

Categorial Logic

Gary Hardegree
Department of Philosophy
University of Massachusetts
Amherst, MA 01003

Part 1: Overview	2
1. Introduction.....	2
2. Standard Categorial-Composition.....	2
3. Expanded Categorial-Composition.....	2
4. What is Categorial Logic?	3
5. Classical Logic (doesn't properly model Grammatical-Composition)	3
6. Intuitionistic Logic (doesn't properly model Grammatical-Composition)	4
7. No Monotonic Logic Properly Models Grammatical-Composition	4
8. Relevance Logic (doesn't properly model Grammatical-Composition)	5
9. Linear Logic (doesn't properly model Grammatical-Composition)	6
10. System G.....	7
Part 2: Detailed Presentation of the Logical Systems	7
1. The Arrow-Fragment of System R	7
2. The Cross-Arrow Fragment of R	12
3. System R is too Strong.....	14
4. Linear Logic.....	14
5. Linear Logic is too Weak.....	15
6. An Aside on the Lambek Calculus	15
7. The Proposed System – G.....	16
8. Summary of Systems Considered	17

Part 1: Overview

1. Introduction

In this chapter, we examine categorial logic, which is intended to be a *calculus of grammatical-composition*,¹ including syntactic-composition and semantic-composition.² We call it a "logic" because, in characterizing it, we utilize logical formalism, and we draw examples from existing logical systems – including classical logic, intuitionistic logic, relevance logic, and linear logic.

As it turns out, no extant logical system adequately models grammatical-composition. Rather, we propose to model grammatical-composition by a novel logical system, dubbed ‘System G’, which lies between relevance logic and linear logic.

2. Standard Categorical-Composition

According to standard categorial-grammar, grammatical-composition consists *exclusively* in *function-application*. In particular, one may combine two items if and only if one item is a function ϕ and the other item serves as an argument for ϕ . This may be depicted as follows.³

ϕ	a function from A into B	$A \rightarrow B$
α	an item of type A	A

$\phi(\alpha)$	an item of type B	B

If we think of arrow as the logical *if-then* connective, then this procedure corresponds to the following argument form,

if A , then B
A

B

which is the inference-principle known as *modus ponens*.

3. Expanded Categorical-Composition

Whereas standard categorial-grammar admits only one mode of composition, corresponding to *modus ponens*, expanded categorial-grammar admits infinitely-many modes of composition, each one corresponding to a valid-inference of categorial logic. This principle is officially presented as follows.

¹ The idea of a calculus of (pure) syntactic-composition traces to Joachim Lambek's "The Mathematics of Sentence Structure", *American Mathematical Monthly*, 65 (1958), 154–170. The system he proposed, the *Lambek Calculus*, is examined in Section 6.

² Here, by syntax, we actually mean just the syntax-semantics interface. In particular, our syntactic-rules mimic our semantic-rules in being completely indifferent to the left-right structure of parse-trees. This is in marked contrast to other approaches to categorial grammar that are more narrowly syntactic in orientation.

³ In depicting function-types, we employ the arrow symbol ‘ \rightarrow ’ rather than the division-symbol ‘/’, which many authors use. The arrow symbol accords with category theory, and also exhibits the strong parallel between functions and conditionals, a parallel that forms the basis of our theory of composition.

Let $\mathcal{A}_0, \dots, \mathcal{A}_k$ be types; then $\mathcal{A}_1, \dots, \mathcal{A}_k$ may be combined to form \mathcal{A}_0 precisely if:

$$\mathcal{A}_1, \dots, \mathcal{A}_k \vdash \mathcal{A}_0$$

Here, ‘ \vdash ’ is a meta-linguistic predicate corresponding to logical entailment.⁴

4. What is Categorial Logic?

The obvious question then is – what logical system do we use to judge entailment (\vdash)? Which logical system best models grammatical-composition? In this connection, there are several prominent extant logical systems that serve as candidates, including

Classical Logic
Intuitionistic Logic
Relevance Logic
Linear Logic

which we examine in the next few sections.

5. Classical Logic (doesn't properly model Grammatical-Composition)

Since it is the strongest logic of the group,⁵ Classical Logic authorizes the maximum number of compositions, but unfortunately it also authorizes compositions that are grammatically inadmissible. For example, the following is a principle of Classical Logic.⁶

$$(c1) \quad A ; B \vdash A \rightarrow B$$

Accordingly, using CL to judge grammatical-composition, the following composition-rule is authorized.

$$S ; S \vdash S \rightarrow S$$

We read this saying that one may combine two sentences (S) to form a sentential-adverb (S-operator). Since this is grammatically highly implausible, categorial logic must reject (c1).

In an important sense, to be explained shortly, (c1) follows from the following inference principle, which is also valid in Classical Logic.

$$(c2) \quad B \vdash A \rightarrow B$$

This is an inference principle found especially obnoxious by many pioneers of alternative logical systems, including the strict-entailment logics of C.I. Lewis⁷ and the relevant-entailment logics of Anderson and Belnap.⁸

⁴ Logicians use various turnstile-symbols, including ‘ \vdash ’ and ‘ \models ’, to depict various manners of judging entailment; in particular, usually ‘ \vdash ’ pertains to proof-theoretic entailment, ‘ \models ’ pertains to model-theoretic entailment.

⁵ If a weaker logic validates an argument form \mathbb{A} , then a stronger logic also validates \mathbb{A} , granting the two logics both pass judgment on \mathbb{A} .

⁶ Although nothing hinges on this, we use one arrow-symbol ‘ \rightarrow ’ for logic, and another arrow-symbol ‘ \rightarrow ’ for category theory.

Grammatically-understood, (c2) is equally obnoxious, since the following instance

$$S \vdash S \rightarrow S$$

authorizes transforming⁹ a sentence into a sentential-adverb (S-operator), which seems grammatically implausible.

In addition to (c1) and (c2), the following CL-principles also yield implausible composition principles.

$$(c3) \quad (A \rightarrow B) \rightarrow B \vdash (B \rightarrow A) \rightarrow A$$

$$(c4) \quad (A \rightarrow B) \rightarrow A \vdash A$$

In light of numerous examples of inadmissible grammatical-compositions based on CL-principles, we conclude that Classical Logic does not properly model grammatical composition.

6. Intuitionistic Logic (doesn't properly model Grammatical-Composition)

Intuitionist Logic is weaker than Classical Logic – every argument deemed valid by IL is also deemed valid by CL, but not conversely. For example, (c3) and (c4) above are rejected by IL. On the other hand, both (c1) and (c2) are accepted by IL. Since the latter yield grammatically-inadmissible compositions, so we must conclude that Intuitionistic Logic also does not properly model grammatical composition.

7. No Monotonic Logic Properly Models Grammatical-Composition

A principle that is fundamental to nearly every logical system that has been considered over the past few millennia is the principle of *monotonicity*.¹⁰ The basic idea is that adding premises to a valid argument does not result in an invalid argument. The following meta-principle is a special case of monotonicity.¹¹

$$\mathcal{B} \vdash C \Rightarrow \mathcal{A} ; \mathcal{B} \vdash C$$

In other words, if C follows from \mathcal{B} , C also follows from \mathcal{A} and \mathcal{B} .

Next, suppose we also grant the following *identity* principle

$$\mathcal{B} \vdash \mathcal{B}$$

which is generally regarded as a minimum requirement of any formal system that presumes to model reasoning. Putting monotonicity and identity together, we obtain the following *simplification principle*.

⁷ C.I. Lewis, 'Implication and the Algebra of Logic', *Mind* N.S., 21 (1912), 522-31.

⁸ Alan Anderson and Nuel Belnap, *Entailment*, Princeton University Press, 1975.

⁹ A transformation corresponds to a single-premise argument.

¹⁰ This usage comes from the definition of monotonic (increasing) functions. Specifically, a function ϕ is said to be monotonic precisely if $\phi(x) \leq \phi(y)$ whenever $x \leq y$. In the context of logical systems, the relevant order-relation is set-inclusion, and the relevant function is the consequence function \mathbb{C} , defined so that $\mathbb{C}(\Gamma) =_{\text{def}} \{\alpha \mid \Gamma \vdash \alpha\}$. Then a logical system is monotonic precisely if its associated consequence-function is monotonic.

¹¹ Here, we use the symbol ' \Rightarrow ' as the meta-language's if-then connective.

$$\mathcal{A} ; \mathcal{B} \vdash \mathcal{B}$$

Suppose we include this principle in the logic of grammatical-composition. Then the following composition is authorized.

$$D ; S \vdash S$$

In the proposed composition, we compose a sentence out of a definite-noun-phrase (D) and a sentence (S) – presumably by simply discarding the DNP. However, it seems manifestly plausible that a valid grammatical-composition must meaningfully utilize *all* its inputs in constructing its output. Accordingly, in order for a logic to model grammatical-composition, it cannot contain the simplification principle, and so it cannot be monotonic.

In what follows, we examine two prominent non-monotonic logics – Relevance Logic, and Linear Logic.

8. Relevance Logic (doesn't properly model Grammatical-Composition)

As its name suggests, Relevance Logic is characterized by sensitivity to matters of *relevance* – in particular, between premises and conclusions of arguments, and between antecedents and consequents of conditional (if-then) statements. In particular, for Relevance Logic, a conditional statement cannot be true unless there is a connection between the antecedent and the consequent, and an argument cannot be valid unless there is a connection between the premises and the conclusion. Most well-known systems of logic do not satisfy either of these *desiderata*.

At the same time, it seems that relevance *desiderata* are tailor-made for modeling grammatical-composition. For example, it is presumed that a grammatical functor actually uses its input in generating its output, and it is also presumed that a grammatical-composition uses all its input in producing its output.

Relevance Logic does a very decent job of modeling grammatical-composition, but it also authorizes several problematic compositions, which we now review.

1. Contraction

$$A \rightarrow (A \rightarrow B) \vdash A \rightarrow B$$

If we use this to model grammatical-composition, then we obtain the following grammatical principle

$$S \rightarrow (S \rightarrow S) \vdash S \rightarrow S$$

according to which a two-place connective automatically transforms into a one-place connective, which seems implausible.

2. Duplication

$$A \vdash A \times A$$

Here, \times is the multiplicative-counterpart of \rightarrow , which we examine in more detail in Section 2.1. If we use this to model grammatical-composition, then we obtain the following grammatical principle.

$$D \vdash D \times D$$

This amounts to saying that a DNP can duplicate itself (repeatedly) and accordingly serve as the input for an unlimited number of functors (e.g., VPs), which seems implausible.

3. Law of Assertion

$$(A \rightarrow A) \rightarrow B \vdash B$$

If we use this to model grammatical-composition, then we obtain the following grammatical principle, where C is the type of common-noun-phrases.

$$(C \rightarrow C) \rightarrow (C \rightarrow C) \vdash C \rightarrow C$$

From this, we obtain the following composition principle.

$$(C \rightarrow C) \rightarrow (C \rightarrow C) ; C \vdash C$$

A *modifier* is a phrase of type $\mathcal{A} \rightarrow \mathcal{A}$, where \mathcal{A} is any type. For example, a common-noun modifier (adjective) is a phrase of type $C \rightarrow C$, and an adjective-modifier is a phrase of type $(C \rightarrow C) \rightarrow (C \rightarrow C)$. According to the grammatical principle proposed above, an adjective-modifier like ‘very’ can be combined with a common-noun like ‘dog’ to produce a common-noun ‘very dog’, which seems implausible.

4. Tautology

A tautology is a formula that is logically-true, alternatively a formula that follows from nothing. Every logic has valid argument forms, but not every logic has tautologies. Relevance logic has tautologies, the simplest of which is depicted in the following principle, which we call ‘Tautology’.

$$\vdash \mathcal{A} \rightarrow \mathcal{A}$$

When the turnstile ‘ \vdash ’ acts as a monadic predicate, it is understood that the premise-set is empty.

Note that Assertion follows from Tautology,¹² so the latter must be rejected by categorial logic insofar as the former is rejected.

9. Linear Logic (doesn't properly model Grammatical-Composition)

Given that Relevance Logic is too strong, we next consider a weaker logical system, Linear Logic, which is founded on the idea of resource-usage.¹³ In particular, whereas Relevance Logic requires that every supposition be used *at least once*, Linear Logic permits a supposition to be used *only once*. This adjustment gets rid of Contraction and Duplication, but it does not get rid of Assertion or Tautology. To accomplish this, we also ban arguments with no premises. This restriction makes sense from a grammatical point of view, since we do not want phrases entering a grammatical-construction "out of thin air". The resulting system is sometimes called Linear Logic without Identity, but we will simply refer to it as Linear Logic.

By moving to Linear Logic (without Identity), we get rid of many inference principles that make no sense grammatically, but unfortunately we also get rid of inference principles that seem desirable, including the following.

¹² Granting transitivity of \vdash .

¹³ Linear Logic originates with Jean-Yves Girard, "Linear Logic", *Theoretical Computer Science*, 50:1, pp. 1-102, 1987.

- | | | |
|-----|---|------------------------------|
| (1) | $A \rightarrow B ; A \rightarrow C \vdash A \rightarrow (B \times C)$ | [Conditional Multiplication] |
| (2) | $A \rightarrow (B \rightarrow C) ; A \rightarrow B \vdash A \rightarrow C$ | [Conditional Modus Ponens] |
| (3) | $A \rightarrow B ; A \rightarrow C ; (B \times C) \rightarrow D \vdash A \rightarrow D$ | [Generalized-Conjunction] |

10. System G

It thus appears that we are in a "Goldilocks" situation – we have one system, R, which is too strong, and we have another system, L, which is too weak, but like Goldilocks we seek a system that is "just right". The system we seek – which we dub "G"¹⁴ – lies logically somewhere between L and R.

In Part 2, we examine the Systems R and L in greater detail, arriving ultimately at the official formulation of System G.

Part 2: Detailed Presentation of the Logical Systems

1. The Arrow-Fragment of System R

We begin with the arrow-fragment of System R, which is to say the fragment of R that pertains exclusively to the conditional operator ' \rightarrow ', which is the starting point in many formal treatments of Relevance Logic.

1. Syntax for R_A

Since it involves only one connective, the syntax of R_A is very easy to describe.

- (1) every atomic formula is a formula;
- (2) if \mathcal{A} and \mathcal{B} are formulas, then so is $(\mathcal{A} \rightarrow \mathcal{B})$;
- (3) nothing else is a formula.

2. Derivation System for R_A

1. Preliminaries

For us, a derivation is a structure of the following form.

line-number	formula	index	annotation
L_1	Φ_1	I_1	A_1
L_2	Φ_2	I_2	A_2
...			
L_k	Φ_k	I_k	A_k

A line's number indicates its location in the derivation, which is used to reference the formula in the annotation-column. A line's annotation indicates how that line is justified according to the rules of inference. A line's index basically lists the *suppositions* (premises, assumptions) on which the line depends. So, for example, the following derivation line

(7)	P	1,2	3,5, MP
-----	---	-----	---------

¹⁴ 'G' stands for 'Goldilocks', 'grammar', and 'grail'.

indicates that formula ‘P’, which is line 7, follows from lines 3 and 5 by the inference-rule *modus ponens*, and depends upon suppositions 1 and 2.¹⁵

2. Indices

Indices encode sub-structural¹⁶ information, and are officially defined as follows.

- (1) \emptyset is an index;
- (2) every sequence of numerals is an index;
- (3) nothing else is an index.

Indices form an algebra under the operation of sequence-addition, $+$, which satisfies various algebraic principles depending upon the specific logical system. In System R, the principles are as follows, where i, j, k are indices.

- (1) $i+(j+k) = (i+j)+k$ + is associative
- (2) $i+j = j+i$ + is commutative
- (3) $i+i = i$ + is contractive
- (4) $i+\emptyset = i$ \emptyset is an identity element for +

In other words, in System R, indices behave just like sets, where $+$ is set-union and \emptyset is the empty-set.

3. Definition of Valid Derivation

Where Φ_0, \dots, Φ_k are formulas, a valid derivation of Φ_0 from $\{\Phi_1, \dots, \Phi_k\}$ is a sequence of lines as follows.

line number	formula	index	annotation
L_1	Φ_1	1	Pr
...
L_k	Φ_k	k	Pr
...
L_0	Φ_0	$1, \dots, k$...

In particular:

- (1) the first k -many lines are the premises – Φ_1, \dots, Φ_k , each one indexed by the line-number k .
- (2) every remaining line is either
 - (1) an assumption, or
 - (2) follows from previous lines by an inference-rule.
- (3) the last line is Φ_0 , which depends upon *all and only* the premises.

¹⁵ There are two reasonable ways of indexing suppositions; (1) one can simply use the supposition's line-number as its index. (2) one can number suppositions independently of formulas. We employ the latter approach.

¹⁶ For a general presentation, see Greg Restall, *An Introduction to Substructural Logics*, Routledge, 2000.

4. Assumptions

There are two kinds of suppositions allowed in derivations – premises, and assumptions. Whereas premises correspond to the input expressions of a grammatical-composition, assumptions arise in sub-derivations in connection with various inference-rules (see below). The following schematically exhibits the assumption-insertion rule.

line-number	formula	index	annotation
L	Φ	m (new)	As

Here, Φ is any formula, and m is any numeral that is new, which is to say it does not occur earlier in the derivation.

5. Inference-Rules

We follow a derivation scheme according to which every logical-operator is characterized by two rules – a construction-rule and a deconstruction-rule. In the literature, these are generally known as *introduction-rules* and an *elimination-rules*; we prefer the more succinct terms ‘in’ and ‘out’.

1. Arrow-Out (\rightarrow O)

L_1	$\mathcal{A} \rightarrow \mathcal{B}$	i	...
L_2	\mathcal{A}	j	...
L_3	\mathcal{B}	$i+j$	L_1, L_2, \rightarrow O

2. Arrow-In (\rightarrow I)

L_1	\mathcal{A}	j	...
L_2	\mathcal{B}	$i+j$...
L_3	$\mathcal{A} \rightarrow \mathcal{B}$	i	L_1, L_2, \rightarrow I

Here, i and j are indices (sequences of numerals), and $i+j$ is the sequence-sum of i and j . Whereas arrow-out is just *modus ponens*, arrow-in corresponds to the following key principle about conditionals.

$$\mathcal{A}_1; \dots; \mathcal{A}_k; \mathcal{A} \vdash \mathcal{B} \Rightarrow \mathcal{A}_1; \dots; \mathcal{A}_k \vdash \mathcal{A} \rightarrow \mathcal{B}$$

Note that, in arrow-in, most derivations introduce \mathcal{A} by way of the assumption-rule.

6. Examples of Derivations in R_A

In what follows, we write sequences without commas.¹⁷ We also take associativity and commutativity for granted, which allows us to write all indices in simple numerical order.

1. Transitivity

$B \rightarrow C ; A \rightarrow B \vdash A \rightarrow C$

(1)	$B \rightarrow C$	1	Pr
(2)	$A \rightarrow B$	2	Pr
(3)	A	3	As
(4)	B	23	2,3, \rightarrow O
(5)	C	123	1,4, \rightarrow O
(6)	$A \rightarrow C$	12	3,5, \rightarrow I

2. Permutation

$A \rightarrow (B \rightarrow C) \vdash B \rightarrow (A \rightarrow C)$

(1)	$A \rightarrow (B \rightarrow C)$	1	Pr
(2)	B	2	As
(3)	A	3	As
(4)	$B \rightarrow C$	13	1,3, \rightarrow O
(5)	C	123	2,4, \rightarrow O
(6)	$A \rightarrow C$	12	3,5, \rightarrow I
(7)	$B \rightarrow (A \rightarrow C)$	1	2,6, \rightarrow I

3. Secondary Modus Ponens

$A \rightarrow (B \rightarrow C) ; B \vdash A \rightarrow C$

(1)	$A \rightarrow (B \rightarrow C)$	1	Pr
(2)	B	2	Pr
(3)	A	3	As
(4)	$B \rightarrow C$	13	1,3, \rightarrow O
(5)	C	123	2,4, \rightarrow O
(6)	$A \rightarrow C$	12	3,5, \rightarrow I

4. Conditional Modus Ponens

(1)	$A \rightarrow (B \rightarrow C)$	1	Pr
(2)	$A \rightarrow B$	2	Pr
(3)	A	3	As
(4)	$B \rightarrow C$	13	1,3, \rightarrow O
(5)	B	23	2,3, \rightarrow O
(6)	C	1233 [=123]	4,5, \rightarrow O
(7)	$A \rightarrow C$	12	3,6, \rightarrow I

¹⁷ Except when this practice conflicts with the usual numeral morphology; for example, we may need to distinguish the numeral '23' (twenty-three) from the sequence '2,3' (two, three).

5. Montague's Law¹⁸ $A \vdash (A \rightarrow B) \rightarrow B$

(1)	A	1	Pr
(2)	$A \rightarrow B$	2	As
(3)	B	12	1,3, \rightarrow O
(4)	$(A \rightarrow B) \rightarrow B$	1	2,4, \rightarrow I

6. Inflection¹⁹ $(A \rightarrow C) \rightarrow D ; A \rightarrow B \vdash (B \rightarrow C) \rightarrow D$

(1)	$(A \rightarrow C) \rightarrow D$	1	Pr
(2)	$A \rightarrow B$	2	Pr
(3)	$B \rightarrow C$	3	As
(4)	A	4	As
(5)	B	24	2,4, \rightarrow O
(6)	C	234	3,5, \rightarrow O
(7)	$A \rightarrow C$	23	4,6, \rightarrow I
(8)	D	123	1,7, \rightarrow O
(9)	$(B \rightarrow C) \rightarrow D$	12	3,8, \rightarrow I

7. Contraction $A \rightarrow (A \rightarrow B) \vdash A \rightarrow B$

(1)	$A \rightarrow (A \rightarrow B)$	1	Pr
(2)	A	2	As
(3)	$A \rightarrow B$	12	1,2, \rightarrow O
(4)	B	122 [=12]	2,3, \rightarrow O
(5)	$A \rightarrow B$	1	2,4, \rightarrow I

8. Tautology $\vdash A \rightarrow A$

(1)	A	1	As
(2)	$A \rightarrow A$	\emptyset	1,1, \rightarrow I

9. Law of Assertion $(A \rightarrow A) \rightarrow B \vdash B$

(1)	$(A \rightarrow A) \rightarrow B$	1	Pr
(2)	A	2	As
(3)	$A \rightarrow A$	\emptyset	2,2, \rightarrow I
(4)	B	1 \emptyset [=1]	1,3, \rightarrow O

¹⁸ This particular transformational principle is so called because one instance of it $[D \vdash (D \rightarrow S) \rightarrow S]$ corresponds to Montague's proposal to treat DPs like 'Kay' as second-order predicates.

¹⁹ We call this principle 'inflection' because it authorizes numerous compositions involving case-inflection.

2. The Cross-Arrow Fragment of R

The proposed type-theory includes two type-operators – \rightarrow , \times . So far, we have only characterized \rightarrow . We now turn to \times .

1. An Aside on Residuation

A conditional connective \rightarrow is said to be *residuated*²⁰ precisely if there is an associated multiplication-operator \times satisfying the following residual law.²¹

$$(A \times B) \rightarrow C \dashv\vdash A \rightarrow (B \rightarrow C)$$

For example, in Classical and Intuitionistic Logic, the multiplication-operator is simply logical-and.

$$A \times B = A \& B$$

On the other hand, in Relevance Logic²² and Quantum Logic²³, multiplication corresponds to "compossibility", which is defined as follows.²⁴

$$A \times B \quad =_{df} \quad \sim(B \rightarrow \sim A) \quad [\neq A \& B]$$

2. Syntax for R_{CA}

- (1) every atomic formula is a formula;
- (2) if \mathcal{A} and \mathcal{B} are formulas, then so is $(\mathcal{A} \rightarrow \mathcal{B})$;
- (3) if \mathcal{A} and \mathcal{B} are formulas, then so is $(\mathcal{A} \times \mathcal{B})$;
- (4) nothing else is a formula.

3. Derivation System for R_{CA}

1. Cross-In ($\times I$)

L_1	\mathcal{A}	i	...
L_2	\mathcal{B}	j	...
L_3	$\mathcal{A} \times \mathcal{B}$	$i+j$	$L_1, L_2, \times I$

²⁰ For a general account of residuation, see T.S. Blyth and M.F. Janowitz, *Residuation Theory*, Pergamon Press, Oxford, 1972.

²¹ One can also call this a division principle; to see why, rewrite $A \rightarrow B$ as B/A . Then the residual law amounts to the following.

$$A/(B \times C) = (A/B)/C$$

A divided by (B times C) equals (A divided by B) divided by C

The word 'residual' pertains to subtraction, in which case the residual law is written thus.

$$A - (B + C) = (A - B) - C$$

²² Dunn, *The Algebra of Intensional Logics*, PhD Dissertation, University of Pittsburgh, 1966.

²³ Hardegree, "Material Implication in Orthomodular (and Boolean) Lattices", *Notre Dame Journal of Formal Logic*, 22 (1981), 163-82.

²⁴ The order doesn't matter for relevance logic, in which \times is commutative, but it does matter for quantum logic, in which \times is not commutative. Also note that, in relevance logic, \times is referred to as "fusion", and is usually written using ' \circ '.

2. Cross-Out ($\times O$)

L_1	$\mathcal{A} \times \mathcal{B}$	h	...
L_2	\mathcal{A}	i	...
L_3	\mathcal{B}	j	...
L_4	C	$i+j+k$...
L_5	C	$h+k$	$L_1, L_2-L_4, \times O$

This rule states that, if one has a cross-formula $\mathcal{A} \times \mathcal{B}$, and one has a sub-derivation of C from $\{\mathcal{A}, \mathcal{B}\}$, then one can infer C . See example below.

4. Examples of Derivations in R_{CA}

1. Duplication

$A \vdash A \times A$

(1)	A	1	Pr
(2)	$A \times A$	11 [= 1]	1,1, $\times I$

2. Conditional-Multiplication

$A \rightarrow B ; A \rightarrow C \vdash A \rightarrow (B \times C)$

(1)	$A \rightarrow B$	1	Pr
(2)	$A \rightarrow C$	2	Pr
(3)	A	3	As
(4)	B	13	1,3, $\rightarrow O$
(5)	C	23	2,3, $\rightarrow O$
(6)	$B \times C$	1233 [= 123]	4,5, $\times I$
(7)	$A \rightarrow (B \times C)$	12	3,6, $\rightarrow I$

3. Schönfinkel's Law²⁵

$(A \times B) \rightarrow C \dashv\vdash A \rightarrow (B \rightarrow C)$

(1)	$(A \times B) \rightarrow C$	1	Pr
(2)	A	2	As
(3)	B	3	As
(4)	$A \times B$	23	2,3, $\times I$
(5)	C	123	1,4, $\rightarrow O$
(6)	$B \rightarrow C$	12	3,5, $\rightarrow I$
(7)	$A \rightarrow (B \rightarrow C)$	1	2,6, $\rightarrow I$

²⁵ We name this principle after Moses Schönfinkel, who first proposed that one can treat a two-place function as a family of one-place functions.

2. Sub-Structural Rules (remove contraction and identity)

- (1) $i+(j+k) = (i+j)+k$ + is associative
 (2) $i+j = j+i$ + is commutative

5. Linear Logic is too Weak

By moving to L, we get rid of many inference principles that do not properly model grammatical-composition, but unfortunately we also get rid of inference principles that seem desirable, including the following.

- (1) $A \rightarrow B ; A \rightarrow C \vdash A \rightarrow (B \times C)$ [Conditional-Multiplication]
 (2) $A \rightarrow (B \rightarrow C) ; A \rightarrow B \vdash A \rightarrow C$ [Conditional Modus Ponens]
 (3) $A \rightarrow B ; A \rightarrow C ; (B \times C) \rightarrow D \vdash A \rightarrow D$ [Generalized Conjunction]

Thus, we find ourselves in a "Goldilocks situation" – we have one system, R, which is too strong, and we have another system, H, which is too weak – but, like Goldilocks, we seek a system that is "just right".

6. An Aside on the Lambek Calculus

So far we have examined well-known logical systems in search of a calculus of grammatical-composition. As noted earlier, the idea of a grammatical-calculus traces to Joachim Lambek, who proposed such a calculus in 1958.²⁶ In this section, we briefly compare our approach to Lambek's.

A very small difference is that Lambek uses a slash-symbol to depict the category-division operator, whereas we use an arrow-symbol, in accord with category theory and logic.²⁷ A very big difference is that Lambek concentrates on syntax, and takes word-order seriously, which we do not. The resulting logic is accordingly non-commutative, and accordingly contains two division operators – left-division (forward slash /) and right-division (backward slash \).

There is no doubt that word-order is important in the *presentation* of meanings/content, especially in languages that use word-order to encode functional-roles (as in SVO, SOV, etc.) . However, this is purely syntactic. Semantic-composition consists (fundamentally) in function-application, which is completely indifferent to whether the function is to the "left" or "right" of its argument.²⁸ Accordingly, a calculus of semantic-composition has no use for a left-right distinction.

Logically speaking, the Lambek Calculus falls neatly into the sub-structural hierarchy as follows.

system	sub-structural features			
intuitionistic logic	monotonicity	contraction	commutation	association
relevance logic	–	contraction	commutation	association
linear logic	–	–	commutation	association
Lambek calculus	–	–	–	association

²⁶ Joachim Lambek, "The Mathematics of Sentence Structure", *American Mathematical Monthly* 65 (1958) , 154–170.

²⁷ The order of arguments is accordingly different; \mathcal{A}/\mathcal{B} corresponds to $\mathcal{B} \rightarrow \mathcal{A}$.

²⁸ The history of mathematics includes an interesting syntactic conflict between algebraists, who *traditionally* place the argument first, and analysts, who *traditionally* place the function first.

2. Contraction (※)

$A \rightarrow (A \rightarrow B) \vdash A \rightarrow B$

(1)	$A \rightarrow (A \rightarrow B)$	1	Pr		
(2)	A	2	As		
(3)	$A \rightarrow B$	12	$1,2, \rightarrow O$		
(4)	B	122	$2,3, \rightarrow O$	※	$\{2\} \subseteq \{2,3\}$
(5)	$A \rightarrow B$	1	$2,4, \rightarrow I$		

3. Generalized-Conjunction

$A \rightarrow B ; A \rightarrow C ; (B \times C) \rightarrow D \vdash A \rightarrow D$

(1)	$A \rightarrow B$	1	Pr		
(2)	$A \rightarrow C$	2	Pr		
(3)	$(B \times C) \rightarrow D$	3	Pr		
(4)	A	4	As		
(5)	B	14	$1,4, \rightarrow O$		$\{1\} \not\subseteq \{4\}$
(6)	C	24	$2,4, \rightarrow O$		$\{2\} \not\subseteq \{4\}$
(7)	$B \times C$	1244 [=124]	$5,6, \times I$		$\{1,4\} \not\subseteq \{2,4\}$
(8)	D	1234	$3,7, \rightarrow O$		$\{3\} \not\subseteq \{1,2,4\}$
(9)	$A \rightarrow D$	123	$4,8, \rightarrow I$		

8. Summary of Systems Considered

1. Examples of Arguments Valid in all Systems [G, L, R, I, C]

(1)	$B \rightarrow C ; A \rightarrow B \vdash A \rightarrow C$	[Transitivity]
(2)	$A \rightarrow (B \rightarrow C) \vdash B \rightarrow (A \rightarrow C)$	[Permutation]
(3)	$A \rightarrow (B \rightarrow C) ; B \vdash A \rightarrow C$	[Secondary <i>Modus Ponens</i>]
(4)	$A \vdash (A \rightarrow B) \rightarrow B$	[Montague's Law]
(5)	$(A \rightarrow C) \rightarrow D ; A \rightarrow B \vdash (B \rightarrow C) \rightarrow D$	[Inflection]
(6)	$(A \rightarrow C) \rightarrow D ; A \rightarrow (B \rightarrow C) \vdash B \rightarrow D$	[Permutivity]
(7)	$B \rightarrow C ; A \times B \vdash A \times C$	[Addition]
(8)	$(A \times B) \rightarrow C \dashv\vdash A \rightarrow (B \rightarrow C)$	[Schönfinkel's Law]

2. G-Valid Arguments Rejected by L

(1)	$A \rightarrow B ; A \rightarrow C \vdash A \rightarrow (B \times C)$	[Conditional Multiplication]
(2)	$A \rightarrow (B \rightarrow C) ; A \rightarrow B \vdash A \rightarrow C$	[Conditional Modus Ponens]
(3)	$A \rightarrow B ; A \rightarrow C ; (B \times C) \rightarrow D \vdash A \rightarrow D$	[Generalized Conjunction]

3. R-Valid Arguments Rejected by G

(1)	$A \rightarrow (A \rightarrow B) \vdash A \rightarrow B$	[Contraction]
(2)	$A \vdash A \times A$	[Duplication]
(3)	$\vdash A \rightarrow A$	[Tautology]
(4)	$(A \rightarrow A) \rightarrow B \vdash B$	[Law of Assertion]

4. I-Valid Arguments Rejected by R

- | | | |
|-----|--|--------------------|
| (1) | $A \vdash B \rightarrow A$ | [Positive Paradox] |
| (2) | $A \rightarrow B \vdash A \rightarrow (A \rightarrow B)$ | [Expansion] |
| (3) | $A \vdash B \rightarrow B$ | [Irrelevance] |
| (4) | $A \times B \vdash A$
$A \times B \vdash B$ | [Simplification] |

5. C-Valid Arguments Rejected by I

- | | | |
|-----|--|------------------------------------|
| (1) | $(A \rightarrow B) \rightarrow B \vdash (B \rightarrow A) \rightarrow A$ | [Lukasiewicz's Law ³⁰] |
| (2) | $(A \rightarrow B) \rightarrow A \vdash A$ | [Peirce's Law ³¹] |

³⁰ We name this principle after Jan Lukasiewicz (1878-1956), who invented multi-valued logic. In his system, one can define disjunction in terms of conditional via: $A \vee B =_{\#} (A \rightarrow B) \rightarrow B$.

³¹ Named after Charles Sanders Peirce (1839-1914) first presented in "On the Algebra of Logic: A Contribution to the Philosophy of Notation", *American Journal of Mathematics* 7, 180–202 (1885).