schemes of which they are the primes. We can easily define a circle governed by a primed scheme with a circle governed by the corresponding unprimed scheme, or vice versa. As Belnap points out in relation to the **A** and **A′** scheme, "one could obtain the **A′** scheme by starting out with the **A** scheme and defining a new operation $\circ'$ in consequent position by $X \circ' Y =_{\text{df}} Y \circ X$".[8] The **Q′** scheme is derivable from the **Q** scheme in exactly the same way. And the **P′** scheme is derivable from the **P** scheme by defining a new operation $\circ'$ in antecedent position by $X \circ' Y = Y \circ X$.

This means that the basic distinction between the four-four-four schemes is whether the circle commutes in the antecedent (**Q** schemes) or consequent (**P** schemes).[9] These are also two important properties for the **GA** based schemes and seem to be two of the most potentially

[7](Belnap, 1996, p. 89)

[8](Belnap, 1996, p. 90)

[9]The further distinction between the scheme and its prime describes how the $\circ$ interacts with negation. The **Q** scheme (**P** scheme) is distinguished from its prime by whether a structure being moved from the left to the right (right to left) side of the turn-style goes on the inside or outside of the previously displayed structure.

interesting properties the schemes can have. Neither of the **A** schemes have either property, nor does the **GA** scheme. But the $\mathbf{C}_{\boldsymbol{a}}$ scheme commutes in the antecedent, while the $\mathbf{C}_{\boldsymbol{b}}$ scheme commutes in the consequent, and the **B** scheme is the only scheme besides the **easy** scheme that commutes in both.

Although the **A** schemes lack commutativity, if we wanted to press the truth-functional analogy, the equivalence classes of the **A** scheme allow something like a structural De Morgan's rule. That is, they postulate that a negated circle in the consequent disjoining two structures is equivalent to a circle in the antecedent conjoining those same structures negated.

The **GA** scheme has the fewest properties built into it: it neither commutes in the antecedent nor consequent, nor does it have the De Morgan like property. As such, it allows the most control over the properties of the languages built using it. Nonetheless, the **GA** scheme and all of the schemes built out of its classes, share a common form or technique by which they can be constructed, that the **P** and **Q** schemes lack. Accordingly, there may be languages that can be built out of the latter schemes that cannot be built from the **GA** scheme.

The form or technique I have in mind is that each of the three members of any of the GA classes can be arrived at by taking one of the members, replacing X with Y, Y with Z, and Z with X twice, in order to get each of the other two members. (You can of course also go the other direction: replacing Y with X, Z with Y, and X with Z.) It is noteworthy that this technique of generating classes keeps the classes defining circle in the antecedent distinct from those defining circle in the consequent.

Similarly, all of the classes of **P** and **Q** schemes also share an underlying form or technique for construction. First, take any consecution and α, β re-letter it, for some α, β. This will be the second member of the class. For the third member pick a consecution displaying a variable not displayed by either of the other two, which involves the kind of circle that the other two do not (since all of these schemes inter-define the antecedent and consequent circle). To get the fourth and final member of the class, for the same α and β, α, β re-letter it. Once you have one class, the way to arrive at the other three classes in the scheme is by re-lettering this class twice (this will be explained further below).

The **easy** scheme has commutativity in the antecedent and consequent, and the De Morgan like property. Because of this, it is the scheme in which circle most closely approximates disjunction in the antecedent and conjunction in the consequent.

6 The completeness of the list

Although this is easy to verify through brute combinatorics, given that we only have twelve consecutions, in this section I will show this list exhausts the possible schemes using circle and star, while keeping the star scheme fixed. Before plunging in, it will help to note that because the letters are schematic we can actually express each of the schemes more simply because many of their equivalence classes are actually identical. We can see, for example, that the three equivalence classes of the **P** scheme are identical through re-lettering. A X, Y re-lettering of P_1 yields P_2, and vice versa. A X, Z re-lettering of P_1 yields P_3, and vice versa. A Y, Z re-lettering of P_2 yields P_3, and vice versa. A similar procedure can be followed with the rest of the **P** and **Q** schemes to show their classes are identical, and with the **GA** scheme to show GA_{1a} is identical with GA_{2a} and GA_{1b} is identical with GA_{2b}. As a result many of the schemes can be expressed more simply, if not more perspicuously as follows:

$$\begin{array}{rlrl} \mathbf{GA}\text{ scheme} &= \{\, GA_{1a}, GA_{1b} \,\} & \mathbf{A}\text{ scheme} &= \{\, A_1 \,\} \\ \mathbf{P}\text{ scheme} &= \{\, P_1 \,\} & \mathbf{Q}\text{ scheme} &= \{\, Q_1 \,\} \end{array}$$

(And similarly for the prime schemes.)

This fact is relevant here because it points to how inflexible the schemas are. If we switched some of the consecutions in one equivalence class with those of another in their scheme, then we would have to make corresponding switches in the other equivalence classes. Otherwise, by re-lettering the altered equivalence classes we would get an equivalence class that is for the most part identical with the corresponding unaltered one, except for the consecution that is the re-lettered newly introduced one. This consecution will belong to the other unaltered equivalence class, and so any consecution in either of the two unaltered equivalence classes will be inter-derivable from one another via the re-lettered altered equivalence class.

For example, suppose we switched (3) and (6) in the equivalence classes of the **GA** scheme to get the equivalence classes $GA^{+}_{1a} =_{\mathrm{df}}$

$[(6),(7),(11)]$ and $GA^{+}_{1b} =_{\mathrm{df}} [(2),(3),(10)]$. Now the X, Z re-lettering of GA^{+}_{1a} has members of both GA_{2a} and GA_{2b}. Accordingly, using this re-lettering and these two classes, we can derive any member of GA_{2a} from a member of GA_{2b} and vice-versa. Thus, on the hypothesis GA_{2a} and GA_{2b} collapse into A_2. Then, using a re-lettering of A_2, we can derive any member of GA^{+}_{1a} from GA^{+}_{1b} and vice-versa, so these collapse together into A_1 and we have the **A** scheme.

In general, the principle that this gives us is that the re-letterings of an equivalence class must be identical to another class (maybe themselves) already in the scheme. Otherwise, the scheme is unstable and we can use the re-lettered equivalence classes to collapse other equivalence classes together. With this principle what remains to be shown, in order to show the completeness of the list, is that these schemes are the only stable ones.

For this, it will help to have another principle governing the classes: if two members of a class are some kind of α, β re-lettering of one another, then there must be at least two other members of the class that are that same kind of re-lettering of one another. The schemes of the second type (**P**, **Q**, etc.) exhibit this rule, but now it can be shown as a corollary of the above principle. Take two consecutions that are some α, β re-lettering of one another. To these, at least one consecution displaying a not yet displayed variable must be added for the class to secure the display property. Now if we α, β re-letter all three members, the re-lettered first will be identical to the non-re-lettered second, and vice-versa, but the re-lettered third will be a new consecution that is an α, β re-lettering of the original third. But since the α, β re-lettering and the original proposed class share members, they collapse together by the above, and the principle is shown. For example, since (4) is a YZ re-lettering of (11) let $GA^{+}_{2a} =_{\mathrm{df}} [(4),(8),(11)]$. Its YZ re-lettering will be: (4), (7), (11). Since (4) is shared, the two classes collapse together, and the original class must include (7) as well.

To see that the scheme's canvassed are the only stable ones first note that no other schemes can be built from the classes of the **GA**, **P**, and **Q** schemes. Since there is only really one equivalence class in the **A**, **P**, and **Q** schemes adding an additional class to any of these would cause them to collapse into the easy scheme. Otherwise we have built all of the schemes it is possible to build from the **GA** scheme alone. This is because there are only really two distinct classes in that scheme, and the only scheme with more than one class that results from taking their unions is the **B** scheme, the classes of which are then used to build the **C** schemes.

Now the **GA** scheme and the schemes of the second type (**P**, **Q**, etc.) are minimal solutions[10] to the display problem in that there is no further refinements of either of them that also solve it. If there were other stable schemes then either they would have to be minimal solutions to the display problem, or be built by joining the classes of some other minimal solution to the display problem. So in order to show that the list of schemes presented is complete, it suffices to show there are no more minimal solutions to the display problem.

Since the circle in the antecedent or consequent can be defined independently, there will be those minimal schemes that define them separately (the **GA** scheme) and those that don't (the second type schemes). If a proposed new minimal solution defined them separately, then it would have to have at least two non-identical classes, dealing with the six consecutions governing each circle separately. These classes could either be composed of six or three members, because if they were composed of four or five they would be obviously unstable. If they were composed of all six consecutions, then they would not be minimal because they could be divided into the classes of the **GA** scheme. If they were composed of three member classes, however, then they must be those of the **GA** scheme because otherwise two of the members would be α, β re-letterings of one another, for some α and some β, which would entail that a fourth member belonged to the class by the above corollary, but this would cause collapse.

If the proposed minimal solution defined the two circles together, then it would have to include consecutions governing each circle in at least one of its classes. This class, like any, must solve the display problem, and so must have at least one member displaying each of the three variables. Either the third member of the class will be an α, β re-lettering of one of the other members or it will not be. If it is, then by the above corollary there will be a fourth member of the class which is an α, β re-lettering of the third, for the same α and β. Now since the class defines the antecedent and consequent circles simultaneously, it is obvious from this and the technique I gave for generating the schemes of the second type, that the possible classes

[10]Borrowing the term from (Belnap, 1996, pp. 89–90).

which could ground those schemes are the same as the ones that satisfy this description, and all of the schemes that could be generated out of these have already been discussed.

If, however, the third member of the class is not an α, β re-lettering of one of the other members, it at least must be the same kind of circle (antecedent or consequent) as one of the other members. Since it is not an α, β re-lettering of this consecution, however, of the other five consecutions with this same kind of circle there are only two consecutions it could be, and it will be in the same class of the **GA** scheme as this consecution. If we now re-letter this class in all three ways, in each re-lettering we will have two out of the three consecutions in the other GA class governing the type of circle that two of the three members of the original class share, but which they are not themselves in. That this would be so should be clear from the fact that no matter how one re-letters a member of the GA classes, one always gets a member of the other GA class governing the same circle. From this fact it also follows that the three re-letterings of the third member of the class will all be different members of the same GA class. Now since these three classes all share members, they will collapse together and the resulting class will be the union of the two classes of the **GA** scheme that the first three consecutions do not belong to (which will be one of the **A** or **A′** scheme classes).

An example will help. Suppose the class we start from is:

$$GA^{*}_{2b} =_{\text{df}} [(1)\ Z \vdash *X \circ *Y,\ (5)\ X \vdash *Y \circ *Z,\ (11)\ Y \vdash *(X \circ Z)]$$

XY, YZ, and ZX Re-lettering this we get:

$$GA^{*XY}_{1b} =_{\text{df}} [(2)\ Z \vdash *Y \circ *X,\ (7)\ X \vdash *(Y \circ Z),\ (10)\ Y \vdash *X \circ *Z]$$
$$GA^{*YZ}_{1b} =_{\text{df}} [(4)\ Z \vdash *(Y \circ X),\ (6)\ X \vdash *Z \circ *Y,\ (10)\ Y \vdash *X \circ *Z]$$
$$GA^{*ZX}_{1b} =_{\text{df}} [(2)\ Z \vdash *Y \circ *X,\ (6)\ X \vdash *Z \circ *Y,\ (12)\ Y \vdash *(X \circ Z)]$$

Now since GA^{*XY}_{1b} & GA^{*YZ}_{1b} share (10), and GA^{*YZ}_{1b} & GA^{*ZX}_{1b} share (6) the three classes will collapse together, yielding A'_2.

Since the two classes of the **A′** (or **A**) scheme are identical by re-lettering, this, or the **A** scheme, is the scheme we are committed to, given that in our original class both circles are present and the third member was not an α, β re-lettering of one of the other members. But the **A** and **A′** schemes are not minimal solutions to the display problem, so there is no other minimal solution besides the ones canvassed. Thus, the list of schemes given is complete.

Accordingly, this list forms a join semi-lattice on the twelve consecutions:

$$\begin{tikzcd}
 & & \textbf{Easy Scheme} \arrow[dll, no head] \arrow[d, no head] \arrow[dddr, no head] \arrow[dddrr, no head] & & \\
\textbf{A} \text{ and } \textbf{A}' \arrow[ddr, no head] & & \textbf{B} \arrow[dl, no head] \arrow[d, no head] & & \\
 & \textbf{C}_a \arrow[d, no head] & \textbf{C}_b \arrow[dl, no head] & & \\
 & \textbf{GA} & & \textbf{P} \text{ and } \textbf{P}' & \textbf{Q} \text{ and } \textbf{Q}'
\end{tikzcd}$$

7 Conclusion

These properties of the behavior of the structural-connectives given the different schemes are, however, rather superficial. Since the languages formulable in display logic are individuated by the differences between the structural rules and display equivalences governing them, the deeper properties of the schemes (as well as the structural rules) are the ones they give the languages built using them. But so far I don't think anyone has a clear grasp on how each of the individual structural rules or schemes of equivalences effects the language families they are a part of or the extent to which the specific properties of the families can be traced back to individual rules. Although this paper has not tackled this difficult question, I hope that by providing a complete semi-lattice of display equivalence schemes for the standard connectives our understanding of the variety of structural recourses within display logic for formulating interesting languages has been slightly improved.

References

Anderson, A., Jr, N. B., & Dunn, J. (1992). *Entailment. the logic of relevance and necessity* (Vol. 2). Princeton: Princeton University Press.

Belnap, N. (1982). Display logic. *Journal of Philosophical Logic*, *11*(4), 375–417.

Belnap, N. (1996). The display problem. In H. Wansing (Ed.), *Proof theory of modal logic* (pp. 79–92). Dordrecht, Boston, and London: Kluwer Academic Publishers.

Gentzen, G. (1969). Investigations into logical deduction. In M. E. Szabo (Ed.), *The Collected Works of Gerhard Gentzen* (pp. 68–131). Amsterdam: North-Holland Publishing Company.

Tyke Nunez
Department of Philosophy
University of Pittsburgh
1001 Cathedral of Learning
Pittsburgh, PA 15260
e-mail: `asn13@pitt.edu`