Display Calculi meet Categorial Type Logics

Rajeev P. Goré

Automated Reasoning Group
Computer Sciences Laboratory
Research School of Information Sciences and Engineering
Australian National University
Canberra, ACT, 0200, Australia
ANU CRICOS Provider Number - 00120C
Rajeev.Gore@anu.edu.au

and

Raffaella Bernardi

KRDB, Faculty of Computer Science
Free University of Bolzano-Bozen
P.zza Domenicani 3
39100 Bolzano-Bozen, Italy
bernardi@inf.unibz.it

The authors thank the Free University of Bolzano-Bozen for the travel and accommodation subsidizes provided to the second teacher and for the financial support given to the first teacher while visiting the KRDB group in the preparatory phase of this course. Furthermore, a special thank goes to the ESSLLI local organizers for their financial and technical support.

Contents

Abstract

Content The work presented in this course follows the *parsing as deduction* approach to Linguistics. We use the tools of Categorial Type Logic (CTL) to study the interface of natural language syntax and semantics. Our aim is to investigate the mathematical structure of CTL and explore its expressivity for analyzing natural language structures.

Standard presentations of CTL using Gentzen's Sequent Calculus or Proofnets are prototypical examples of a logical account of natural language analysis. Nevertheless, the limitations of these proof systems surface when one tries to fully explore the 'theoretical space' for type-logical connectives. We show how the generalized sequent framework of Display Calculi (DC) improves upon the traditional accounts, and in particular, how it opens new perspectives on the treatment of negative structure.

Such expressivity is required if one wants to enrich the type-logical vocabulary with antitone connectives, in addition to the more familiar isotone operations. Phenomena of polarity sensitivity, pervasive in natural languages, suggest that such extensions are desirable.

Structure The course consists of two main components: The proof theoretical & model theoretical background of CTL and the modeling of linguistic phenomena. Through the course these two components will be mixed after a first separate introduction to each of them providing the technical background and the motivations of the approach.

Pre-requisite The course is addressed to any student interested in logic, language and their connections, and only basic background on propositional logic is required. Students with knowledge of categorial grammar and proof theory will, of course, obtain further profit.

ESSLLI's related courses The course can be seen as a continuation of the foundational course on "Substructural Logics" given by Greg Restall at ESSLLI 01 (Helsinki). In particular, we will show substructural logics at work by focusing on a specific one (CTL) and by zooming in on the linguistic applications only mentioned by Restall.

Three closely related courses given at ESSLLI 04 are the introductory courses on "Type logical grammar" and "Proof automation for type-logical grammars" (by Glyn

Morrill and Pierre Casteran & Richard Moot, respectively.) and the advanced course on "Multi-modal combinatory categorial grammar" (by Geert-Jan Kruijff).

References The course notes have been mostly extracted from [Ber02] and [Gor98c, Gor97]. In the course, at the end of each lecture we will provide further reading material that can be downloaded from the CoLogNET portal of the "Logic and Natural Language Processing Area": `http://colognet.let.uu.nl`.

Exercises For the linguistic applications, we will propose some exercises to check your understanding of the presented material. You will be able to check them your selves, via the on-line version of `Grail` at: `http://grail.let.uu.nl`.

Part I

Introduction to DC and CTL

In this part of the notes, we introduce the formal background behind the Grammar Logic we will put at work in the second part of the notes. In particular, we focus attention on the type-forming operations of Categorial Type Logic (CTL), which build structured complex categories out of a small set of basic types. We show how the analytical force of these type-forming operations derives from the fact that they come in pairs of opposites —residuated or Galois connected pairs, to be more precise. We have all become acquainted with these notions in our elementary math classes, when we learned how to solve algebraic equations like $3 \times x \leq 5$ by 'isolating' the unknown x using the laws connecting $(\times, \div)$, producing the solution $x \leq \frac{5}{3}$. In categorial grammar, such pair of opposites is used to put together and take apart linguistic expressions, both syntactically and semantically.

In Chapter 1, we try to set CTL within the "Language & Logic" big ombrella, briefly pointing to its main pecularities.

In Chapter 2, we provide the algebraic notions of *residuation* and *Galois* that govern the behavior of the type-forming operations of CTL and introduce Display Calculi (DC) as a unified framework for presenting resource and structure sensitive reasoning.

In Chapter 3, we introduce CTL by means of DC sheding lights on the proof theoretical beahvior of CTL operators. Moreover, we study the derivability relations we will see at work in Part II.

Chapter 1

Where do we stand?

This course is part of the "Logic & Language" Session of ESSLLI'04. As such, we would like to start in this preliminary and informal chapter from the very broad area and zoom into the framework we will be introducing in the course. In this way we hope to help the reader grasping the main properties of the framework and spotting similarities and differences with what he/she labels as "Logic & Language". We will start from Computational Linguistics, move to Formal Grammars and end with Logical Grammars.

1.1 Formal Grammars

Formal Grammars are an area of investigation of *Computational Linguistics*. The field is as diverse as linguistics itself, in general it could been seen as the study of *formal devices* which can be used to distinguish grammatical and ungrammatical strings of a given language [CCCSS00] and build their semantic interpretation. These formal devices can be automata, which are abstract computing machines, and formal grammars, which are string (or structure) rewriting systems. Formal Grammars are used by parsers to determine the "syntactic structure" of string of words. See [Mit03] for an overview of the field.

The object of this course is a formal grammar. Here we point out the major tasks of formal grammars and the devices that are used to carry them out by briefly comparing Context-Free Phrase Structure Grammars with the framework object of our studying, Categorial Grammars (CG)[1].

Phrase Structure Grammars. In formal language theory a language is defined as a set of strings, i.e. a set of finite sequences of vocabulary items. A grammar for a language then is a formal device that defines which strings belong to that language. One particular kind of formal system that is used to define a language is commonly known as a *context-free phrase structure grammar*. Besides deciding whether a string

[1]This section is extracted (and adapted) from [Hey99] with the author's permission. We thank Dirk for this.

belongs to a given language, this grammar deals with phrase structures represented as trees.

An important difference between strings and phrase structures is that whereas string concatenation is assumed to be associative, trees are bracketed structures. The latter thus preserve aspects of the compositional (constituent) structure or derivation which is lost in the string representations. We illustrate the behavior of this grammar by means of the example below.

EXAMPLE 1.1. [Grammars and Languages]We consider a small fragment of English defined by the following grammar G ($= \langle \mathsf{LEX}, \mathsf{RULES} \rangle$), with vocabulary Σ and categories CAT.

$\Sigma = \{\text{Sara}, \text{dress}, \text{wears}, \text{the}, \text{new}\}$,
$\mathsf{CAT} = \{det, n, np, s, v, vp, adj\}$,
$\mathsf{LEX} = \{\langle \text{Sara}, np\rangle, \langle \text{the}, det\rangle, \langle \text{dress}, n\rangle, \langle \text{new}, adj\rangle, \langle \text{wears}, v\rangle\}$
$\mathsf{RULES} = \{s \rightarrow np\ vp,\ np \rightarrow det\ n,\ vp \rightarrow v\ np, n \rightarrow adj\ n\}$

Among the elements of the language recognized by the grammar, $L(G)$, are $\langle$the, $det\rangle$ because this is in the lexicon, and $\langle$Sara wears the new dress, $s\rangle$ which is in the language by the repeated application rules.

(1) $\langle$new dress, $n\rangle \in L(G)$ because
$n \rightarrow adj\ n \in \mathsf{RULES}$,
$\langle$new, $adj\rangle \in L(G)$ (LEX), and
$\langle$dress, $n\rangle \in L(G)$ (LEX)

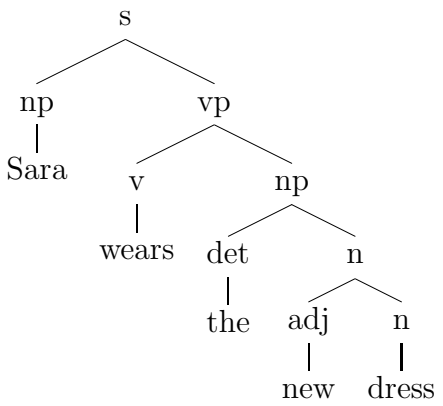
(2) $\langle$the new dress, $np\rangle \in L(G)$ because
$np \rightarrow det\ n \in \mathsf{RULES}$,
$\langle$the, $det\rangle \in L(G)$ (LEX), and
$\langle$new dress, $n\rangle \in L(G)$ (1)

(3) $\langle$wears the new dress, $vp\rangle \in L(G)$ because
$vp \rightarrow v\ np \in \mathsf{RULES}$,
$\langle$wears, $v\rangle \in L(G)$ (LEX), and
$\langle$the new dress, $np\rangle \in L(G)$ (2)

(4) $\langle$Sara wears the new dress, $s\rangle \in L(G)$ because
$s \rightarrow np\ vp \in \mathsf{RULES}$,
$\langle$Sara, $np\rangle \in L(G)$ (LEX), and
$\langle$wears the new dress, $vp\rangle \in L(G)$ (3)

In a similar way we can show that the set of phrase structure trees ($T(G)$) contains the following tree which we also depict in the usual graphical way.

$\langle\langle \text{Sara}, np\rangle, \langle\langle \text{wears}, v\rangle, \langle\langle \text{the}, det\rangle, \langle\langle \text{new}, adj\rangle, \langle \text{dress}, n\rangle, n\rangle, np\rangle, vp\rangle, s\rangle$



As the following table shows, we can also derive that the sentence is in the language by means of a rewriting procedure. The left column represents the sequence of categories and words that is arrived at by replacing one of the categories (c) (identical to the left-hand side of the rule in the second column) on the line above by the right-hand side of the rule or by a word that is assigned the category c by the lexicon.

s	$s \rightarrow np\ vp$
np vp	$vp \rightarrow v\ np$
np v np	$np \rightarrow det\ n$
np v det n	$n \rightarrow adj\ n$
np v det adj n	$\langle$Sara, $np\rangle$
Sara v det adj n	$\langle$wears, $v\rangle$
Sara wears det adj n	$\langle$the, $det\rangle$
Sara wears the det adj n	$\langle$new, $adj\rangle$
Sara wears the new n	$\langle$dress, $n\rangle$
Sara wears the new dress	

This example recapitulates the basic definitions of grammar and language isolating some of the functions a grammar is required to serve. In particular, it shows that the principal tasks of a grammar are the definition of several sets of objects: (i) a set of expressions (the string set), (ii) a set of pairs of expressions and categories (language), (iii) a set of phrase structures. More abstractly, we can say that a grammar is a device that is concerned with two aspects.

(1) Defining the membership of elements to some (sub)set: *classification* or categorisation.

(2) Specifying the *compositional* structure of complex elements.

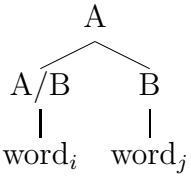
The grammars that we introduced categorise expressions in two steps. The lexicon component LEX of a grammar deals with the categorisation of the atomic expressions (the words), whereas the RULES determine the category of complex expressions. The rules also take care of the definition of compositional structure. They define which parts can combine to form larger structures. Categories play an important role in defining the compositional structure because they group together the set of expressions that behave similarly with respect to compositional structure. In other words, the rules make reference to the categories, not to the individual expressions.

Besides classification, another important function of a grammar is to define structure. Analysing this aspect, we can say that the context-free grammar fixes the order of (sub)expressions (precedence) and a part-whole structure (dominance). Phrase structure trees make this compositional structure and the categorisation of wholes and parts explicit. We will now turn to another type of grammar that performs the same tasks in a different way and which is the ancestor of the logical grammar presented in the course.

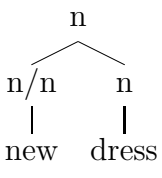
Categorial Grammars. The formal grammar we work with, Categorial Grammars (CG), adopts an alternative way to define classification and composition. In this type of grammar the lexical component takes over the role of the rule component in fixing the compositional structure of a language. It is important to notice how in order to effect this, the notion of category is modified.

CG have a richer notion of category than Phrase Structure Grammars. Besides atomic categories, they also use complex categories of the form A/B and $(B\backslash A)$ (where A and B are again categories, possibly complex themselves.) A language is defined solely by a lexicon that assigns categories to words. There are no phrase structure rules. Instead, a pair of general combination operations is used: (1) an expression in A/B concatenated with an expression of category B gives an expression in A (see (a) below); (2) an expression in B concatenated with an expression in $B\backslash A$ gives an expression in A (the schema is symmetric to (a)). Notice that these schemata do not mention specific categories, but use variables over arbitrary categories A, B. The information about precisely which expressions combine with others is provided by the lexicon. For instance, an adjective, such as *new*, might be assigned the category n/n, which, by the general schema, means that it combines with expressions in n to form expressions in n. Representing this composition in a tree we have (a') that plays the role of the phrase structure in (b).

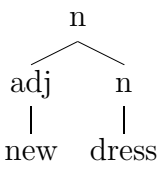
(a)



(a')



(b)



Note that CG categories could be seen as trees themselves and therefore are close to the elementary trees of Tree Adjoining Grammar (TAG). The reader is referred to the foundational course given by B. Gaiffe and G. Perrier "Basic Parsing Techniques for natural language" (ESSLLI 04) for an introductory comparison of CG and TAG and to [Moo02] for a formal embedding of (a version of) the former into the latter.

From this elementary summary, it should be clear how classification and composition are defined in this alternative framework. One thing that is important to underline is that the classification of expressions by the categories directly expresses their combination properties. These aspects will be spelled out in precise terms in Chapter 4 where it is also shown how moving from Classical CG to Categorial Type Logic (CTL) corresponds to move from a formal grammar to a logic grammar where categories are

logical formula and rules are logical rules. Here we try to briefly explain the place CTL occupies within the "Logic and Natural Language Processing" (LNLP) area.

1.2 Logic and NLP

In several fields Logic is used to grasp the global features of reality abstracting away from the empirical details and providing a model of the observed objects. In our case, the object is natural language. Logic has been recognized to be an important tool for understanding the structure of language (both its syntax and semantics) since [Fre87]. Moreover, inference tools are a crucial component of Natural Language Processing (NLP) systems that require deep analysis which integrate semantic analysis of natural language utterances to their general cognitive processing. Valuable references for foundational and advanced introductions to the field are [PMW90, vBtM97].

Since 1995 an European Conference on *Logical Aspects of Computational Linguistics* (LACL) is organized every two years. In the introduction of LACL'96 proceedings the field is classified in (i) logical semantics of natural language, (ii) grammar and logic, (iii) mathematics with linguistic motivations, and (iv) computational perspectives. Following this division, if we have to chose one label, we can say that the framework presented in the course follows under the "grammar and logic" one. As before, we would like to highlight the main aspects in which CTL differs from other logical views on grammars.

Logic Based Grammars. One known example of a family of logics used to accomplish NLP tasks are Feature Logics [Rou97]. They are known to be especially useful in classifying and constraining the linguistic objects known as feature structures. Here, the logic is used to *formalize* linguistic theories. For instance, the feature co-occurrence restriction used in Generalized Phrase Structure Grammar (GPSG) to say that any category classified as a verb must have a negative "noun" and a positive "verb" quality

$$[VFORM] \supset [-N, +V]$$

can be represented in modal feature logic by means of the universal modality $\Box$ as following

$$\Box(vform : ture \rightarrow n : - \wedge v : +)$$

In other words, a linguistics rule is "translated" into a logical language and then the formal system is used to reason with the formalized linguistic objects.

Contrary to this, in CTL there is no such distinction between an a-priori grammar (with its set of grammatical rules) and a logic in which the former is translated. The grammar is in itself a logic, and the only rules at work are the (logical and structural) rules of the system obtained by investigating its algebraic structure. The reader is invited to compare the use of the universal modality above with its application in CTL explained in Part II.

As fully explained in these course notes, CTL is a (modal) logic used to reason on linguistic signs. Syntactic categories are formulas, grammatical rules are logical rules, and structural (re)ordering corresponds to structural rules (or frame constraints). As

such, *soundness* and *completeness* properties do not simply characterize the logic, but rather are main properties of the grammar itself. The challenge is to investigate the 'theoretical space' of the family of (modal) logics and select the proper model of natural language grammar suitable for the specific natural language to be generated. In the course, we will show how Display Calculi help carrying out this task.

Resources Sensitivity. A second crucial point is the *resource sensitivity* notion that governs the manner in which the operations of a system can utilize its resources: how often they may be used, how they can be assembled together to create larger structures, and how they can be reconfigured into other equivalent structures.

From a logical perspective, resource sensitivity is nicely exemplified in Linear Logic [Gir87] where the implicational connective $-\!\circ$ is used to indicate that it consumes the resource that is needed to prove the consequent; thus, the formula $p-\!\circ q$ is read as 'consume p yielding q'. After the rule for $-\!\circ$ is applied, the resource p is no longer available for further inferential steps.

In Linear Logic, Contraction and Multiplication of resources are reintroduced locally by means of unary operators $(!, ?)$ which are used to mark those resources for which the corrisponding structural rules are adequate. The use of these "markers" introduces the idea of structural rules that are controlled rather then globally available.

The logical concerns behind this logic have direct parallels in natural language grammar and are reflected in CTL which is a resource-sensitive logic (see [Moo99] for more details). Clearly, the multiplicity of linguistics material is important, since linguistic elements must generally be used once and only once during an analysis. Thus, we cannot ignore or waste linguistic material (1-a), nor can we indiscriminately duplicate it (1-b).

(1) a. *The coach smiled the ball. $\neq$ The coach smiled.

b. *The coach smiled smiled. $\neq$ The coach smiled.

As in other domains, in Linguistic as well, there is the need of locally controlling structural reasoning and account for the different compositional relations linguistic phenomena may exhibit. As in Linear Logic, in CTL as well this control is expressed by means of unary operators. However, the unary operators of CTL belong to the same algebraic structure of the implication operators used for simple composition of structures. This algebraic property and its encoding into sequent calculi is the topic of Chapter 3.

Statistical and Logic Based Approaches. We would like to conclude this part with a last comment on the possible application of logical grammars. As pointed out in [Car03], while, logic oriented linguists are more interested in selecting the proper logic for modeling natural language, computer scientists typically worry about efficiency of processing in terms of time and space, often side-stepping cognitive issues in the interest of building effective software. Within the computational linguistic community, this interest is reflected in the development of statistical models that have largely supplanted logical ones. Statistical algorithms search for the "best" structure which is assumed to be the one with the highest likelihood and perform much better than any logic-based

grammar system when put at work with large data, as needed in several applications. Rather than seeing this as a failure of the logical approach, in the LNLP area the integration of the inductive and the deductive styles of grammatical reasoning is perceived as an exciting challenge for the field. (See [Per00] for an interesting discussion.) As for the computational aspects of CTL and related grammars, the reader is refered to the courses given at ESSLLI'04 by P. Casteran and R. Moot (Proof automation for type-logical grammars) and by G. Jaeger and J. Michaelis (An introduction to mildly context-sensitive grammar).

Chapter 2

Introduction to DC

We introduce Display Calculi (DC) as a unified framework for presenting resource and structure sensitive reasoning. We use the term "Display Calculi" rather than "Display Logic" because display logic is really not a logic per se but a proof-theoretic framework for generalised sequent calculi.

Since this is an advanced course, we are going to assume that you know about Hilbert calculi, about $(n+1)$-ary relational semantics for n-ary non-classical connectives a là Jónsson and Tarski, and about sequent calculi. But don't worry: more background information will be provided in the lectures themselves if necessary.

2.1 Preliminaries

We first introduce some notions from algebra which play a central role in display calculi but which are important in their own right. Most of this is based upon the work of J Michael Dunn [Dun93].

Let $\mathcal{A} := (A, \leq_A)$ and $\mathcal{B} := (B, \leq_B)$ be two partially ordered sets (so that both $\leq_A$ and $\leq_B$ are reflexive and transitive). A pair (f, g) of unary functions $f : A \mapsto B$ and $g : B \mapsto A$ form a:

Residuated Pair if	$fa \leq_B b$	iff	$a \leq_A gb$
Dual Residuated Pair if	$b \leq_B fa$	iff	$gb \leq_A a$
Galois Connection if	$b \leq_B fa$	iff	$a \leq_A gb$
Dual Galois Connection if	$fa \leq_B b$	iff	$gb \leq_A a$

We shall assume that $\mathcal{A} = \mathcal{B}$ since our domain will always be the set of "formulae", thereby dropping the subscripts on the order $\leq$. A dual residuated pair (f, g) is then just a residuated pair (g, f) so we shall not consider dual residuated pairs further.

Given a partially ordered set $\langle A, \leq \rangle$, an n-ary function f is isotone (sometimes called monotonic) in its i-th position if for all a_i and all a and b

$$a \leq b \text{ implies } f(a_1, \cdots, a_{i-1}, a, a_{i+1}, \cdots, a_n) \leq f(a_1, \cdots, a_{i-1}, b, a_{i+1}, \cdots, a_n)$$

It is antitone (sometimes called anti-monotonic) if

$$a \leq b \text{ implies } f(a_1, \cdots, a_{i-1}, b, a_{i+1}, \cdots, a_n) \leq f(a_1, \cdots, a_{i-1}, a, a_{i+1}, \cdots, a_n)$$

Dunn also gives equivalent definition for characterising residuated pairs and Galois connections using the "tonicity" of the functions involved as:

Residuated Pair: Both f and g are isotone, and $fgb \leq_B b$ and $a \leq_A gfa$

Galois Connection: Both f and g are antitone, and $b \leq_B fgb$ and $a \leq_A gfa$

Dual Galois Connection: Both f and g are antitone, and $fgb \leq_B b$ and $gfa \leq_A a$.

As we shall see, unary modalities often appear in pairs which form residuated pairs, while negations often appear in pairs which form Galois connections and dual Galois connections. These notions generalise to binary functions (connectives) as follows.

A partially ordered groupoid is a triple $\langle A, \circ, \leq \rangle$ where A is a non-empty set, $\circ$ is a binary operation on A which is isotonic in each place, and $\leq$ is a partial order on A. The isotonicity condition is simply this: if $a \leq b$ then $a \circ c \leq b \circ c$ and $c \circ a \leq c \circ b$.

A partially ordered groupoid is

Left Residuated: when an element $c \leftarrow b$ always exists such that $a \circ b \leq c$ iff $a \leq c \leftarrow b$

Right Residuated: when an element $a \rightarrow c$ always exists such that $a \circ b \leq c$ iff $b \leq a \rightarrow c$

Residuated: when it is both left and right residuated.

An equivalent definition is:

Left Residuated when $\circ$ is isotonic in each place and $a \circ (a \rightarrow b) \leq b$ and $b \leq a \rightarrow (a \circ b)$.

Right Residuated when $\circ$ is isotonic in each place and $(b \leftarrow a) \circ a \leq b$ and $b \leq (b \circ a) \leftarrow a$.

2.2 Introduction

2.2.1 Residuation and Galois Connected Logical Connectives

We now give a informal account of the logical connectives that correspond to the algebraic principles described above. Their main features are that they are based on residuation and galois connections, and that they are a "resource sensitive", a notion that is explained in more detail later.

We assume the existence of a semantic consequence relation $\Gamma \models A$ which captures the notion that formula A follows semantically from a collection of formulae Γ. This depends upon the semantics of the logic in question. As we shall see, resource sensitivity means that we must account for the order of formulae in Γ, the number of copies of a formula in Γ as well as whether that formula is really needed in order to conclude A from Γ.

2.2.2 Unary Modalities

By unary modalities we mean isotonic unary connectives usually called "box" and "diamond".

Syntax: We shall use $\Box$ and $\Diamond$. We shall also use $\Diamond^{\downarrow}$ and $\Box^{\downarrow}$.

Semantics: The Kripke semantics of such (unary) modalities is given by a binary relation R over the underlying set of "worlds" or "points" or "situations" W. To such frames $\langle W, R\rangle$, we add a valuation v mapping atomic formulae like p to the subsets of W where the atomic formulae are true. For any $x \in W$, we write $x \Vdash p$ when $x \in v(p)$ and extend this to the modalities in the following way:

$x \Vdash \Diamond A$ iff $\exists y.\ R(x,y)$ and $y \Vdash A$
$x \Vdash \Box A$ iff $\forall y.$ if $R(x,y)$ then $y \Vdash A$
$x \Vdash \Diamond^{\downarrow} A$ iff $\exists y.\ R^{-1}(x,y)$ and $y \Vdash A$
$x \Vdash \Box^{\downarrow} A$ iff $\forall y.$ if $R^{-1}(x,y)$ then $y \Vdash A$

Correspondence: Certain formula shapes correspond to certain first-order properties (frame conditions) of R. For example, the reflexivity condition $\forall x.R(x,x)$ corresponds to the shape $\Box A \Rightarrow A$. That is, a frame $\mathfrak{F}$ validates all instances of the formula shape $\Box A \Rightarrow A$ iff the underlying binary relation R has the property of being reflexive.

Residuation: The diamond and box modalities are connected via residuation as follows:

$\Diamond^{\downarrow} A \models B$ iff $A \models \Box B$
$\Diamond A \models B$ iff $A \models \Box^{\downarrow} B$

We have already mentioned that the notion of dual residuation does not add anything of value when we have the same underlying partially ordered set, so unary modalities can also be seen to be related by dual residuation.

2.2.3 Unary Negations

By unary negations we mean anti-tonic unary connectives usually called "not".

Syntax: We shall use $\neg$ and $\sim$.

Semantics: The Kripke semantics of such (unary) negations is given by a binary relation R (of incompatibility) over the underlying set of "worlds" or "points" or "situations" W. To such frames $\langle W, R\rangle$, we add a valuation v mapping atomic formulae like p to the subsets of W where the atomic formulae are true. For any $x \in W$, we write $x \Vdash p$ when $x \in v(p)$ and extend this to the modalities in the following way:

$x \Vdash \neg A$ iff $\forall y.$ if $R(x,y)$ then $y \nVdash A$
$x \Vdash \sim A$ iff $\forall y.$ if $R^{-1}(x,y)$ then $y \nVdash A$

Correspondence: Certain formula shapes correspond to certain first-order properties (frame conditions) of R.

Galois Connection: The two negations are connected via the laws of Galois connections as follows:

$$A \models \neg B \quad \text{iff} \quad B \models\!\sim A$$

The fact that the connectives are anti-tonic is reflected in the fact that the subformulae A and B switch sides under the Galois laws while the negation itself changes but remains on the same side of $\models$.

The negations described above have a "box"-like flavour to them since they are couched in terms of all R-successors and all R-predecessors.

There are also negations that can be captured by dual Galois connections, and these have a "diamond" like flavour. We omit details because these take us beyond the scope of our lectures.

2.2.4 Binary Connectives

By binary connectives we mean non-classical conjunctions, disjunction and implications. They are sometimes called intensional conjunction and disjunction.

Syntax: We shall use $\otimes$ for intensional conjunction, $\oplus$ for intensional disjunction, and $\rightarrow$ and $\leftarrow$ for the two implications which arise naturally from the Lambek calculus.

Semantics: The Kripke semantics of such (binary) modalities is given by a ternary relation R over the underlying set of "worlds" or "points" or "situations" W. Again, adding a valuation gives a model in which we evaluate the intensional connectives at any particualr $x \in W$ as below:

$$x \Vdash A \otimes B \quad \text{iff} \quad \exists y, z.\ R(x, y, z) \text{ and } y \Vdash A \text{ and } z \Vdash B$$
$$x \Vdash A \rightarrow B \quad \text{iff} \quad \forall y, z.\ \text{if } R(x, y, z) \text{ and } y \Vdash A \text{ then } z \Vdash B$$
$$x \Vdash A \leftarrow B \quad \text{iff} \quad \forall y, z.\ \text{if } R(x, y, z) \text{ and } z \Vdash A \text{ then } y \Vdash B$$

Correspondence: Certain formula shapes correspond to certain first-order properties (frame conditions) of R.

Residuation: These three connectives are related via residuation as set out below:

$$A \models C \leftarrow B \quad \text{iff } A \otimes B \models C \quad \text{iff } B \models A \rightarrow C.$$

2.2.5 Resource Sensitivity

The connectives of classical logic have some rather strong properties such as the fact that $A \wedge A \Leftrightarrow A$. Thus one occurrence of the formula A is equal in logical power to many occurrences of A by universal substitution of logical equivalents. Resource sensitive logics break such strong properties and count each occurrence of A.

More formally, suppose we consider a Hilbert derivation of A from a collection of formulae Γ as a finite sequence of formulae $\langle A_0, A_1, \cdots, A_n \rangle$ where A_n is A and each formula in the sequence is either an instance of an axiom schema, or is obtained from previous members of the sequence via a rule of inference like modus ponens or necessitation. We can then count the number of times that a particular formula is "used" in

such inference rules. For example, if we use p and $p \Rightarrow q$ to infer q and then use p and $p \Rightarrow r$ to infer r then we have used p twice.

If we want to track such usage then we must speak of formula occurrences and we must keep track of the usage of such occurrences. Such logics are traditionally known as resource sensitive logics, but can be categorised into three broad classes:

Use At Most Once: In these logics, the formula $A \Rightarrow (A \wedge A)$ is not valid but $A \wedge B \Rightarrow A$ is valid. Consequently, each occurrence of A can be utilised at most once since we cannot produce more occurrences of A simply by utilising $A \Rightarrow (A \wedge A)$ to produce more occurrences.

Use At Least Once: In these logic, the formula $A \wedge B \Rightarrow A$ is typically not valid, but $A \Rightarrow (A \wedge A)$ is valid. In particular, $A \wedge A \Rightarrow A$ is not valid. Consequently, each occurrence of $A \wedge A$ must be utilised at least once since we cannot gain access to only a portion of it via $A \wedge A \Rightarrow A$.

Use Exactly Once: If neither $A \Rightarrow (A \wedge A)$ nor $A \wedge B \Rightarrow A$ is valid then we obtain the motivation of linearity, each formula occurrence must be utilised exactly once.

Bounded Logics: Once we can count the number of uses of a formula occurrence, it is natural to consider logics where each occurrence can be used exactly n times for some finite n.

2.3 Proof Theory

2.3.1 Axiomatic Presentation

As is usual in Hilbert calculi, there are many axiom schema and very few inference rules. For modalities we typically need the K axiom

$$\Box(A \Rightarrow B) \Rightarrow (\Box A \Rightarrow \Box B) \qquad \Box^{\downarrow}(A \Rightarrow B) \Rightarrow (\Box^{\downarrow} A \Rightarrow \Box^{\downarrow} B)$$

and axioms which guarantee the residuation conditions such as:

$$A \Rightarrow \Box\Diamond^{\downarrow} A \qquad \Diamond\Box^{\downarrow} A \Rightarrow A$$

together with the necessitation rules:

$$\text{Nec-}\Box \ \frac{\vdash A}{\vdash \Box A} \qquad \text{Nec-}\Box^{\downarrow} \ \frac{\vdash A}{\vdash \Box^{\downarrow} A}$$

For the binary connectives we typically require axioms like $A \wedge B \Rightarrow A$ and $A \wedge B \Rightarrow B$, which capture resource sensitivity. But often, Hilbert axiomatisations are given in terms of implicational axioms only. We omit details as this would take us too far afield.

The main problem with Hilbert calculi is that they do not lend themselves to computation or automation. It is not at all obvious how to find derivations because the right instantiation of a particular axiom often requires an "Eureka" step.

$$\text{(ctrL)}\ \frac{\Gamma, A, A \vdash \Delta}{\Gamma, A \vdash \Delta} \qquad \text{(ctrR)}\ \frac{\Gamma \vdash A, A, \Delta}{\Gamma \vdash A, \Delta}$$

$$\text{(wkL)}\ \frac{\Gamma, A_i \vdash \Delta}{\Gamma, A_1, A_2 \vdash \Delta}\ (i \in \{1,2\}) \qquad \text{(wkR)}\ \frac{\Gamma \vdash A_i, \Delta}{\Gamma \vdash A_1, A_2, \Delta}\ (i \in \{1,2\})$$

$$\text{(prmL)}\ \frac{\Gamma, B, A \vdash \Delta}{\Gamma, A, B, \vdash \Delta} \qquad \text{(prmR)}\ \frac{\Gamma \vdash B, A, \Delta}{\Gamma \vdash A, B, \Delta}$$

Figure 2.1: Sequent Structural Rules

2.3.2 Sequent Calculi

The traditional way to examine the subtle nature of resource sensitivity is to use Gentzens's sequent calculi. Here, the building blocks are sequents of the form $\Gamma \vdash \Delta$ where the Γ and Δ are comma-separated sequences of formula occurrences. Thus, we have built-in the notion that the sequence A, B is different from the sequence B, A, and that the sequence A is different from the sequence A, A.

A sequent calculus then consists of a finite collection of rules of the form

$$\text{(name)}\ \frac{\Gamma_1 \vdash \Delta_1 \quad \Gamma_2 \vdash \Delta_2 \quad \cdots \quad \Gamma_n \vdash \Delta_n}{\Gamma \vdash \Delta}$$

where the sequents above the line are called the premises and the sequent below the line is called the conclusion.

There is a finite collection of initial sequents which are rules without premises such as the rule

$$p \vdash p$$

since this sequent requires no premise as justification.

A derivation of $\Gamma \vdash \Delta$ is a tree of sequents where the root is the sequent $\Gamma \vdash \Delta$, each leaf is an instance of an initial sequent, and every sequent in the tree is obtained from its children by instantiating a rule of the calculus.

For every logical connective, there is a rule that introduces (creates) that connective on the left of the sequent arrow, and another that introduces (creates) that connective on the right of the sequent arrow. For example, the introduction rules for conjunction

are usually given as:

$$(\wedge\text{L})\ \frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta} \qquad (\wedge\text{R})\ \frac{\Gamma \vdash A, \Delta \quad \Gamma \vdash B, \Delta}{\Gamma \vdash A \wedge B, \Delta}$$

There are three rules which alter only the structure of sequents by adding, deleting or permuting the order of formulae called "contraction", "weakening" and "permutation" as shown in Figure 2.1.

The permutation rule allows us to re-arrange the order of formula occurrences in a sequence. It is usually present in most sequent calculi. Resource sensitivity is gained by including or omitting the other rules of contraction and weakening, which alter the number of formula occurrences in sequents. The three broad classes of resource sensitive logics are captured as explained below:

Use At Most Once: In these logics, the rules

$$(\text{ctrL})\ \frac{\Gamma, A, A \vdash \Delta}{\Gamma, A \vdash \Delta} \qquad (\text{ctrR})\ \frac{\Gamma \vdash A, A, \Delta}{\Gamma \vdash A, \Delta}$$

are omitted. If we now track the usage of A from the conclusion to the premise, then each occurrence of A can be utilised at most once since we cannot produce more occurrences of A as we go up the derivation by using the contraction rules, and the leaves must be of the form $p \vdash p$.

Use At Least Once: In these logics, the rules

$$(\text{wkL})\ \frac{\Gamma, A_i \vdash \Delta}{\Gamma, A_1, A_2 \vdash \Delta}(i \in \{1,2\}) \qquad (\text{wkR})\ \frac{\Gamma \vdash A_i, \Delta}{\Gamma \vdash A_1, A_2, \Delta}(i \in \{1,2\})$$

are omitted. Thus, as we go up the derivation, formulae occurrences cannot disappear. Every formula occurrence must be used at least once in order to end up with leaves of the form $p \vdash p$.

Use Exactly Once: If we omit both weakening and contraction rules then we obtain the basis of linearity, each formula occurrence must be utilised exactly once.

Bounded Logics: Bounded versions of weakening and contraction can be used where we are allowed to add or delete n occurrences of formulae.

2.3.3 Cut And Its Elimination

Most sequent calculi come equipped with a rule called the cut rule:

$$(\text{cut})\ \frac{\Gamma_1 \vdash A, \Delta_1 \quad \Gamma_2, A \vdash \Delta_2}{\Gamma_2, \Gamma_1 \vdash \Delta_2, \Delta_1}$$

where the formula A is known as the cut-formula.

We could write a whole book on the various ways of viewing this rule but here are some of its salient points:

Transitivity of $\vdash$: If Δ_1 and Γ_2 are both empty, then this rule captures the fact that the derivability relation $\vdash$ is transitive viz:

$$(\text{cut})\ \frac{\Gamma_1 \vdash A \quad A \vdash \Delta_2}{\Gamma_1 \vdash \Delta_2}$$

Mimicking Modus Ponens: An easy way to prove the completeness of a sequent calculus w.r.t. an existing Hilbert calculus is to use the cut rule to mimic modus ponens. To see this, we put Γ_1, Δ_1 and Γ_2 to empty and put Δ_2 equal to B:

$$(\text{cut})\ \frac{\vdash A \quad A \vdash B}{\vdash B}$$

Bi-valency of A: In all the logics we study, a formula will take one of two truth values. Viewed upwards, the cut rule can be seen to capture this bi-valency: that is, in order to find a derivation for $\Gamma_2, \Gamma_1 \vdash \Delta_2, \Delta_1$ we try the left premise assuming that A is false, and try the right premise assuming that A is true.

Unfortunately, the cut rule is bad for proof search since the move from the conclusion up to the two premises involves a formula A which may not appear at all in the conclusion. That is, we have to guess the right A in order to use the cut rule effectively.

The cut rule is also bad from a resource sensitive reading since it causes formula occurrences to disappear from a derivation as we go down. The contraction rule also has this effect, but we do not want this effect in "logics without contraction".

The main requirement of a good sequent calculus is then the ability to eliminate cut. There are two ways to view this and it is important to fix the nomenclature:

Cut Admissibility: We assume that the sequent calculus does not contain the cut rule and assume we are given two cut-free derivations of $\Gamma_1 \vdash A, \Delta_1$ and $\Gamma_2, A \vdash \Delta_2$ respectively. We then produce a cut-free derivation of $\Gamma_2, \Gamma_1 \vdash \Delta_2, \Delta_1$ from these two cut-free derivations.

Cut Elimination: We assume that the sequent calculus does contain cut and assume that we are given two derivations of $\Gamma_1 \vdash A, \Delta_1$ and $\Gamma_2, A \vdash \Delta_2$ respectively (which may contain multiple uses of cut). We further assume that these two derivations are the premises of a bottom-most instance of the cut rule with conclusion $\Gamma_2, \Gamma_1 \vdash \Delta_2, \Delta_1$. We then produce a cut-free derivation of $\Gamma_2, \Gamma_1 \vdash \Delta_2, \Delta_1$. One way to do this is to always attack the top-most cut in the given derivation, and in any intermediate derivations we produce. In which case, we mimick the cut-admissibility procedure since the sub-derivations above this top-most cut are guaranteed to be cut-free.

Both of these methods traditionally involve a double induction on the size of the cut-formula and the depth from the leaves of the occurrence of cut we are trying to eliminate. Often, the majority of a paper on a sequent calculus is taken up in proving this theorem. Needless to say, the intricacies of cut-elimination are beyond the scope of these lectures.

2.3.4 Why Move To Other Proof Systems?

Because Gentzen's comma is too limited to capture the diversity of the logical phenomena we are trying to captue in a disciplined and uniform way. The literature abounds with sequent calculi for many of these logics, but often, a small change in one rule necessitates alterations to other rules since sequent calculi are not modular. Moreover, the cut-elimination procedures are not modular, so that a small change in the sequent rules can necessitate a huge increase in the complexity of the cut-elimination procedure.

Various alternative sequent formulations have been studied and we give a flavour of what they involve, but a detailed study would take us too far afield.

Hyper-sequents: $\Gamma_1 \vdash \Delta_1 \mid \Gamma_2 \vdash \Delta_2 \mid \cdots \mid \Gamma_n \vdash \Delta_n$

A hyper-sequent is an ordered list of traditional sequents. The rules are built from hyper-sequents instead of sequents but keep the traditional n premises and one conclusion structure. There are extra rules to allow us to add or delete or permute members of these lists, and even rules that allow us to move formula occurrences from one member to another. Avron [Avr96] shows that many non-classical logics can be captured this way.

Multi-sequents: $\Gamma_e; \Gamma_i \vdash \Delta_i; \Delta_e$

A sequent is now composed of at least two compartments on each side: the "extensional" and "intensional" compartments. Each compartment is a comma-separated set, multiset or sequence of formulae occurrences. We can also have extra rules that move formula occurrences from the extensional to the intensional compartments, or vice-versa. See [Gir93, Cro0X] for details.

Higher-level Sequents: $\sigma \vdash^i \delta$

Sequents are now composed of sub-sequents in a tree-like fashion with each sequent arrow marked by a level. There are rules that allow us to move formulae from one level to another. See Dosen [Došš85] and Masini [Mas92] for details.

Proof Nets: If we trace a particular formula up a traditional sequent calculus derivation until it becomes principal, then trace its sub-formulae, we will eventually reach the leaves and find its constituent atomic formulae. Girard saw that the tree-like structure of sequents could be replaced by graphs where each node contains a formula, and the edges of the graphs represent the rules applied to obtain larger formulae from their sub-formulae. A certain subset of such graphical nets can be shown to capture sequent derivations. Again, the details are beyond the scope of this course. See [TS96] for details.

Display Calculi: The basic structural apparatus of sequents is made more complicated by allowing complex n-ary structural connectives in addition to Gentzen's comma. These are added and used in a disciplined way to ensure that each structural connective captures only one logical aspect of the logic, and that each such logical aspect can be included or omitted at will without affecting the other logical aspects of the logic.

2.4 Display Calculi

As we hope to show in this section, Display Calculi provide a powerful and unified proof-theoretical framework for Logic Engineering. The main advantage of display calculi is that they provide much finer control over the properties of individual connectives in a truly modular manner. This modularity is a by-product of a clear separation of logical and structural aspects of the logic in question using logical and structural rules which are couched in such a way that all logical aspects are captured by the logical rules and all other aspects are captured by structural rules. This is only possible by allowing complex unary and structural connectives which act as meta-level proxies for their logical counterparts in a manner that generalises the overloading of Gentzen's polyvalent "comma" as standing for conjunction on the left of the sequent arrow and disjunction on the right of the sequent arrow.

The name "display calculus" comes from a defining property of display calculi which allows us to "display" every substructure of a complex structure as the whole of one side of the sequent.

The intuition of display calculi can be given in terms of high school algebra using the familiar method of solving algebraic equations like $2x^2 - 4 \leq 0$ by manipulating the equation to "make x the subject" as follows:

the equation	$(2 \times x^2) - 4 \leq 0$	
becomes	$(2 \times x^2) \leq 0 + 4$	by adding 4 to both sides
becomes	$x^2 \leq (0 + 4)/2$	by dividing both sides by 2
becomes	$x \leq \sqrt{(0 + 4)/2}$	by taking (positive) square root of both sides
becomes	$x \leq \sqrt{2}$	by simplification (or rewriting).

Most of these manipulations are only possible because the operations come in "opposite" (residuated or Galois connected) pairs that undo the effects of each other; namely $(+, -)$, $(\times, /)$, $(x^2, \sqrt{x})$. The ability to "display the x" by "making x the subject" allows us to unravel the context surrounding the x, thereby shedding some light on the meaning of x in the given context. Once we have a handle on x, we can replace all occurrences of x in other equations by the right hand side, thereby reducing the number of variables in the problem at hand.

The modularity of display calculi is also manifested by one general cut-elimination theorem, which applies to all display calculi whose rules obey eight conditions stipulated by Belnap [Bel82].

As many people have observed, Display Calculi are a very low-level tool for automated reasoning. That is, there are many rules and no real systematic procedure for finding derivations. Nevertheless, as we hope to show, display calculi are an important tool for designing new logics.

By way of introduction, we now look at a particular display calculus for classical propositional logic from [Gor97]. A much more general account which will be useful later can be found in [Gor98d, Gor98b].

Since this is supposed to be an introduction to display calculi we cover the various aspects of display calculi by comparing and contrasting a display calculus $\delta\mathbf{PC}$ for classical propositional logic from [Gor97] with traditional sequent calculi for classical propositional logic.

2.4.1 Formulae and Structures

As with all sequent calculi there are two **disjoint** sorts of syntactic entities:

Formulae: which are built from atomic formulae like p and q and logical constants like $\top$ and $\bot$ using logical connectives like $\wedge$. Formulae are written using formula variables like A and B and C. For classical propositional logic we use the following syntax in BNF form:

$$p ::= \top \mid \bot \mid p_0 \mid p_1 \mid \cdots$$
$$A ::= p \mid \neg A \mid A \wedge B \mid A \vee B \mid A \Rightarrow B$$

Structures: which are built from formulae (i.e. formulae are atomic structures) and structural constants (say I) using structural connectives like comma. Structures are written using structure variables like X and Y and Z. For classical propositional logic we use the following syntax in BNF form:

$$X ::= A \mid I \mid *X \mid X \, , \, Y$$

Notice the hierarchy: formulae are atomic structures, but structures are not formulae. In traditional sequent calculi we already make a distinction like this using Γ for a set, multiset or sequence of formula occurrences while using A for a single formula occurrence. But unlike in traditional sequent calculi, there are no initial assumptions about structures: they are not sets, multisets or lists for example.

Moreover, there is no underlying assumption about commutativity or associativity of structural connectives. Whereas Gentzen's comma is polyvalent, the structural connectives from display calculi come with a fixed arity, usually unary or binary. Thus structures are complex trees with internal nodes containing structural connectives and leaf nodes containing formulae.

2.4.2 Sequents and Sequent Rules

A sequent $X \vdash Y$ consists of an **antecedent** X on the left of "turnstile" and a **succedent** Y on the right of "turnstile". Sequent rules are of the form:

$$\text{(name)} \quad \frac{X_1 \vdash Y_1 \quad X_2 \vdash Y_2 \quad \cdots \quad X_n \vdash Y_n}{X \vdash Y}$$

where the sequents above the horizontal line are the **premisses** and the sequent below the horizontal line is the **conclusion**.

The rules come in two flavours: structural rules and logical rules.

2.4.3 Structural Rules

Structural rules are built from structure variables like X, structural constants, and structural connectives: that is, no formula variables, no logical constants and no formula connectives are allowed in their statement.

$$(I^+_- \vdash) \quad \dfrac{\dfrac{X \ , \ I \vdash Y}{\dfrac{}{X \vdash Y}}}{\dfrac{}{I \ , \ X \vdash Y}} \qquad (\vdash I^+_-) \quad \dfrac{\dfrac{X \vdash I \ , \ Y}{\dfrac{}{X \vdash Y}}}{\dfrac{}{X \vdash Y \ , \ I}}$$

$$(\text{VerI}) \quad \dfrac{I \vdash X}{Y \vdash X} \qquad (\text{ExI}) \quad \dfrac{X \vdash I}{X \vdash Y}$$

$$(M \vdash) \quad \dfrac{X \vdash Z}{X \ , \ Y \vdash Z} \qquad (\vdash M) \quad \dfrac{Z \vdash X}{Z \vdash X \ , \ Y}$$

$$(C \vdash) \quad \dfrac{X \ , \ X \vdash Z}{X \vdash Z} \qquad (\vdash C) \quad \dfrac{Z \vdash X \ , \ X}{Z \vdash X}$$

$$(A \vdash) \quad \dfrac{X \ , \ (Y \ , \ Z) \vdash W}{\dfrac{}{(X \ , \ Y) \ , \ Z \vdash W}} \qquad (\vdash A) \quad \dfrac{W \vdash X \ , \ (Y \ , \ Z)}{\dfrac{}{W \vdash (X \ , \ Y) \ , \ Z}}$$

$$(P \vdash) \quad \dfrac{Y \ , \ X \vdash Z}{X \ , \ Y \vdash Z} \qquad (\vdash P) \quad \dfrac{Z \vdash Y \ , \ X}{Z \vdash X \ , \ Y}$$

Figure 2.2: Basic Structural Rules

In traditional sequent calculi we are used to seeing a structural rule like contraction or weakening expressed using formula variables:

$$(\text{Ctr}) \quad \dfrac{\Gamma, A, A \vdash \Delta}{\Gamma, A \vdash \Delta} \qquad (\text{Wk}) \quad \dfrac{\Gamma, \vdash \Delta}{\Gamma, A \vdash \Delta}$$

but now we insist that only structure variables be used in structural rules. As we shall see, this is because of a substitution condition that is required of all display calculi. So the display calculus analogues for contraction and weakening are of the form:

$$(\text{Ctr}) \quad \dfrac{X, Z, Z \vdash Y}{X, Z \vdash Y} \qquad (\text{Wk}) \quad \dfrac{X \vdash Y}{X, Z \vdash Y}$$

The rest of the structural rules for $\delta\mathbf{PC}$ are shown in Figure 2.2. Some rules are clearly just "rewrite" rules where a structural connective turns into a logical connective, possibly on the other side of turnstile. In fact, for each connective, at least one of its rules is a rewrite rule.

The other rule can be viewed bottom-up as a constraint that limits the upward application of that rule. For example, the $(\vdash \wedge)$ rule can only be applied bottom-up if the left hand side can be put into a form $(X, \ , Y)$.

THEOREM 2.1. *Every "rewrite" rule is invertible: if the conclusion is derivable then so is the premiss.*

By way of example, here is how to obtain a derivation of the premiss $A \ , \ B \vdash Z$ of

$$(\bot\vdash)\ \ \bot\vdash I \qquad\qquad (\vdash\bot)\ \ \frac{X\vdash I}{X\vdash\bot}$$

$$(\vdash\top)\ \ \frac{I\vdash Z}{\top\vdash Z} \qquad\qquad (\vdash\top)\ \ I\vdash\top$$

$$(\vee\vdash)\ \ \frac{A\vdash X \quad B\vdash Y}{A\vee B\vdash X\ ,\ Y} \qquad\qquad (\vdash\vee)\ \ \frac{Z\vdash A\ ,\ B}{Z\vdash A\vee B}$$

$$(\wedge\vdash)\ \ \frac{A\ ,\ B\vdash Z}{A\wedge B\vdash Z} \qquad\qquad (\vdash\wedge)\ \ \frac{X\vdash A \quad Y\vdash B}{X\ ,\ Y\vdash A\wedge B}$$

$$(\supset\vdash)\ \ \frac{X\vdash A \quad B\vdash Y}{A\supset B\vdash *X\ ,\ Y} \qquad\qquad (\vdash\supset)\ \ \frac{A\ ,\ X\vdash B}{X\vdash A\supset B}$$

$$(\neg\vdash)\ \ \frac{*A\vdash Z}{\neg A\vdash Z} \qquad\qquad (\vdash\neg)\ \ \frac{Z\vdash *A}{Z\vdash\neg A}$$

Figure 2.3: Logical Introduction Rules

the $(\vdash\wedge)$ rule from a given derivation of its conclusion $A\wedge B\vdash Z$:

$$\dfrac{\dfrac{\dfrac{A\vdash A \quad B\vdash B}{A\ ,\ B\vdash A\wedge B}\,(\vdash\wedge) \qquad A\wedge B\vdash Z}{A\ ,\ B\vdash Z}\,(\text{cut})$$

2.4.4 Logical Rules

A logical rule introduces exactly one formula connective, either as the whole of the antecedent, or as the whole of the succedent, of its conclusion. Clearly, there should be two logical rules for each connective: one rule introducing that connective into the antecedent and the other into its succedent. The introduced formula is known as the principal formula, and the subformulae from which it is composed are known as the side formulae.

In traditional sequent calculi we are used to introducing logical connectives into a context viz:

$$(\wedge\text{L})\ \ \frac{\Gamma, A, B\vdash\Delta}{\Gamma, A\wedge B\vdash\Delta}$$

But now we must use a rule like:

$$(\wedge\text{L})\ \ \frac{A, B\vdash Y}{A\wedge B\vdash Y}$$

with no context like Γ. Notice that the introduced formula is then displayed!

The rest of the logical rules for $\delta\mathbf{PC}$ are shown in Figure 2.3.

$$\text{(id)}\ \ p \vdash p \qquad\qquad \text{(cut)}\ \ \frac{X \vdash A \quad A \vdash Y}{X \vdash Y}$$

Figure 2.4: Identity and Cut Rules

$$\frac{*X \vdash Y}{*Y \vdash X} \qquad \frac{X \vdash *Y}{Y \vdash *X} \qquad \frac{**X \vdash Y}{X \vdash Y}$$

$$\frac{\dfrac{X \vdash Z\ ,\ *Y}{X\ ,\ Y \vdash Z}}{Y \vdash *X\ ,\ Z} \qquad \frac{\dfrac{Z\ ,\ *Y \vdash X}{Z \vdash X\ ,\ Y}}{*X\ ,\ Z \vdash Y}$$

Figure 2.5: Display Postulates

Finally, we need a rule which introduces atomic formulae, and a cut rule as shown in Figure 2.4. Note that we can cut on formulae only.

2.4.5 Display Postulates, Antecedent Parts and Succedent Parts

A display calculus must also have a collection of (usually bidirectional and unary) structural rules called "display postulates" which allow us to "unravel" each structure.

The display postulates we need to unravel structure for $\delta\mathbf{PC}$ are shown in Figure 2.4.5.

For example, the following rules are enough to allow us to unravel any structure built only from formulae and the $*$ structural connective:

$$\text{(dgc)}\ \frac{*X \vdash Y}{*Y \vdash X} \qquad \text{(gc)}\ \frac{X \vdash *Y}{Y \vdash *X}$$

If you squint your eyes and pretend that $*$ is a unary function on some underlying set, and pretend that $\vdash$ is a partial order then these are precisely the conditions for a dual Galois connection $(*, *)$ and a Galois connection $(*, *)$, whence the names "dgc" and "gc". Thus $*$ is an antitonic operation, and as we shall see, is an exact proxy for classical negation wherever it appears in a sequent.

The rule which allows us to cancel two occurrences of $*$ is not really required for unraveling occurrences of $*$ and could be put into the basic structural rules as a property of $*$.

Since we also have a binary comma which we want to be isotonic in both argument positions we also need the following display postulates which will later reveal themselves

to capture residuation:

$$\frac{X \vdash Z \ , \ *Y}{X \ , Y \vdash Z} \qquad \frac{Z \ , \ *Y \vdash X}{Z \vdash X \ , Y}$$

$$\frac{X \ , Y \vdash Z}{Y \vdash *X \ , Z} \qquad \frac{Z \vdash X \ , Y}{*X \ , Z \vdash Y}$$

That is, since $*$ is an antitonic operation, we can use it in the appropriate place to unravel structures build from commas.

As we have seen, the tonocity of operators often plays an important role in determining their other properties. Consequently, in the general case of display calculi, each structural connective comes with a tonicity vector which tells us whether that structural connective is isotonic or antitonic in that position. Using this notion, it is possible to assign a polarity of "antecedent part" or "succedent part" to each substructure of a structure. Intuitively, we count the number of polarity switches that are necessary to reach the sub-structure occurrence in question. If the sub-structure occurrence is of the same polarity then it has the same "cedency" as the structure in which it appears, otherwise, it has the opposite "cedency".

For the particular calculus considered here, the following definition suffices. In any structure Z, the substructure X occurs **negatively** (respectively **positively**) if X appears in the scope of an odd number (respectively zero or an even number) of $*$ connectives [Bel82]. In a sequent $V \vdash W$, a particular occurrence of X is an **antecedent part** (respectively **succedent part**) if it occurs positively in the antecedent V (respectively positively in the succedent W) or if it occurs negatively in the succedent W (respectively negatively in the antecedent V) [Bel82].

For example, if we use a unary anti-tonic structural connective $*$ then the occurrence of X in $*X$ has a "negative" polarity with respect to the polarity of $*X$. Thus the occurrence of X in $*X \vdash Y$ is a succedent part since it has the opposite "cedency" to $*X$ which is an antecedent part since $*X$ appears positively in the antecedent in $*X \vdash Y$.

We now have all the ingredients to state the defining feature of a display calculus. The display calculus δ**PC** is shown in Figures 2.4.5-2.4.

2.4.6 Display Theorem

Two sequents σ and σ' are **structurally equivalent** if we can pass from one to the other (and back) using only the display postulates. The name Display Calculus comes from the following theorem (which does not hold for subformulae of a formula!).

THEOREM 2.2. [Theorem (Belnap82):] *For every sequent $X \vdash Y$ and every substructure Z of $X \vdash Y$, there is a structurally equivalent sequent $Z \vdash Y'$ or $X' \vdash Z$ that has Z displayed as the whole of its antecedent if Z is an antecedent part or succedent if Z is a succedent part of $X \vdash Y$.*

LEMMA 2.3. *For any formula A, the sequent $A \vdash A$ is provable.*

Proof: By induction on structure of A.

2.4.7 Soundness and Completeness

There are two main ways to prove the soundness of a display calculus:

Semantics: If the logic in question has a semantics then we need only show that the rules of the display calculus map valid formulae to valid formulae.

Syntactic: If the logic in question has a Hilbert calculus then we need only show that the rules of the display calculus map theorems to theorems.

In both cases, it is useful to translate sequents into formulae, we therefore give a translation τ from sequents to formulae using two mutually recursive functions τ_1 and τ_2 where:

$$\tau(X \vdash Y) := \tau_1(X) \Rightarrow \tau_2(Y)$$

$$\begin{array}{lcllcl} \tau_1(X\ ,\ Y) & := & \tau_1(X) \wedge \tau_1(Y) & \tau_2(X\ ,\ Y) & := & \tau_2(X) \vee \tau_2(Y) \\ \tau_1(*X) & := & \neg\tau_2(X) & \tau_2(*X) & := & \neg\tau_1(X) \\ \tau_1(I) & := & \top & \tau_2(I) & := & \bot \\ \tau_1(A) & := & A & \tau_2(A) & := & A \end{array}$$

The following lemma is often useful when proving completeness but will not been needed here. It shows how the structural connectives are overloaded to capture their logical counter-parts in antecedent and succedent positions.

LEMMA 2.4. *For every structure Z, both $Z \vdash \tau_1(Z)$ and $\tau_2(Z) \vdash Z$ are provable in δ**PC**.*

Proof: By simultaneous induction on the structure of Z.

To prove soundness of δ**PC** we follow the semantic route.

THEOREM 2.5. [Soundness] *If the sequent $X \vdash Y$ is provable in δ**PC** then the formula $\tau(X \vdash Y)$ is* **PC***-valid.*

Proof: By induction on the length of the proof of $X \vdash Y$.
Base Cases: If the proof of $X \vdash Y$ is of length one then $X \vdash Y$ must be one of the axioms of δ**PC** shown on top of the dotted lines below:

$$\frac{p \vdash p}{p \Rightarrow p} \qquad \frac{I \vdash \top}{\top \Rightarrow \top} \qquad \frac{\bot \vdash I}{\bot \Rightarrow \bot}$$

And each τ-translated formula shown below the dotted line is trivially **PC**-valid.
Induction Step: For the induction step we show that for each rule of δ**PC**, if the τ-translation of the premiss is **PC**-valid then so is the τ-translation of the conclusion. This allows us to extend the theorem to proofs of $X \vdash Y$ of arbitrary lengths.

For example, the soundness of the display postulates for $*$ reduces to showing the following easy (vertical) equivalences in **PC** where objects of the form $\tau_1(X)$ are just formulae of **PC**:

$$\begin{array}{ccc} \neg\tau_2(X) \Rightarrow \tau_2(Y) & \tau_1(X) \Rightarrow \neg\tau_1(Y) & \neg\neg\tau_1(X) \Rightarrow \tau_2(Y) \\ \text{iff} & \text{iff} & \text{iff} \\ \neg\tau_2(Y) \Rightarrow \tau_2(X) & \tau_1(Y) \Rightarrow \neg\tau_1(X) & \tau_1(X) \Rightarrow \tau_2(Y) \end{array}$$

The τ-translates of the display postulates for comma reduce to the (residuation) laws shown (horizontally) below by putting $A \Rightarrow B := (\neg A \vee B)$, and artificially, $A \Leftarrow B := (A \vee \neg B)$. These residuation laws are easily seen to hold for classical propositional logic where $(A \Rightarrow B)$ is $(B \Leftarrow A)$:

$$\tau_1(X) \Rightarrow \tau_2(Z) \vee \neg\tau_1(Y) \quad \text{iff} \quad \tau_1(X) \wedge \tau_1(Y) \Rightarrow \tau_2(Z) \quad \text{iff} \quad \tau_1(Y) \Rightarrow \neg\tau_1(X) \vee \tau_2(Z)$$

$$\tau_1(Z) \wedge \neg\tau_2(Y) \Rightarrow \tau_2(X) \quad \text{iff} \quad \tau_1(Z) \Rightarrow \tau_2(X) \vee \tau_2(Y) \quad \text{iff} \quad \neg\tau_2(X) \wedge \tau_1(Z) \Rightarrow \tau_2(Y)$$

The easiest syntactic way to prove completeness is to use "Axiom Chopping" whereby we show that every axiom of a Hilbert calculus for our logic is derivable in our display calculus and that each of the Hilbert calculus rules of inference is derivable in our display calculus. We can then mimic any Hilbert calculus derivation within our display calculus.

For example, we now show that the traditional rule of Modus Ponens (from $\vdash A$ and $\vdash A \Rightarrow B$ infer $\vdash B$) is derivable in $\delta\mathbf{PC}$. As our assumptions we are given purported $\delta\mathbf{PC}$ derivations of the sequents $I \vdash A$ and $I \vdash A \Rightarrow B$. Our task is to produce a $\delta\mathbf{PC}$ derivation of the sequent $I \vdash B$. This we show below:

$$\dfrac{\dfrac{\dfrac{I \vdash A \Rightarrow B \quad \dfrac{A \vdash A \qquad B \vdash B}{A \Rightarrow B \vdash *A \,,\, B}\,(\supset\vdash)}{I \vdash *A \,,\, B}\,(\text{cut})}{A \,,\, I \vdash B}\,(\text{dp})}{A \vdash B}\,(\text{dp}) \qquad\qquad \dfrac{I \vdash A \quad A \vdash B}{I \vdash B}\,(\text{cut})$$

An alternative way would be to show that every rule of a traditional sequent calculus for classical propositional logic is derivable in $\delta\mathbf{PC}$. Here is how to mimic one form of the traditional cut rule shown below left:

$$\dfrac{\Gamma \vdash A, \Delta \quad \Pi, A \vdash \Sigma}{\Pi, \Gamma \vdash \Sigma, \Delta}\,(\text{cut}) \qquad\qquad \dfrac{\dfrac{\dfrac{\dfrac{\tau_1(\Gamma) \vdash A \,,\, \tau_2(\Delta)}{\tau_1(\Gamma) \,,\, *\tau_2(\Delta) \vdash A}\,(\text{dp}) \quad \dfrac{\tau_1(\Pi) \,,\, A \vdash \tau_2(\Sigma)}{A \vdash *\tau_1(\Pi) \,,\, \tau_2(\Sigma)}}{\tau_1(\Gamma) \,,\, *\tau_2(\Delta) \vdash *\tau_1(\Pi) \,,\, \tau_2(\Sigma)}\,(\text{cut})}{\tau_1(\Pi) \,,\, (\tau_1(\Gamma) \,,\, *\tau_2(\Delta)) \vdash \tau_2(\Sigma)}\,(\text{dp})}{\tau_1(\Pi) \,,\, \tau_1(\Gamma) \vdash \tau_2(\Sigma) \,,\, \tau_2(\Delta)}\,(\text{A}\vdash),(\text{dp})$$

Finally, a third way is to prove completeness is to show that the Lindenbaum algebra that results from taking the equivalence classes of provably equivalent formulae, where $A \equiv B$ iff $A \vdash B$ and $B \vdash A$, is an algebra of the form appropriate for the logic in question. See [Gor97] for details.

THEOREM 2.6. [Completeness] *If the formula $\tau(X \vdash Y)$ is* $\mathbf{PC}$*-valid then the sequent $X \vdash Y$ is provable in* $\delta\mathbf{PC}$.

2.4.8 Cut Elimination

The beauty of display calculi becomes apparent when we consider cut-elimination. Belnap stipulates eight conditions on the rules of the calculus which guarantee cut-elimination. The version of the rules below is taken from Kracht [Kra96].

For every sequent rule Belnap [Bel82, page 388] first defines the following notions: in an application *Inf* of a sequent rule (ρ), "constituents occurring as part of occurrences of structures assigned to structure-variables are defined to be **parameters** of *Inf* ; all other constituents are defined as **nonparametric**, including those assigned to formula-variables. Constituents occupying similar positions in occurrences of structures assigned to the same structure-variable are defined as **congruent** in *Inf* ".

For example, in an instance of the associative rule (A), all constituents of structures assigned to the structure variables are parameters. As usual, the introduction rules for the logical connectives like ($\vdash\supset$) involve constituents (side formulae) in the premisses and constituents (principal formulae) in the conclusion that are not parameters.

The eight (actually seven) conditions shown below are from [Kra96]:

(C1) Each formula which is a constituent of some premiss of a rule ρ is a subformula of some formula in the conclusion of ρ.

Intuition: formulae cannot disappear from premisses to conclusion as in the cut rule.

(C2) Congruent parameters are occurrences of the same structure.

Intuition: true by definition since we use structural variables rather than formula variables.

(C3) Each parameter is congruent to at most one constituent in the conclusion. Equivalently, no two constituents of the conclusion are congruent to each other.

Intuition: This forbids rules like

$$\frac{X \vdash Z}{X \ , \ X \vdash Z}$$

and leads to the condition about formulae appearing only once in the antecedents of primitive formulae in the section on encoding of axioms using structural rules.

(C4) Congruent parameters are either all antecedent parts or all succedent parts of their respective sequent.

Intuition: this forbids rules like

$$\frac{X \ , \ Y \vdash Z}{X \vdash Y \ , \ Z}$$

where Y switches from being an antecedent part into a succedent part.

(C5) If a formula is non-parametric in the conclusion of a rule ρ, it is either the entire antecedent, or the entire succedent. Such a formula is called a **principal** formula.

Intuition: all introduced formulae are displayed.

(C6/7) Each rule is closed under simultaneous substitution of arbitrary structures for congruent parameters.

Intuition: the traditional contraction rule would not obey this condition since it is couched in terms of formula variables, not structure variables. That is, we could only substitute formulae into these variables, not arbitrary structures.

(C8) If there are inference rules ρ_1 and ρ_2 with respective conclusions $X \vdash A$ and $A \vdash Y$ with A principal in both inferences (in the sense of C5), and if (cut) is applied to yield $X \vdash Y$ then, either $X \vdash Y$ is identical to $X \vdash A$ or to $A \vdash Y$; or it is possible to pass from the premisses of ρ_1 and ρ_2 to $X \vdash Y$ by means of inferences falling under (cut) where the cut-formula is always a proper subformula of A. If A satisfies the "if" part of this condition it is known as a "matching principal constituent".

The conditions C1-C7 can be verified simply by inspection of the rules. Only C8 requires some work but it can be proved, case by case, using induction on the structure of the cut formula. For example, below is a cut on the formula $A \wedge B$ where $A \wedge B$ is *principal* in both premisses of (cut). And below that are two derivations (split due to typesetting reasons) showing that the *same* conclusion can be derived from the *same* premisses using a cut on (the strictly smaller) formula A and B instead, at the cost of some extra display postulate (dp) moves:

$$\dfrac{\dfrac{X \vdash A \quad Y \vdash B}{X \, , \, Y \vdash A \wedge B}\,(\vdash \wedge) \qquad \dfrac{A \, , \, B \vdash Z}{A \wedge B \vdash Z}\,(\wedge \vdash)}{X \, , \, Y \vdash Z}\,(\text{cut})$$

original cut on (left and right) principal $A \wedge B$

$$\dfrac{X \vdash A \quad \dfrac{\dfrac{A \, , \, B \vdash Z}{A \vdash Z \, , \, *B}\,(\text{dp})}{X \vdash Z \, , \, *B}\,(\text{cut})}{\dfrac{X \, , \, B \vdash Z}{B \vdash *X \, , \, Z}\,(\text{dp})}\,(\text{dp})$$

$$\dfrac{Y \vdash B \quad B \vdash *X \, , \, Z}{\dfrac{Y \vdash *X \, , \, Z}{X \, , \, Y \vdash Z}\,(\text{dp})}\,(\text{cut})$$

replace by cuts on A and B

C8: reduction of cut degree for principal cuts on $A \wedge B$

We thus have access to Belnap's cut-elimination theorem.

THEOREM 2.7. [Cut-elimination] *Any display calculus whose rules obey conditions C1-C8 enjoys cut-elimination: if there is a proof of the sequent $X \vdash Y$ in $\delta\mathbf{PC}$, then there is a cut-free proof of $X \vdash Y$ in $\delta\mathbf{PC}$ [Bel82].*

The proof of this theorem is beyond the scope of this course as it falls squarely in the realm of proof theory. Suffice to say that the cut-elimination theorem has actually been checked by using an interactive theorem prover [DG02, DG03].

2.4.9 Gentzen Overloading

Usually, comma is a proxy for $\wedge$ when comma appears on the left hand side of turnstile and a proxy for $\vee$ when it appears on the right hand side of turnstile. This phenomenon of overloading also appears in display calculi, but there is greater complexity because structural connectives can have a polarity associated with them.

For example, by perusing the translation τ it is obvious that in $\delta\mathbf{PC}$, the structural connective $*$ is really a proxy for classical negation $\neg$ in both antecedent and succedent positions.

2.4.10 Encoding Axioms Via Structural Rules

Kracht [Kra96] shows that modal display calculi capture certain axiomatic extensions of a Hilbert calculi for tense logic **Kt** by the addition of purely structural rules. This phenomenon can be applied to any display calculus. Below we consider the case for $\delta\mathbf{PC}$.

Let $\delta\mathbf{L}$ be the display calculus obtained by deleting from $\delta\mathbf{PC}$ all the structural rules from Figure 2.2. Let ρ be a structural rule and let $\delta\mathbf{L}\rho$ be the display calculus $\delta\mathbf{L}$ extended by rule ρ. Let Ax be a formula of the form $A \Rightarrow B$ and let $\mathbf{LAx}$ be the logic obtained by adding Ax as an axiom schema. The display calculus $\delta\mathbf{L}\rho$ **properly characterises LAx** iff $\delta\mathbf{L}\rho$ satisfies C1-C8 and is sound and complete with respect to $\mathbf{LAx}$.

A formula $A \Rightarrow B$ is **primitive** iff A and B are built from propositional variables and the constants $\top$ with the help of $\wedge$ only, and if no propositional variable appears in A more than once [Kra96]. The definition of primitivity changes with the structural connectives that are available so this definition suffices for the structural connectives we have for $\delta\mathbf{PC}$.

A structural rule ρ is **special** if it is of the form shown below:

$$(\rho)\ \frac{X \vdash Z}{Y \vdash Z}$$

Again, this definition has been simplified for $\delta\mathbf{PC}$: a more general definition would take us too far afield.

Following Kracht [Kra96], define the following translation σ from positive formulae into structures:

$$\begin{array}{lcllcl} \sigma(\top) & = & I & \qquad \sigma(p) & = & X_p \\ \sigma(A \wedge B) & = & (\sigma(A)\ ,\ \sigma(B)) & & & \end{array}$$

Now, given a primitive formula Ax of the form $A \Rightarrow B$ we calculate the special structural rule (ρ), given below:

$$(\rho)\ \frac{\sigma(B) \vdash Z}{\sigma(A) \vdash Z}$$

That is, instead of adding new rules couched in terms of *formulae*, we add *purely structural rules* couched in terms of their translation under σ.

The next two theorems show that these rules capture the equation exactly, and vice-versa.

THEOREM 2.8. *If $A \Rightarrow B$ is a primitive formula Ax then the display calculus $\delta\mathbf{L}\rho$ properly characterises* $\mathbf{LAx}$.

THEOREM 2.9. *If (ρ) is a special structural rule that meets C1-C8 then $\delta\mathbf{L}\rho$ properly characterises the logic* $\mathbf{LAx}$ *where Ax is the primitive formula $\tau_1(Y) \Rightarrow \tau_1(X)$.*

For example the primitive axioms for contraction, weakening, associativity and commutativity are shown below with their corresponding special structural rules.

contraction	weakening	associativity	commutativity
$p \Rightarrow p \wedge p$	$p \wedge q \Rightarrow p$	$p \wedge (q \wedge r) \Rightarrow (p \wedge q) \wedge r$	$p \wedge q \Rightarrow q \wedge p$

$$\text{(Ctr)}\ \frac{X\ ,\ X \vdash Z}{X \vdash Z} \quad \text{(Wk)}\ \frac{X \vdash Z}{X\ ,\ Y \vdash Z} \quad \text{(A)}\ \frac{(X\ ,\ Y)\ ,\ W \vdash Z}{X\ ,\ (Y\ ,\ W) \vdash Z} \quad \text{(P)}\ \frac{X\ ,\ Y \vdash Z}{Y\ ,\ X \vdash Z}$$

Chapter 3

Displaying CTL

In this chapter, we take a close look at the mathematical structure which we use to model natural language and introduce the grammatical base logic we will see at work in Part II. We spell out the logical connection between the binary operators of the system NL introduced by Lambek in [Lam61] and the unary ones of NL($\Diamond$) proposed by Kurtonina and Moortgat [Kur95, KM95, Moo96b]. Moreover, following Dunn [Dun91] and Goré [Gor98c], we show that the algebraic structure of these systems can accommodate downward monotone unary operators as well. We present the extended system NL($\Diamond$,$\cdot^{\mathbf{0}}$) and study its formal properties. A very useful survey of the field of substructural logics is given in [Res00].

By means of Display Calculi (Chapter 2), we clarify how the logical rules of NL, NL($\Diamond$) and NL($\Diamond$,$\cdot^{\mathbf{0}}$) actually encode the definition of residuated and Galois operators. We then give the axiomatic presentation of the logical system of residuated and Galois connected operators NL($\Diamond$,$\cdot^{\mathbf{0}}$), proving its soundness and completeness with respect to Kripke frame semantics. Furthermore, we study the proof theoretical behavior of these systems beginning with Display Calculi, compiling in the Galois connection law and then moving to a cut-free Gentzen sequent presentation. Finally, we investigate the abstract derivability patterns that arise in NL($\Diamond$,$\cdot^{\mathbf{0}}$). We refer to the whole family of logic as Categorial Type Logics (CTLs).

The results presented here draw on work done in collaboration with Carlos Areces and Michael Moortgat [AB04, ABM01].

3.1 Capturing Residuation

3.2 The Logic of Residuation NL

The base system of the CTL family of the binary type forming operators is the non-associative and non-commutative Lambek calculus NL [Lam61]. We present its axiomatic presentation and the original sequent calculus given by Lambek.

Definition 3.1. [Formula Language of NL] Given a set ATOM of atomic propositional

formula, the language of NL is defined recursively as

$$\mathsf{FORM} ::= \mathsf{ATOM} \mid \mathsf{FORM}/\mathsf{FORM} \mid \mathsf{FORM}\backslash\mathsf{FORM} \mid \mathsf{FORM} \bullet \mathsf{FORM}.$$

An axiomatic presentation of NL is given as follows.

Definition 3.2. [NL: Axiomatic System] The system NL is defined by the axioms below. Given $A, B, C \in \mathsf{FORM}$

$$\begin{array}{rl} [\text{REFL}] & \vdash A \longrightarrow A, \\ [\text{TRANS}] & \text{If} \vdash A \longrightarrow B \text{ and} \vdash B \longrightarrow C, \text{ then} \vdash A \longrightarrow C, \\ [\text{RES}_2] & \vdash A \longrightarrow C/B \text{ iff} \vdash A \bullet B \longrightarrow C \text{ iff} \vdash B \longrightarrow A\backslash C. \end{array}$$

NL is commonly called the pure logic of residuation, and rightly so as we can see from its axiomatic presentation. [REFL] and [TRANS] define essential properties for the derivability relation $\longrightarrow$ while [RES_2] characterizes $(\backslash, \bullet, /)$ as a residuated triple[1].

The axiomatic presentation of NL clearly shows that residuation directly governs the behavior of its type forming operators. Unfortunately, it is not well-suited proof theoretically. In particular, the [TRANS] and [RES_2] rules above violate the subformula property, introducing non determinism in the proof search. As with classical propositional logic, an alternative is the formulation of an equivalent Gentzen presentation, in which the use of the counterpart of [TRANS], the cut-rule, can be shown to be redundant (cut-elimination) and the [RES_2] is compiled in the logical rules. The sequent presentation is well behaved proof theoretically: it enjoys the subformula property and it yields backward-chaining decision procedure [Lam58, Lam61].

Sequent Calculus

While in the axiomatic presentation the derivability relation holds between formulas of the logical language, in a Gentzen system it is stated in terms of *sequents*: pairs $\Gamma \Rightarrow A$ where Γ is a structured configuration of formulas or *structural term* and A is a logical formula. The set STRUCT of structural terms needed for a sequent presentation of NL is very simple.

$$\mathsf{STRUCT} ::= \mathsf{FORM} \mid (\mathsf{STRUCT} \circ \mathsf{STRUCT}).$$

The logical rules of the Gentzen system for NL are given in Figure 3.1. In the figure, A, B, C are formulas, Γ, Δ are structural terms and the notation $\Gamma[\Delta]$ is used to single out a particular instance of the substructure Δ in Γ.

As we can see from inspecting these rules, it is not immediately obvious that they are characterizing the same derivability relation as the one characterized by the axiomatic

[1] [RES_2] is the axiomatic presentation of the definition of residuated partially ordered groupoid given in Chapter 2 – the implications are here written as $\backslash$ and $/$ as it is usual in the Lambek tradition. [RES_2] could also be understood as a kind of deduction theorem. But while a deduction theorem is better seen as a link between the meta-language and the object language, [RES_2] relates three operators in the object language.

$$\frac{}{A \Rightarrow A}\ [\text{axiom}] \qquad \frac{\Delta \Rightarrow A \quad \Gamma[A] \Rightarrow C}{\Gamma[\Delta] \Rightarrow C}\ [\text{cut}]$$

$$\frac{\Delta \Rightarrow B \quad \Gamma[A] \Rightarrow C}{\Gamma[(A/B \circ \Delta)] \Rightarrow C}\ [/\text{L}] \qquad \frac{\Gamma \circ B \Rightarrow A}{\Gamma \Rightarrow A/B}\ [/\text{R}]$$

$$\frac{\Delta \Rightarrow B \quad \Gamma[A] \Rightarrow C}{\Gamma[(\Delta \circ B\backslash A)] \Rightarrow C}\ [\backslash\text{L}] \qquad \frac{B \circ \Gamma \Rightarrow A}{\Gamma \Rightarrow B\backslash A}\ [\backslash\text{R}]$$

$$\frac{\Gamma[(A \circ B)] \Rightarrow C}{\Gamma[A \bullet B] \Rightarrow C}\ [\bullet\text{L}] \qquad \frac{\Gamma \Rightarrow A \quad \Delta \Rightarrow B}{(\Gamma \circ \Delta) \Rightarrow A \bullet B}\ [\bullet\text{R}]$$

Figure 3.1: Gentzen sequent calculus for NL.

presentation of NL. To establish the equivalence between the two formats, define the translation $\cdot^t : \mathsf{STRUCT} \to \mathsf{FORM}$ as

$$\begin{aligned} (\Gamma_1 \circ \Gamma_2)^t &= (\Gamma_1^t \bullet \Gamma_2^t), \\ A^t &= A, \text{ for } A \in \mathsf{FORM}. \end{aligned}$$

Proposition 3.3. [See [Lam58, Lam61]] If $\vdash A \longrightarrow B$ is a theorem of the axiomatic presentation of NL then there is a Gentzen proof of $A \Rightarrow B$. And for every proof of a sequent $\Gamma \Rightarrow B$, $\vdash \Gamma^t \longrightarrow B$ is a theorem.

The system presented in Figure 3.1 includes the cut-rule, but Lambek proved also in [Lam58], that the rule is admissible, in the sense that it does not increase the set of theorems that can already be derived using just the other rules.

Proposition 3.4. [Cut-elimination and Decidability] The cut-rule is admissible in NL, and the system is decidable.

3.2.1 The Residuated Unary Operators NL($\Diamond$)

The system NL($\Diamond$) introduced in [Moo96b, Moo97] is obtained by adding unary residuated operators $\Diamond$ and $\Box^{\downarrow}$ to NL. The logical language of NL is extended with formulas formed by $\Diamond$ and $\Box^{\downarrow}$ and consequently the $\langle\cdot\rangle$ is added to the structural language.

Definition 3.5. [Formulas and Structures of NL($\Diamond$)] Given a set ATOM of atomic propositional symbols, the logical and structural languages of NL($\Diamond$) are obtained extending the set of FORM and STRUCT of NL.

$$\begin{aligned} \mathsf{FORM} \quad ::= \quad & \mathsf{ATOM} \mid \mathsf{FORM}/\mathsf{FORM} \mid \mathsf{FORM}\backslash\mathsf{FORM} \mid \mathsf{FORM} \bullet \mathsf{FORM} \mid \\ & \Diamond\mathsf{FORM} \mid \Box^{\downarrow}\mathsf{FORM}. \end{aligned}$$

$$\mathsf{STRUCT} \quad ::= \quad \mathsf{FORM} \mid (\,\mathsf{STRUCT} \circ \mathsf{STRUCT}) \mid \langle \mathsf{STRUCT} \rangle.$$

Its axiomatic presentation is obtained by simply adding to the axioms in Definition 3.2 the one below.

$$[\mathrm{RES}_1] \qquad \vdash \Diamond A \longrightarrow B \;\text{ iff }\; \vdash A \longrightarrow \Box^{\downarrow} B$$

which defines the pair $(\Diamond, \Box^{\downarrow})$ as a residuated pair of operators (Chapter 2).

As in the case of $[\mathrm{RES}_2]$, the axiom above is compiled in the logical rules to obtain a well behaved proof system. The logical rules in Gentzen sequent format are as in Figure 3.2.

$$\frac{\Gamma[A] \Rightarrow B}{\Gamma[\langle \Box^{\downarrow} A\rangle] \Rightarrow B}\,[\Box^{\downarrow}\mathrm{L}] \qquad \frac{\langle\Gamma\rangle \Rightarrow A}{\Gamma \Rightarrow \Box^{\downarrow} A}\,[\Box^{\downarrow}\,\mathrm{R}]$$

$$\frac{\Gamma[\langle A\rangle] \Rightarrow B}{\Gamma[\Diamond A] \Rightarrow B}\,[\Diamond\mathrm{L}] \qquad \frac{\Gamma \Rightarrow A}{\langle\Gamma\rangle \Rightarrow \Diamond A}\,[\Diamond\mathrm{R}]$$

Figure 3.2: Logical rules for residuated unary operators of $\mathsf{NL}(\Diamond)$.

3.2.2 Kripke Models

The Lambek calculi and their modern extensions are modal logics. Standard models for modal logics are Kripke models, or relational structures. These structures are rather simple, they only consist of a set together with a collection of relations on that set, but they turn out to be extremely expressive and have found several interesting applications (see [BRV01] for an introduction to modal logic and an overview of the field). In this course we will use Kripke models to reason with linguistic resources, we here repeat the definitions given in Chapter 2,

Definition 3.6. [Kripke Models] A model for $\mathsf{NL}(\Diamond)$ is a tuple $\mathcal{M} = (W, R^3_\bullet, R^2_\Diamond, V)$ where W is a non-empty set, $R^3_\bullet \subseteq W^3$, $R^2_\Diamond \subseteq W^2$, and V is a valuation $V : \mathsf{ATOM} \to \mathcal{P}(W)$. The $R^3_\bullet$ relation governs the residuated triple $(\backslash, \bullet, /)$, the $R^2_\Diamond$ relation governs the residuated pair $(\Diamond, \Box^{\downarrow})$. Given a model $\mathcal{M} = (W, R, V)$ and $x, y \in W$, the *satisfiability relation* is inductivly defined as follows[2].

$$\begin{aligned}
\mathcal{M}, x \Vdash A \quad &\text{iff} \quad x \in V(A) \;\text{ where } A \in \mathsf{ATOM}.\\
\mathcal{M}, x \Vdash \Diamond A \quad &\text{iff} \quad \exists y[R_\Diamond xy \;\&\; \mathcal{M}, y \Vdash A].\\
\mathcal{M}, y \Vdash \Box^{\downarrow} A \quad &\text{iff} \quad \forall x[R_\Diamond xy \to \mathcal{M}, x \Vdash A].\\
\mathcal{M}, x \Vdash A \bullet B \quad &\text{iff} \quad \exists y \exists z[R_\bullet xyz \;\&\; \mathcal{M}, y \Vdash A \;\&\; \mathcal{M}, z \Vdash B].\\
\mathcal{M}, y \Vdash C/B \quad &\text{iff} \quad \forall x \forall z[(R_\bullet xyz \;\&\; \mathcal{M}, z \Vdash B) \to \mathcal{M}, x \Vdash C].\\
\mathcal{M}, z \Vdash A\backslash C \quad &\text{iff} \quad \forall x \forall y[(R_\bullet xyz \;\&\; \mathcal{M}, y \Vdash A) \to \mathcal{M}, x \Vdash C].
\end{aligned}$$

[2]Note that the unary operators $\Diamond$ and $\Box^{\downarrow}$ can be thought of as the possibility in the past (P) and the necessity in the future (G) operators of temporal logic [Pri67], therefore their interpretation moves in the opposite directions along the accessibility relation $R_\Diamond$. The downarrow on the universal operator is there to highlight this fact.

Given an arrow $A \longrightarrow B$, a model $\mathcal{M} = (W, R, V)$ and $x \in W$, we say that $\mathcal{M}, x \models A \longrightarrow B$ iff $\mathcal{M}, x \Vdash A$ implies $\mathcal{M}, x \Vdash B$. $\mathcal{M} \models A \longrightarrow B$ iff for all $x \in W$, $\mathcal{M}, x \Vdash A \longrightarrow B$. We say that $A \Rightarrow B$ is *valid* (notation $\models A \longrightarrow B$) iff for any model $\mathcal{M}$, $\mathcal{M} \models A \longrightarrow B$.

THEOREM 3.7. [Soundness and Completeness [Dos92, Kur95]] $\mathsf{NL}(\Diamond) \vdash A \longrightarrow B$ iff $\models A \longrightarrow B$

It is easy to show that the [RES$_1$], and [RES$_2$] preserve validity in all Kripke models establishing soundness. The proof is by induction on the length of the derivation of $A \longrightarrow B$. For completeness, one uses a simple *canonical* model, which effectively falsifies non-theorems. The *canonical model* $\mathcal{M}^c = (W^c, R^c_\bullet, R^c_\Diamond, V^c)$ is as below

$$\begin{array}{rcl} W^c & = & \mathsf{FORM} \text{ (the set of all formulas in the language),} \\ R^c_\bullet(ABC) & \text{iff} & \vdash A \longrightarrow B \bullet C \\ R^c_\Diamond(AB) & \text{iff} & \vdash A \longrightarrow \Diamond B, \text{ and} \\ A \in V^c(p) & \text{iff} & \vdash A \longrightarrow p. \end{array}$$

To show that the canonical model is adequate, one proves the Truth Lemma below.

LEMMA 3.8. [Truth Lemma] For any formula B, $\mathcal{M}^c, A \Vdash B$ iff $\vdash A \longrightarrow B$.

With this lemma, we can prove completeness with respect to a class of models. Suppose $\nvdash A \longrightarrow B$. Then by Lemma 3.8 $\mathcal{M}^c, A \nVdash B$. As $\mathcal{M}^c, A \Vdash A$, we have $\mathcal{M}^c \not\models A \longrightarrow B$ and hence $\not\models A \longrightarrow B$.

3.2.3 Structural Constraints

The system introduced above can accommodate different modes of accessibility relations and impose restrictions on the interpretation of the accessibility relations $R_\bullet$ and $R_\Diamond$. On the proof theoretical level this correspond to introduce structural rules.

Completeness. The above completeness result is extended to stronger logics by restricting the attention to the relevant classes of frames. In [Kur95] it is shown that one can use the tools of modal Correspondence Theory [Ben84] to generalize the completeness result above to a family of logics. A useful class of structural rules with pleasant completeness properties is characterized by Weak Sahlqvist structural rules.

DEFINITION 3.9. [Weak Sahlqvist Structural Rules]A weak Sahlqvist structural rule is a rule of the form

$$\frac{\Gamma[\Sigma'[\Phi_1, \ldots, \Phi_m]] \vdash C}{\Gamma[\Sigma[\Delta_1, \ldots, \Delta_n]] \vdash C}$$

subject to the following conditions:

i. both Σ and Σ' contains only the structural operators $\circ$, $\langle\cdot\rangle$;
ii. Σ' contains at least one structural operator;
iii. there is no repetition of variables in $\Delta_1, \ldots, \Delta_n$;

iv. all variables in $\Phi_1, \ldots, \Phi_m$ occur in $\Delta_1, \ldots, \Delta_n$.

PROPOSITION 3.10. [Sahlqvist Completeness ([Kur95])] If P is a weak Sahlqvist structural rule, then (i) $\mathsf{NL}(\Diamond)$ $+P$ is frame complete for the first order frame condition corresponding to P, and (ii) $\mathcal{L} + P$ has a canonical model whenever $\mathcal{L}$ does.

Complexity. Structural rules have an effect on the generative capacity of CTL systems. The base logic NL is strictly context free. By allowing structural rules to copy or delete type formulas, the systems become Turing-complete [Car99]. But it is shown in [Moo02] that with a linearity restriction on structural rules, one stays within PSPACE, the complexity class of context-sensitive grammars. The linearity constraint requires structural rules to be non-expanding in the sense defined below.

DEFINITION 3.11. [Non-Expanding Structural Rules] Given an antecedent configuration Σ, the length of Σ is defined as follows:

$$\begin{array}{lcl} length(\Delta_1 \circ \Delta_2) & = & length(\Delta_1) + length(\Delta_2) + 2 \\ length(\langle \Delta \rangle) & = & length(\Delta) + 1 \\ length(\Delta) & = & 0. \end{array}$$

A structural rule

$$\frac{\Gamma[\Sigma'[\Delta_1, \ldots, \Delta_n]] \vdash C}{\Gamma[\Sigma[\Delta_{\pi_1}, \ldots, \Delta_{\pi_n}]] \vdash C}$$

where Σ' is non empty, is *non-expanding* if

$$length(\Sigma[\Delta_{\pi_1}, \ldots, \Delta_{\pi_n}]) \leq length(\Sigma'[\Delta_1, \ldots, \Delta_n]).$$

3.3 Displaying Residuation and Galois Connection

In this section by means of Display Calculi (DCs), we clarify the residuation principle hidden in the logical rules of $\mathsf{NL}(\Diamond)$ and study the proof theoretical behavior of Galois unary operators obtaining $\mathsf{NL}(\Diamond, \cdot^{\mathbf{0}})$.

As explained in Chapter 2, the fundamental property of DCs which gives them their name, is the 'display property': any particular constituent of a sequent can be turned into the whole of the right or left side by moving other constituents to the other side. This property is strongly used in the general cut-elimination method. But for our approach more interesting than the display property is the ability of DCs to define the behavior of their logical operators in terms of *structural properties* —sequent rules involving only structural operators.

3.3.1 Residuation

Here we look at Binary and Unary Residuated operators.

Binary Operators

Let us introduce the appropriate logical and structural language for the DC we want to investigate. We start by the logic of residuation based only on binary operators.

Definition 3.12. [DC Language] Given a set ATOM of atomic propositional symbols and the sets $\mathsf{OP}_s = \{;, <, >\}$ and $\mathsf{OP}_l = \{\bullet, /, \backslash\}$ of structural and logical operators respectively, the set FORM of logical formulas and the set STRUCT of structural formulas are defined as

$$\begin{array}{ll} \mathsf{FORM} ::= & \mathsf{ATOM} \mid \mathsf{FORM} \bullet \mathsf{FORM} \mid \\ & \mathsf{FORM}/\mathsf{FORM} \mid \mathsf{FORM}\backslash\mathsf{FORM}. \\ \mathsf{STRUCT} ::= & \mathsf{FORM} \mid \mathsf{STRUCT}; \mathsf{STRUCT} \mid \\ & \mathsf{STRUCT} < \mathsf{STRUCT} \mid \mathsf{STRUCT} > \mathsf{STRUCT}. \end{array}$$

The behavior of the structural operators is explicitly expressed by means of display postulates. In what follows, we will use variables $\Delta, \Gamma, \Sigma, \Phi, \Psi$ to denote structures, and reserve A, B, C for logical formulas. In the case of residuation, we can directly express that $(;, <, >)$ is a residuated triple by the following structural rule [rp]. In order to avoid confusion between the logical rules of CTL and DC we mark the latter as L' and R'.

$$\frac{\Gamma \Rightarrow \Delta > \Sigma}{\Delta; \Gamma \Rightarrow \Sigma}\ [\mathrm{rp}] \qquad \frac{\Gamma \Rightarrow \Delta > \Sigma}{\Delta \Rightarrow \Sigma < \Gamma}\ [\mathrm{rp}]$$

What remains to be done is to project the residuation behavior of $(;, <, >)$ into the corresponding logical operators $(\bullet, /, \backslash)$. The general methodology is described in detail in [Gor98a]. In a nutshell, it works as follows. We are in search of a right and left introduction rule for each of the logical operators, we can obtain [•L'], [/R'] and [\R'] directly from [rp] by projection. In the literature on DCs these rules are usually called *rewrite rules* (see Figure 3.3).

To obtain the still missing rules we have to work only slightly harder. As we pointed out in Chapter 2, from the fact that $(;, <, >)$ are residuated we know their monotonicity behavior, and this is exactly what we need.

Let s be a structural operator and l its corresponding logical counterpart. In the schemata below we will select whether the consequent of the rule is $s(\Delta, \Gamma) \Rightarrow l(\Phi, \Psi)$ or $l(\Delta, \Gamma) \Rightarrow s(\Phi, \Psi)$ depending on the rule needed.

$$\frac{\Phi \Rightarrow \Delta \quad \Psi \Rightarrow \Gamma}{[l, s](\Delta, \Gamma) \Rightarrow [s, l](\Phi, \Psi)}\ \text{if } s \text{ is } [\downarrow, \downarrow] \qquad \frac{\Delta \Rightarrow \Phi \quad \Gamma \Rightarrow \Psi}{[l, s](\Delta, \Gamma) \Rightarrow [s, l](\Phi, \Psi)}\ \text{if } s \text{ is } [\uparrow, \uparrow]$$

$$\frac{\Delta \Rightarrow \Phi \quad \Psi \Rightarrow \Gamma}{[l, s](\Delta, \Gamma) \Rightarrow [s, l](\Phi, \Psi)}\ \text{if } s \text{ is } [\uparrow, \downarrow] \qquad \frac{\Phi \Rightarrow \Delta \quad \Gamma \Rightarrow \Psi}{[l, s](\Delta, \Gamma) \Rightarrow [s, l](\Phi, \Psi)}\ \text{if } s \text{ is } [\downarrow, \uparrow]$$

Applying the schemata above, we obtain [•R'], [/L'], and [\L']. The full set of rules is shown in Figure 3.3.

The rules will immediately encode the proper tonicity of the operator. It is also easy to prove that the logical operators indeed satisfy the residuation property. We show two of the required four derivations below.

$$\frac{A \Rightarrow \Delta \quad \Gamma \Rightarrow B}{A/B \Rightarrow \Delta < \Gamma}\ [/\text{L'}] \qquad \frac{\Sigma \Rightarrow A < B}{\Sigma \Rightarrow A/B}\ [/\text{R'}]$$

$$\frac{A; B \Rightarrow \Sigma}{A \bullet B \Rightarrow \Sigma}\ [\bullet\text{L'}] \qquad \frac{\Gamma \Rightarrow B \quad \Delta \Rightarrow A}{\Gamma; \Delta \Rightarrow B \bullet A}\ [\bullet\text{R'}]$$

$$\frac{\Delta \Rightarrow A \quad B \Rightarrow \Gamma}{A\backslash B \Rightarrow \Delta > \Gamma}\ [\backslash\text{L'}] \qquad \frac{\Sigma \Rightarrow A > B}{\Sigma \Rightarrow A\backslash B}\ [\backslash\text{R'}]$$

Figure 3.3: DC logical rules for residuated binary operators.

$$\dfrac{B \Rightarrow A\backslash C \quad \dfrac{A \Rightarrow A \quad C \Rightarrow C}{A\backslash C \Rightarrow A > C}\ [\backslash\text{L'}]}{\dfrac{\dfrac{B \Rightarrow A > C}{A; B \Rightarrow C}\ [\text{rp}]}{A \bullet B \Rightarrow C}\ [\bullet\text{L'}]}\ [\text{cut}] \qquad \dfrac{\dfrac{A \Rightarrow A \quad B \Rightarrow B}{A; B \Rightarrow A \bullet B}\ [\bullet\text{R'}] \quad A \bullet B \Rightarrow C}{\dfrac{\dfrac{A; B \Rightarrow C}{B \Rightarrow A > C}\ [\text{rp}]}{B \Rightarrow A\backslash C}\ [\backslash\text{R'}]}\ [\text{cut}]$$

And in a similar way we can prove the "composition property" we mentioned in Chapter 2.

As we can see, DC provides guidance in our logic engineering task of designing a sequent calculus characterizing the behavior of a triple of residuated operators. Moreover, we can readily verify the conditions specified by Belnap and conclude that the cut is admissible.

If we compare the calculus just obtained with the one introduced in Figure 3.1 we immediately notice similarities, but also important differences, the most relevant being the presence of only one structural operator, and the restriction to a single formula in the right hand side of sequents. It is not too difficult to restrict the language to obtain a perfect match (but of course, in doing so we would be giving up the display property, and "abandoning" DC and its general theorem concerning cut-elimination). Consider, for example, the [\L'] rule

$$\frac{\Delta \Rightarrow A \quad B \Rightarrow C}{A\backslash B \Rightarrow \Delta > C}\ [\backslash\text{L'}]\ \text{ by [rp] }\ \frac{\dfrac{\Delta \Rightarrow A \quad B \Rightarrow C}{A\backslash B \Rightarrow \Delta > C}\ [\backslash\text{L'}]}{\Delta; A\backslash B \Rightarrow C}\ [\text{rp}]\ \text{ hence }\ \frac{\Delta \Rightarrow A \quad B \Rightarrow C}{\Delta; A\backslash B \Rightarrow C}\ .$$

By replacing ; by $\circ$ and adding structural contexts (which are now required given that we have lost the display property) we obtain [\L]

$$\frac{\Delta \Rightarrow A \quad \Gamma[B] \Rightarrow C}{\Gamma[\Delta \circ A\backslash B] \Rightarrow C}\ [\backslash\text{L}].$$

Unary Operators

By spelling out the law of residuation for unary functions, we derive a sequent calculus that can be compiled into the standard calculus for NL($\Diamond$). Let us start by defining the proper logical and structural languages.

DEFINITION 3.13. [Logical and Structural Languages for a DC Presentation of NL($\Diamond$))] Given a set ATOM of atomic propositional symbols and the sets $\mathsf{OP}_s = \{\bullet, \circ\}$ and $\mathsf{OP}_l = \{\Diamond, \Box^{\downarrow}\}$ of structural and logical operators[3], the set FORM of logical formulas and the set STRUCT of structures for a display calculus presentation of NL($\Diamond$) are defined as

$$\mathsf{FORM} ::= \mathsf{ATOM} \mid \Diamond\mathsf{FORM} \mid \Box^{\downarrow}\mathsf{FORM}.$$

$$\mathsf{STRUCT} ::= \mathsf{FORM} \mid \bullet\,\mathsf{STRUCT} \mid \circ\,\mathsf{STRUCT}.$$

Again we start by specifying the residuated behavior of the structural pair $(\circ, \bullet)$,

$$\frac{\bullet\Delta \Rightarrow \Gamma}{\Delta \Rightarrow \circ\Gamma}\ [\mathrm{rp}].$$

And we obtain the rules for the logical operators by projection and monotonicity behavior. The full set of rules is given in Figure 3.4.

$$\frac{A \Rightarrow \Delta}{\Box^{\downarrow}A \Rightarrow \circ\Delta}\ [\Box^{\downarrow}\mathrm{L'}] \qquad \frac{\Delta \Rightarrow \circ A}{\Delta \Rightarrow \Box^{\downarrow}A}\ [\Box^{\downarrow}\mathrm{R'}]$$

$$\frac{\bullet A \Rightarrow \Delta}{\Diamond A \Rightarrow \Delta}\ [\Diamond\mathrm{L'}] \qquad \frac{\Delta \Rightarrow A}{\bullet\Delta \Rightarrow \Diamond A}\ [\Diamond\mathrm{R'}]$$

Figure 3.4: DC logical rules for residuated unary operators.

We can prove that $(\Diamond, \Box^{\downarrow})$ is a residuated pair.

$$\dfrac{A \Rightarrow \Box^{\downarrow}B \quad \dfrac{\dfrac{B \Rightarrow B}{\Box^{\downarrow}B \Rightarrow \circ B}\ [\Box^{\downarrow}\mathrm{L'}]}{A \Rightarrow \circ B}\ [\mathrm{cut}]}{\dfrac{\bullet A \Rightarrow B}{\Diamond A \Rightarrow B}\ [\mathrm{rp}]}\ [\Diamond\mathrm{L'}] \qquad \dfrac{\dfrac{\dfrac{A \Rightarrow A}{\bullet A \Rightarrow \Diamond A}\ [\Diamond\mathrm{R'}] \quad \Diamond A \Rightarrow B}{\bullet A \Rightarrow B}\ [\mathrm{cut}]}{A \Rightarrow \circ B}\ [\mathrm{rp}]}{A \Rightarrow \Box^{\downarrow}B}\ [\Box^{\downarrow}\mathrm{R'}]$$

Now we "compile" the structural postulate [rp] to obtain the logical rules in the standard Gentzen presentation of NL($\Diamond$) as we did in the case of binary operators. We spell out the needed steps for the $\Box^{\downarrow}$ operator and obtain the rules $[\Box^{\downarrow}\mathrm{L}]$ and $[\Box^{\downarrow}\mathrm{R}]$ as presented in [Moo97] —by replacing $\bullet$ by $\langle\cdot\rangle$.

$$\frac{A \Rightarrow B}{\Box^{\downarrow}A \Rightarrow \circ B}\ [\Box^{\downarrow}\mathrm{L'}]\ \text{ by [rp] }\ \frac{\dfrac{A \Rightarrow B}{\Box^{\downarrow}A \Rightarrow \circ B}\ [\Box^{\downarrow}\mathrm{L'}]}{\bullet\Box^{\downarrow}A \Rightarrow B}\ [\mathrm{rp}]\ \text{ by compilation }\ \frac{\Gamma[A] \Rightarrow B}{\Gamma[\langle\Box^{\downarrow}A\rangle] \Rightarrow B}\ [\Box^{\downarrow}\mathrm{L}].$$

$$\frac{\Gamma \Rightarrow \circ A}{\Gamma \Rightarrow \Box^{\downarrow}A}\ [\Box^{\downarrow}\mathrm{R'}]\ \text{ by [rp] }\ \frac{\dfrac{\bullet\Gamma \Rightarrow A}{\Gamma \Rightarrow \circ A}\ [\mathrm{rp}]}{\Gamma \Rightarrow \Box^{\downarrow}A}\ [\Box^{\downarrow}\mathrm{R'}]\ \text{ by compilation }\ \frac{\langle\Gamma\rangle \Rightarrow A}{\Gamma \Rightarrow \Box^{\downarrow}A}\ [\Box^{\downarrow}\mathrm{R}].$$

The logical rules of the $\Diamond$ are obtained straightforwardly.

[3]Note that the $\circ$ of DC is not the same as the binary one used in NL($\Diamond$).

3.3.2 Galois Connected Operations

Galois connected operators have been also studied in the context of Linear Logic, see [Lam93, Abr91, Gor98c, Res00], where they are intended to exhibit negation-like behavior. This means that the Galois properties have to be mixed with extra features guaranteeing, for example, a double negation law ${}^{\mathbf{0}}({A}^{\mathbf{0}}) = A = ({}^{\mathbf{0}}A)^{\mathbf{0}}$. In related work, Lambek [Lam99, Lam01] considers algebraic structures he calls *pregroups*, where each element a has a left and a right *adjoint*, written a^l and a^r. Also in these structures, one has $a^{lr} = a = a^{rl}$. In this course, we do not consider these stronger notions, but we concentrate on the pure Galois properties and investigate the effects of adding ${}^{\mathbf{0}}\cdot, \cdot^{\mathbf{0}}$ to the base multimodal logic $\mathsf{NL}(\diamond)$. Rember that we are interested in the base logic because we think it opens a window on the *invariants* of grammatical composition —the laws of the base logic are universals in the sense that they do not depend on structural postulates (that is, non-logical axioms).

3.3.3 Axiomatic Presentation of $\mathsf{NL}(\diamond,\cdot^{\mathbf{0}})$

There are two ways to extend the standard axiomatic presentation of $\mathsf{NL}(\diamond)$ with Galois operators to obtain $\mathsf{NL}(\diamond,\cdot^{\mathbf{0}})$. A system in Hilbert style $\mathsf{Hil\text{-}NL}(\diamond,\cdot^{\mathbf{0}})$ can be obtained by extending $\mathsf{NL}(\diamond)$ with the axioms [A1], [A2] and the rules [R1], [R2] below. It is easy to show that [GC] is a derived rule in this setting. Alternatively, one adds [GC] to $\mathsf{NL}(\diamond)$. It can be shown then that [A1], [A2] and the rules [R1], [R2] are derivable[4].

$$
\begin{array}{ll}
[\mathrm{A1}] & \vdash A \longrightarrow {}^{\mathbf{0}}({A}^{\mathbf{0}}). \\
[\mathrm{A2}] & \vdash A \longrightarrow ({}^{\mathbf{0}}A)^{\mathbf{0}}. \\
[\mathrm{R1}] & \text{From } \vdash A \longrightarrow B \text{ infer } \vdash B^{\mathbf{0}} \longrightarrow A^{\mathbf{0}}. \\
[\mathrm{R2}] & \text{From } \vdash A \longrightarrow B \text{ infer } \vdash {}^{\mathbf{0}}B \longrightarrow {}^{\mathbf{0}}A. \\
[\mathrm{GC}] & \vdash A \longrightarrow {}^{\mathbf{0}}B \text{ if and only if } \vdash B \longrightarrow A^{\mathbf{0}}.
\end{array}
$$

The Kripke style semantics of $\mathsf{NL}(\diamond)$ can be straightforwardly extended to $\mathsf{NL}(\diamond,\cdot^{\mathbf{0}})$. A model for $\mathsf{NL}(\diamond,\cdot^{\mathbf{0}})$ is a tuple $\mathcal{M} = (W, R^3_\bullet, R^2_\diamond, R^2_{\mathbf{0}}, V)$, where W, V and the accessibility relations $R^3_\bullet$ and $R^2_\diamond$ are as before. The new binary relation $R^2_{\mathbf{0}}$ governs the Galois connected pair $({}^{\mathbf{0}}\cdot, \cdot^{\mathbf{0}})$. For simplicity, in what follows we will restrict ourselves to models $\mathcal{M} = \langle W, R, V\rangle$ where R is the relation governing the Galois operators.

Given a model $\mathcal{M} = (W, R, V)$ and $x, y \in W$ we define

$$
\begin{array}{lll}
\mathcal{M}, x \Vdash A^{\mathbf{0}} & \text{iff} & \forall y(Rxy \rightarrow \mathcal{M}, y \nVdash A). \\
\mathcal{M}, x \Vdash {}^{\mathbf{0}}A & \text{iff} & \forall y(Ryx \rightarrow \mathcal{M}, y \nVdash A).
\end{array}
$$

It is easy to show that the axioms [A1], and [A2] are true in all Kripke models, and that the rules [R1] and [R2] preserve validity, establishing soundness. For completeness, we can extend the formula-based canonical construction for $\mathsf{NL}(\diamond)$. The *canonical model* $\mathcal{M}^c = (W^c, R^c, V^c)$ has

[4]Note that a similar alternative presentation could have been given while introducing $\mathsf{NL}(\diamond)$. There as well, we could obtain a Hilbert style system based on the composition of residuated type forming operators and on their monotonicity properties.

$$\begin{array}{rcl} W^c & = & \mathsf{FORM} \text{ (the set of all formulas in the language),} \\ \neg R^c(AB) & \text{iff} & \vdash A \longrightarrow B^{\mathbf{0}}, \text{ and} \\ A \in V^c(p) & \text{iff} & \vdash A \longrightarrow p. \end{array}$$

Notice that we define when two elements of W are *not* related by R. This, of course, defines also which elements are related. But we can do even better than $\mathcal{M}^c$. Given an arrow $A \longrightarrow B$, we can restrict W^c to be simply $W^c = \mathsf{Sub}(A) \cup \mathsf{Sub}(B)$ (the set of subformulas of A and B) and prove the following truth lemma.

Lemma 3.14. [Truth Lemma] Given $A \longrightarrow B$, then for all $C, D \in \mathsf{Sub}(A) \cup \mathsf{Sub}(B)$ $\mathcal{M}^c, C \Vdash D$ iff $\vdash C \longrightarrow D$.

With this lemma, we can prove completeness with respect to a class of finite models, and hence obtain also decidability (actually, even an upper bound on complexity).

Proof. The proof proceeds by induction on the complexity of the consequent formula. For $B \in \mathsf{ATOM}$, $\mathcal{M}^c, A \Vdash B$ iff $A \in V^c(B)$ iff, by definition of V^c, $\vdash A \longrightarrow B$. We assume as induction hypothesis (IH) that the lemma is true for formulas of lower or equal complexity than B.

We consider ${}^{\mathbf{0}}B$ (the case for $B^{\mathbf{0}}$ being even simpler).

[$\Rightarrow$] direction. $\mathcal{M}^c, A \Vdash {}^{\mathbf{0}}B$ iff for all $D \in W^c$ if R^cDA then $\mathcal{M}^c, D \nVdash B$. By contraposition and definition of R^c, for all D, $\mathcal{M}^c, D \Vdash B$ implies $\vdash D \longrightarrow A^{\mathbf{0}}$. By definition of W^c, D is in $\mathsf{Sub}(A) \cup \mathsf{Sub}(B)$ and we can apply IH to obtain that for all $D \in W^c$, $\vdash D \longrightarrow B$ implies $\vdash D \longrightarrow A^{\mathbf{0}}$. In particular, $B \in W^c$ and by [REFL] $\vdash B \longrightarrow B$, hence $\vdash B \longrightarrow A^{\mathbf{0}}$. By [GC], $\vdash A \longrightarrow {}^{\mathbf{0}}B$.

[$\Leftarrow$] direction. Assume $\vdash A \longrightarrow {}^{\mathbf{0}}B$ to prove $\mathcal{M}^c, A \Vdash {}^{\mathbf{0}}B$. Take D such that R^cDA, we should prove $\mathcal{M}^c, D \nVdash B$. Notice that by definition of R^c, we have that $\nvdash D \longrightarrow A^{\mathbf{0}}$. For contradiction, suppose $\mathcal{M}^c, D \Vdash B$, then by IH, $\vdash D \longrightarrow B$, but then we can prove $\vdash D \longrightarrow A^{\mathbf{0}}$ as follows

$$\dfrac{\vdash A \longrightarrow {}^{\mathbf{0}}B \quad \dfrac{\vdash D \longrightarrow B}{\vdash {}^{\mathbf{0}}B \longrightarrow {}^{\mathbf{0}}D}\,[R2]}{\dfrac{\vdash A \longrightarrow {}^{\mathbf{0}}D}{\vdash D \longrightarrow A^{\mathbf{0}}}\,[\text{GC}]}\,[\text{TRANS}] \tag{3.1}$$

QED

Theorem 3.15. [Completeness] Given $A \longrightarrow B$, then $\models A \longrightarrow B$ implies $\vdash A \longrightarrow B$.

Proof. Suppose $\nvdash A \longrightarrow B$. Then by Lemma 3.14 $\mathcal{M}^c, A \nVdash B$. As $\mathcal{M}^c, A \Vdash A$, we have $\mathcal{M}^c \not\models A \longrightarrow B$ and hence $\not\models A \longrightarrow B$. QED

As we already said, Lemma 3.14 actually establishes a strong finite model property (an arrow $A \longrightarrow B$ is valid iff B is satisfied in $\mathcal{M}^c, A$, a (pointed) model whose size is polynomial in $|A| \cup |B|$). From this, an NP upper bound in the complexity of the validity problem for $\mathsf{NL}(\Diamond, \cdot^{\mathbf{0}})$ follows.

Theorem 3.16. Given $A \longrightarrow B \in \mathsf{NL}(\Diamond, \cdot^{\mathbf{0}})$, deciding whether $A \longrightarrow B$ is valid can be done in non-deterministic polynomial time.

Displaying Galois Connected Operations

The method we have used above when looking at the logics of residuation can handle other kinds of algebraic properties, assuming that they can be encoded in terms of display rules. In this section we apply this method to the Galois connections.

The steps we will take to provide a DC encoding [GC] should be familiar by now. We start by explicitly writing the algebraic property characterizing a Galois connection for a pair of structural operators $(\natural, \flat)$.

$$\frac{\Gamma \Rightarrow \flat\Delta}{\Delta \Rightarrow \natural\Gamma}\ [\text{gc}].$$

We now project this behavior into the logical operators $({}^{\mathbf{0}}\cdot, \cdot^{\mathbf{0}})$ as it is shown in Figure 3.5.

$$\frac{\Sigma \Rightarrow A}{{}^{\mathbf{0}}A \Rightarrow \flat\Sigma}\ [{}^{\mathbf{0}}\cdot\text{L'}] \qquad \frac{\Sigma \Rightarrow \flat A}{\Sigma \Rightarrow {}^{\mathbf{0}}A}\ [{}^{\mathbf{0}}\cdot\text{R'}]$$

$$\frac{\Sigma \Rightarrow A}{A^{\mathbf{0}} \Rightarrow \natural\Sigma}\ [\cdot^{\mathbf{0}}\text{L'}] \qquad \frac{\Sigma \Rightarrow \natural A}{\Sigma \Rightarrow A^{\mathbf{0}}}\ [\cdot^{\mathbf{0}}\text{R'}]$$

Figure 3.5: DC logical rules for Galois connected unary operators.

To move closer to standard sequent presentations of CTL, we need to compile [gc] into the logical rules. We can take $[{}^{\mathbf{0}}\cdot\text{L}]$ and $[\cdot^{\mathbf{0}}\text{L}]$ as they are. To obtain $[{}^{\mathbf{0}}\cdot\text{R}]$ and $[\cdot^{\mathbf{0}}\text{R}]$ we need to apply [gc].

$$\frac{\Sigma \Rightarrow \flat A}{\Sigma \Rightarrow {}^{\mathbf{0}}A}\ [{}^{\mathbf{0}}\cdot\text{R'}] \quad \text{by [gc]} \quad \frac{\dfrac{A \Rightarrow \natural\Sigma}{\Sigma \Rightarrow \flat A}\ [\text{gc}]}{\Sigma \Rightarrow {}^{\mathbf{0}}A}\ [{}^{\mathbf{0}}\cdot\text{R'}] \quad \text{by compilation} \quad \frac{A \Rightarrow \natural\Sigma}{\Sigma \Rightarrow {}^{\mathbf{0}}A}\ [{}^{\mathbf{0}}\cdot\text{R}].$$

$$\frac{\Sigma \Rightarrow \natural A}{\Sigma \Rightarrow A^{\mathbf{0}}}\ [\cdot^{\mathbf{0}}\text{R'}] \quad \text{by [gc]} \quad \frac{\dfrac{A \Rightarrow \flat\Sigma}{\Sigma \Rightarrow \natural A}\ [\text{gc}]}{\Sigma \Rightarrow A^{\mathbf{0}}}\ [\cdot^{\mathbf{0}}\text{R'}] \quad \text{by compilation} \quad \frac{A \Rightarrow \flat\Sigma}{\Sigma \Rightarrow A^{\mathbf{0}}}\ [\cdot^{\mathbf{0}}\text{R}].$$

The full set of rules obtained is shown in Figure 3.6. Notice that given the nature of Galois connections (which involves a permutation in the order of the poset), it is not possible to eliminate the structural operators from the right hand side of the sequents. This is an important difference with respect to what we obtained in the previous section. The proofs below show that the ${}^{\mathbf{0}}\cdot$ and $\cdot^{\mathbf{0}}$ operators are indeed Galois connected,

$$\dfrac{A \Rightarrow B^{\mathbf{0}} \quad \dfrac{B \Rightarrow B}{B^{\mathbf{0}} \Rightarrow \natural B}\ [\cdot^{\mathbf{0}}\text{L}]}{\dfrac{A \Rightarrow \natural B}{B \Rightarrow {}^{\mathbf{0}}A}\ [{}^{\mathbf{0}}\cdot\text{R}]}\ [\text{cut}] \qquad \dfrac{A \Rightarrow {}^{\mathbf{0}}B \quad \dfrac{B \Rightarrow B}{{}^{\mathbf{0}}B \Rightarrow \flat B}\ [{}^{\mathbf{0}}\cdot\text{L}]}{\dfrac{A \Rightarrow \flat B}{B \Rightarrow A^{\mathbf{0}}}\ [\cdot^{\mathbf{0}}\text{R}]}\ [\text{cut}]$$

That is, the rule [gc] holds for ${}^{\mathbf{0}}\cdot$ and $\cdot^{\mathbf{0}}$. Moreover, the operators satisfy the appropriate Galois composition laws [gcl].

$$\frac{\Sigma \Rightarrow A}{{}^{\mathbf{0}}A \Rightarrow \flat\Sigma}\ [{}^{\mathbf{0}}\cdot\mathrm{L}] \qquad \frac{A \Rightarrow \natural\Sigma}{\Sigma \Rightarrow {}^{\mathbf{0}}A}\ [{}^{\mathbf{0}}\cdot\mathrm{R}]$$

$$\frac{\Sigma \Rightarrow A}{A^{\mathbf{0}} \Rightarrow \natural\Sigma}\ [\cdot^{\mathbf{0}}\mathrm{L}] \qquad \frac{A \Rightarrow \flat\Sigma}{\Sigma \Rightarrow A^{\mathbf{0}}}\ [\cdot^{\mathbf{0}}\mathrm{R}]$$

$$\frac{\Delta \Rightarrow \Gamma \quad \Gamma \Rightarrow \Sigma}{\Delta \Rightarrow \Sigma}\ [\mathrm{cut}]$$

Figure 3.6: Compiled logical rules for Galois connected unary operators.

$$\frac{\dfrac{A \Rightarrow A}{A^{\mathbf{0}} \Rightarrow \natural A}\ [\cdot^{\mathbf{0}}\mathrm{L}]}{A \Rightarrow {}^{\mathbf{0}}(A^{\mathbf{0}})}\ [{}^{\mathbf{0}}\cdot\mathrm{R}] \qquad \frac{\dfrac{A \Rightarrow A}{{}^{\mathbf{0}}A \Rightarrow \flat A}\ [{}^{\mathbf{0}}\cdot\mathrm{L}]}{A \Rightarrow ({}^{\mathbf{0}}A)^{\mathbf{0}}}\ [\cdot^{\mathbf{0}}\mathrm{R}]$$

From these, the fact that the operators are $[\downarrow]$-functions follows immediately.

$$\frac{\dfrac{A \Rightarrow B}{A \Rightarrow {}^{\mathbf{0}}(B^{\mathbf{0}})}\ [\mathrm{gcl}]}{B^{\mathbf{0}} \Rightarrow A^{\mathbf{0}}}\ [\mathrm{gc}] \qquad \frac{\dfrac{A \Rightarrow B}{A \Rightarrow ({}^{\mathbf{0}}B)^{\mathbf{0}}}\ [\mathrm{gcl}]}{{}^{\mathbf{0}}B \Rightarrow {}^{\mathbf{0}}A}\ [\mathrm{gc}]$$

The system so obtained does not enjoy cut-elimination. When interested in computational aspect of the system, this is an essential property to achieve. In the next section we present a solution to this problem.

Cut-Free Sequent Calculus

In this section we show how the cut-rule of the system we have reached by applying our method can be eliminated yielding a decidable proof search.

In the Gentzen presentation, we want to compile away the display postulate [gc], but also part of the cut-rule, so to obtain a cut-free system. Because the Galois connected operators are order-reversing, we have to distinguish positive and negative contexts in the statement of the cut-rule. We write $\Gamma[\Delta]$ for a structure Γ with a substructure Δ in an isotone position (dominated by an even number of occurrence of $\flat$ or $\natural$), and $\Gamma\{\Delta\}$ for a structure Γ with Δ in an antitone position (dominated by an odd number of $\flat$ or $\natural$). In (3.2) we give the four instances of the cut-rule we have to consider.

$$\begin{array}{cc} \dfrac{\Delta \Rightarrow A \quad \Gamma[A] \Rightarrow \Delta'}{\Gamma[\Delta] \Rightarrow \Delta'}\ [\mathrm{cut}_1] & \dfrac{\Delta' \Rightarrow \Gamma[A] \quad A \Rightarrow \Delta}{\Delta' \Rightarrow \Gamma[\Delta]}\ [\mathrm{cut}_2] \\[2ex] \dfrac{\Delta \Rightarrow A \quad \Delta' \Rightarrow \Gamma\{A\}}{\Delta' \Rightarrow \Gamma\{\Delta\}}\ [\mathrm{cut}_3] & \dfrac{\Gamma\{A\} \Rightarrow \Delta' \quad A \Rightarrow \Delta}{\Gamma\{\Delta\} \Rightarrow \Delta'}\ [\mathrm{cut}_4] \end{array} \tag{3.2}$$

The logical rules we have obtained in the previous section (Figure 3.6) swap around antecedent and succedent of a sequent. For cut-elimination to go through, we also need contextual versions of the rules, compiling in the axiom schemata [A1]/[A2] with

$[\mathrm{cut}_2]/[\mathrm{cut}_4]$. The full system $\mathsf{NL}(\Diamond,\cdot^{\mathbf{0}})$ is given in Figure 3.7. We will call $\mathsf{Seq\text{-}NL}(\Diamond,\cdot^{\mathbf{0}})$ the Gentzen system introduced in this section, to distinguish it from its Hilbert-style axiomatization $\mathsf{Hil\text{-}NL}(\Diamond,\cdot^{\mathbf{0}})$. When the difference on the presentation is irrelevant we will use the unmarked $\mathsf{NL}(\Diamond,\cdot^{\mathbf{0}})$.

THEOREM 3.17. [Cut-Elimination] In $\mathsf{Seq\text{-}NL}(\Diamond,\cdot^{\mathbf{0}})$, every valid sequent $A \longrightarrow B$ has a cut-free proof.

The proof proceeds by induction on the complexity of the cut-inferences. Below, we present the principal cases of the cut-elimination transformation: the cases where a cut on a complex cut-formula is replaced by a cut on its sub-formula, thus decreasing the complexity. The other cases follow the same ideas.

In (3.3) and (3.4), isotone cuts $[\mathrm{cut}_1]$, $[\mathrm{cut}_3]$ on the complex formula $A^{\mathbf{0}}$ are replaced by antitone cuts $[\mathrm{cut}_4]$, $[\mathrm{cut}_2]$. Similarly for cuts on ${}^{\mathbf{0}}A$. (We use double lines for the instantiation of the premise that makes a logical rule applicable.)

$$\dfrac{\dfrac{A \Rightarrow \flat\Delta}{\Delta \Rightarrow A^{\mathbf{0}}}\,[\cdot^{\mathbf{0}}\mathrm{R}] \quad \dfrac{\dfrac{\Gamma\{A\} \Rightarrow \Delta'}{\Gamma\{\flat(A^{\mathbf{0}})\} \Rightarrow \Delta'}\,[\cdot^{\mathbf{0}}\mathrm{L}^{+}]}{\Gamma[A^{\mathbf{0}}] \Rightarrow \Delta'}}{\dfrac{\Gamma[\Delta] \Rightarrow \Delta'}{\Gamma\{\flat\Delta\} \Rightarrow \Delta'}}\,[\mathrm{cut}_1] \quad \rightsquigarrow \quad \dfrac{\Gamma\{A\} \Rightarrow \Delta' \quad A \Rightarrow \flat\Delta}{\Gamma\{\flat\Delta\} \Rightarrow \Delta'}\,[\mathrm{cut}_4] \tag{3.3}$$

$$\dfrac{\dfrac{A \Rightarrow \flat\Delta}{\Delta \Rightarrow A^{\mathbf{0}}}\,[\cdot^{\mathbf{0}}\mathrm{R}] \quad \dfrac{\dfrac{\Delta' \Rightarrow \Gamma[A]}{\Delta' \Rightarrow \Gamma[\flat(A^{\mathbf{0}})]}\,[\cdot^{\mathbf{0}}\mathrm{R}^{-}]}{\Delta' \Rightarrow \Gamma\{A^{\mathbf{0}}\}}}{\dfrac{\Delta' \Rightarrow \Gamma\{\Delta\}}{\Delta' \Rightarrow \Gamma[\flat\Delta]}}\,[\mathrm{cut}_3] \quad \rightsquigarrow \quad \dfrac{\Delta' \Rightarrow \Gamma[A] \quad A \Rightarrow \flat\Delta}{\Delta' \Rightarrow \Gamma[\flat\Delta]}\,[\mathrm{cut}_2] \tag{3.4}$$

In (3.5) and (3.6), antitone cuts $[\mathrm{cut}_4]$, $[\mathrm{cut}_2]$ are replaced by isotone cuts $[\mathrm{cut}_1]$, $[\mathrm{cut}_3]$.

$$\dfrac{\dfrac{\Gamma\{\natural A\} \Rightarrow \Delta'}{\Gamma\{A^{\mathbf{0}}\} \Rightarrow \Delta'}\,(\cdot^{\mathbf{0}}\mathrm{L}^{-}) \quad \dfrac{\Delta \Rightarrow A}{A^{\mathbf{0}} \Rightarrow \natural\Delta}\,[\cdot^{\mathbf{0}}\mathrm{L}]}{\Gamma\{\natural\Delta\} \Rightarrow \Delta'}\,[\mathrm{cut}_4] \quad \rightsquigarrow \quad \dfrac{\Delta \Rightarrow A \quad \Gamma\{\natural A\} \Rightarrow \Delta'}{\Gamma\{\natural\Delta\} \Rightarrow \Delta'}\,[\mathrm{cut}_1] \tag{3.5}$$

$$\dfrac{\dfrac{\Delta' \Rightarrow \Gamma[\natural A]}{\Delta' \Rightarrow \Gamma[A^{\mathbf{0}}]}\,[\cdot^{\mathbf{0}}\mathrm{R}^{+}] \quad \dfrac{\Delta \Rightarrow A}{A^{\mathbf{0}} \Rightarrow \natural\Delta}\,[\cdot^{\mathbf{0}}\mathrm{L}]}{\Delta' \Rightarrow \Gamma[\natural\Delta]}\,[\mathrm{cut}_2] \quad \rightsquigarrow \quad \dfrac{\Delta \Rightarrow A \quad \Delta' \Rightarrow \Gamma[\natural A]}{\Delta' \Rightarrow \Gamma[\natural\Delta]}\,[\mathrm{cut}_3] \tag{3.6}$$

We can also establish soundness and completeness on the basis of the sequent presentation.

$$\frac{}{A \Rightarrow A}\ [\text{axiom}]$$

$$\frac{\Delta \Rightarrow B \quad \Gamma[A] \Rightarrow C}{\Gamma[(A/B \circ \Delta)] \Rightarrow C}\ [/\text{L}] \qquad \frac{\Gamma \circ B \Rightarrow A}{\Gamma \Rightarrow A/B}\ [/\text{R}]$$

$$\frac{\Delta \Rightarrow B \quad \Gamma[A] \Rightarrow C}{\Gamma[(\Delta \circ B\backslash A)] \Rightarrow C}\ [\backslash\text{L}] \qquad \frac{B \circ \Gamma \Rightarrow A}{\Gamma \Rightarrow B\backslash A}\ [\backslash\text{R}]$$

$$\frac{\Gamma[(A \circ B)] \Rightarrow C}{\Gamma[A \bullet B] \Rightarrow C}\ [\bullet\text{L}] \qquad \frac{\Gamma \Rightarrow A \quad \Delta \Rightarrow B}{(\Gamma \circ \Delta) \Rightarrow A \bullet B}\ [\bullet\text{R}]$$

$$\frac{\Gamma[A] \Rightarrow B}{\Gamma[\langle \Box^{\downarrow} A\rangle] \Rightarrow B}\ [\Box^{\downarrow}\text{L}] \qquad \frac{\langle \Gamma\rangle \Rightarrow A}{\Gamma \Rightarrow \Box^{\downarrow} A}\ [\Box^{\downarrow}\ \text{R}]$$

$$\frac{\Gamma[\langle A\rangle] \Rightarrow B}{\Gamma[\Diamond A] \Rightarrow B}\ [\Diamond\text{L}] \qquad \frac{\Gamma \Rightarrow A}{\langle \Gamma\rangle \Rightarrow \Diamond A}\ [\Diamond\text{R}]$$

$$\frac{\Delta \Rightarrow A}{{}^{\mathbf{0}}A \Rightarrow \flat\Delta}\ [{}^{\mathbf{0}}\cdot\text{L}] \qquad \frac{A \Rightarrow \natural\Delta}{\Delta \Rightarrow {}^{\mathbf{0}}A}\ [{}^{\mathbf{0}}\cdot\text{R}]$$

$$\frac{\Delta \Rightarrow A}{A^{\mathbf{0}} \Rightarrow \natural\Delta}\ [\cdot^{\mathbf{0}}\text{L}] \qquad \frac{A \Rightarrow \flat\Delta}{\Delta \Rightarrow A^{\mathbf{0}}}\ [\cdot^{\mathbf{0}}\text{R}]$$

$$\frac{\Gamma\{A\} \Rightarrow \Delta}{\Gamma\{\natural^{\mathbf{0}} A\} \Rightarrow \Delta}\ [{}^{\mathbf{0}}\cdot\text{L}^{+}] \qquad \frac{\Delta \Rightarrow \Gamma[A]}{\Delta \Rightarrow \Gamma[\natural^{\mathbf{0}} A]}\ [{}^{\mathbf{0}}\cdot\text{R}^{-}]$$

$$\frac{\Gamma\{A\} \Rightarrow \Delta}{\Gamma\{\flat A^{\mathbf{0}}\} \Rightarrow \Delta}\ [\cdot^{\mathbf{0}}\text{L}^{+}] \qquad \frac{\Delta \Rightarrow \Gamma[A]}{\Delta \Rightarrow \Gamma[\flat A^{\mathbf{0}}]}\ [\cdot^{\mathbf{0}}\text{R}^{-}]$$

$$\frac{\Delta \Rightarrow \Gamma[\flat A]}{\Delta \Rightarrow \Gamma[{}^{\mathbf{0}}A]}\ [{}^{\mathbf{0}}\cdot\text{L}^{-}] \qquad \frac{\Gamma\{\flat A\} \Rightarrow \Delta}{\Gamma\{{}^{\mathbf{0}}A\} \Rightarrow \Delta}\ [{}^{\mathbf{0}}\cdot\text{R}^{+}]$$

$$\frac{\Delta \Rightarrow \Gamma[\natural A]}{\Delta \Rightarrow \Gamma[A^{\mathbf{0}}]}\ [\cdot^{\mathbf{0}}\text{L}^{-}] \qquad \frac{\Gamma\{\natural A\} \Rightarrow \Delta}{\Gamma\{A^{\mathbf{0}}\} \Rightarrow \Delta}\ [\cdot^{\mathbf{0}}\text{R}^{+}]$$

Figure 3.7: Logical rules of $\mathsf{NL}(\Diamond,\cdot^{\mathbf{0}})$.

Soundness and completeness Seq-NL($\Diamond$,$\cdot^{\mathbf{0}}$)

We start by proving the following.

PROPOSITION 3.18. Let $\mathcal{M} = \langle W, R, V\rangle$ be a model, and $x \in W$ then

1. $\mathcal{M}, x \models \Delta \longrightarrow \Delta'[A]$ and $\models A \longrightarrow B$ then $\Delta \longrightarrow \Delta'[B]$.
2. $\mathcal{M}, x \models \Delta[A] \longrightarrow \Delta'$ and $\models B \longrightarrow A$ then $\Delta[B] \longrightarrow \Delta'$.
3. $\mathcal{M}, x \models \Delta \longrightarrow \Delta'\{A\}$ and $\models B \longrightarrow A$ then $\Delta \longrightarrow \Delta'\{B\}$.
4. $\mathcal{M}, x \models \Delta\{A\} \longrightarrow \Delta'$ and $\models A \longrightarrow B$ then $\Delta\{B\} \longrightarrow \Delta'$.

PROOF. By induction on the number of operators surrounding A. QED

Now define the following forgetting function.

DEFINITION 3.19. We define the translation $Tr : \mathsf{STRUCT} \to \mathsf{FORM}$ as follows,

$$Tr(p) = p \text{ for } p \in \mathsf{ATOM}$$

$$\begin{array}{rclcrcl} Tr(\mathbf{0}(A)) & = & \mathbf{0}(Tr(A)) & & Tr(\flat(A)) & = & \mathbf{0}(Tr(A)) \\ Tr((A)\mathbf{0}) & = & (Tr(A))\mathbf{0} & & Tr(\natural(A)) & = & (Tr(A))\mathbf{0}. \end{array}$$

THEOREM 3.20. [Soundness of Seq-NL($\Diamond$,$\cdot^{\mathbf{0}}$)] The sequent presentation of NL($\Diamond$,$\cdot^{\mathbf{0}}$) is sound.

PROOF. Given a rule

$$\frac{A \Rightarrow B}{C \Rightarrow D}$$

we prove that if $\models Tr(A) \longrightarrow Tr(B)$ then $\models Tr(C) \longrightarrow Tr(D)$, and similarly for rules with two premises.

Notice that Proposition 3.18 proves soundness of the cut-rules. For rules [$\mathbf{0}\cdot$R$^+$], [$\cdot\mathbf{0}$R$^+$] it is trivial, [$\mathbf{0}\cdot$L] For rules [$\mathbf{0}\cdot$R], [$\cdot\mathbf{0}$R], [$\mathbf{0}\cdot$L] and [$\cdot\mathbf{0}$L] use the fact that the [GC] rule is sound. For rules [$\mathbf{0}\cdot$R$^-$], [$\cdot\mathbf{0}$R$^-$], [$\mathbf{0}\cdot$L$^+$] and [$\cdot\mathbf{0}$L$^+$] use Proposition 3.18 plus the fact that axioms [A1] and [A2] are valid. QED

THEOREM 3.21. [Equivalence of Seq-NL($\Diamond$,$\cdot^{\mathbf{0}}$) and Hil-NL($\Diamond$,$\cdot^{\mathbf{0}}$)] If $A \longrightarrow B$ is a theorem of Hil-NL($\Diamond$,$\cdot^{\mathbf{0}}$) then there is a proof of $A \longrightarrow B$ in Seq-NL($\Diamond$,$\cdot^{\mathbf{0}}$). And for every proof of a sequent $\Gamma \Rightarrow \Delta$ in Seq-NL($\Diamond$,$\cdot^{\mathbf{0}}$), $Tr(\Gamma) \Rightarrow Tr(\Delta)$ is a theorem of Hil-NL($\Diamond$,$\cdot^{\mathbf{0}}$).

Relating Galois connected and Residuated Operations

The families of residuated unary and binary operators and unary Galois connected operators of $\mathsf{NL}(\diamond,\cdot^{\mathbf{0}})$ (Figure 3.7) are totally independent and there is no interaction among them. Each family is interpreted by its own accessibility relation $R^3_{\bullet}$, $R^2_{\diamond}$ and $R^2_{\mathbf{0}}$. If we want to create some interaction among the operators there could be different ways of relating them. The interaction could be established by means of structural postulates or as in $\delta\mathbf{PC}$. (See Dunn's work on Gaggle Theory [Dun91] for a discussion of semantics for residuation and (dual) Galois connections, and [Gor98a] for a general presentation in the framework of display calculi). In these course notes we leave this question open and investigate the expressivity of the pure logic of Galois connected and residuated operators. In the next section we discuss the derivability relation among types that we will explore in Part II.

3.4 Derivability Patterns

We have seen that the binary and unary logical connectives of $\mathsf{NL}(\diamond,\cdot^{\mathbf{0}})$ are governed by the same algebraic principles of residuation and of the related ones of Galois connections. In this section, we highlight some useful derivability relations among types determined by these algebraic properties. Let us start presenting some theorems of the part of the system which is already well known, namely NL.

First of all, notice that $(X/\cdot, \cdot\backslash X)$ is a pair of Galois connected operators, therefore the algebraic principle [GC] discussed in this chapter was already hidden in the base logic of the binary residuated operators. All the derivability relations which hold for this pair hold for the unary Galois connected operators and *vice versa*. In particular, in [Lam88] it is pointed out that the lifting of a category to a higher order type, $A \longrightarrow B/(A\backslash B)$, is a closure operation, as it obeys Definition 3.22 below.

Definition 3.22. [Closure] Let $\mathcal{A} = (A, \sqsubseteq)$ be a partially ordered set. Any correspondence

$$a \sqsubseteq a^*$$

which associates with each element a some other element a^* in A shall be called a *closure operation* provided it satisfies the three conditions below:

$$a \sqsubseteq a^*, \quad a^* \sqsubseteq b^* \text{ if } a \sqsubseteq b, \quad a^{**} \sqsubseteq a^*.$$

By exploring $\mathsf{NL}(\diamond,\cdot^{\mathbf{0}})$ one soon realizes that the definition above characterizes the behavior not only of $X/(\cdot\backslash X)$, $(X/\cdot)\backslash X$, but also of ${}^{\mathbf{0}}(\cdot^{\mathbf{0}})$, $({}^{\mathbf{0}}\cdot)^{\mathbf{0}}$, $\Box^{\downarrow}\diamond(\cdot)$, when considering $\sqsubseteq$ as the derivability relation $(\longrightarrow)$ and A as the set of types FORM.

First of all, we have already seen how residuated and Galois connected operators compose (Section 3.1), *viz.* $A \longrightarrow \Box^{\downarrow}\diamond A$, $A \longrightarrow {}^{\mathbf{0}}(A^{\mathbf{0}})$ and $A \longrightarrow ({}^{\mathbf{0}}A)^{\mathbf{0}}$. Morever, we know that $\diamond\cdot$ and $\Box^{\downarrow}\cdot$ are upward monotone, whereas $\cdot^{\mathbf{0}}$ and ${}^{\mathbf{0}}\cdot$ are downward monotone. By the monotonicity calculus it follows that their compositions, $\Box^{\downarrow}\diamond\cdot$ and ${}^{\mathbf{0}}(\cdot^{\mathbf{0}})$, $({}^{\mathbf{0}}\cdot)^{\mathbf{0}}$ are upward monotone operators. Finally, the derivability relations below can easily be proved,

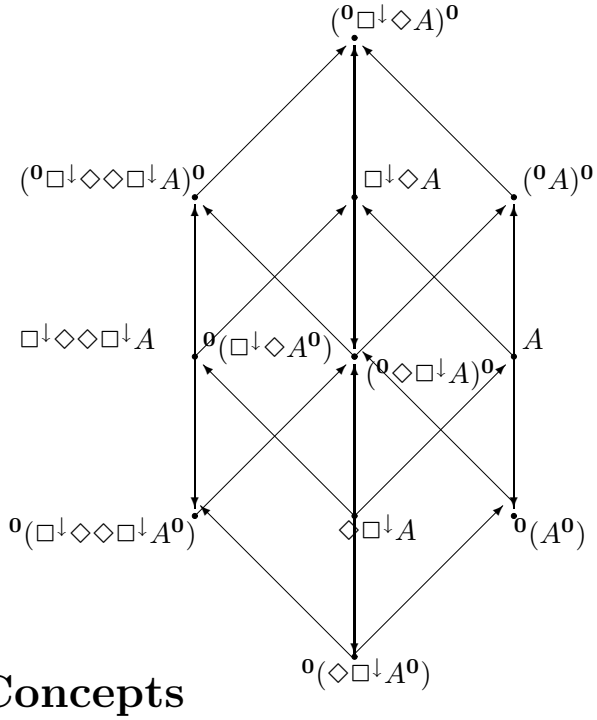
$$\Box^{\downarrow}\Diamond\Box^{\downarrow}\Diamond A \longrightarrow \Box^{\downarrow}\Diamond A \text{ and } ({}^{\mathbf{0}}(({}^{\mathbf{0}}A)^{\mathbf{0}}))^{\mathbf{0}} \longrightarrow ({}^{\mathbf{0}}A)^{\mathbf{0}}.$$

Note that since closure operations are upward monotone the above derivability relation are in fact equivalences.

Another interesting property regards triples of Galois connected operators, as commented in [Ore44]. The same behavior is exhibited by residuated pairs. Let (f_1, f_2) be either the residuated or Galois connected operators, $f_1 f_2 f_1 A$ iff $f_1 A$, and similarly $f_2 f_1 f_2 A$ iff $f_2 A$.

$${}^{\mathbf{0}}(({}^{\mathbf{0}}A)^{\mathbf{0}}) \longleftrightarrow {}^{\mathbf{0}}A \quad \text{and} \quad ({}^{\mathbf{0}}(A^{\mathbf{0}}))^{\mathbf{0}} \longleftrightarrow A^{\mathbf{0}} \quad \Diamond\Box^{\downarrow}\Diamond A \longleftrightarrow \Diamond A \text{ and } \Box^{\downarrow}\Diamond\Box^{\downarrow}A \longleftrightarrow \Box^{\downarrow}A.$$

From these simple relations an interesting net of derivability patterns can be derived. The ones we will explore in this thesis are summarized in the picture below.



3.5 Key Concepts

In this chapter by explaining why CTLs are also known as "logics of residuation", we have introduced modern extensions of the Lambek calculi. The fundamental points to be emphasized are:

1. NL($\Diamond$) is the pure logic of residuation. In other words, its operators are governed by the algebraic principle of residuation. All theorems provable in NL($\Diamond$) are consequences of this principle.

2. The algebraic structure of NL($\Diamond$) provides room for a pair of order-reversing operators, Galois connected operators $({}^{\mathbf{0}}\cdot, \cdot^{\mathbf{0}})$. The whole system NL($\Diamond$,$\cdot^{\mathbf{0}}$) is sound and complete with respect to Kripke models and is at least in NP.

3. Similarly, the same algebraic structure could accommodate dual Galois connected operators. One could extend the logic presented here with those operators.

Part II

Linguistic Analyses

The assembly of meaning is governed by a systematic correspondence between semantic types and the domains of denotation where expressions find their semantic value. Many natural language phenomena show that successful composition is dependent on finer distinctions within the denotation domains standardly assumed. In this part of the notes, we show that the extended vocabulary of type-logical constants introduced in Chapter 3 provides the means to encode the required distinctions in lexical type assignment.

In Chapter 4, we introduce the background assumptions of the categorial approach in linguistics, and we sketch the developments that have led to the introduction of CTL. Moreover, we review the different uses of unary residuated operators proposed in the CTL tradition.

In Chapter 5, we see look at unary operators as logical features and employ them in the analysis of Generalized Quantifiers and Polarity Items. We develop a type-logical account of scope construal which recasts the minimalist feature-checking mechanisms in purely deductive terms. Moreover, we study the distribution of *polarity-sensitive* expressions. We investigate compatibility and incompatibility relations from a cross-linguistic perspective, showing how we reduce distributional differences between polarity-sensitive items in Dutch, Greek and Italian to differences in the lexical type assignments of these languages.

Chapter 4

The Logical Approach in Linguistics

The framework of categorial type logic (CTL) [Moo97] developed out of earlier work in the tradition of categorial grammar. In this chapter, we briefly present these ancestral lines of research, and we give the reader an idea of the kind of problems that have led to the introduction of CTL.

The present chapter is organized as follows. We start by introducing classical and combinatory categorial grammars, two formalisms closely related to CTL (Section 4.1). Then, by highlighting the differences between these frameworks and the logical approach assumed in these course notes, we introduce the main aspects of CTL (Section 4.2). Moreover, we discuss the proof theoretical perspective on form-meaning assembly of linguistic expressions (Section 4.3).

4.1 Rule-Based Categorial Grammars

The categorial tradition of natural language analysis goes back to the pioneering works of Lesniewski [Les29] and Ajdukiewicz [Ajd35]. The ingredients of a categorial grammar are extremely simple: a system of syntactic categories (or types), and a set of rules to compute with these types. The categories are either atomic, or they are structured as 'fractions' $\frac{a}{b}$. Atomic types categorize expressions that in some intuitive sense are 'complete'; incomplete expressions are assigned a fractional category. The basic combinatory rule schema takes the form of a kind of 'multiplication': from $\frac{a}{b} \times b$ one obtains the category a. The algebraic nature of the schemata for category combination was emphasized by Bar-Hillel in [BH53].

In this section, we discuss two categorial frameworks: the classical categorial grammars of Ajdukiewicz and Bar-Hillel (CG, also known as AB grammars), and the combinatory categorial grammars of Steedman (CCG, [Ste00]). These frameworks have the same category concept, but they have different sets of rule schemata for category combination: the CCG rule set extends the schemata of CG in order to overcome certain expressive limitations of the classical categorial approach.

4.1.1 Classical Categorial Grammar

The type language and the rules of classical Categorial Grammar (CG) are defined as below.

Definition 4.1. [Type Language and Rules of CG] The language of CG is recursively built over atomic categories by means of the category forming operators $\backslash$ and $/$. The combinatorial behavior of categories is captured by the left/right application rules.

CG language. Given a set of basic categories ATOM, the set of categories CAT is the smallest set such that:

i. if $A \in \mathsf{ATOM}$, then $A \in \mathsf{CAT}$;
ii. if A and $B \in \mathsf{CAT}$, then A/B and $B\backslash A \in \mathsf{CAT}$.

There are two schemata for category combination, *backward application* (BA) and *forward application* (FA) *CG rules.*

$$A/B, B \Rightarrow A \quad [\mathtt{FA}]$$
$$B, B\backslash A \Rightarrow A \quad [\mathtt{BA}].$$

[FA] (resp. [BA]) says that when an expression of category A/B (resp. $B\backslash A$) is concatenated with an expression of category B on its right (resp. on its left), it yields a structure of category A.

To facilitate the comparison between CG and the categorial systems developed by Lambek (Section 4.2), we present CG as a deductive system (cf. Buszkowski [Bus97]). Below we define the *derives* relation, holding between a finite sequence of categories Γ and a category A.

Definition 4.2. [Derivability Relation] Let $\Rightarrow$ be the *derivability* relation between a finite non-empty sequence of categories Γ and a category B ($\Gamma \Rightarrow B$), fulfilling the following conditions:

$$A \Rightarrow A \qquad [\text{id}]$$
$$\Gamma, A, \Gamma' \Rightarrow B \text{ and } \Delta \Rightarrow A, \text{ then } \Gamma, \Delta, \Gamma' \Rightarrow B. \quad [\text{cut}]$$

In CG $\Rightarrow$ is the smallest relation containing the logical axioms [id], the application rules [BA] and [FA] as non-logical axioms, and it is closed under [cut].

To obtain a grammar G, we add a lexicon to the deductive part. Let Σ be the terminal alphabet, *i.e.* the set of basic natural language expressions. The lexicon LEX assigns a finite number of types to the elements of Σ, *i.e.* $\mathsf{LEX} \subseteq \Sigma \times \mathsf{CAT}$. We say that G generates a string $w_1 \dots w_n \in \Sigma^+$ as an expression of category B if and only if there are categories $A_1, \dots, A_n$ such that $(w_i, A_i) \in \mathsf{LEX}$ and $A_1, \dots, A_n \Rightarrow B$. $L(G)$, the language of G, is the set of strings generated by G for some designated category, the start symbol of G.

It was shown in [BGS60] that CG has the weak generative capacity of Context Free Grammar (CFG). But conceptually, CG already improves on CFG. The structured category format allows one to replace a stipulated set of rewrite rules by two simple combinatory schemata. In phrase structure grammar, this categorial idea later resurfaced in the form of the X-Bar Theory [Jac77].

In order to get a feeling for the kind of phenomena that can be handled by CG, and for the limitations of this framework, we introduce an extremely elementary fragment of English in Example 4.3. We will use the phrases given there as a checklist throughout this chapter, and come back to them later to see how the descendants of CG improve on the original framework.

EXAMPLE 4.3. [English Toy Fragment] The fragment contains simple declarative sentences, with intransitive or transitive verbs; proper names and full noun phrases introduced by determiners; nominal and adverbial modifiers; relative clauses with subject and object relativization.

(1) a. Lori left.
b. Lori knows Sara.
c. Sara wears the new dress.

(2) a. The student left.
b. Some student left.

(3) a. No student left yet.
b. Some student left already.

(4) a. who knows Lori.
b. which Sara wrote.
c. which Sara wrote there.

(5) a. Every student knows one book.
b. Every student knows some book.
c. No student knows any book.

Let us see whether we can come up with a CG that generates the phrases of our toy fragment.

EXAMPLE 4.4. [CG Grammar for the Toy Fragment] Let ATOM be $\{n, s, np\}$ (for common nouns, sentences and names, respectively) and LEX as given below:

Lori, Sara	np	the	np/n
student, book, dress	n	left	$np\backslash s$
knows, wrote, wears	$(np\backslash s)/np$	some, every, one, any, no	$(s/(np\backslash s))/n$
which, who	$(n\backslash n)/(np\backslash s)$	there, yet, already	$(np\backslash s)\backslash(np\backslash s)$
new, tall	n/n		

Given the lexicon above, our sample grammar recognizes the strings in (1), (2) and (3) as expressions of category s; the relative clause in (4-a) is recognized as an expression of type $n\backslash n$. By way of illustration, we give the derivations of (1-c) and (4-a). We use the familiar *parse tree* format, with vocabulary items as leaves and the types assigned to them in the lexicon as preterminals.

Sara wears the new dress $\in s$? $\rightsquigarrow$ $np, (np\backslash s)/np, np/n, n/n, n \Rightarrow s$?

$$\dfrac{\dfrac{\underline{\text{Sara}}}{np} \quad \dfrac{\underline{\text{wears}}}{(np\backslash s)/np} \quad \dfrac{\underline{\text{the}}}{np/n} \quad \dfrac{\dfrac{\underline{\text{new}}}{n/n} \quad \dfrac{\underline{\text{dress}}}{n}\,[\texttt{FA}]}{n}\,[\texttt{FA}]}{np}\,[\texttt{FA}]}{np\backslash s}\,[\texttt{BA}]}{s}$$

who knows Lori $\in n\backslash n?$ $\quad\rightsquigarrow\quad$ $(n\backslash n)/(np\backslash s), (np\backslash s)/np, np \Rightarrow n\backslash n?$

$$\dfrac{\dfrac{\underline{\text{who}}}{(n\backslash n)/(np\backslash s)} \quad \dfrac{\dfrac{\underline{\text{knows}}}{(np\backslash s)/np} \quad \dfrac{\underline{\text{Lori}}}{np}\,[\texttt{FA}]}{np\backslash s}\,[\texttt{FA}]}{n\backslash n}$$

Turning to the remaining examples, our CG runs into problems. Let us look at the relative clauses first. The case of subject relativization (4-a) is derivable from the assignment $(n\backslash n)/(np\backslash s)$ to the relative pronoun, but this type will not do for object relativization (4-b), or for (4-c) where the relativized position is a non-peripheral constituent of the relative clause body. To generate these structures, our CG would have to multiply lexical assignments in an *ad hoc* way for each case. Writing tv as an abbreviation for $(np\backslash s)/np$, the assignment $((n\backslash n)/tv)/np$ to the relative pronoun would produce (4-b); for the non-peripheral case of relativization, yet another type would be needed —obviously, not a very satisfactory situation. In a similar way, multiple lexical assignments would be needed to obtain the examples in (5), with full noun phrases in direct object position: the lexicon, as it stands, only covers the subject case. Writing iv as an abbreviation for $np\backslash s$, the determiners *some, every, one, any, no* could be assigned a second type $(tv\backslash iv)/n$ for their occurrence in direct object position.

One way of dealing with this failure to express structural generalizations in lexical type assignments is to extend the inventory of combinatory rules of CG. The framework of Combinatory Categorial Grammar, developed by Mark Steedman, offers the most elaborate proposal for this strategy.

4.1.2 Combinatory Categorial Grammar

For an detailed exposition of Combinatory Categorial Grammar (CCG), we refer the reader to [Ste00, Bal02][1]. The architecture of CCG is the same as that of CG: we can take over the definitions of the category language, the derives relation, lexicon, grammar G and the language generated by $L(G)$ from the previous section, with one important change: instead of having just the forward/backward application rules as non-logical axioms, CCG introduces a larger set of rule schemata. The name CCG derives from the fact that these extra schemata are inspired by the combinators of Curry's Combinatory Logic [CF68].

[1] In order to avoid confusion with the notation and facilitate the comparison between CCG and CTL we replace the "left-result" notation used in CCG, with the "result on top" one we have being using so far.

Below we present some of the rule schemata that have been proposed in the CCG framework, and we return to our toy fragment, to see how they can help in the cases where CG failed.

Lifting	$A \Rightarrow B/(A\backslash B)$	[T]
Forward Composition	$A/B, B/C \Rightarrow A/C$	[B]
Backward Crossed Composition	$A/B, A\backslash C \Rightarrow C/B$	[$\mathtt{B}_\times$]

Example 4.5. [Wh-Dependencies] Let us look first at the cases of direct object relativization in (4-b) and (4-c). Suppose we extend the lexicon given in Example 4.4 with a second type for *which* and *who*: $(n\backslash n)/(s/np)$. Intuitively, this type says that the relative pronoun looks for a clause with an np missing at the right edge. With the combinators [T] and [B], we can compose subject and transitive verb in (4-b), and produce the required type s/np for combination with the relative pronoun as shown in the derivation below. The [T] combinator lifts the subject np type into a fractional type $s/(np\backslash s)$ which can then combine with the transitive verb by means of Forward Composition.

$$\dfrac{\dfrac{\text{which}}{(n\backslash n)/(s/np)} \quad \dfrac{\dfrac{\dfrac{\text{Sara}}{np}}{s/(np\backslash s)}\,[\mathtt{T}] \quad \dfrac{\text{wrote}}{(np\backslash s)/np}}{s/np}\,[\mathtt{B}]}{n\backslash n}\,[\mathtt{FA}]$$

The combinators [B] and [T] are not enough to parse the phrase in (4-c): *which Sara wrote there*. Here, the missing np in the relative clause body comes from a non-peripheral position, whereas our lexical entry for non-subject relativization insists on a peripheral missing np, as indicated by the argument subtype s/np for the relative pronoun. To derive the non-peripheral case of relativization, our CCG grammar has to relay on the combinator [$\mathtt{B}_\times$] as illustrated below.

$$\dfrac{\dfrac{\text{which}}{(n\backslash n)/(s/np)} \quad \dfrac{\dfrac{\text{Sara}}{s/(np\backslash s)}\,[\mathtt{T}] \quad \dfrac{\dfrac{\text{wrote}}{(np\backslash s)/np} \quad \dfrac{\text{there}}{(np\backslash s)\backslash(np\backslash s)}}{(np\backslash s)/np}\,[\mathtt{B}_\times]}{s/np}\,[\mathtt{B}]}{n\backslash n}\,[\mathtt{FA}]$$

Example 4.6. [Object generalized quantifiers] The next set of examples are the sentences with full noun phrases in direct object position. In our discussion of CG, we already noticed that the noun phrase *some book* can be assigned a type which allows it to combine with a transitive verb by means of Backward Application producing $np\backslash s$ as a result. A derivation is given in (i) below. In CCG , there is a second option for typing the direct object: $(s/np)\backslash s$. This type requires the combination of the subject and the transitive verb into a constituent of type s/np. This combination, as we have already seen in the derivation of relative clauses, can be obtained by means of the Composition combinator [B]. By way of illustration, we present the derivations of (5-b) in (i) and (ii), the reader is referred to [Ste00] for a proper analysis of quantifiers within CCG. In the discussion of meaning assembly in Section 4.3, we will come back to these two options for object generalized quantifiers.

(i)

$$\dfrac{\dfrac{\text{every\_student}}{s/(np\backslash s)} \quad \dfrac{\dfrac{\text{knows}}{(np\backslash s)/np} \quad \dfrac{\text{some\_book}}{((np\backslash s)/np)\backslash(np\backslash s)}\,[\mathtt{BA}]}{np\backslash s}\,[\mathtt{FA}]}{s}$$

(ii)

$$\dfrac{\dfrac{\dfrac{\text{every\_student}}{s/(np\backslash s)} \quad \dfrac{\text{knows}}{(np\backslash s)/np}\,[\mathtt{B}]}{s/np} \quad \dfrac{\text{some\_book}}{(s/np)\backslash s}\,[\mathtt{BA}]}{s}$$

Let us evaluate the CCG strategy. We notice first of all that a combinator like $[\mathtt{B}_\times]$, which was used in the derivation of non-peripheral cases of extraction, implicitly involves a form of commutativity. It is obvious that such a combinator, if it would be available in its full generality, would lead to problems of overgeneration. CCG avoids such problems by restricting the application of combinatory rules to certain categories. Different languages could impose their individual restrictions on the rules; also, they can make their individual choices as to which combinators they allow. As for generative capacity, it is shown in [VW90] that an appropriately restricted version of CCG is weakly equivalent to linear indexed grammars, which means CCG belongs to the class of mildly context-sensitive formalisms. Important questions that remain are: What is the set of combinatory schemata allowed by Universal Grammar? and: Could we refine schemata in such a way that side conditions on their applicability can be avoided? (See [Bal02] for an answer to the last question within the CCG framework.) These questions will be addressed in the next two sections.

4.2 A Logic of Types

At the beginning of this chapter, we commented on the resemblance between complex categories and fractions in arithmetic, and between the Application schemata and multiplication. The crucial insight of Lambek [Lam58] was that one can also see the categories as *logical formulas*. The changes introduced by this logical perspective with respect to the rule-based approach are summarized in Table 4.1. To start with, categories are seen as *formulas* and their type forming operators as connectives, i.e. *logical constants*. As a result, the rules for category combination can now be formulated as rules of inference for these connectives, rather than as the non-logical axiom schemata we had in CG and CCG. Parsing literally becomes a process of deduction in the logic of the categorial type formulas.

The logical perspective introduces another important theme: the distinction between proof theory and model theory. In the logical setup, formulas will be assigned a modeltheoretic interpretation. The syntactic side of derivations (the prooftheoretic machinery) can then be judged in terms of its soundness and completeness with respect to the proposed interpretation. (See Chapter 3 for the formal details.)

CG & CCG	L
Categories	Formulas
Type forming operators	Logical constants
Rule schemata	Inference Rules
Parsing	Deduction

Table 4.1: Rules-based approach vs. logical approach.

4.2.1 Parsing as Deduction

Let us look at the syntax of the Lambek calculus (L) first. Lambek himself presented his type logic in the format of a Gentzen-style Sequent Calculus [Gen38]. An alternative (equivalent) presentation[2], which is closer to the format we have used in the previous sections, is the Natural Deduction (N.D.) format.

Definition 4.7. [Natural Deduction Rules for L] Let Γ, Δ stand for finite non-empty sequences of formulas and A, B, C for logical formulas. The logical rules of L are:

$$\frac{}{A \vdash A}\ [\text{axiom}]$$

$$\frac{\Delta \vdash B/A \quad \Gamma \vdash A}{\Delta, \Gamma \vdash B}\ [/\text{E}] \qquad \frac{\Gamma \vdash A \quad \Delta \vdash A\backslash B}{\Gamma, \Delta \vdash B}\ [\backslash\text{E}]$$

$$\frac{\Delta, B \vdash C}{\Delta \vdash C/B}\ [/\text{I}] \qquad \frac{B, \Delta \vdash C}{\Delta \vdash B\backslash C}\ [\backslash\text{I}]$$

The rules of Forward and Backward Application in this format take the form of the familiar inference patterns of Modus Ponens, where we see the 'fractional' categories now as 'implicational' formulas. Compiling in the Cut rule of our definition of the 'derives' relation, we obtain the Elimination rules for '/' and '\'. But the elimination rules capture only one half of the inferential possibilities of these connectives: they tell us how we can *use* an implicational formula in a derivation. To obtain the other half, we need inference rules to *derive* an implicational formula. These are the Introduction rules for the '/' and '\' connectives. As rules of inference, they give our grammar logic access to *hypothetical reasoning*: to obtain a formula C/B ($B\backslash C$), we withdraw a hypothesis B as the rightmost (leftmost) assumption of the antecedent sequence of formulas.

On the modeltheoretic side, we want to interpret formulas (i.e. syntactic categories) as sets of expressions, and the 'derives' relation as settheoretic inclusion at the interpretive level. In the systems considered so far, categorial combination was intuitively interpreted as concatenation. We can make this interpretation precise by considering semigroup models. It was shown by Pentus in [Pen95] that the calculus of [Lam88] is indeed sound and complete with respect to this interpretation.

Definition 4.8. [Semigroup Interpretation]

[2]See [Res00] for a detailed comparison of the two presentations.

$$\begin{array}{lcl} A, B & = & \{xy \in M \mid x \in A \wedge y \in B\} \\ C/B & = & \{x \in M \mid \forall y(y \in B \rightarrow xy \in C)\} \\ B\backslash C & = & \{y \in M \mid \forall x(x \in B \rightarrow xy \in C)\}. \end{array}$$

A pleasant consequence of the shift to the logical perspective is that a number of combinators that have the status of non-logical axioms in CCG now turn out to be theorems of our type logic.

Example 4.9. [Hypothetical Reasoning] We show that the combinatory rules [T] and [B] of CCG considered above are theorems of L.

The combinator T of CCG. The lifting theorem, which raises a type to a higher order one[3], is a typical application of hypothetical reasoning. Its derivation is illustrated below.

$$\dfrac{\dfrac{\Delta \vdash A \quad [(A\backslash B) \vdash (A\backslash B)]^1}{\Delta, (A\backslash B) \vdash B}\,[\backslash\mathrm{E}]}{\Delta \vdash B/(A\backslash B)}\,[/\mathrm{I}]^1$$

The derivation proves that if a structure Δ is of type A, then it is of type $B/(A\backslash B)$ as well. The proof is given by hypothetical reasoning: Assume a structure of type $A\backslash B$, given $\Delta \vdash A$, then Δ composed with $A\backslash B$ is of type B. Then by withdrawing the hypothesis by means of the coindexed rule, Δ is proved to be of the higher order type. Note that the introduction rule can discharge one hypothesis at a time since we are in a resource sensitive system.

The combinator B of CCG. The forward composition added in CCG to the function application of CG is derivable in L as shown below:

$$\dfrac{\dfrac{\Delta \vdash A/B \qquad \dfrac{\Gamma \vdash B/C \quad [C \vdash C]^1}{\Gamma, C \vdash B}\,[/\mathrm{E}]}{\Delta, \Gamma, C \vdash A}\,[/\mathrm{E}]}{\Delta, \Gamma \vdash A/C}\,[/\mathrm{I}]^1$$

Similarly to the previous derivation, the combinator is inferred by means of the logical rules of L. In particular, the derivation is based on the hypothetical reasoning: it starts by assuming a hypothesis C and it withdraws it once the functions are composed.

Let us turn to the examples of our toy fragment, and present some Lambek derivations in the sequent-style Natural Deduction format introduced above. The leaves of the N.D. derivations are axioms $A \vdash A$. Some of these leaves correspond to lexical assumptions, others to hypothetical assumptions that will have to be withdrawn in the course of the derivation. To make the derivations more readable, we replace the formula on the left of $\vdash$ by the lexical item in the case of lexical assumptions.

[3]The order of the categories is defined as following: $\mathtt{order}(A) = 0$, if $A \in \mathsf{ATOM}, \mathtt{order}(A/B) = \mathtt{max}(\mathtt{order}(A), \mathtt{order}(B) + 1)$ and the same holds for $(B\backslash A)$.

EXAMPLE 4.10. [Function Application in L] Given the lexicon of our toy grammar, the expression in (4-a), *who knows Lori*, is shown to be an expression of type $n\backslash n$ as follows.

$$\dfrac{\text{who} \vdash (n\backslash n)/(np\backslash s) \qquad \dfrac{\text{knows} \vdash (np\backslash s)/np \quad \text{Lori} \vdash np}{\text{knows, Lori} \vdash np\backslash s}\,[/\text{E}]}{\text{who, knows, Lori} \vdash n\backslash n}\,[/\text{E}]$$

As we discussed above, hypothetical reasoning is applied in the derivation of the combinator [B] which is required to account for right-peripheral extraction. We show how the structure *which Sara wrote* is proved to be grammatical in L.

EXAMPLE 4.11. [Right-Peripheral Extraction in L] The string *which Sara wrote* is derived as an expression of type $n\backslash n$, by starting from the lexical entries it consists of and by assuming a hypothetical np taken as object by the transitive verb.

$$\dfrac{\text{which} \vdash (n\backslash n)/(s/np) \qquad \dfrac{\text{Sara} \vdash np \qquad \dfrac{\text{wrote} \vdash (np\backslash s)/np \quad [np \vdash np]^1}{\text{wrote}, np \vdash np\backslash s}\,[/\text{E}]}{\text{Sara, wrote}, np \vdash s}\,[\backslash\text{E}]}{\text{Sara, wrote} \vdash s/np}\,[/\text{I}]^1}{\text{which, Sara, wrote} \vdash n\backslash n}\,[/\text{E}]$$

First, the string 'Sara, wrote, np' is proved to be of category s. Then, the hypothesis np is withdrawn. This is done by means of [/I] which produces the formula s/np required by the type assigned to the relative pronoun.

The type logic L does not succeed in producing a derivation for the case of non-peripheral extraction *which Sara wrote there*. As we saw in our discussion of Backward Crossed Composition [$\text{B}_\times$], this combinator involves a form of commutativity. This combinator, in other words, is not a valid theorem of L —it would violate the concatenation interpretation. Summing up, by making the shift to a type *logic*, we have gained a better understanding of the CCG combinators, seeing which ones are indeed valid given the interpretation of the type-forming connectives and which ones are not. But as a linguistic framework, L is not expressive enough to deal with the phenomena illustrated by our toy fragment. The proof by Pentus [Pen93] that L grammars are context free provides the formal underpinnings for this claim.

4.2.2 Logical Rules and Structural Rules

The presentation of the antecedent part Γ in a sequent $\Gamma \vdash A$ as a sequence of formulas hides an implicit structural assumption about grammatical composition, *viz.* that it is an associative operation, which ignores the hierarchical constituent structure of type formulas. Lambek in his [Lam61] paper was the first to notice that this assumption is too strong, and that it leads to overgeneration. The formulation of his [Lam61] system removes the implicit structural assumption, which means that structural rules have to be introduced in a fully explicit fashion. The type logics so obtained have a combination

of logical rules for the connectives (Introduction and Elimination rules), plus structural rules of inference for the manipulation of antecedent configurations. Structures are built from the set of formulas FORM by means of the *binary* structural operator $\circ$ as follows.

i. If $A \in \mathsf{FORM}$, then $A \in \mathsf{STRUCT}$;

ii. If Γ and $\Delta \in \mathsf{STRUCT}$, then $(\Gamma \circ \Delta) \in \mathsf{STRUCT}$.

The separation of logical and structural rules makes it possible to generate a family of logics with the same logical rules, but different structural rules. We refer to this family as Categorial Type Logics (CTLs). The base logic for this family is the system presented in [Lam61]: the type logic with absolutely no structural rules. It is usually abbreviated as NL($\Diamond$), because it is obtained from L by dropping associativity.

Definition 4.12. [The Lambek Family]. **Logical rules** for the base logic NL($\Diamond$):

$$\frac{}{A \vdash A}\ [\text{axiom}]$$

$$\frac{\Delta \vdash B/A \quad \Gamma \vdash A}{(\Delta \circ \Gamma) \vdash B}\ [/\text{E}] \qquad \frac{\Gamma \vdash A \quad \Delta \vdash A\backslash B}{(\Gamma \circ \Delta) \vdash B}\ [\backslash\text{E}]$$

$$\frac{(\Delta \circ B) \vdash C}{\Delta \vdash C/B}\ [/\text{I}] \qquad \frac{(B \circ \Delta) \vdash C}{\Delta \vdash B\backslash C}\ [\backslash\text{I}]$$

As we have shown in Chapter 3, the algebraic structure interpreting these operators has space for n-ary operators, and in particular for the unary ones studied in [Moo96b, Moo97]. The non associative Lambek calculus extended with these operators is usually referred to as NL($\Diamond$) and is obtained by adding to NL the logical rules below.

$$\frac{\Gamma \vdash t\Box^{\downarrow}A}{\langle\Gamma\rangle \vdash A}\ [\Box^{\downarrow}\text{E}] \qquad \frac{\langle\Gamma\rangle \vdash A}{\Gamma \vdash \Box^{\downarrow}A}\ [\Box^{\downarrow}\text{I}]$$

$$\frac{\Delta \vdash \Diamond A \quad \Gamma[\langle A\rangle] \vdash B}{\Gamma[\Delta] \vdash B}\ [\Diamond\text{E}] \qquad \frac{\Gamma \vdash A}{\langle\Gamma\rangle \vdash \Diamond A}\ [\Diamond\text{I}]$$

The relation between the unary structural operator, $\langle . \rangle$, and the unary logical operator, $\Diamond$, is the same one holding between the binary structural and logical operator ($\circ$ and $\bullet$, respectively). This can be clearly seen by observing rule [$\Diamond$I] where the introduction of the $\Diamond$ in the right side (i.e. the logical side) of the sequent corresponds to the introduction of $\langle . \rangle$ in the left side (i.e. the structural side). This correspondence explains also the [$\Diamond$E] rule, which performs a cut (within a structure Γ of the formula A headed by the diamond operator (where the latter is a logical when occurs in the right side of the sequent and a structural diamond when it's on its left side). In the remaining part of this chapter we will try to give more linguistic intuition on the role of the unary structural operator and of its interplay with its residual (the $\Box^{\downarrow}$). As for now, notice that relation between them is established by means of the [$\Box^{\downarrow}$E] and [$\Box^{\downarrow}$I].

In CTL, unary operators have been used mainly in two ways: to control structural reasoning and to encode partial orders among elements of the same domain of interpretation. In this Chapter we will focus attention on the first use by briefly reviewing the literature, whereas in Chapter 5 we will describe the partial ordering strategy.

Structural rules. Let us write $\Gamma[\Delta]$ for a structure Γ contaning a distinguished occurrence of the substructure Δ. Adding a structural rule of Associativity [ass] to NL , one obtains L . By adding commutativity [per] to L one obtains LP [Ben88]. The picture is completed with the non associative and commutative Lambek calculus NLP.

$$\frac{\Gamma[\Delta_1 \circ (\Delta_2 \circ \Delta_3)] \vdash C}{\Gamma[(\Delta_1 \circ \Delta_2) \circ \Delta_3] \vdash C}\ [\text{ass}] \quad \frac{\Gamma[(\Delta_2 \circ \Delta_1)] \vdash C}{\Gamma[(\Delta_1 \circ \Delta_2)] \vdash C}\ [\text{per}]$$

$$\frac{\Gamma[\langle \Delta_1 \circ \Delta_2 \rangle] \vdash C}{\Gamma[\langle \Delta_1 \rangle \circ \langle \Delta_2 \rangle] \vdash C}\ dis_\diamond$$

Multimodal systems. The structural rules above apply in a *global* fashion. While discussing the linguistic application of L and of CCG, we have noted that we need *control* over structural options. In the so-called multimodal version of CTL (introduced in Chapter 3), the required control is achieved by distinguishing different *modes* of composition, which can then live together and interact within one grammatical logic. In the notation, we keep the different modes apart by indexing the logical and the structural connectives, *i.e.* we now write $(\backslash_i, \bullet_i, /_i)$ and $(\diamond_i, \Box_i^\downarrow)$ and $\circ_i$ and $\rangle \dot{\langle}_i$, where $i \in I$ and I is a set of mode indices. The different modes have the same logical rules, but they can differ in their structural properties. Thus, one can introduce structural rules *locally* by restricting them to a certain family of logical constants. Finally, the addition of modes increases the number of logics which can be obtained from the base logic. Besides associativity and/or commutativity options for individual composition modes, one can formulate inclusion and interaction rules for configurations involving multiple modes.

i. *Inclusion* structural rules (also known as entropy principles), e.g. if $\Gamma[\Delta \circ_1 \Delta'] \vdash A$ then $\Gamma[\Delta \circ_2 \Delta'] \vdash A$; if $\Gamma[\langle \Delta \rangle_1] \vdash A$ then $\Gamma[\langle \Delta \rangle_2] \vdash A$

ii. *Interaction* structural rules which mix distinct modes and operators.

For an illustration of interaction principles, we can return to the non-peripheral extraction example in our toy fragment. Suppose we have the structural rules below for the interaction between two modes, $\circ$ and $\circ_a$.

$$\frac{\Gamma[\Delta_1 \circ (\Delta_2 \circ_a \Delta_3)] \vdash C}{\Gamma[(\Delta_1 \circ \Delta_2) \circ_a \Delta_3] \vdash C}\ [\text{mixass}] \quad \frac{\Gamma[(\Delta_1 \circ_a \Delta_2) \circ \Delta_3] \vdash C}{\Gamma[(\Delta_1 \circ \Delta_3) \circ_a \Delta_2] \vdash C}\ [\text{diss}]$$

Example 4.13. [Non-Peripheral Extraction] We modify the lexicon in such a way that $\circ$ is used for regular phrasal composition, and $\circ_a$ for extraction. We need a type assignment to introduce a *wh* dependency, and a type assignment to eliminate it. In this example, these are $(n\backslash n)/(s/_a np)$ for the relative pronoun, and $(np\backslash s)/_a np$ for the transitive verb, The derivation of *which Sara wrote there* is then as follows.

$$\dfrac{\text{which} \vdash (n\backslash n)/(s/_a np) \qquad \dfrac{\text{Sara} \vdash np \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\text{wrote} \vdash (np\backslash s)/_a np \quad [np \vdash np]^1}{\text{wrote} \circ_a np \vdash np\backslash s}\,[/_a\text{E}] \qquad \text{there} \vdash (np\backslash s)\backslash(np\backslash s)}{((\text{wrote} \circ_a np) \circ \text{there}) \vdash np\backslash s}\,[\backslash\text{E}]}{\text{Sara} \circ ((\text{wrote} \circ_a np) \circ \text{there}) \vdash s}\,[\text{diss}]}{\text{Sara} \circ ((\text{wrote} \circ \text{there}) \circ_a np) \vdash s}\,[\text{mixass}]}{(\text{Sara} \circ (\text{wrote} \circ \text{there})) \circ_a np \vdash s}\,[/_a\text{I}]^1}{(\text{Sara} \circ (\text{wrote} \circ \text{there})) \vdash s/_a np}\,[\backslash\text{E}]}{\text{which} \circ (\text{Sara} \circ (\text{wrote} \circ \text{there})) \vdash n\backslash n}\,[/\text{E}]$$

Note that the application of the structural rules is *lexically anchored*. The modes labelling the connectives of the types assigned to the transitive verb *wrote* and the relative pronoun *which* drive the structural reasoning in the derivation. The structural rule [diss] brings the np in the peripheral position and [mixass] makes it available to the abstraction. The application of these rules is restricted to the environments requiring them.

We have seen that in CCG the above expression is parsed by applying the combinator $[\mathtt{B}_\times]$. The latter is derivable in NL extended with the structural rules above. However, the use of modes to account for long distance phenomena is still not completely satisfactory since the application of the structural rules is tied to the lexical entries both of the relative pronoun and the transitive verb, which now gets a special lexical entry that allows its direct object to be extracted: $(np\backslash s)/_a np$ in contrast with the linguistic facts.

Suppose now, we have the interaction structural rule below, in [Moo99], it is shown that they perform the required task, together with a lexical type assignment $(n\backslash n)/(s/\Diamond\Box^{\downarrow}np)$.

$$\dfrac{\Gamma[\Delta_1 \circ (\Delta_2 \circ \langle\Delta_3\rangle)] \vdash C}{\Gamma[(\Delta_1 \circ \Delta_2) \circ \langle\Delta_3\rangle] \vdash C}\,[\text{ass}_\Diamond] \qquad \dfrac{\Gamma[(\Delta_1 \circ \langle\Delta_3\rangle) \circ \Delta_2] \vdash C}{\Gamma[(\Delta_1 \circ \Delta_2) \circ \langle\Delta_3\rangle] \vdash C}\,[\text{diss}_\Diamond] \tag{4.1}$$

Note, that these rules are available only for marked formulas, where the latter are introduced only due to information stored in the lexical entries. Let us first look at the derivation of (4-b) where the extraction is performed from a peripheral position.

Example 4.14. [Right-Peripheral Extraction in $\mathsf{NL}(\Diamond)$ plus $[\text{ass}_\Diamond]$]

$$\dfrac{\text{which} \vdash (n\backslash n)/(s/\Diamond\Box^{\downarrow}np) \qquad \dfrac{\dfrac{\text{Sara} \vdash np \qquad \dfrac{\dfrac{\dfrac{\dfrac{\text{wrote} \vdash (np\backslash s)/np \qquad \dfrac{[\Box^{\downarrow}np \vdash \Box^{\downarrow}np]^1}{\langle\Box^{\downarrow}np\rangle \vdash np}\,[\Box^{\downarrow}\text{E}]}{(\text{wrote} \circ \langle\Box^{\downarrow}np\rangle) \vdash np\backslash s}\,[/\text{E}]}{\text{Sara} \circ (\text{wrote} \circ \langle\Box^{\downarrow}np\rangle) \vdash s}\,[\backslash\text{E}]}{(\text{Sara} \circ \text{wrote}) \circ \langle\Box^{\downarrow}np\rangle \vdash s}\,[\text{ass}_\Diamond]}{(\text{Sara} \circ \text{wrote}) \circ \Diamond\Box^{\downarrow}np \vdash s}\,[\Diamond\text{E}]^1}{(\text{Sara} \circ \text{wrote}) \vdash s/\Diamond\Box^{\downarrow}np}\,[/\text{I}]^2}{\text{which} \circ (\text{Sara} \circ \text{wrote}) \vdash n\backslash n}\,[/\text{E}]$$

Note how the re-bracketing is controlled by the pronoun type assignment which requires a sentence missing a noun phrase occurring in a special position $\Diamond\Box^{\downarrow}np$. This forces the assumption of a marked noun phrase $\Box^{\downarrow}np$. This marker is then passed from the logical to the structural language by means of $[\Box^{\downarrow}\mathrm{E}]$, to allow the required re-bracketing $[\mathrm{ass}_{\Diamond}]$. However, abstraction can take place only from atomic structural formulas (*i.e.* logical formulas). Therefore, once $\langle\Box^{\downarrow}np\rangle$ has fulfilled its task on the structural level, it is replaced by the corresponding logical formula by means of $[\Diamond\mathrm{E}]^1$. This rule substitutes the co-indexed hypothesis with a second one, which is finally discharged building the type suitable for the pronoun.

The interaction structural rule $[\mathrm{ass}_{\Diamond}]$ required by the derivation of right-branch extraction in peripheral position in itself is not enough to express the proper structural generalization of wh-dependencies in English. In particular, this structural rule does not help deriving (4-c). To account for right-branch extraction from a non-peripheral position one needs the interaction structural rule $[\mathrm{diss}_{\Diamond}]$ as well.

Example 4.15. [Non-Peripheral Extraction in $\mathsf{NL}(\Diamond)$ plus $[\mathrm{ass}_{\Diamond}]$ and $[\mathrm{diss}_{\Diamond}]$]

$$\dfrac{\mathrm{which} \vdash wh/(s/\Diamond\Box^{\downarrow}np) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[\Box^{\downarrow}np \vdash \Box^{\downarrow}np]^1}{\vdots}}{\mathrm{Sara} \circ ((\mathrm{wrote} \circ \langle\Box^{\downarrow}np\rangle) \circ \mathrm{there}) \vdash s}\,[\mathrm{diss}_{\Diamond}]}{\mathrm{Sara} \circ ((\mathrm{wrote} \circ \mathrm{there}) \circ \langle\Box^{\downarrow}np\rangle) \vdash s}\,[\mathrm{ass}_{\Diamond}] \qquad [\Diamond\Box^{\downarrow}np \vdash \Diamond\Box^{\downarrow}np]^2}{(\mathrm{Sara} \circ (\mathrm{wrote} \circ \mathrm{there})) \circ \langle\Box^{\downarrow}np\rangle \vdash s}\,[\Diamond\mathrm{E}]^1}{(\mathrm{Sara} \circ (\mathrm{wrote} \circ \mathrm{there})) \circ \Diamond\Box^{\downarrow}np \vdash s}\,[/\mathrm{I}]^2}{\mathrm{Sara} \circ (\mathrm{wrote} \circ \mathrm{there}) \vdash s/\Diamond\Box^{\downarrow}np}\,[/\mathrm{E}]}{\mathrm{which} \circ (\mathrm{Sara} \circ (\mathrm{wrote} \circ \mathrm{there})) \vdash n\backslash n}$$

This derivation can be read in a similar way than the previous one. The only difference is the application of $[\mathrm{diss}_{\Diamond}]$ which brings the hypothesis in a peripheral position.

These examples are meant only as an illustration of the method used in CTL to account for long distance phenomena. A detailed discussion can be found in [Moo99], where unary operators and structural reasoning are also exploited to deal with crosslinguistic variations. In particular, the lexical type assignments and structural packages required to model English relative clauses are compared with the ones required by subject-object-verb language like Dutch. In a few words, the structural variation between the two languages with respect to relative clauses is captured by combining a universal base logic with different structural packages.[4]

The categorial account of structural control is surprisingly close to 'feature-driven' structural reasoning in generative grammar, especially within the minimalism program

[4]See [Bal02] for a detailed analysis of long distance dependencies in CCG extended with modes. The reader is invited to compare the two approaches for an understanding of their differences and similarities. The multi-modal CCG is the topic of the course given at ESSLLI'04 by Geert-Jan Kruijff, further references to this work will be provided there.

[Cho95] formalized in [Sta97]. For some discussion of this correspondence and a study of the connection between Stabler's Minimalist Grammar and CTL see [Ber02] and [Ver99], respectively.
The example above illustrates how modes and structural rules can be used to account for differences among contexts *within* the same languages. Similarly, these logical tools are used to account for differences holding *across* languages. By way of illustration, we look at Italian and English adjectives.

EXAMPLE 4.16. [Italian vs. English Adjectives] English and Italian adjectives may differ in their ordering possibilities with respect to a noun.

(6) a. Sara wears a new dress.
b. *Sara wears a dress new.

(7) a. Sara indossa un nuovo vestito.
Sara wears a new dress
tr. Sara wears a new dress.
b. Sara indossa un vestito nuovo.
Sara wears a dress new
tr. Sara wears a new dress.

As the examples show, some adjectives in Italian require more freedom with respect to word order than their English counterparts. This crosslinguistic difference can be expressed by assigning different logical types to Italian and English adjectives. Since the exhibited structural property is not shared by all Italian phrases, the structural freedom of the adjectives must have been lexically anchored. This restriction can be expressed by means of modes. Let us try to make things more concrete by looking at the derivation of the relevant structures in (6) and (7). Let qp abbreviate the type of quantifier phrases.

(i)
$$\dfrac{\mathrm{a} \vdash qp/n \qquad \dfrac{\mathrm{new} \vdash n/n \quad \mathrm{dress} \vdash n}{\mathrm{new} \circ \mathrm{dress} \vdash n}\,[/\mathrm{E}]}{\dfrac{\mathrm{a} \circ (\mathrm{new} \circ \mathrm{dress}) \vdash qp}{\mathrm{a} \circ (\mathrm{dress} \circ \mathrm{new}) \vdash qp}\,[\mathrm{per}_\bullet]^*}\,[/\mathrm{E}]$$

(ii)
$$\dfrac{\mathrm{un} \vdash qp/n \qquad \dfrac{\mathrm{nuovo} \vdash n/_c n \quad \mathrm{vestito} \vdash n}{\mathrm{nuovo} \circ_c \mathrm{vestito} \vdash n}\,[/_c\mathrm{E}]}{\dfrac{\mathrm{un} \circ (\mathrm{nuovo} \circ_c \mathrm{vestito}) \vdash qp}{\mathrm{un} \circ (\mathrm{vestito} \circ_c \mathrm{nuovo}) \vdash qp}\,[\mathrm{per}_\bullet]}\,[/\mathrm{E}]$$

The $*$ on the last step of the derivation in (i) marks where the derivation fails in accounting for (6). On the other hand, the use of a commutative composition operator, introduced by the lexical assignment of *nuovo*, allows the permutation required to build the structures in (7).

Finally, unary operators have also been employed to encode morphological information. The type logical analysis of morphological agreement worked out by Heylen [Hey99] provides a categorial alternative for the unification and subsumption based approach of framework like Head Driven Phrase Structure Grammar, and Lexicalized Functional Grammar. In order to recast their mechanisms in logical terms, underspecification and

overspecification are expressed via inclusion postulates and the law of residuation is exploited to account for the subsumption relation among expressions of the same syntactic category. Moreover, interaction postulates involving unary and binary operators govern the way information is distributed through phrase structure. In our exposition of the linguistic applications of the unary operators, Heylen's proposal is of particular interest for the use of the inclusion postulates and more generally for the application of inclusion relations among types.

The following postulates exemplify the encoding of underspecification. We indicate the modes as indexes, where 'pl' and 'sg' stand for 'plural' and 'singular', respectively and 'num' identifies underspecification.

Inclusion Postulates

$$[\text{PL}]\ \Diamond_{\text{num}} A \longrightarrow \Diamond_{\text{pl}} A \quad \text{and} \quad [\text{SG}]\ \Diamond_{\text{num}} A \longrightarrow \Diamond_{\text{sg}} A$$

These postulates can be read as saying that a phrase of syntactic category A underspecified for its number, $\Diamond_{\text{num}} A$, could be either plural $\Diamond_{\text{pl}} A$, or singular, $\Diamond_{\text{sg}} A$. The alternative presentation with structural rules is given below.

$$\frac{\Gamma[\langle\Delta\rangle^{\text{pl}}] \vdash C}{\Gamma[\langle\Delta\rangle^{\text{num}}] \vdash C}\ [\text{pl}] \qquad \frac{\Gamma[\langle\Delta\rangle^{\text{sg}}] \vdash C}{\Gamma[\langle\Delta\rangle^{\text{num}}] \vdash C}\ [\text{sg}]$$

Morphological agreement is required, for instance, for the combination of the definite article with its noun in Italian. Differently from English, Italian uses definitive articles sensitive to the number of the noun they combine with, e.g. *i pomodori* (tr. the tomatoes) is correct, whereas *i pomodoro* (tr. the tomato) is not. On the other hand, transitive verbs are underspecified regarding the number of their object, e.g. both *il gatto mangia i pomodori* (tr. the cat eats the tomatoes) and *il gatto mangia il pomodoro* (tr. the cat eats the tomato) are correct Italian sentences. When encoding morphological information into lexical assignments, the relation holding among expressions of the same syntactic category but with different morphological properties, must be taken into account. For this specific case, it must be stated that the expression taken as argument by the transitive verb can be either plural or singular. In [Hey99], this information would be expressed by the unary operators labelling the lexical type assignments as shown by the example below.

$$\begin{array}{llll} \textit{il} & \in \Box^{\downarrow}_{\text{sg}} np/\Box^{\downarrow}_{\text{sg}} n & \textit{pomodori} & \in \Box^{\downarrow}_{\text{pl}} n \\ \textit{i} & \in \Box^{\downarrow}_{\text{pl}} np/\Box^{\downarrow}_{\text{pl}} n & \textit{mangia} & \in (\Box^{\downarrow}_{\text{sg}} np\backslash s)/\Box^{\downarrow}_{\text{num}} np \end{array}$$

The type assignment of the article *i* (resp. *il*) specifies that it combines with a plural (resp. singular) noun, to give a plural (resp. singular) noun phrase, whereas the verb *mangia* is sensitive to the number of its subject, but is underspecified for the number of the noun phrase taken as object.

The assembly of the plural article *i* with the singular noun *pomodoro* is blocked simply by the mismatch of their types. On the other hand, the possibility of the verb *mangia* to combine both with the plural and singular noun phrases *i pomodori* and *il*

pomodoro, is carried out by means of the inclusion relations [pl] and [sg]. In a top-down reading the derivation below can be read as saying that a structure which is specified for its number can also be underspecified if required by the constituent it composes with.

$$\dfrac{\text{mangia} \vdash (\square^{\downarrow}_{\text{sg}} np \backslash s)/\square^{\downarrow}_{\text{num}} np \quad \dfrac{\dfrac{\dfrac{\dfrac{\text{i} \circ \text{pomodori} \vdash \square^{\downarrow}_{\text{pl}} np}{\langle \text{i} \circ \text{pomodori} \rangle^{\text{pl}} \vdash np}\,[\square^{\downarrow}_{\text{pl}}\text{E}]}{\langle \text{i} \circ \text{pomodori} \rangle^{\text{num}} \vdash np}\,[\text{pl}]}{\text{i} \circ \text{pomodori} \vdash \square^{\downarrow}_{\text{num}} np}\,[\square^{\downarrow}_{\text{num}}\text{I}]}{\text{mangia} \circ (\text{i} \circ \text{pomodori}) \vdash \square^{\downarrow}_{\text{sg}} np \backslash s}\,[/\text{E}]$$

Note how once again the residuation law makes possible a division of labor between the logical and structural languages; the former takes care of feature checking, whereas the subsumption relation is checked by the latter. Moreover, notice how in the enforcing of the agreement relation a crucial role is played by the monotonicity of the binary operators. This can be better understood by abstracting away from the details of the derivation and looking at the general schema below. Let $C \longrightarrow B$, then in natural deduction there is a derivation from $\Gamma \vdash C$ to $\Gamma \vdash B$, therefore

$$\dfrac{\Delta \vdash A/B \quad \begin{array}{c}\Gamma \vdash C \\ \vdots \\ \Gamma \vdash B\end{array}}{\Delta \circ \Gamma \vdash A}\,[/\text{E}]$$

Put differently, one could say that since / is downward monotone in its second argument position, a structure of type A/B will combine with any structure of a type C smaller than or equal to B.

4.3 The Composition of Meaning

Linguistic signs have a form and a meaning component. The discussion so far has concentrated on the form aspect of grammatical composition. Let us turn now to meaning assembly and the relation between natural language form and meaning. See [Gam91] for an introduction to the field of formal semantics. Montague's Universal Grammar program [Tho74] provides a general framework to study these issues. The core of this program is an algebraic formulation of Frege's principle of compositionality [Fre84]. Intuitively, the principle says that the meaning of a complex syntactic expression is a function of the meaning of its constituent parts and of the derivational steps that have put them together. Montague formalizes the principle as a mapping between a syntactic and a semantic algebra. The mapping is a *homomorphism*, *i.e.* it preserves structure in the following sense [Jan97].

Definition 4.17. [Homomorphism] Let $\mathcal{A} = (A, F)$ and $\mathcal{B} = (B, G)$ be algebras. A mapping $m : \mathcal{A} \to \mathcal{B}$ is called a *homomorphism* if there is a mapping $m' : F \to G$ s.t. for all $f \in F$ and all $a_1, \ldots, a_m \in A$ holds $m(f(a_1, \ldots, a_n)) = m'(f)(m(a_1), \ldots, m(a_n))$.

4.3.1 Semantic Types and Typed Lambda Terms

The definition above requires the syntactic algebra and the semantic algebra of a grammar to work in tandem. Syntactic combinatorics is determined by the syntactic categories, similarly the semantic laws of composition are governed by semantic types. To set up the form-meaning correspondence, it is useful to build a language of semantic types in parallel to the syntactic type language.

Definition 4.18. [Types] Given a non-empty set of *basic types* Base, the set of types TYPE is the smallest set such that

i. $\mathsf{Base} \subseteq \mathsf{TYPE}$;
ii. $(a, b) \in \mathsf{TYPE}$, if a and $b \in \mathsf{TYPE}$.

Note that this definition closely resembles the one of the syntactic categories of CG. The only difference is the lack of directionality of the functional type (a, b). A function mapping the syntactic categories into TYPE can be given as follows.

Definition 4.19. [Categories and Types] Let us define a function $\mathtt{type} : \mathsf{CAT} \to \mathsf{TYPE}$ which maps syntactic categories to semantic types.

$$\begin{array}{ll} \mathtt{type}(np) = e; & \mathtt{type}(A/B) = (\mathtt{type}(B), \mathtt{type}(A)); \\ \mathtt{type}(s) = t; & \mathtt{type}(B\backslash A) = (\mathtt{type}(B), \mathtt{type}(A)); \\ \mathtt{type}(n) = (e, t). & \end{array}$$

To represent meaning assembly, we use the tools of the typed λ-calculus. Terms are built out of variables and constants of the various types.

Definition 4.20. [Typed λ-terms] Let VAR_a be a countably infinite set of *variables* of type a and CON_a a collection of *constants* of type a. The set TERM_a of λ-terms of type a is defined by mutual recursion as the smallest set such that the following holds:

i. $\mathsf{VAR}_a \subseteq \mathsf{TERM}_a$,
ii. $\mathsf{CON}_a \subseteq \mathsf{TERM}_a$,
iii. $(\alpha(\beta)) \in \mathsf{TERM}_a$ if $\alpha \in \mathsf{TERM}_{(a,b)}$ and $\beta \in \mathsf{TERM}_b$,
iv. $\lambda x.\alpha \in \mathsf{TERM}_{(a,b)}$, if $x \in \mathsf{VAR}_a$ and $\alpha \in \mathsf{TERM}_b$.

We represent with α_a a term α of type a.

The relevant items are *iii.* and *iv.* The former defines function application, the latter abstraction over variables. The λ is an operator which binds variables following specific constraints for which it is important to distinguish free and bound variables.

Definition 4.21. [Free and Bound Variables] The set $\mathrm{Free}(\alpha)$ of *free variables* of the λ-term α is defined by

i. $\mathrm{Free}(x_b) = \{x_b\}$ if $x_b \in \mathsf{VAR}_b$,
ii. $\mathrm{Free}(c_b) = \{\}$ if $c_b \in \mathsf{CON}_b$,
iii. $\mathrm{Free}(\alpha_{(a,b)}(\beta_a)) = \mathrm{Free}(\alpha_{(a,b)}) \cup \mathrm{Free}(\beta_a)$,
iv. $\mathrm{Free}(\lambda x_a.\alpha_b) = \mathrm{Free}(\alpha_b) - \{x_a\}$.

A variable v' is *free for* v *in* the expression β iff no free occurrence of v in β is within the scope of $\lambda v'$.

Reduction rules determine the equivalence among λ-terms.

DEFINITION 4.22. [Reduction Rules] The λ-calculus is characterized by the following *reduction rules*, where $\alpha_b([\beta_a/x_a]))$ stands for the result of substituting a term β_a for x_a in α_b.

$(\lambda x_a.\alpha_b)(\beta_a) \Rightarrow \alpha_b[\beta_a/x_a]$	x_a is free for β_a in α_b	β-reduction
$\lambda x_a.\alpha_{(a,b)}(x_a) \Rightarrow \alpha_{(a,b)}$	x_a is not free in $\alpha_{(a,b)}$	η-reduction

These rules reduce a term into a simpler one. Applying this re-writing system we can determine whether two terms are logically equivalent, *viz.* whether they reduce to a common result. An important theorem concerning λ-calculus is that reduction eventually terminates with a term that can no longer be reduced using the above reduction rules. Such a term is said to be in β, η *normal form.*

The main novelty introduced by Montague is that the interpretation of the type-theoretical logical system may also serve as the interpretation of natural language expressions. To this end, he adopted a model theoretic semantics. When applied to natural language, model theory can be thought of as a theory designed to explain entailment relations among sentences and consequently to account for truth conditions of meanings. In order to capture these relations, meanings are seen as objects in an abstract model. A bit more formally, this is expressed by saying that natural language sentences refer to or *denote* objects in the model. In other words, the denotation assigned to typed lambda terms serve as a bridge to interpret linguistic expressions. Models are pairs consisting of a frame and a valuation. They are defined below.

DEFINITION 4.23. [Frame] A *frame* D consists of the collection of basic domains, *i.e.* $\cup_{\alpha \in \mathsf{Base}} Dom_\alpha$ and the domains for functional types. The latter are as follows

$$Dom_{(a,b)} = Dom_b^{Dom_a} = \{f \mid f : Dom_a \rightarrow Dom_b\}.$$

In words, expressions corresponding to functional types, like verb phrases, denote in the set of functions from the domain of their argument to the domain of their value. In our case, given the set of individuals E, the domains of functions are built up from the primitive ones below:

$$Dom_e = E \quad \text{and} \quad Dom_t = \{1, 0\}.$$

Besides the set of typed domains, a model must include an interpretation function I mapping the items of the lexicon to elements of the domains.

DEFINITION 4.24. [Model] A *model* is a pair $\mathcal{M} = \langle D, I \rangle$ in which the interpretation of the constant terms `lex` in the lexicon `Lexicon` of a given language are obtained as follow

i. D is a frame;
ii. The interpretation function is $I : \texttt{Lexicon} \rightarrow D$, s.t. if α is of type a, $I(\alpha) \in Dom_a$.

The interpretation function over lexical expressions is extended by the denotation function which recursively assigns an interpretation to all expressions.

DEFINITION 4.25. [Denotation] The *denotation* $[\![\alpha_a]\!]^f_{\mathcal{M}}$ of a λ-term α_a with respect to the model $\mathcal{M} = \langle D, I\rangle$ and assignment f, where $f : \mathsf{VAR}_a \rightarrow Dom_a$, is given by

i. $[\![x_a]\!]^f_{\mathcal{M}} = f(x_a)$ if $x_a \in \mathsf{VAR}_a$.
ii. $[\![\alpha_a]\!]^f_{\mathcal{M}} = I(\alpha_a)$ if $\alpha_a \in \mathsf{CON}_a$.
iii. $[\![\alpha_{(a,b)}(\beta_a)]\!]^f_{\mathcal{M}} = [\![\alpha_{(a,b)}]\!]^f_{\mathcal{M}}([\![\beta_a]\!]^f_{\mathcal{M}})$.
iv. $[\![\lambda x_a.\alpha_b]\!]^f_{\mathcal{M}} = g$ such that $g(d) = [\![\alpha_b]\!]^{f[x_a:=d]}_{\mathcal{M}}$.

where $f[x_a := d]$ stands for the assignment that maps x_a to $d \in Dom_a$ and maps $y_a \neq x_a$ to $f(y_a)$.

Intuitively, the denotation of a term formed by the λ-operator says that applying the denotation of a functional term $\lambda x.\alpha$ to an object d is the result of evaluating α in an assignment where x takes the value d.

REMARK 4.26. The form and meaning components of linguistic signs are inhabitants of their corresponding syntactic and semantic types, respectively. The definitions above say that two signs may differ in their form (belong to different syntactic types) despite being similar in their meaning (belonging to the same semantic type). Consequently, the two signs receive the same interpretation denoting the same object in the domain. For instance, this is the case of signs whose forms are in the syntactic type A/B and $B\backslash A$ and, therefore, their meanings are in the semantic type $(\texttt{type}(B), \texttt{type}(A))$ and are interpreted in the domain $Dom_{(b,a)}$.

4.3.2 Interpretations for the Sample Grammar

Natural language expressions can be interpreted by assuming either a relational or a functional perspective. We briefly illustrate the two approaches and their connection by discussing some examples. As a notational convention, we represent the constants in TERM with special fonts. For the ease of presentation, we do not indicate the semantic types unless necessary. For instance, the individual *Lori* is assigned a denotation in the domain of entities, and is represented by the term `lori`. The meaning of complex phrases is built out of the meaning of the lexical items. Thus we must start by adding the semantic information in the lexicon.

DEFINITION 4.27. [Term Labelled Lexicon] Given a set of basic expressions of a natural language Σ, a term labeled categorial lexicon is a relation,

$$\mathsf{LEX} \subseteq \Sigma \times (\mathsf{CAT} \times \mathsf{TERM}) \text{ such that if } (w, (A, \alpha)) \in \mathsf{LEX}, \text{ then } \alpha \in \mathsf{TERM}_{\texttt{type}(A)}$$

This constraint on lexical entries enforces the requirement that if the expression w is assigned a syntactic category A and term α, then the term α is of the appropriate type for the category A.

EXAMPLE 4.28. [Extended Lexical Entries] Labelled lexical entries are for instance the ones below,

Sara	$np:$	`sara`	which	$(n\backslash n)/(np\backslash s):$	$\lambda xyz.x(z) \wedge y(z)$
Pim	$np:$	`pim`	which	$(n\backslash n)/(np\backslash s):$	$\lambda xyz.x(z) \wedge y(z)$
Lori	$np:$	`lori`	some	$(s/(np\backslash s))/n:$	$\lambda xy.\exists z(x(z) \wedge y(z))$
knows	$(np\backslash s)/np:$	`know`	some	$((s/np)\backslash s)/n:$	$\lambda xy.\exists z(x(z) \wedge y(z))$
student	$n:$	`student`	some	$(tv\backslash(np\backslash s))/n:$	$\lambda xyu.\exists z(x(z) \wedge y(z)(u))$
professor	$n:$	`professor`	every	$(s/(np\backslash s))/n:$	$\lambda xy.\forall z(x(z) \rightarrow y(z))$
tall	$n/n:$	`tall`	every	$((s/np)\backslash s)/n:$	$\lambda xy.\forall z(x(z) \rightarrow y(z))$

Notice the different term assignment for the logical (the determiners and the relative pronoun) and the non-logical constants.

The denotations of the linguistic expressions are illustrated by the examples below.

EXAMPLE 4.29. [Relational Interpretation of Non-Logical Constants] Let our model be based on the set of entities $E = \{\texttt{lori}, \texttt{ale}, \texttt{sara}, \texttt{pim}\}$ which represent *Lori, Ale, Sara* and *Pim*, respectively. Assume that they all know themselves, plus *Ale* and *Lori* know each other, but they do not know *Sara* or *Pim*; *Sara* does know *Lori* but not *Ale* or *Pim*. The first three are students whereas *Pim* is a professor, and both *Lori* and *Pim* are tall. This is easily expressed set theoretically. Let $[\![\texttt{w}]\!]$ indicate the interpretation of w:

$$
\begin{array}{lcl}
[\![\texttt{sara}]\!] & = & \text{sara;} \\
[\![\texttt{pim}]\!] & = & \text{pim;} \\
[\![\texttt{lori}]\!] & = & \text{lori;} \\
[\![\texttt{know}]\!] & = & \{\langle\text{lori, ale}\rangle, \langle\text{ale,lori}\rangle, \langle\text{sara, lori}\rangle, \\
 & & \langle\text{lori, lori}\rangle, \langle\text{ale, ale}\rangle, \langle\text{sara, sara}\rangle, \langle\text{pim, pim}\rangle\}; \\
[\![\texttt{student}]\!] & = & \{\text{lori, ale, sara}\}; \\
[\![\texttt{professor}]\!] & = & \{\text{pim}\}; \\
[\![\texttt{tall}]\!] & = & \{\text{lori, pim}\}.
\end{array}
$$

which is nothing else to say that, for example, the relation *know* is the set of pairs $\langle\alpha, \beta\rangle$ where α knows β; or that 'student' is the set of all those elements which are a student.

Alternatively, one can assume a functional perspective and interpret, for example, *know* as a function $f : Dom_e \rightarrow (Dom_e \rightarrow Dom_t)$. The shift from the relational to the functional perspective is made possible by the fact that the sets and their characteristic functions amount to the same thing: if f_X is a function from Y to $\{0, 1\}$, then $X = \{y \mid f_X(y) = 1\}$. In other words, the assertion '$y \in X$' and '$f_X(y) = 1$' are equivalent.[5]

The interpretation of complex phrases is obtained by interpreting the corresponding lambda terms. For example, if $\texttt{walk} \in \mathsf{CON}_{(e,t)}$ and $x \in \mathsf{VAR}_e$, then $\texttt{walk}(x)$ expresses the fact that x has the property of walking, whereas $\lambda x.\texttt{walk}(x)$ is an abstraction over

[5]Consquently, the two notations $y(z)(u)$ and $y(u, z)$ are equivalent.

x and it represents the property itself. Moreover, due to the reduction rules of the lambda calculus the constant $\texttt{walk}_{(e,t)}$ is equivalent to the term $\lambda x_e.(\texttt{walk}(x))_t$. Applying Definition 4.25 this term is denoted by a function g such that for each entity $d \in Dom_e$, gives $g(d) = 1$ iff $[\![\texttt{walk}(x)]\!]^{f[x:=d]}_{\mathcal{M}} = 1$, or in other words iff x has the property expressed by walk.

The logical constants are interpreted by using set theoretical operations as illustrated below.

EXAMPLE 4.30. [Logical Constants] By evaluating the lambda expressions in Example 4.28 in a model, one obtains the interpretations below:

$$\begin{array}{lcl} [\![\texttt{no N}]\!] & = & \{X \subseteq E \mid [\![\texttt{N}]\!] \cap X = \emptyset\}. \\ [\![\texttt{some N}]\!] & = & \{X \subseteq E \mid [\![\texttt{N}]\!] \cap X \neq \emptyset\}. \\ [\![\texttt{every N}]\!] & = & \{X \subseteq E \mid [\![\texttt{N}]\!] \subseteq X\}. \\ [\![\texttt{N which VP}]\!] & = & [\![\texttt{N}]\!] \cap [\![\texttt{VP}]\!]. \end{array}$$

Generalized quantifiers have attracted the attention of many researchers working on the interaction between logic and linguistics [KF85, Eij85]. We will come back to them in Chapter 5.

4.4 Putting Things Together

In this section we explain how the syntactic derivations of the formal grammars discussed in Sections 4.1 and 4.2 are associated with instructions for meaning assembly.

4.4.1 Rule-Based Approach vs. Deductive Approach

In CG and CCG , the syntactic rules for category combination have the status of non-logical axioms. To obtain a Montague-style compositional interpretation, we have to associate them with instructions for meaning assembly in a rule-by-rule fashion. Below are the combination schemata we have been using paired rule-by-rule with their semantic interpretation.

Forward Application	$A/B : f \quad B : x \Rightarrow A : f(x)$	[FA]
Backward Application	$B : x \quad B\backslash A : f \Rightarrow A : f(x)$	[BA]
Lifting	$A : x \Rightarrow B/(A\backslash B) : \lambda y.yx$	[T]
Forward Composition	$A/B : f \quad B/C : g \Rightarrow A/C : \lambda x.f(gx)$	[B]
Backward Crossed Composition	$A/B : g \quad A\backslash C : f \Rightarrow C/B : \lambda x.f(gx)$	[$\text{B}_\times$]

EXAMPLE 4.31. [Meaning Assembly in CCG] Given the lexical assignments of the labelled lexicon above, CCG builds the meaning of *which Sara wrote* as follows.

$$\dfrac{\dfrac{\text{which}}{(n\backslash n)/(s/np) : \lambda xyu.x(u) \wedge y(u)} \quad \dfrac{\dfrac{\dfrac{\text{Sara}}{np : \texttt{sara}}}{s/(np\backslash s) : \lambda z.z(\texttt{sara})}\,[\text{T}] \quad \dfrac{\text{wrote}}{(np\backslash s)/np : \lambda yx.\texttt{wrote}(x,y)}}{s/np : \lambda y.\texttt{wrote}(\texttt{sara}, y)}\,[\text{B}]}{n\backslash n : \lambda yu.\texttt{wrote}(\texttt{sara}, u) \wedge y(u)}\,[\text{FA}]$$

Note that in the derivation we have hidden the β-conversion rules.

EXAMPLE 4.32. [Ambiguous Sentences] Let `some, some'` and `every` abbreviate the lambda terms from our labelled lexicon $\lambda x.\exists z\mathtt{student}(z) \wedge x(z)$, $\lambda xu.\exists z\mathtt{student}(z) \wedge x(u,z)$, and $\lambda x.\forall z\mathtt{student}(z) \rightarrow x(z)$, respectively. Considered the toy-grammar of CCG we have given above by way of illustration, the meaning of *every student knows some book* is built as following.

(i)

$$\dfrac{\dfrac{\text{every\_student}}{s/(np\backslash s) : \mathtt{every}} \quad \dfrac{\dfrac{\text{knows}}{(np\backslash s)/np : \mathtt{know}} \quad \dfrac{\text{some\_book}}{((np\backslash s)/np)\backslash(np\backslash s) : \mathtt{some'}}\,[\mathtt{BA}]}{np\backslash s : \mathtt{some'}(\mathtt{know})}\,[\mathtt{FA}]}{s : \mathtt{every}(\mathtt{some'}(\mathtt{know}))}$$

(ii)

$$\dfrac{\dfrac{\dfrac{\text{every\_student}}{s/(np\backslash s) : \mathtt{every}} \quad \dfrac{\text{knows}}{(np\backslash s)/np : \mathtt{know}}\,[\mathtt{B}]}{s/np : \lambda x.\mathtt{every}(\mathtt{know}\ x)} \quad \dfrac{\text{some\_book}}{(s/np)\backslash s : \mathtt{some}}\,[\mathtt{BA}]}{s : \mathtt{some}(\lambda x.\mathtt{every}(\mathtt{know}\ x))}$$

The derivation in (i) (resp. (ii)) gives the subject wide (resp. narrow) scope reading.

4.4.2 Curry-Howard Correspondence

In the Lambek calculus framework, syntactic rules are replaced by logical rules of inference. Therefore, the semantic rules are obtained deductively by exploiting the correspondence between proofs and terms. The famous Curry-Howard correspondence tells us that every proof in the natural deduction calculus for intuitionistic implicational logic can be encoded by a typed λ-term and *vice versa* [How80]. The categorial interpretation of derivations can be modelled directly on the Curry-Howard result, with the proviso that in the absence of structural rules in the categorial systems, the obtainable terms will be a sublanguage of the full λ-calculus.

Let us define the correspondence between the logical rules of NL and the application and abstraction rules of the lambda calculus. In a few words, the elimination of the functional connectives $\backslash$ and $/$ produces functional application terms, whereas the abstraction over variables corresponds to the introduction of the functional operators.

DEFINITION 4.33. [Term Assignment for Natural Deduction] Let $\Gamma \vdash t : A$ stand for a deduction of the formula A decorated with the term t from a structured configuration of undischarged term-decorated assumptions Γ.

$$x : A \vdash x : A$$

$$\dfrac{\Gamma \vdash t : A/B \quad \Delta \vdash u : B}{\Gamma \circ \Delta \vdash t(u) : A}\,[/\mathrm{E}] \qquad \dfrac{(\Gamma \circ x : B) \vdash t : A}{\Gamma \vdash \lambda x.t : A/B}\,[/\mathrm{I}]$$

$$\dfrac{\Delta \vdash u : B \quad \Gamma \vdash t : B\backslash A}{\Delta \circ \Gamma \vdash t(u) : A}\,[\backslash\mathrm{E}] \qquad \dfrac{(x : B \circ \Gamma) \vdash t : A}{\Gamma \vdash \lambda x.t : B\backslash A}\,[\backslash\mathrm{I}]$$

The Lambek calculi are fragments of intuitionistic implicational logic [Abr90]. Consequently, the lambda terms computed by it form a fragment of the full language of lambda terms. First of all, since empty antecedents are not allowed and the Lambek calculi are resource sensitive, *viz.* each assumption is used exactly once, the system reasons about lambda terms with specific properties: (i) each subterm contains a free variable; and (ii) no multiple occurrences of the same variable are present. The latter could seem to be too strong constraint when thinking of linguistic applications. However, this is not the case as we will discuss at the end of this section (Example 4.40). A formal definition of the lambda calculus fragment corresponding to LP is given below[6].

Definition 4.34. [Fragment of the Lambda Terms for LP] Let $\Lambda(\mathsf{LP})$ be the largest $\mathsf{LAMBDA} \subseteq \mathsf{TERM}$ such that

i. each subterm of $\alpha \in \mathsf{LAMBDA}$ contains a free variable;

ii. no subterm of $\alpha \in \mathsf{LAMBDA}$ contains more than one free occurrence of the same variable;

iii. each occurrence of the λ abstractor in $\alpha \in \mathsf{TERM}$ binds a variable within its scope.

Derivations for the various Lambek calculi are all associated with LP term recipes. Therefore, we move from an isomorphism to a weaker correspondence. The correspondence between LP proofs and the lambda calculus was given in [Ben87, Bus87, Wan92].

Theorem 4.35. Given an LP derivation of a sequent $A_1, \ldots, A_n \vdash B$ one can find a corresponding construction $\alpha_a \in \Lambda(\mathsf{LP})$, and conversely. A term $\alpha_a \in \Lambda(\mathsf{LP})$ is called a construction of a sequent $A_1, \ldots, A_n \vdash B$ iff α has exactly the free variable occurrences $x^1_{\mathtt{type}(A_n)}, \ldots, x^n_{\mathtt{type}(A_n)}$.

While introducing the lambda calculus we spoke of terms in normal forms. These terms are obtained proof theoretically by defining normal form derivations as following.

Definition 4.36. [Normal Form for Natural Deduction Derivations)] A derivation in natural deduction format is in *normal form* when there are no detours in it. A *detour* is formed when

i. a connective is introduced and immediately eliminated at the next step.

ii. an elimination rule is immediately followed by the introduction of the same connective.

The rules eliminating these two detours are called *reduction* rules.

Remark 4.37. The reductions of the detours in *i.* and in *ii.* correspond to β-reduction and η-reduction, respectively. Moreover, note that the above rewriting rules hold for all Lambek calculi, regardless of their structural rules.

[6]Again, for the sake of simplicity here we restrict attention to product-free Lambek calculi. See [Moo97] for the definition of the full systems.

By means of example, we give the reduction rule corresponding to η-reduction. The reader is referred to [Res00] for an extensive presentation of normalization.

$$\dfrac{\dfrac{\begin{array}{c}D_1\\ \vdots\\ [B \vdash x : B]^1 \quad \Gamma \vdash t : B\backslash A\end{array}}{B, \Gamma \vdash t(x) : A}\,[\backslash\mathrm{E}]}{\Gamma \vdash \lambda x.t(x) : B\backslash A}\,[\backslash\mathrm{I}]^1 \qquad \text{rewrites to} \qquad \begin{array}{c}D_1\\ \vdots\\ \Gamma \vdash t : B\backslash A\end{array}$$

in the lambda-calculus the reduction above corresponds to the rewrite rule $\lambda x.t(x) \Rightarrow_\eta t$ The correspondence between proofs and lambda terms is completed by the following theorem [Pra65, Gir87, GLT89].

Theorem 4.38. [Normalization] If $\mathcal{D}$ is a normal form derivation of $x_1 : A_1, \ldots x_n : A_n \vdash \alpha : C$, then α is in β, η normal form.

Let us now check how this framework accounts for the assembly of form-meaning pairs.

Starting from the labelled lexicon, the task for the Lambek derivational engine is to compute the lambda term representing the meaning assembly for a complex structure as a by-product of the derivation that establishes its grammaticality. The crucial distinction here is between the *derivational* meaning and the *lexical* meaning. The derivational meaning fully abstracts from lexical semantics: it is a general recipe for meaning assembly from assumptions of the given types.

Practically, one can proceed in two ways: (i) either one starts labeling the axioms of a derivation with the actual lambda terms assigned in the lexicon, or (ii) one labels the leaves of the derivation with variables, computes the proof term for the final structure and then replaces the variables by the actual lambda terms assigned in the lexicon to the basic constituents. We illustrate the two methods below in Examples 4.39 and 4.40, respectively.

Example 4.39. [Lifting] Starting from the type assignment Lori $\in np :$ `lori`, one derives the higher order assignments as following:

$$\dfrac{\dfrac{\text{Lori} \vdash np : \texttt{lori} \quad [np\backslash s \vdash np\backslash s : x]^1}{\text{Lori} \circ np\backslash s \vdash s : x(\texttt{lori})}\,[\backslash\mathrm{E}]}{\text{Lori} \vdash s/(np\backslash s) : \lambda x.x(\texttt{lori})}\,[/\mathrm{I}]^1 \qquad \dfrac{\dfrac{[s/np \vdash s/np : x]^1 \quad \text{Lori} \vdash np : \texttt{lori}}{s/np \circ \text{Lori} \vdash s : x(\texttt{lori})}\,[/\mathrm{E}]}{\text{Lori} \vdash (s/np)\backslash s : \lambda x.x(\texttt{lori})}\,[\backslash\mathrm{I}]^1$$

First of all, note how the system assigns a variable to the hypothesis. The latter is discharged by means of [/I] (or [\I]) which corresponds to the abstraction over the variable. Moreover, note that the higher order types in the two derivations are different, but they correspond to the same lambda terms, *i.e.* the two structures are correctly assigned the same meaning.

This example shows how in the CTL framework, the assembly of meaning is a byproduct of the proof theoretical analysis. In particular, the type-lifting, stipulated in the Montagovian tradition and explicitly expressed by the [T] combinator in CCG, is obtained simply by means of logical rules. See [Oeh99] for a discussion about the advantages of having the lifting as a derivable theorem in the system.

The relative clause examples in our toy fragment offer a nice illustration of the division of labor between lexical and derivational semantics. Intuitively, a relative pronoun has to compute the intersection of two properties: the common noun property obtained from the n that is modified, and the property obtained from the body of the relative clause, a sentence with a np hypothesis missing. In the logical form, this would come down to binding two occurrences of a variable by one λ binder. On the level of *derivational* semantics, one cannot obtain this double binding: the Lambek systems are resource sensitive, which means that every assumption is used exactly once. But on the level of *lexical* semantics, we can overcome this expressive limitation (which is syntactically well-justified!) by assigning the relative pronoun a double-bind term as its lexical meaning recipe: which $\in (n\backslash n)/(s/np) : \lambda xyz.x(z) \wedge y(z)$. In this way, we obtain the proper recipe for the relative clause *which Sara wrote*, namely $\lambda yz.\texttt{wrote}(\texttt{Sara}, z) \wedge y(z)$, as shown below.

Example 4.40. [Relative Clause]

$$\dfrac{\text{which} \vdash (n\backslash n)/(s/np) : X_4 \qquad \dfrac{\dfrac{\text{Sara} \vdash np : X_3 \qquad \dfrac{\text{wrote} \vdash (np\backslash s)/np : X_1 \quad [x \vdash np : X_2]^1}{\text{wrote} \circ x \vdash np\backslash s : X_1X_2}\,[/\text{E}]}{\text{Sara} \circ (\text{wrote} \circ x) \vdash s : (X_1X_2)X_3}\,[\backslash\text{E}]}{(\text{Sara} \circ \text{wrote}) \circ x \vdash s : (X_1X_2)X_3}\,[\text{ass}]}{\text{Sara} \circ \text{wrote} \vdash s/np : \lambda X_3.(X_1X_2)X_3}\,[/\text{I}]^1}{\text{which} \circ (\text{Sara} \circ \text{wrote}) \vdash n\backslash n : X_4(\lambda X_3.(X_1X_2)X_3)}\,[/\text{E}]$$

Note that the structural rules do not effect the meaning assembly. By replacing the variables $X_1, \ldots, X_4$ with the corresponding lexical assignments, and applying the reduction rules, one obtains the proper meaning of the analyzed structure.

Generalizing over the different uses of unary operators we have reviewed in Section 4.2, one could say that unary operators allow us to express distinctions among members of the same semantic type which are relevant for the syntactic composition. In other words, the unary operators express distinctions similar to the ones expressed by the directional functional implications $\backslash$ and $/$ at the level of meaning assembly, where the directionality information plays no role anymore.

In a similar way, the unary operators encode in the type assignments fine-grained distinctions both within and across languages which are not distinguishable in the meaning assembly. For instance, we have seen them at work to encode the different syntactic behavior of plural and singular Italian articles. They are both interpreted in the domain $Dom_{((e,t),e)}$, *viz.* the set of functions from nouns to noun phrases, however, their contributions to the linguistic composition differ: the plural article *i* is unable to compose with a singular noun, whereas the singular *il* can. The unary operators encode this difference which is not visible on the level of the domains of interpretation. Similarly, the crosslinguistic contrast between the way adjectives may combine with nouns in Italian and in English does not play any role in the assignment of the meaning to their composition, but it is relevant for their syntactic assembly in the two languages. This observation about unary operators is at the heart of the analyses presented in the next Chapter.

4.5 Key Concepts

The main points of this chapter to be kept in mind are the following:

1. Linguistic signs are pairs of form and meaning, and composed phrases are structures rather than strings.
2. When employing a logic to model linguistic phenomena, grammatical derivations are seen as theorems of the grammatical logic.
3. Unary operators and modes can be used to lexically anchor linguistic composition.
4. The correspondence between proofs and natural language models, via the lambda terms, properly accounts for the natural language syntax semantics interface.

Chapter 5

Modalities for Partial Orderings

Linguistic composition is affected by several aspects of the constituents involved. In this Chapter, we investigate logico-semantic properties of quantifier phrases (QPs) and the sensitivity of polarity items (PIs) with respect to a certain semantic property shared by other expressions called 'triggers'. We study how these semantic factors influence the different scope behavior of the items involved and cause ill-formedness. To this end, we look for a classification of such expressions reflecting distinctions within the domains of interpretation of the linguistic signs.

By means of CTL we spell out the link between the subset relations holding at the semantic level and the way the interpreted items behave syntactically. Using our extended vocabulary of type-forming operators, the subset relations within semantic domains are captured by syntactic derivability relations between types. As a result, we gain a proof theoretical understanding of the scope construal of QPs and syntactic licensing/antilicensing relations of PIs.[1]

5.1 Zooming in on the Semantic Domains

In this Chapter, we look at items which are in a licensing or antilicensing relation with a certain property. These items are grammatical *only* when in construction with the expressions having this property; or ungrammatical when in construction with these expressions and grammatical in construction with the signs which do not have the property they repel.

A first reason to be interested in sorting out these different composition relations is that they provide a classification of items belonging to the same semantic domain. The property an item can be sensitive to may be shared by several expressions creating a net of licensing relations. In other words, the licensing relation, holding between an item and a trigger having that specific property attracting the item, is inherited by the other expressions sharing that same property with the (direct) trigger. Semantically, the connections among phrases sharing some property is expressed in terms of inclusion relations of the domain of interpretation. Syntactically, the same link can be captured by

[1]The results on the QP analysis presented in this chapter are partially based on joint work with Richard Moot [BM03].

derivability relations among their types and the inheritance relations among the direct trigger and its relatives is determined deductively.

As in the accounts reviewed in the previous Chapter, here as well we employ unary operators to zoom in on the domains of interpretation encoding differences not visible otherwise. However, in this Chapter attention is focused on different sorts of features distinguishing the members of the same semantic type. Moreover, we will use both Residuated and Galois unary operators. In particular, we look at the monotonicity and nonveridicality properties of expressions in the functional domains, as well as at the difference between, for instance, distributive and non-distributive quantifier phrases. We will show how $\mathsf{NL}(\Diamond,\cdot^{\mathbf{0}})$, introduced in Chapter 3, can account for the sentences in our original checklist (Example 4.3) which the grammars discussed in Chapter 4 fail to recognize. We repeat those sentences below indicating their interprations by means of [X > Y], *viz.* X has scope over Y.

(1) a. Every student knows one book. [Every > One], [*One > Every]
b. Every student knows some book. [Every > Some], [Some > Every]
c. No student knows any book. [No > Any], [*Any > No]

(2) a. No student left yet.
b. Some student left already.

The challenge one has to face to account for these sentences is to have a language expressive enough to distinguish the scope behavior of the quantifier phrases, and the different distribution of the adverbs *yet* and *already*. Moreover, the grammar has to account for the ungrammaticality of the sentences below:

(3) a. *A student knows any book.
b. *No student left already.
c. *Some student left yet.

Briefly, we exploit the different compositions of the unary operators to differentiate the sentential levels on which quantifiers may or may not take scope accounting for their different ways of scoping. Furthermore, the same property is used to embody the subset relation holding inside a domain between members enjoying different but related properties. For instance, in the domain of quantifiers one could distinguish the antiadditive quantifier *nobody* and the downward monotone one *few n*, where the set of antiadditive functions is a subset of the downward monotone ones. We take advantage of fine-grained type assignments to model polarity items which are in a licensing condition with some semantic property. Finally, we show how the downward monotone property of the Galois operators could have a role in dealing with antilicensing relations.

The *Leitmotiv* of all these analyses is the simple schema given in the previous Chapter and repeated here. Let $C \longrightarrow B$,

$$\frac{\Delta \vdash A/B \quad \begin{array}{c}\Gamma \vdash C \\ \vdots \\ \Gamma \vdash B\end{array}}{\Delta \circ \Gamma \vdash A}\ [/\mathrm{E}]$$

An important aspect to underline is that differently from the analysis of morphological

agreement, in our approach the derivation from $\Gamma \vdash C$ to $\Gamma \vdash B$ will be carried out only by logical rules with no application of structural reasoning. Therefore, our analysis stays within the borders of the base logic given by the algebraic principles of Galois connections and residuation. In particular, we will exploit the following feature of unary operators that raised in our investigation of the mathematical structure of CTL, namely the way residuated and Gaolis operators compose. Recall the patterns

$$\Diamond\Box^{\downarrow}A \longrightarrow A \quad \text{and} \quad A \longrightarrow \Box^{\downarrow}\Diamond A.$$

Moreover, $\mathsf{NL}(\Diamond,\cdot^{\mathbf{0}})$ offers a second pair of unary operators $({}^{\mathbf{0}}\cdot, \cdot^{\mathbf{0}})$ which exhibit different logical behavior

$$A \longrightarrow {}^{\mathbf{0}}(A^{\mathbf{0}}) \quad \text{and} \quad A \longrightarrow ({}^{\mathbf{0}}A)^{\mathbf{0}}$$

and such that ${}^{\mathbf{0}}(A^{\mathbf{0}}) \not\longleftrightarrow \Box^{\downarrow}\Diamond A$, and $({}^{\mathbf{0}}A)^{\mathbf{0}} \not\longleftrightarrow \Box^{\downarrow}\Diamond A$. Finally, the Galois connected operators introduce a way to reverse the derivability relations holding among types, e.g. if $A \longrightarrow B$, then ${}^{\mathbf{0}}B \longrightarrow {}^{\mathbf{0}}A$. Based on these relations we obtain the cube of derivability relations given in Chapter 3 that offers a rich hierarchy of types allowing us to make fine-grained distinctions among expressions of the same domain of interpretation as we will show in the remaining part of this Chapter. In Figure 5.1 the cube of derivability relation given by one possible combination of the unary Galois, a symmetric level is obtained by means of the other pair.

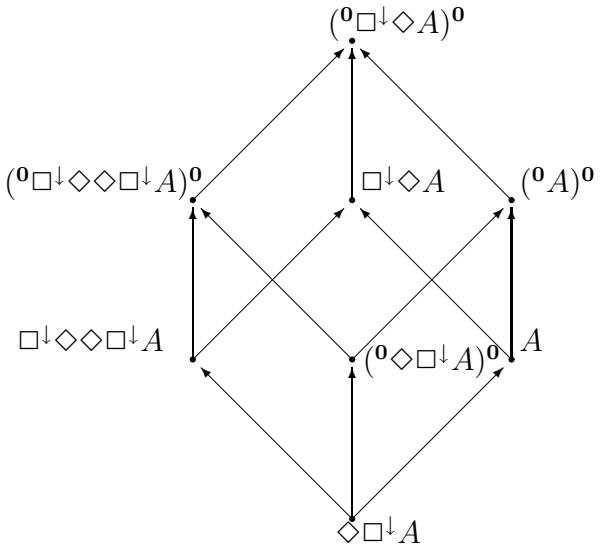


Figure 5.1: Some derivability patterns in $\mathsf{NL}(\Diamond,\cdot^{\mathbf{0}})$.

5.2 QP Classification

In this section we focus attention on the differences exhibited by quantifiers with respect to the ways of scope taking [BS97]. The problem can be exemplified as below,

(4) a. John didn't read a book. [Not > A], [A > Not].
b. John didn't read every book. [Not > Every], [*Every > Not].

(5) a. Every boy read a different book. [Every > A], [*A > Every].
b. *All the boys read a different book.

(6) a. Three referees read few abstracts. [Three > Few], [*Few > Three].
b. Few referees read three abstracts. [Three > Few], [Few > Three].

In (4) the preferred reading is for negation to scope over the existential QP in (4-a) and over the universal QP in (4-b). However, while the existential QP is free to scope over negation (4-a), the universally quantified object can scope over negation only if focussed. The contrast in (5) shows that while *every* has the distributivity property, *all* lacks it as emphasized by the presence of *different*. Finally, (6) illustrates the inability of *few abstracts* to take scope over a QP preceding it in the surface structure (s-structure).

In [BS97] it is shown that (i) scope interaction among quantifier phrases (QPs) is determined by the need to respect the contribution of distributivity and (ii) the availability of the inverse scope reading depends on the interaction between the scope elements involved. Following these criteria, Beghelli and Stowell propose a classification of quantifier phrases.

First of all, QPs can be distinguished by considering whether they introduce a *discourse referent* or not. Counting quantifier phrases (CQPs) like *few referees* belong to the latter class, the other QPs to the former one.

A second distinction concerns the sort of variables introduced: either individual variables denoting groups, or set variables. Indefinites and definite quantifiers like *a book* and *the books* are instances of the first case referred to as group quantifier phrases (GQPs); distributive quantifiers like *every* and *each* form the second group (DQPs). These two classes can be further subcategorized by considering the way their members behave with respect to distributivity and negation.

GQPs are either referentially dependent —they range over individuals whose existence is presupposed— or they are referentially independent (e.g. the definite quantifier *the books*). In the last case, besides introducing a group of referents, they fulfill the function of being the logical subject of predication. Therefore, one could say that the latter subclass has an extra feature with respect to the former one. QPs of the second group cannot work as a distributed share of DQPs since they introduce discourse referents which cannot be multiplied (7-a), while members of the first group can (7-b). Finally, there are indefinites and bare-numeral GQPs which can alternatively be interpreted non-specifically, in this case they lack the feature particular to GQPs and take scope locally like CQPs (7-c).

(7) a. Every student read the books. [The > Every].
b. Every student read a book. [Every > A], [A > Every].
c. Every boys read two books about India. [Every > Two], [Two > Every].

Among the DQPs, Beghelli and Stowell distinguish *each* from *every*. The former is said to introduce a set of variables which must be bound by a distributive operator.

Hence (8-a) is awkward, and (9-a) does not have a generic reading. On the other hand, the set of variables introduced by *every* can be bound by negative (8-b) and generic operators (9-b) as well as by the distributive operator (7-b). This contrast gives rise to different scope possibilities.

(8) a. %John didn't read each book.
b. John didn't read every book. [Not > Every]

(9) a. Each dog has a tail.
b. Every dog has a tail.

Note that *all* is similar to *every* for its universal force, but cannot work as a distributive operator as emphasized by the presence of *different* in (10-a,b). Therefore, it is considered to be part of GQPs group since it behaves like them with respect to negation (11-a,b), though it differs from the other members for not being able to work as distributed share of DQPs (7-b) and (11-c).

(10) a. *All the boys read a different book.
b. Every boy read a (different) book. [Every > A book].
c. All the boys read a book. [A > All], [All > A].

(11) a. John didn't read all the books. [Not > All], [All > Not].
b. John didn't read a book. [Not > A], [A > Not].
c. Every boy read all the books. [All > Every].

Besides these three groups, Beghelli and Stowell consider negative quantifier phrases (NQPs) and interrogative quantifier phrases (WhQPs), where the former needs to be bound by the negative operator and the latter by the interrogative one. The full picture of the different main groups is shown below.

Group-denoting QPs (GQPs):	e.g. *a N*, *some N*, *all N*, *the N*;
Interrogative QPs (WhQPs):	e.g. *what*, *which N*, *how*;
Counting QPs (CQPs):	e.g. *few N*, *exactely n N*, *at most n N*;
Distributive-Universal QPs (DQPs):	e.g. *every N*, *each N*;
Negative QPs (NQPs):	e.g. *nobody*, *no N*.

5.2.1 Feature Checking Theory for QP Scope

Beghelli and Stowell's analysis follows the generative grammar tradition, hence we first briefly introduce its main aspects. The standard theory of quantifier scope in generative grammar (see May [May77], Reinhart [Rei97], among others) is based on two central assumptions: (i) Quantifier scope is determined by the constituent command relation (c-command) holding at the level of Logical Form (LF)[2]; (ii) QPs are assigned scope by undergoing *movement* to their scope positions in the derivation of the LF representations. A generative grammar can produce different LF-structures for the same

[2]A node *a c-commands* a node *b* if the first branching node dominating *a* dominates *b* too.

(ambiguous) well-formed clause. Due to the different c-command relations at the level of LF the surface structure (s-structure) receives different meanings.

In this approach, lexical elements carry two sorts of information: (i) one regarding the category they select, and (ii) one about the features they require to be checked. Consequently, a successful movement must satisfy two requirements: (i) the expression which moves must land to the specifier SPEC of the node dominating the selected sister-node, and (ii) the HEAD of the node must be labelled with the appropriate feature, matching the one carried by the expression to be moved. In the minimalist approach, these two mechanisms correspond to two different operations: (i) MERGE which takes care of the first request (category selection) and MOVE driven by features checking.

In the generative grammar tradition, the standard way of controlling movement operations is by means of features. Beghelli and Stowell [BS97] apply this method to account for different QP distributions. Scope is seen as the by-product of agreement processes checked via SPEC-HEAD agreement, and mismatches in agreement give rise to ungrammatical sentences. They distinguish five classes of QPs and indicate membership to any of the QP-groups by some syntactic properties which are morphologically encoded in the determiner position. They claim that for certain combinations of quantifier types the grammar simply excludes certain logically possible scope construals. Let us summarize their method.

In order to account for the facts illustrated above, Beghelli and Stowell consider the clausal structure as including, among others, a hierarchy of functional projections (FP) which are the landing sites for QPs. Each quantifier acquires its scope by moving into the specifier of that functional projection which suits its semantic and/or morphological properties. For instance, the landing site of DQPs (SPEC-DISTP) must have the distributive operator $\forall$ (hosted in HEAD-DISTP), and the functional category DISTP must select for a distributed share phrase (SHAREP) where GQPs can land. In a similar way, the order among the other FPs is obtained reaching the full functional structure in Figure 5.2 where the HEAD-positions are compiled in for the ease of presentation.

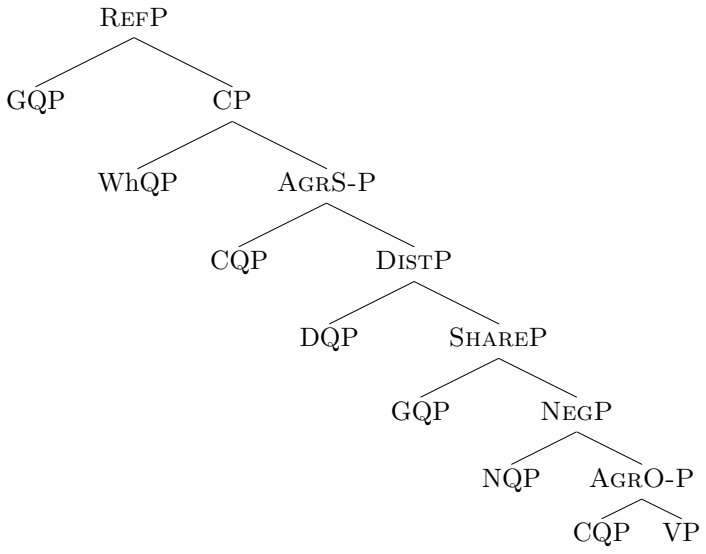


Figure 5.2: Phrase structure for QPs.

Movement is driven by the need of checking the features carried by the QPs. For example, negative quantifiers like *nobody* bear a feature [+Neg] and therefore they must move to the SPEC of the negative phrase (NEGP) hosting the negative operator $\neg$. From this it follows that NQPs cannot have scope over quantifiers which must land on a level higher than the functional category NEGP, e.g. the definite *the books* which moves to the specifier of the referential phrase (SPEC-REFP) (12-a.), but can have scope on QPs which are on a lower level, e.g. *few books* which stay in its case position, SPEC-AGRO-P (12-b).

(12) a. John didn't read the books. [The > Not].
b. John didn't read few books. [Not > Few].

The different positions assigned to the subject agreement phrase (AGRS-P) and the object agreement phrase (AGRO-P), reflect the asymmetric behavior exhibited by CQPs when occurring in the two positions (6-a) and (6-b). Notice that since all quantifiers carry information about their case, they might need to reconstruct under a lower level to check their scope features after having cancelled their case features at SPEC-AGRS-P.

Finally, notice that each of the levels hosts an operator. For example, $\neg, \exists, \forall$ are hosted in the heads Neg^0, Share^0 and Dist^0, respectively[3]. These operators attract the features carried by the QPs. From this, it follows that the different features which characterize the classes of QPs carry logico-semantic information and the functional structure above corresponds to a hierarchy of operators.

5.2.2 Controlling Scope Distribution in CTL

The aim of this section is to obtain the scope constraints discussed above deductively by means of modally refined type assignments. We start by explaining how QPs are dealt with in the CTL framework and introducing the q-operator [Moo91]. We then move to compare the movement operation used within the minimalist framework with the $[q\text{E}]$ rule used in CTL. Via this comparison we learn how to decorate the type assignment of QPs with unary modalities as to control the scope distribution of the different quantifiers. Once the method has been explained, we look back at the possible types at our disposal and start exploring the landscape of natural language quantifiers.

Modalities for Feature Checking

Quantifier Phrases belong to a class of phenomena that are difficult to be analyzed by means of CTL due to the discrepancies between syntactic and semantic composition they exhibit. To tackle this problem in the case of problematic binders, Moortgat provides in [Moo91] a three-place operator $q(A, B, C)$ which captures their behavior "in situ." (See [Hen93, DG01] for alternative solutions within the CTL framework.) The intuitive interpretation is the following. Syntactically, the q-category occupies the position of an

[3]The existential operator $\exists$ is hosted in the referential head (Ref^0) as well. Ref^0 differs from Share^0 in having an extra feature which attracts those quantifiers introducing referentially independent variables, e.g. definite quantifiers.

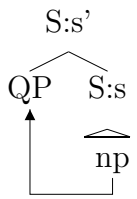
expression of category A within a structural context of category B and while doing so it turns it into one of category C.

Semantically, the category $q(A, B, C)$ maps to $((\mathsf{type}(A), \mathsf{type}(B)), \mathsf{type}(C))$. An expression of that type binds a variable of type $\mathsf{type}(A)$ within a domain of type $\mathsf{type}(B)$, producing a meaning recipe of type $\mathsf{type}(C)$ as a result of functional application. The Natural Deduction rule below encapsules this combined syntactic/semantic behavior.

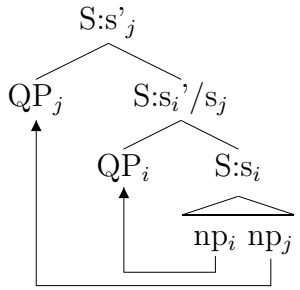
$$\frac{\Gamma \vdash \alpha : q(A, B, C) \quad \Delta[x : A] \vdash \beta : B}{\Delta[\Gamma] \vdash \alpha(\lambda x.\beta) : C}\ [q\mathrm{E}].$$

In the case of QPs, we obtain $q(np, s, s)$ which can be read as saying that (i) the quantifier is a binder of an np type variable, (ii) the binding relation is within a sentential domain, and (iii) the whole resulting structure is a sentence.[4]

Translating this into the minimalist approach, we have that (i) the QP is an expression undergoing movement, (ii) its trace is within a sentential phrase, (iii) the landing site of QP is again a sentential phrase. For the ease of exposition we distinguish the binding domain s by the resulting one s': $q(np, s, s')$. We can think of the $[q\mathrm{E}]$ rule of use as producing the replacement of the np represented below:



where the s' and s can be thought of as features carried by the head of the functional projection. Let us now take a structure containing two quantifiers QP_i and QP_j and let $q(np, s_i, s'_i)$ and $q(np, s_j, s'_j)$ be their types, respectively. Applying what we have said to generate the simple tree above, we obtain



which can be read as saying QP_j selects for s'_i and carries a feature which must be checked against the HEAD of $S : s'_j$ by moving to its SPEC position. Notice that though QP_j does not carry the type (feature) s'_i in its q-type, one could say that it has a type

[4] In the multimodal setting of Chapter 3, the q connective of course cannot be a *primitive* connective: the challenge is to show how it can be synthesized in terms of logical and structural rules for the primitive operations $\Diamond, \Box^{\downarrow}, /, \bullet, \backslash$ with appropriate mode distinctions. Such a decomposition of q is in fact proposed in [Moo96a, AB03, Moo04]. The exact details of the decomposition are not directly relevant to the issues dealt with in this course. Here, we will use the simple format of $[q\mathrm{E}]$ as a derived rule of inference.

(feature) s_j such that s'_i derives (agrees with) it, *i.e.* $s'_i \longrightarrow s_j$. In other words, the scope constraints forced by the functional projection hierarchy are accounted for deductively.

This discussion shows that the different ways of scoping exhibited by QPs can be controlled by differentiating their sentential types: a QP will have scope over a second one if a derivability condition among types is satisfied. In Chapter 3, we have presented the full scale of derivability patterns at our disposal. The simplest cases are repeated in Figure 5.3.

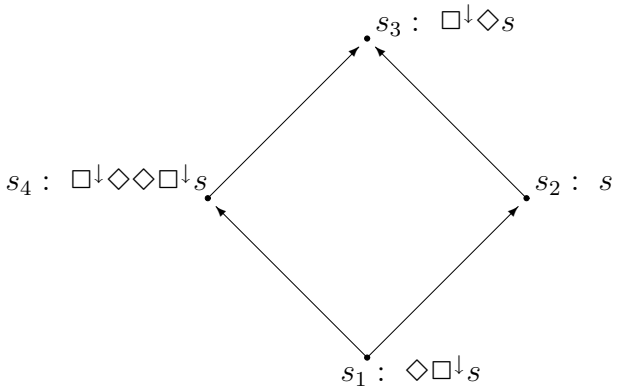


Figure 5.3: Basic sentential levels.

With these four sentential types, sixteen different QP-types can be obtained. Taking into consideration what we said above, one can establish a classification of QP-types based on their scope possibilities. The strongest quantifier type —the one which will have wide scope in most of the cases— is $q(np, \Box^{\downarrow}\Diamond s, \Box^{\downarrow}\Diamond s)$: it is able to take scope over all QP-types (since all sentential types derive $\Box^{\downarrow}\Diamond s$), and there are only four QP-types which can take scope over it, those with $\Box^{\downarrow}\Diamond s$ as binding domain. For symmetric reasons, the weakest QP-type is $q(np, \Diamond\Box^{\downarrow} s, \Diamond\Box^{\downarrow} s)$. Let us now see these QP-types at work by considering the linguistic data presented by Beghelli and Stowell. Our QP-type hierarchy gives us a way to situate the classification proposed within the minimalist program. Moreover, it predicts the existence of further subclasses of quantifier phrases.

Types for Beghelli and Stowell's QP Classification

From the analysis of the linguistic data, it follows that negation and distributivity play a fundamental role in determining the scope possibilities of QPs. Therefore, we start by focusing attention on the functional categories NegP and DistP.

Negative quantifiers and negation can have scope over all quantifiers with the exception of *each* and the ones landing to RefP, e.g. *the books*. We repeat the relevant data below.

(13) a. %John didn't read each book.

b. John didn't read the book. [The > Not].

c. John didn't read all the books. [Not > All], [All > Not].

d. John didn't read a book. [Not > A], [A > Not].

e. John didn't read every book. [Not > Every].

Moreover, as the data show among the QPs considered, *every* and *each* are the only two quantifiers which cannot have scope over negation. Beghelli and Stowell explain this fact by saying that they need the distributed share phrase SHAREP to be filled in, in order to generate grammatical structures. The sentence below is taken as evidence for this claim.

(14) One boy didn't read each book. [Each > One > Not].

The second important borderline in Beghelli and Stowell's functional projection hierarchy is DISTP. Not all the QPs can work as distributed share for DQPs, from this fact scope distinctions follow. In particular, *every* and *each* cannot take scope over NQPs, *all* and those QPs landing to SPEC-REFP.

The scope possibilities, particular to the natural language quantifiers studied so far, are expressed by the QP-types listed in Table 5.1. We refer to the types using the corresponding abbreviations s_i given in Figure 5.3. Recalling what said in Section 5.2.2 about the scoping strength degree of QP-types, the lexical assignments say, for instance, that definite GQPs will always have wide scope: their QP-type can take scope over all the other QP-types and none of the ones used in the lexicon manage to take (immediate) scope over it.

NQPs	$q(np, s_2, s_2)$	e.g. *nobody*;
pure DQPs	$q(np, s_4, s_4)$	e.g. *each_n*;
universal DQPs	$q(np, s_4, s_1)$	e.g. *every_n*;
universal GQPs	$q(np, s_3, s_2)$	e.g. *all_n*;
definite GQPs	$q(np, s_3, s_3)$	e.g. *the_n*;
indefinite and bare numeral GQPs	$q(np, s_3, s_1)$	e.g. *a_n*, *one_n*.

Table 5.1: Lexicon.

Let us check how these scope constraints are actually derived in CTL. We start by considering the interaction of QPs with negation and show how the data in (13) follow from the lexical assignments in Table 5.1 and the types $(np\backslash s_2)/(np\backslash s_2)$ and $(np\backslash s_1)/np$ assigned to *didn't* and *read*, respectively. In the derivations in Figure 5.4, the QP-type is represented by a variable-type $q(np, s_x, s'_y)$ which must be instantiated by the different QP-types given above to check their scope possibilities. A derivation from $\Delta \vdash s_i$ to $\Delta \vdash s_j$ (*i.e.* $s_i \longrightarrow s_j$) is abbreviated as $[D_i]$. Finally, recall that due to the Curry-Howard correspondence (see Section 4.3) while determining the grammaticality of the linguistic structures, the logical rules build their meaning as well and a unique lambda term is assigned to each syntactic derivation. For instance, the derivation in Figure 5.4 builds the lambda terms as shown in Figure 5.5, where the Q is a variable to be replaced by the actual term representing the quantifier in the structure.

[Not > QP]

$$\dfrac{\text{John} \vdash np \qquad \dfrac{\text{didn't} \vdash (np\backslash s_2)/(np\backslash s_2) \qquad \dfrac{\dfrac{\boxed{\text{QP} \vdash q(np, s_x, s'_y)} \qquad \dfrac{\dfrac{[y \vdash np]^2 \qquad \dfrac{\text{read} \vdash (np\backslash s_1)/np \qquad [x \vdash np]^1}{\text{read} \circ x \vdash np\backslash s_1}\,[/\text{E}]}{y \circ (\text{read} \circ x) \vdash s_1}\,[\backslash\text{E}]}{y \circ (\text{read} \circ x) \vdash s_x}\,[D_1]}{y \circ (\text{read} \circ \text{QP}) \vdash \boxed{s'_y}}\,[q\text{E}]^1}{y \circ (\text{read} \circ \text{QP}) \vdash \boxed{s_2}}\,[D_y]}{\text{read} \circ \text{QP} \vdash np\backslash s_2}\,[\backslash\text{I}]^2}{\text{didn't} \circ (\text{read} \circ \text{QP}) \vdash np\backslash s_2}\,[/\text{E}]}{\text{John} \circ (\text{didn't} \circ (\text{read} \circ \text{QP})) \vdash s_2}\,[\backslash\text{E}]}{\text{John} \circ (\text{didn't} \circ (\text{read} \circ \text{QP})) \vdash s_3}\,[D_2]$$

[QP > Not]

$$\dfrac{\dfrac{\boxed{QP \vdash q(np, s_x, s'_y)} \qquad \dfrac{\text{John} \circ (\text{didn't} \circ (\text{read} \circ [x \vdash np]^1)) \vdash \boxed{s_2}}{\text{John} \circ (\text{didn't} \circ (\text{read} \circ x)) \vdash \boxed{s_x}}\,[D_2]}{\text{John} \circ (\text{didn't} \circ (\text{read} \circ \text{QP})) \vdash s'_y}\,[q\text{E}]^1}{\text{John} \circ (\text{didn't} \circ (\text{read} \circ \text{QP})) \vdash s_3}\,[D_y]$$

Figure 5.4: Wide and narrow scope negation.

[Not > QP]

$$\dfrac{\mathtt{j} : np \qquad \dfrac{\lambda P.\neg P : (np\backslash s_2)/(np\backslash s_2) \qquad \dfrac{\dfrac{\boxed{Q : q(np, s_x, s'_y)} \qquad (\mathtt{Read}\ x)\ y : s_x}{Q(\lambda x.(\mathtt{Read}\ x)\ y) : \boxed{s'_y}}\,[q\text{E}]^1}{Q(\lambda x.(\mathtt{Read}\ x)\ y) : \boxed{s_2}}\,[D_y]}{\lambda y.Q(\lambda x.(\mathtt{Read}\ x)\ y) : np\backslash s_2}\,[\backslash\text{I}]^2}{\lambda z.\neg Q(\lambda x.(\mathtt{Read}\ x)\ z) : np\backslash s_2}\,[/\text{E}]}{\neg Q(\lambda.x(\mathtt{Read}\ x)\mathtt{j}) : s_2}\,[\backslash\text{E}]}{\neg Q(\lambda x.(\mathtt{Read}\ x)\mathtt{j}) : s_3}\,[D_2]$$

[QP > Not]

$$\dfrac{\dfrac{\boxed{Q : q(np, s_x, s'_y)} \qquad \dfrac{\neg((\mathtt{Read}\ [x : np]^1)\mathtt{j}) : \boxed{s_2}}{\neg((\mathtt{Read}\ x)\mathtt{j}) : \boxed{s_x}}\,[D_2]}{Q(\lambda x.\neg(\mathtt{Read}\ x)\mathtt{j}) : s'_y}\,[q\text{E}]^1}{Q(\lambda x.\neg(\mathtt{Read}x)\mathtt{j}) : s_3}\,[D_y]$$

Figure 5.5: Meaning assembly.

The first derivation in Figures 5.4 and 5.5 gives the *direct* scope reading [Not > QP]: the QP is in the semantic scope of the negation and the latter c-commands the former at the surface structure. The types given in Table 5.1 block this derivation in case the QP is *each book* or *the book* since s'_y is instantiated as s_4 and s_3, respectively and $s_4 : \Box^{\downarrow}\Diamond\Diamond\Box^{\downarrow}s \not\longrightarrow s_2 : s$, and $s_3 : \Box^{\downarrow}\Diamond s \not\longrightarrow s_2 : s$. Therefore, the derivations fail in applying $[D_y]$. On the other hand, the direct scope is derived when any of the other QPs occur, since in all the other cases the s'_y is instantiated with sentential types deriving s_2. Notice that $[D_1]$ will be a correct inference for any type instantiating s_x, since s_1 is the lower type in the patterns we are considering.

The second derivation gives the *inverse* scope reading [QP > Not]: the negation is in the semantic scope of the quantifier, but the latter does not c-command the former at the s-structure. First of all notice, that the relevant point here is the derivation holding between s_2 and s_x —all the sentential levels we are considering derive s_3, hence the inference $[D_y]$ $s'_y \longrightarrow s_3 : \Box^{\downarrow}\Diamond s$, holds for any QP-types. This derivation fails at $[D_2]$ in case the QP we are considering is either *every book* or *each book*—since s_x is instantiated by s_4 and $s_2 : s \not\longrightarrow s_4 : \Box^{\downarrow}\Diamond\Diamond\Box^{\downarrow}s$; while it is derivable when considering the other QPs. We have shown that the derivations in Figure 5.4 correctly account for the data in (13), and we can now move to consider multiple quantifiers contexts sharing the structure [QP [TV QP]].

Comparing the derivations in Figure 5.6 with the tree given in Section 5.2.2 one sees that similar results are obtained: given two quantifiers QP_i, QP_j, QP_i has scope over QP_j [$\mathrm{QP}_j > \mathrm{QP}_i$] iff $s'_i \longrightarrow s_j$ and the s'_j is a grammatical sentential level, $s'_j \longrightarrow s_3$. Again by replacing the variable-types used in the derivations one can easily check that the data in (7) and (10-c) are correctly predicted. Now that we have illustrated how CTL assigns scope to constituents which in the generative grammar undergo movement, we can consider the class of those QPs which take scope locally.

Accordingly to Beghelli and Stowell, CQPs differ from the other QPs since they must take scope in their case position. Moreover, they show an asymmetric behavior when occurring in subject/object position, which motivates the different placements of AgrS-P and AgrO-P in the phrase structure in Figure 5.2.

(15) a. Some student visited few girls. [Some > Few].
b. Every student visited few girls. [Every > Few].

(16) a. Few girls visited some student. [Few > Some], [Some > Few].
b. Few girls visited every student. [Few > Every].

These sentences show that CQPs are unable to take inverse scope. For instance, we cannot construe (15-a) to mean that for few girls it is the case that some student visited her. On the other hand, the structures in (16) with the CQP in subject position do allow for the reading with *few girls* having wide scope.

Before studying the type assignment for those QPs, notice that linguistic reality seems to be more complex than we could express differentiating subject and object types. In [Swa98] de Swart points out that at least in the case of negative polarity items, e.g. *anything*, the difference in the scope possibilities with respect to negation cannot be explained in terms of subject/object asymmetries as shown by the cases

$[\mathrm{QP}_{\mathrm{sub}} > \mathrm{QP}_{\mathrm{obj}}]$

$$\dfrac{\boxed{\mathrm{QP} \vdash q(np, s_x, s'_y)} \qquad \dfrac{\dfrac{\boxed{\mathrm{QP} \vdash q(np, s_u, s'_v)} \qquad \dfrac{[x \vdash np]^2 \quad [y \vdash np]^1 \quad \vdots \quad x \circ (\mathrm{TV} \circ y) \vdash s_1}{x \circ (\mathrm{TV} \circ y) \vdash s_u}\,[D_1]}{x \circ (\mathrm{TV} \circ \mathrm{QP}) \vdash \boxed{s'_v}}\,[q\mathrm{E}]^1}{x \circ (\mathrm{TV} \circ \mathrm{QP}) \vdash \boxed{s_x}}\,[D_v]}{\mathrm{QP} \circ (\mathrm{TV} \circ \mathrm{QP}) \vdash s'_y}\,[q\mathrm{E}]^2$$

$$\dfrac{\mathrm{QP} \circ (\mathrm{TV} \circ \mathrm{QP}) \vdash s'_y}{\mathrm{QP} \circ (\mathrm{TV} \circ \mathrm{QP}) \vdash s_3}\,[D_y]$$

$[\mathrm{QP}_{\mathrm{obj}} > \mathrm{QP}_{\mathrm{sub}}]$

$$\dfrac{\boxed{\mathrm{QP} \vdash q(np, s_u, s'_v)} \qquad \dfrac{\dfrac{\boxed{\mathrm{QP} \vdash q(np, s_x, s'_y)} \qquad \dfrac{[x \vdash np]^2 \quad [y \vdash np]^1 \quad \vdots \quad x \circ (\mathrm{TV} \circ y) \vdash s_1}{x \circ (\mathrm{TV} \circ y) \vdash s_x}\,[D_1]}{\mathrm{QP} \circ (\mathrm{TV} \circ y) \vdash \boxed{s'_y}}\,[q\mathrm{E}]^2}{\mathrm{QP} \circ (\mathrm{TV} \circ y) \vdash \boxed{s_u}}\,[D_y]}{\mathrm{QP} \circ (\mathrm{TV} \circ \mathrm{QP}) \vdash s'_v}\,[q\mathrm{E}]^1$$

$$\dfrac{\mathrm{QP} \circ (\mathrm{TV} \circ \mathrm{QP}) \vdash s'_v}{\mathrm{QP} \circ (\mathrm{TV} \circ \mathrm{QP}) \vdash s_3}\,[D_v]$$

Figure 5.6: Structures with multiple QPs.

below.

(17) a. Phil would not give me anything.
b. *Anything Phil would not give me.

The fact that CQPs cannot have wide scope over expressions preceding them at s-structure can be rephrased in CTL terms, by saying that the type of these QPs should not have the same freedom as the other quantifiers, freedom given by the q-operator. Therefore, a proper type assignment for them can be given by using the classical functional types. The behavior of CQPs is described by $s_2/(np\backslash s_4)$ where the directional implication blocks them to take inverse scope, and the sentential types express their direct scope possibilities summarized in the example below.[5]

(18) a. Few students read each book. [Few > Each].
b. Few students read the books. [*Few > The], [The > Few].
c. A student read few books. [A > Few], [*Few > A].
d. John didn't read few books. [Not > Few].

In order to account for the composition of the verb phrase with these QPs in postverbal position, transitive verbs are assigned a lifted type which enables them to compose with local scoping QPs. Given that $s_2/(np\backslash s_4) \longrightarrow s_2/(np\backslash s_1)$ and $np \longrightarrow s_2/(np\backslash s_1)$ a proper type for transitive verbs is $(np\backslash s_1)/(s_2/(np\backslash s_1))$: they will compose with both CQPs when occurring in a postverbal position and with simple noun phrases. Moreover, since the types assigned to *each_n* and *the_n* don't derive $s_2/(np\backslash s_1)$, the analysis discussed so far is not modified by the introduction of the new type for the verb phrases. The quantifiers *each_n* and *the_n* are the only QPs which might cause ungrammaticality as narrow scope takers.

Exploring the Landscape of QP-types

Now that we have identified the QP-types deriving the behavior of the quantifiers studied in [BS97], we can start exploring the full set of types generated by the basic derivability patterns in Figure 5.3 and see which other QP-(sub)classes they predict to exist. We look back at the types assigned to the subclasses of the GQP group, which is the one more extensively studied by Beghelli and Stowell.

As we have seen, GQPs can be divided in three subclasses distinguishing (i) the QP-*a_n* type which can work as the share distributed phrase for DQPs, but can also have wide scope over them moving to Spec-RefP, (ii) QP-*all_n* type which cannot land into Spec-ShareP, (iii) the QP-*the_n* which must land into Spec-RefP. These three subclasses correspond to the QP-types, (i) $q(np, s_3, s_1)$ (ii) $q(np, s_3, s_2)$, (iii) $q(np, s_3, s_3)$, respectively. From this it follows that the QP-classification we have obtained could

[5] If our understanding of Beghelli and Stowell's analysis is correct, we should also have the sentence *Few students didn't go to the party.* with reading [Few > Not]. However, the type assigned to CQP (p.c. by Anna Szabolsci) does not account for this. A technical solution could be to use the extended derivability patterns given in Figure 5.1, or to re-think about the behavior of CQPs and *each*. We believe this is not effecting the general method we are proposing but rather it raises questions on the (sub-set) relations holding among the QP types.

express a fourth type with s_3 as sentential binder, namely $q(np, s_3, s_4)$. This type can be in the immediate scope of DQPs and CQPs in subject position, since $s_4 \longrightarrow s_4$, and will be ungrammatical in the scope of NQPs, since $s_4 \not\longrightarrow s_2$. In minimalist terms, this would mean that the new GQP can move to Spec-ShareP, but cannot have scope locally in AgrO-P. This behavior is indeed exhibited by the positive polarity item *some_n* as illustrated by the example below.

(19) a. Each student read some book. [Each > Some], [Some > Each].
b. No student read some book. [Some > No].
c. Few students read some book. [Few > Some], [Some > Few].
d. At most five students read some book. [At most 5 > Some], [Some > At most].

The full cases of QP-types with sentential binder s_3 are now exhausted and they express the behavior of the (sub)classes of GQPs. Similarly, one could explore the other classes and search for quantifiers matching the predicted behavior. For instance, the contrast in (20) between *few_n* and *exactly five_n* [SZ97] seems to suggest that CQPs should be further subcategorized.

(20) a. *How did few people think that you behaved?
b. How did exactly five people think that you behaved?

Finally, notice that in Beghelli and Stowell's QP-classification interrogative phrases are considered as carrying a feature which must be checked against the Head-CP by moving to its Spec position. The whole class is referred to as WhQPs and no subclasses are considered. However, wh-phrases exhibit different behavior with respect to weak islands. In particular, while *who* can escape islands formed by negation (21-a), *how* cannot (21-b). We repeat the data below [SZ97].

(21) a. Who didn't Fido see?
b. *How didn't Fido behave?

Again, these data seem to suggest that a family of WhQP-types could be used to represent interrogative quantifier behaviors. We now move on to consider the type classification obtained by extending the derivability patterns used so far. The whole picture given in Chapter 3 offers types which do not derive the sentential type assigned to grammatical sentences s_3. Therefore, our analysis predicts the existence of QPs which require to be in the scope of another scope element returning s_3 (or a lower type) for grammaticality. This is the case of QP-types like $q(np, s_x, (^{\mathbf{0}}s_y)^{\mathbf{0}})$ or $q(np, s_x, {}^{\mathbf{0}}(s_y^{\mathbf{0}}))$, where s_x and s_y stand for any of the sentential types in our derivability patterns.

Natural languages make use of these sorts of quantifiers. A classical example is given by negative polarity items like *anybody* which is ungrammatical in *anybody left* but grammatical if the same structure is a subordinate clause preceded by a proper licensor, e.g. *doubts*: *John doubts that anybody left.* In the next Section, we will explore the whole landscape of polarity items and derive their distribution properties.

Finally, CTL types predict that QPs might behave differently with respect to coordination. In particular, it is predicted that NPIs cannot occur in any of the two

constituents of the conjunction whereas the other QPs can (22) and (23) (see [Ber02] for further details). On the other hand, the standard c-command analysis fails to predict these data as discussed in [Pro00, Hoe00].

(22) a. *No student and any professor came to the party.
b. *Any professor and no student came to the party.
c. *Mary chased nobody and anybody's dog.

(23) a. No student and no professor came to the party.
b. A student and some professor came to the party.
c. Every student and some professor came to the party.
d. Every student and/but no professor came to the party.
e. Mary chased nobody and nobody's dog.

5.3 Classification of Polarity Items

The study of negative polarity items (NPIs) started with the work by Klima [Kli64] who looks at them as expressions which must be 'in construction with' a *trigger* or *licensor*, where the latter is either negation or an "affective element", e.g. a verb like *surprised*. Then, Ladusaw [Lad79] gave a precise semantic interpretation to the vague idea of affective licensors proposed by Klima, identifying them with downward monotone expressions.

Let us look at some data, to clarify the phenomenon. NPIs can be either in the same clause of their trigger, or in an embedded sentence while the trigger is in the matrix sentence (24-c). Moreover, the general claim about the relation of a negative polarity item and its licensor (or trigger) is that the former is licensed by the latter when occurring in its immediate scope [Lin81]. However, there are also harmless interveners, like *think*, which function as a *bridge* between the NPI and its licensor (24-f) [ES73].

(24) a. *Anybody left.
b. John didn't read anything. [Not > Any], [*Any > Not].
c. John doubts anybody left. [Doubt > Any], [*Any > Doubt].
d. *John didn't doubt that anybody left.
e. *John didn't shout that anybody left.
f. John didn't think that anybody left. [Not > Think > Any].

The connection between negation and monotonicity has been deeply studied [KF85, Zwa83] and it turns out that the set of antimorphic functions (AM) —negation-like expressions— is a subset of the set of downward monotone functions (DM). Moreover, it is possible to identify in the set DM the subset of antiaddive functions (AA), satisfing the first De Morgan law and half of the second one. This classification of downward monotone expressions is summarized in Table 5.2 together with the part of the De Morgan's laws they satisfy. Clearly, an inclusion relation holds among the sets of functions

of different negative strength: AM $\subseteq$ AA $\subseteq$ DM[6].

antimorphic	antiadditive	downward monotone
$f(X \cap Y) = f(X) \cup f(Y)$ $f(X \cup Y) = f(X) \cap f(Y)$	$f(X) \cup f(Y) \subseteq f(X \cap Y)$ $f(X \cup Y) = f(X) \cap f(Y)$	$f(X) \cup f(Y) \subseteq f(X \cap Y)$ $f(X \cup Y) \subseteq f(X) \cap f(Y)$
not	nobody, never, nothing	few, seldom, hardly

Table 5.2: Monotone functions classification.

In [Wou94], it is shown that a classification of both Dutch positive and negative polarity items can be given in terms of their sensitivity to (downward) monotonicity properties.

The whole picture is summarized in Table 5.3 taken from [Wou94]. The + and – indicate grammaticality and ungrammaticality, respectively.

Negation	NPIs			PPIs		
	strong	medium	weak	strong	medium	weak
Minimal (DM)	–	–	+	–	+	+
Regular (AA)	–	+	+	–	–	+
Classical (AM)	+	+	+	–	–	–
	mals (tender)	ook maar (anything)	hoeven (need)	allerminst (not-at-all)	een beetje (a bit)	nog (still)

Table 5.3: Polarity items distribution in Dutch.

Table 5.3 can be read as saying that NPIs are licensed, where PPIs are antilicensed by a certain property among the ones characterizing downward monotone functions. From this it follows that a NPI licensed by the property of a function in DM will be grammatical also when composed with any function belonging to a stronger set. On the other hand, if a PPI is 'allergic' to one specific property shared by the functions of a certain set, it will be ungrammatical when composed with them, but compatible with any other function in a weaker set which does not have this property. In the next section we introduce the general method to reach a CTL analysis of licensing and antilicensing relations. (See [Ber02] for more details.)

The classification of licensors has been extended to non-veridical contexts in [Zwa95]. Within this larger frame, we obtain the classification given in Table 5.4 that includes anti-veridical (AV), intentional downward monotone (IDM), intentional upward-monotone (IUM) and upward monotone (UM) functions. In [Gia97], new distinctions among PI have been identifies by means of this new lens as we will summarize below after explaining the general deductive method.

[6]Notice that the table could include also a fourth subset, namely the one characterized by the second De Morgan law and half of the first one (antimultiplicative). However, these functions seem to have no relevant role in the distribution of polarity items in Dutch [Wou94].

Nonveridical				
AV: not p without p neither p nor q	AM: $\subseteq$ not p or not q	AA: $\subseteq$ nobody	DM: $\supseteq$ few p	IDM: impossible p forbid p exclude p
			UM: $\supseteq$ p or q if $\cdot$, p	IUM: will p may p suggest p before p usually p perhaps p

Table 5.4: (Non)veridical contexts.

5.3.1 Licensing Relations in CTL

In this section, we show how the unary operators of $\mathsf{NL}(\Diamond,\cdot^{\mathbf{0}})$ can be used to account for the linguistic typologies presented in the previous section, and clarify their differences and similarities. To express the relation between a trigger and a sensitive item schematically in sequent notation, we use the following convention: $\Delta\lceil NPI_i\rceil$ stands for $\Delta[\mathrm{NPI}_i] \vdash C : \alpha'(\delta')$, where NPI_i is the logical type assigned to the negative polarity item occuring in the whole structure Δ, α' is the lambda term representing it and δ' the lambda term corresponding to the structure on which the negative polarity has wide scope. The * marks ungrammatical compositions.

Let Li_1, Li_2 stand for two functions such that the set of functions represented by Li_1 is included in the one formed by the function represented by L_2: $\mathsf{L_1} \subseteq \mathsf{L_2}$. And let NPI_i stand for a negative polarity item which requires the property enjoyed by Li_i.

(a) $\mathrm{Li}_1 \circ \Delta\lceil NPI_1\rceil$ (c) $\mathrm{Li}_2 \circ \Delta\lceil NPI_2\rceil$
(b) $\mathrm{Li}_1 \circ \Delta\lceil NPI_2\rceil$ (d) $^*\mathrm{Li}_2 \circ \Delta\lceil NPI_1\rceil$

Since the polarity item has scope over the structure it occurs in, it determines the type assigned to it. We represent this fact by assigning npi_i to the whole structure $\Delta\lceil\mathrm{NPI}_i\rceil \vdash npi_i$. The inclusion relation holding among the sets of licensors is expressed by considering the type of Li_1 and Li_2 so that the former derives the latter (but not *vice versa*). Furthermore, since the Li_i can take $\Delta\lceil\mathrm{NPI}_i\rceil \vdash npi_i$ as argument, $\mathrm{Li}_1 : A/npi_1$ and $\mathrm{Li}_2 : A/npi_2$, where $A/npi_1 \longrightarrow A/npi_2$, and hence, from the monotonicity property of the functional operator $/$, $npi_2 \longrightarrow npi_1$. We leave the general formula A on the value of the licensors' type, since it is not relevant for the understanding of the main idea. Summing up, the derivability relation $\longrightarrow$ among the logical types, simply encodes the inclusion relation $\subseteq$ among the sets of expressions of the same semantic type. From this encoding the compositions above derive as follows:

(b)

$$\dfrac{\mathrm{Li}_1 \vdash A/npi_1 \quad \begin{array}{c}\Delta[\mathrm{NPI}_2] \vdash npi_2 \\ \vdots \\ \Delta[\mathrm{NPI}_2] \vdash npi_1\end{array}}{\mathrm{Li}_1 \circ \Delta[\mathrm{NPI}_2] \vdash A}\ [/\mathrm{E}]$$

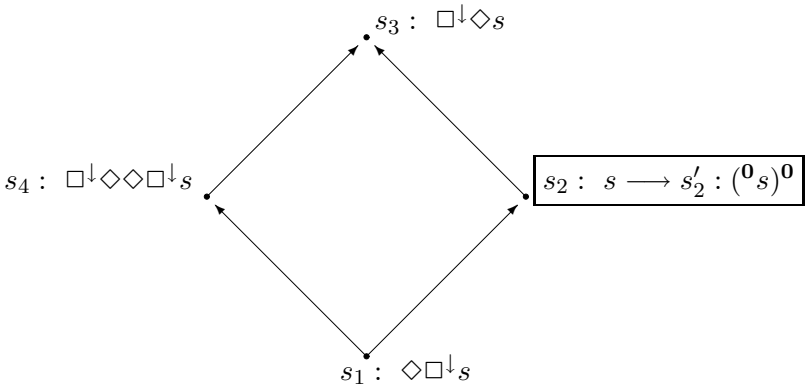
(d)

$$\dfrac{\mathrm{Li}_2 \vdash A/npi_2 \quad \dfrac{\Delta[\mathrm{NPI}_1] \vdash npi_1}{\Delta[\mathrm{NPI}_1] \vdash npi_2}\ *}{{}^{*}\mathrm{Li}_2 \circ \Delta[\mathrm{NPI}_1] \vdash A}\ [/\mathrm{E}]$$

The $*$ marks where the derivation fails. The compositions in (a) and (c) above are obtained simply by functional application. Let us now make things more concrete by means of an example and start exploiting the logical properties of $\mathsf{NL}(\Diamond,\cdot^{\mathbf{0}})$ to model the analysis sketched above.

The QPs *a student*, *few student* and *nobody* all have their denotation in the domain $D_t^{D_{(e,t)}}$. Hence, their semantic type is $((e,t),t)$. However, they differ with respect to monotonicity: *a student* is an upward monotone function, whereas *at most three woman* $\in \mathsf{DM}$ and *nobody* $\in \mathsf{AA}$, where $\mathsf{AA} \subseteq \mathsf{DM}$.

In order to account for NPIs distribution, we need to differentiate these expressions and introduce a sentential level which is ungrammatical. The ungrammatical construction becomes grammatical if a licensor occurs. We will use s_2' to represent this ungrammatical sentential level which stand for $({}^{\mathbf{0}}s)^{\mathbf{0}}$. We repeat below, the derivability relation involving this type. The box in the figure emphasizes the new derivability relation we exploit in this section.



In Section 5.2.2, we have seen that unary operators give us the right expressivity to account for such distinctions. We can consider *nodoby* as $q(np, s_2', s_2)$, and *a student* as $q(np, s_3, s_1)$. The type of *at most three women* has to be derivable from the one of *nobody* encoding the subset relation; we consider it to be of type $q(np, s_1', s_2)$. Hence, npi_2, npi_1 are $s_1' : ({}^{\mathbf{0}}(\Diamond\Box^{\downarrow}s))^{\mathbf{0}}$ and $s_2' : ({}^{\mathbf{0}}s)^{\mathbf{0}}$, respectively.

The derivability relations sketched above follow from the logical properties of the unary operators. These types correctly block a structure containing a strong NPI or a weaker one to compose with the upward monotone expression *a student* (25-a-b), and predict the different behavior between the idiomatic expression *say a word* and the weaker negative polarity item *anybody* with respect to contexts like *at most three women* (26-a-b).

(25) a. *A *student* saw anybody.
b. *A *student* said a word.

(26) a. *At most three women* said anything.
b. **At most three women* said a word.

Here we are in the case of an item in a licensing relation with a function. The inference schema involved is the one given below. Given $g \longrightarrow f$,

$$\frac{\Delta[f] \vdash C}{\Delta[g] \vdash C}.$$

For instance, in the case of *anybody* the type of its direct licensor *at most three women* is derivable from the type of *nobody*, then from (26-a) it follows that *Nobody said anything* is derivable as well. Schematically, given that $AA : q(np, s'_2, s_2) \longrightarrow DM : q(np, s'_1, s_2)$,

$$\frac{\Delta[q(np, s'_1, s_2)] \vdash C}{\Delta[q(np, s'_2, s_2)] \vdash C}.$$

5.3.2 Cross-linguistic differences

The distribution of negative polarity items differs within and across languages. Within a same language, it is possible to reach a classification of negative polarity items based on the property licensing them. On a crosslinguistic level one can obtain natural language typologies based on the licensing relations they satisfy. We look at Dutch, Greek and Italian by way of example and show the differences among them that the CTL analysis shed lights on.

Types for Dutch Negative Polarity Items

A classification of Dutch NPIs can be given based on the strength of the downward monotone functions they require as licensor [Wou94]. Strong negative polarity items (SNPIs) are licensed by functions characterized by the two laws of De Morgan[7], *i.e.* antimorphic functions (AM); their medium relatives (MNPIs) are felicitous also in 'less negative' contexts, being licensed by functions which satisfy the first De Morgan law and half of the second —antiadditive functions (AA); finally, their weaker versions (WNPIs) require half of both laws, hence they are licensed in the scope of all downward monotone functions (DM). The full array of Dutch NPIs is illustrated below. The idiomatic *mals* (tr. 'tender'), the quantifier *ook maar iets* (tr. anything) and the predicate *hoeven* (tr. need) are taken as representative of SNPIs, MNPIs, and WNPIs, respectively. The quantifiers *weinig_n* (tr. few_n) and *niemand* (tr. nobody) represent the DM and AA functions respectively, while *niet* (tr. not) is an antimorphic function. The data are summarized in Table 5.5.

[7]The laws of De Morgan are: 1. $f(X \cup Y) = f(X) \cap f(Y)$, 2. $f(X \cap Y) = f(X) \cup f(Y)$.

(27) a. *Weinig* studenten hoeven hard te studeren. [DM].
Few students need hard to study
Few students need to study hard.

b. *Niemand* hoeft te fietsen. [AA].
Nobody needs to bike.
Nobody has to bike.

c. Hij hoeft *niet* te roepen. [AM].
He needs not to shout
He doesn't need to shout.

(28) a. **Weinig* monniken zullen ook maar iets bereiken. [*DM].
Few monks will anything achieve.
tr. Few monks will achieve something.

b. *Niemand* zal ook maar iets bereiken. [AA].
Nobody will anything achieve.
tr. Nobody will achieve anything.

c. Ik denk *niet* dat er ook maar iemand zal komen. [AM > ook maar]
I think not that anybody will come
tr. I don't think that anybody will come.

(29) a. *Van *weinig* monniken was de kritiek mals. [*DM].
Of few monks was the criticism tender.
tr. The criticism of few monks was tender.

b. *De kritiek van vader abt was *nooit* mals. [*AA].
The criticism of father abbot was never tender.
tr. The criticism of father abbot was never tender.

c. De kritiek zal *niet* mals zijn. [AM].
The criticism will not tender be.
tr. The criticism will be harsh.

	NPIs		
	strong	medium	weak
Positive	–	–	–
Minimal (DM)	–	–	+
Regular (AA)	–	+	+
Classical (AM)	+	+	+
	mals (tender)	ook maar (anything)	hoeven (need)

Table 5.5: Negative polarity items distribution in Dutch.

The analysis of Dutch data shows that the inclusion relation relevant to describe a classification of NPIs is the one holding among antimorphic, antiadditive and downward monotone functions: $\mathsf{AM} \subseteq \mathsf{AA} \subseteq \mathsf{DM}$. The order relation among the sets of NPIs is

$\mathsf{WNPI} \subseteq \mathsf{MNPI} \subseteq \mathsf{SNPI}$, where the inclusion should be read in terms of the demands of the items, e.g. a weak negative polarity item requires a weaker property than a medium one. Applying the type logical method illustrated above this means that in this case we need a linear derivability relation among three types. Let us take, $s'_1 : ({}^{\mathbf{0}}\Diamond\Box^{\downarrow}s)^{\mathbf{0}} \longrightarrow s'_2 : ({}^{\mathbf{0}}s)^{\mathbf{0}} \longrightarrow s'_3 : ({}^{\mathbf{0}}\Box^{\downarrow}\Diamond s)^{\mathbf{0}}$. For the sake of simplicity now we do not take into account the different ways Dutch quantifiers may scope, and assign a uniform output sentential type $s_2 : s$ to all of them; the same holds for the clause negation *niet* (tr. not). For the same reason, we take grammatical sentences to be of type $s_2 : s$.

Lexicon

WNPI: $q(np, s'_1, s'_1)$, *hoeven* (tr. need)	DM: $q(np, s'_1, s_2)$, *weinig* (tr. few);
MNPI: $q(np, s'_2, s'_2)$, *ook maar iets* (tr. anything)	AA: $q(np, s'_2, s_2)$, *niemand* (tr. nobody);
SNPI: $np\backslash s'_3$, *is mals* (tr. is tender)	AM: $(np\backslash s_2)/(np\backslash s'_3)$, *niet* (tr. not).

The full derivability patterns expressing the inclusion relations holding among the sets of downward monotone functions and the ones among the sets of Dutch negative polarity items are as in Figure 5.7.

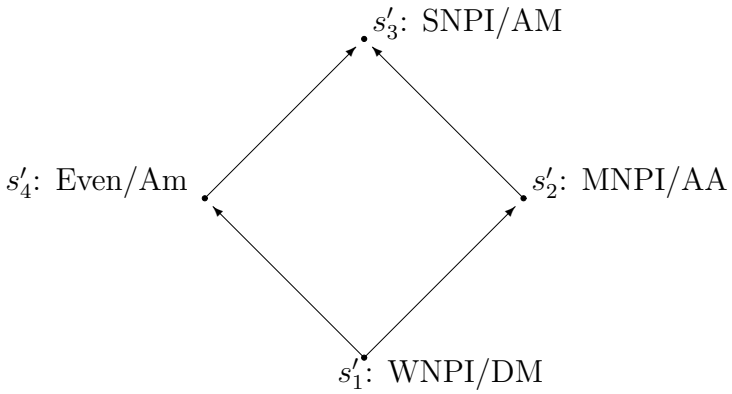


Figure 5.7: Types for Dutch NPIs.

Notice, that though the inclusion relation among the licensors' types seems to be reversed (e.g. DM $\longrightarrow$ AA), this is not the case due to the fact that the sentential types in point occur in a downward monotone position in the QP-types.

Types for Greek Negative Polarity Items

Greek negative polarity items are shown to be in a licensing condition with nonveridicality [Gia97], where intuitively a nonveridical expression (NV) is such that when composed with a proposition p it does not entail the truth of p. Here as well, a classification of negative polarity items can be given based on the inclusion relations holding among nonveridical functions. In particular, the required distinction, inside the set

of nonveridical functions, is among negation-like operators referred to as antiveridical functions (AV), e.g. *dhen* (tr. not), and the intensional ones creating *opaque* contexts [Qui60, Qui61, Tho74][8], e.g. *isos* (tr. perhaps) and *bori* (tr. may). These two groups of functions form subsets of the set of nonveridical functions, but do not exhaust it. Examples of nonveridical contexts which are neither antiveridical nor intensional are questions, or downward monotone functions like *few*.

In [Gia97], it is shown that a classification of Greek polarity items can be given based on their different behavior with respect to nonveridical functions. In particular, one can distinguish (i) 'emphatic negative polarity items' which can only occur in AV contexts and therefore are referred to as NPIs; (ii) 'Idiomatic expressions' (or minimizers (Min)) like *ipe leksi* (tr. say a word) also qualify as NPIs, though they are more flexible as we will comment when discussing their type; (iii) 'Affective polarity items' (APIs), e.g. *kanena* (tr. anybody) which are felicitous in construction with (all) NV contexts; and (iv) free choice items (FCIs) (*i.e.* items with a universal force) which are ungrammatical in AV contexts, and felicitous in nonveridical opaque contexts. Differently from English, Greek employs special words for FCIs, e.g. *opjosdhipote* (tr. anybody) [Gia01]. For our comparative purposes and study of licensing conditions the emphatic polarity items are not interesting since they involve prosodic aspects which interfere with the licensing relation. Therefore, we concentrate on APIs, FCIs, and Min.

(30) a. *Dhen* idha kanenan. [AV].
not saw.perf.1sg any-person.
tr. I didn't see anybody.

b. *Isos* na irthe kanenas. [Opaque].
perhaps subj came.perf.3sg anybody.
tr. Perhaps somebody came.

c. *Pote* ekanes esi tipota ja na me voithisis? [Question].
when did.2sg you anything for subj me help.perf.2sg
tr. Have you ever done anything to help me?

(31) a. **Dhen* idha opjondhipote. [AV].
not saw.perf.1sg FCI-anybody
tr. I didn't see FCI-anybody.

b. *Isos* o Pavlos milise me opjondhipote. [Opaque].
perhaps the Paul subj talked.3sg with anybody-FCI.
tr. *Perhaps Paul talked to anybody.

c. *Aghorases opjodhipote vivlio? [Question].
bought.perf.2sg FCI-any book
tr. Did you buy any book?

[8] *Opaque* contexts are those contexts having different denotations depending on the point of reference (or: situation, possible world, index). They do not satisfy the extensionality principle, where the latter says that given $s = t$ then $\models \phi \leftrightarrow \phi'$ where $\phi = \phi'[t/s]$. The name 'opaque' is meant to distinguish these contexts from the *transparent* ones for which this substitution principle holds [Gam91]. Example of 'opaque contexts' are those formed by modal verbs, habituals, generics, imperatives, intensional verbs, future particle.

(32) a. *Dhen* ipe leksi oli mera. [AV].
not say word all day
tr. He didn't say.perf a word all day.

b. *Bori na pi leksi. [Opaque].
may subj say a word
tr. May say a word.

c. **Pjos* ipe leksi? [Question].
who said.perf.3sg word
tr. Who said a word?

The full picture is summarized in Table 5.6.

	NPI	API	FCI	Min
Veridical	–	–	–	–
Antiveridical	+	+	–	+
Opaque	–	+	+	–
Nonveridical	–	+	–	–

Table 5.6: Negative polarity distribution in Greek.

Note that FCIs must always occur in contexts which provide them alternatives (worlds or situation). This motivates the fact that they are felicitous in opaque contexts, whereas are ungrammatical in veridical and episodic contexts [Gia01]. This point can be clarified by the contrast between (33-a) and (33-b).

(33) a. *Elaxisti fitites ipan otidhipote.
Very few students say.perf FCI-anything.
Few students said nothing.

b. Elaxisti fitites *lene* otidhipote sto mathima.
Very few students say.imp FCI-anything in class
Few students usually say anything in class.

The grammaticality of (b) is due to the use of the imperfective *lene* which gives the habitual interpretation licensing the FCI.

In contrast to the situation in Dutch, the licensors of negative polarity items in Greek are not in a linear order relation. The relevant inclusion relations in this case is among antiveridical functions and nonveridical ones, and among nonveridical intensional functions and the nonveridical ones: $\mathsf{AV} \subseteq \mathsf{NV}$ and $\mathsf{NVI} \subseteq \mathsf{NV}$, where $\mathsf{NVI} \not\subseteq \mathsf{AV}$ and $\mathsf{AV} \not\subseteq \mathsf{NVI}$. This is reflected on the relations among the items licensed by these functions, namely $\mathsf{API} \subseteq \mathsf{FCI}$, and $\mathsf{API} \subseteq \mathsf{Min}$: affective polarity items are felicitous in contexts with less or weaker properties than the others two. Moreover, FCIs are not felicitous in antiveridical contexts, and minimizers are ungrammatical in opaque contexts. This split in the demands of the polarity items is captured by the types: $s'_1 : ({}^{\mathbf{0}}\Diamond\Box^{\downarrow}s)^{\mathbf{0}} \longrightarrow s'_4 : ({}^{\mathbf{0}}\Box^{\downarrow}\Diamond\Diamond\Box^{\downarrow}s)^{\mathbf{0}}$, and $s'_1 : ({}^{\mathbf{0}}\Diamond\Box^{\downarrow}s)^{\mathbf{0}} \longrightarrow s'_2 : ({}^{\mathbf{0}}s)^{\mathbf{0}}$ where $s'_4 : ({}^{\mathbf{0}}\Box^{\downarrow}\Diamond\Diamond\Box^{\downarrow}s)^{\mathbf{0}} \not\longrightarrow s'_2 : ({}^{\mathbf{0}}s)^{\mathbf{0}}$ (Figure 5.1). Again, we do not pay attention to the different ways quantifiers may take

scope in Greek and consider $s_2 : s$ to be the sentential type of grammatical sentences. For the sake of simplicity, we give a simplified type for the intensional functions which does not take into consideration the intensional category.

Lexicon

API: $q(np, s'_1, s'_1)$, *kanenan* (tr. anybody) Min: $np\backslash s'_2$, *ipe leksi* (tr. say a word)
AV: $s_2/(np\backslash s'_2)$, *dhen* (tr. not) NVI: s_2/s'_4, *isos* (tr. perhaps)
FCI: $q(np, s'_4, s'_4)$, *opjosdhipote* (anybody-FCI)

If we look back at the derivability relations holding within the logic, we notice that the split of the types converges in $s'_3 : ({}^{\mathbf{0}}\Box^{\downarrow}\Diamond s)^{\mathbf{0}}$. Therefore, there could be room for a context where all the different items would be grammatical. This prediction is satisfied by the if-clauses as illustrated by the following examples from [Gia97].

(34) a. *An* dhis kanenan, na tu pis na me permeni.
if see.2sg anybody, subj him say.2sg subj me wait.3sg
tr. If you see anybody, tell him to wait for me.

b. *An* pis leksi tha se skotoso.
if say.perf.2sg word will you kill.perf.1sg
tr. If you say a word, I will kill you.

c. *An* kimithis me opjondhipote, tha se skotoso.
if you sleep.2sg with FC-person fut you kill.1sg
If you sleep with FCI-anybody, I'll kill you.

The type for the conditional *an* (tr. if) is $(s_2/s'_3)/s_3$: it can have any kind of negative polarity item in its antecedent. Our type logical analysis of the Greek data is summarized in Figure 5.8.

Negative Polarity Items in Italian

Italian is a negative concord language (NC), *viz.* it allows double negation. In the literature there has been an ongoing debate over the exact classification of its negative constituents. The reason is that they exhibit the behavior of both NPIs and negative quantifiers (NQs). In other words, Italian uses a single negative constituent *nessuno* as both an existential quantifier NPI on par with English *anyone* and as a NQ on par with English *no one*. See [Lad79, Lin81, Lin87, Mug90, Pro94, Gia00], for an approach in which n-words across languages are considered as NPIs since like other NPIs can occur in the polarity environments. The problem of this approach is to explain how negative constituents, like *nessuno*, can also occur outside the traditional polarity environments and yield negative context, thereby behaving like NQ. On the other hand [Zan91, Riz82, Acq92] treat negative constituents of NC languages as negative quantifiers, like English *nobody*. The problem this approach has to solve is to explain why n-words can also occur in NPI environments where they do not yield negative force and are interpreted existentially.

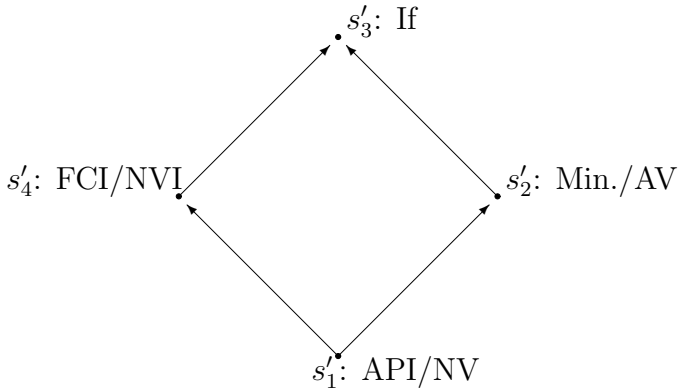


Figure 5.8: Types for Greek NPIs.

We do not enter in this discussion here, since what matter to us is the licensing relation holding among items like *nessuno* when behaving as NPIs and their licensors, *non* (tr. not). In Italian the negative polarity *mai* (tr. ever) shows a different strength than the NPIs *nessuno*, *granché* (tr. all that much) and *mica* (tr. at all). Following the analysis of the Dutch data we can refer to them as WNPI and SNPI, respectively. Furthermore, like in Greek, the quantifier *chiunque* (tr. anybody) as well as the determiner *qualsiasi* (tr. any) express the universal force of FCIs. Similarly to its Greek counterpart, *chiunque* and *qualsiasi* cannot occur in the scope of AV functions, and can be in opaque contexts like the modal *può* (tr. can), but they are ungrammatical in questions. On the other hand, *mai* which is felicitous in questions is ungrammatical in construction with *può*, and in general in opaque contexts.

(35) a. *Non* gioco mai. [AV > WNPI].
Not play ever
tr. I never play.

b. *Non* ho visto nessuno. [AV > SNPI].
Not saw.1sg1 nobody
tr. I didn't seen anybody.

c. **Non* ho visto chiunque. [*AV > FCI].
Not saw.sg1p anybody-FCI.
tr. I didn't seen anybody.

(36) a. *Puoi giocare mai. [*Modal > WNPI].
Can play ever.
tr. You can never play.

b. *Puoi prendere in prestito nessun libro. [*Modal > SNPI].
Can borrow.1pl no book

tr. You cannot borrow any book.

c. Chiunque *può* risolvere questo problema. [Modal > FCI].
Anybody-FCI can solve this problem.
tr. Anybody can solve this problem.

(37) a. *Se* verrai mai a trovarmi, portami Sara. [If > WNPI].
If come.sub1pl ever to visit me, bring me Sara.
tr. If you ever come to visit me, bring me Sara.

b. **Se* vedrai nessuno, torna qui. [*If > MNPI].
If see.sub2sg2 nobody, come back here.
tr. If you don't see anybody, come back here.

c. **Se* vedrai chiunque, torna qui. [*If > FCI].
If see.sub2sg2 anybody-FCI, come back here.
tr. If you see anybody-FCI, come back here.

(38) a. Hai sognato mai la luna? [Question].
Have dreamed.sg2 ever the moon?
tr. Have you ever dreamed the moon?

b. Hai visto nessuno? [Question].
Have saw.2sg nobody?
Have you seen anybody?

c. *Hai visto chiunque? [Question].
Have saw.2sg anybody-FCI?
Have you seen anybody?

	Chiunque (FCI)	Mai (WNPI)	Nessuno (SNPI)
Veridical	–	–	–
AV	–	+	+
Opaque	+	–	–
Conditional	–	+	–
Question	–	+	+

The Italian lexicon items given above are expressed by different types than the ones used for Dutch and Greek. In particular, they require the use of the third sentential level given by $A \longrightarrow {}^{\mathbf{0}}(A^{\mathbf{0}})$. Before looking at the type assignments it is interesting to note that a positive polarity item like *qualcuno* (tr. somebody) refuses to be in the scope of *non*, but is felicitous when composed with the others nonveridical contexts as exemplified below.

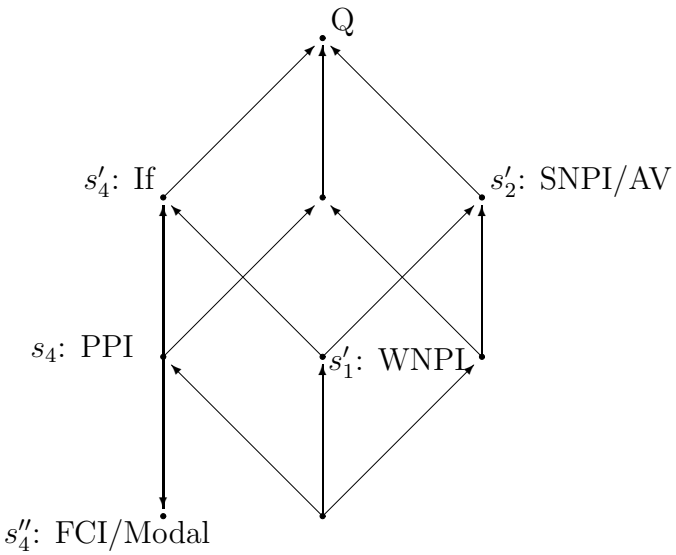
(39) a. **Non* ho visto qualcuno. [*AV > PPI].
Not have seen.sg1 somebody
I didn't see anybody.

b. *Se* vedrai qualcuno, fallo venire qui. [If > PPI].
If see.subjsg2 somebody, let him come here.
If you see anybody, let him come here.

c. Qualcuno di noi *può* risolvere questo problema. [Modal > PPI].
Somebody of us, can solve this problem.
Some of us can solve this problem.

d. Hai visto qualcuno? [Question].
Have you seen somebody.
Did you see anybody?

Again, the types are summarized by labelling the cube of the derivability relations and the lexical assignments are given below using the standard abbreviations.



Lexicon

PPI: $q(np, s_4, s_4)$, *qualcuno*
SNPI: $q(np, s'_2, s'_2)$, *nessuno*
FCI: $q(np, s''_4, s''_4)$, *chiunque*
WNPI: $(np\backslash s_1)\backslash(np\backslash s'_1)$, *mai*
AV: $(np\backslash s_1)/(np\backslash s'_2)$, *non*
If: $(s_1/s'_1)/s'_4$, *se*
Modal: $(((s''_4/np)\backslash s''_4)\backslash s_1)/(np\backslash s''_4)$, *può*

5.3.3 Antilicensing Relations in CTL

In addition to the composition relations we have been studying so far, in natural languages there are expressions which are in an antilicensing relation with some semantic property. In other words, expressions which 'must not' occur in construction with some other items because they are allergic to some of their properties. This relation can hold either between a functional type sensitive to the property of its argument or between an item sensitive to the property of a function. In the first case, the sensitive item cannot have (immediate) scope over its trigger, in the second case the sensitive item cannot be in the (immediate) scope of its trigger. The schema representing the antilicensing relation is given below.

Let Δ be a structure containing a sensitive item in construction with an expression `f` which does not have the property the item is allergic to. Let f be the type of this expression and g the type of an expression `g` having a weaker property than `f`. Given $f \longrightarrow g$

$$\frac{\Delta\{f\} \vdash C}{\Delta\{g\} \vdash C}$$

where $\{\cdot\}$ stands for a negative context (Section 3.3.2). The distribution of English wh-phrases [Sza97] and Dutch positive polarity items are an example of these phenomena. The former are sensitive to the semantic property of the scope element forming a weak island, whereas the latter are sensitive to the semantic property of the function which can take them in its scope. We look at Dutch PPIs by means of example.

Positive Polarity Items in Dutch

Dutch positive polarity items are in an antilicensing relation with downward monotone functions. The following data from [Wou94] illustrate this statement and Table 5.7 summarizes the PPIs' distribution. The triggers are emphasized, whereas the PPIs are underlined.

(40) a. **Weinig* monniken zijn allerminst gelukkig. [*DM > allerminst].
Few monks are not-at-all happy.
tr. Few monks are not-at-all happy.

b. **Niemand* is allerminst gelukkig. [*AA > allerminst].
Nobody is not-at-all happy
tr. Everybody is at least a bit happy.

c. *De schoolmeester is *niet* allerminst gelukkig. [*AM > allerminst].
The teacher is not not-at-all happy
The teacher is quite happy.

(41) a. *Weinig* monniken zijn een beetje gellukkig. [DM > een beetje].
Few monks are a bit happy.
tr. Few monks are a bit happy.

b. %*Niemand* is een beetje gelukkig. [%AA > een beetje].
Nobdy is a bit happy.
tr. Nobody is a bit happy.

c. *De schoolmeester is *niet* een beetje gelukkig. [*AM > een beetje].
The teacher is not a bit happy.
tr. The teacher is happy.

(42) a. *Weinig* kinderen wil nog Donne lezen. [DM > nog].
Few children want still Donne read
tr. Few children still want to read Donne.

b. *Niemand* wil nog Donne lezen. [AA > nog].
Nobody wants still Donne read.
tr. Nobody wants to read Donne anymore.

c. *Jan wil *niet* nog Donne lezen. [*AM > nog].
Jan wants not still Donne read.

tr. Jan does not want to read Donne anymore.

	PPIs		
	strong	medium	weak
Positive	+	+	+
Minimal (DM)	–	+	+
Regular (AA)	–	–	+
Classical (AM)	–	–	–
	allerminst (not-at-all)	een beetje (a bit)	nog (still)

Table 5.7: Positive polarity items distribution in Dutch.

From Section 5.3, we know that the types of the PPIs' triggers are logically related. Moreover, we have seen that the subset relation $\mathsf{AM} \subseteq \mathsf{AA} \subseteq \mathsf{DM}$ is captured by the derivability relation among the types assigned to antimorphic, antiadditive and downward monotone functions. A PPI antilicensed by a certain property is ungrammatical when constructed with the functions having such a property, but is compatible with any functions of a weaker set. In CTL terms this means that the PPIs *reverse* the subset relation holding among monotone functions. This requirement must be expressed in their type logical assignments.

Recall that to account for the licensing relation we have used the downward monotonicity property of the / and \, namely the fact that a function of type A/B ($B\backslash A$) composes with any expression of type $C \longrightarrow B$. Now, notice that a function of type $A/{}^{\mathbf{0}}C$ composes with any expression of type ${}^{\mathbf{0}}B$ such that $C \longrightarrow B$. This is due to the downward monotonicity of the Galois operator ${}^{\mathbf{0}}\cdot$ which reverses the derivability relation among types. We exploit this logical property to obtain the effect required by the antilicensing relation.

Let AM, AA, DM be the types of the functions in the sets AM, AA and DM, where $AM \longrightarrow AA \longrightarrow DM$. A weak PPI is antilicensed by antimorphicity, therefore it can be constructed with any expression in a set equal to or bigger than AA, $B/{}^{\mathbf{0}}AA$. A medium PPI is antilicensed by antiadditivity, therefore it can be in construction with any expression in a set equal to or bigger than DM, $B/{}^{\mathbf{0}}DM$. From these types the following inferences derive.

$$\frac{\text{MPPI} \vdash B/{}^{\mathbf{0}}DM \quad \dfrac{DM \vdash DM}{{}^{\mathbf{0}}DM \vdash {}^{\mathbf{0}}DM}\,[\downarrow \text{Mon}]}{\text{MPPI} \circ {}^{\mathbf{0}}DM \vdash A} \qquad \frac{\text{MPPI} \vdash B/{}^{\mathbf{0}}(DM) \quad \dfrac{AA \vdash AA}{{}^{\mathbf{0}}AA \vdash {}^{\mathbf{0}}DM}\,*}{{}^{*}\text{MPPI} \circ {}^{\mathbf{0}}AA \vdash B}$$

$$\frac{\text{WPPI} \vdash B/{}^{\mathbf{0}}AA \quad \dfrac{AA \vdash AA}{{}^{\mathbf{0}}AA \vdash {}^{\mathbf{0}}AA}\,[\downarrow \text{Mon}]}{\text{WPPI} \circ {}^{\mathbf{0}}AA \vdash A} \qquad \frac{\text{WPPI} \vdash B/{}^{\mathbf{0}}AA \quad \begin{array}{c}\dfrac{DM \vdash DM}{{}^{\mathbf{0}}DM \vdash {}^{\mathbf{0}}DM}\,[\downarrow \text{Mon}] \\ \vdots \\ {}^{\mathbf{0}}DM \vdash {}^{\mathbf{0}}AA\end{array}}{\text{WPPI} \circ {}^{\mathbf{0}}DM \vdash B}$$

Note that the type of WPPIs derives the one of MPPIs. This correspond to an inclusion relation among the corresponding sets: $\mathsf{WPPI} \subseteq \mathsf{MPPI}$. In line with the interpretation

assigned to the order holding among NPIs, this inclusion can be read as holding among sets of expressions allergic to stronger properties. Finally, since the PPIs are sensitive to the property of the functions they are in the scope of, the lambda terms assigned to them have to express this relation. Thus, the term, e.g., of a MPPI is $\lambda P.(P\ \texttt{MPPI})$.

This linguistic application of the Galois operators seems to be promising. However, to reach a better understanding of their use to model linguistic composition two aspects should be further studied. First of all, it is not known yet what would be an appropriate Curry-Howard interpretation for these connectives. Moreover, the antilicensing analysis given here seems to suggest the need of an interaction between the accessibility relation of the binary operators with the one of the Galois connections. We leave these two problems open for further research.

5.4 Key Concepts

In this chapter, we have given a type logical account of the different scope distribution of quantifier phrases and of polarity items (PIs). We have seen that

1. Linguistic theories offer classifications of items based on semantic differences or on the different interactions of syntactic and semantic properties. In particular,

2. Items can deviate in their ways of scope taking, e.g. quantifier phrases.

3. Composition of linguistic signs may be driven by licensing or antilicensing conditions, e.g. negative and positive polarity items with respect to downward monotone functions. The compositional relation is shown to be inherited by expressions in subset or superset relation to the (direct) trigger.

4. Semantic types are seen as sets of expressions. Among such sets there can be an inclusion relations. The grammaticality of constructions involving sensitive items depends on the inclusion relation holding among the triggers.

Based on these linguistic analyses, we have

1. given modally decorated types to account for the different ways QPs take scope and shed light on new classes of QPs not considered in the linguistic theory.

2. captured the inclusion relation among the sets of triggers of PIs in terms of derivability relation among the type assignments achieving a deductive account of licensing relations.

Bibliography

[AB03] C. Areces and R. Bernardi. In situ binding: An hybrid approach. In *Proceedings of ICoS-4*, 2003.

[AB04] C. Areces and R. Bernardi. Analyzing the core of categorial grammar. *Journal of Logic, Language and Information (JoLLI)*, 13(2):121–137, 2004.

[ABM01] C. Areces, R. Bernardi, and M. Moortgat. Galois connections in categorial type logic. In R. Oehrle and L. Moss, editors, *Electronic Notes in Theoretical Computer Science. Proceedings of FGMOL'01*, volume 47. Elsevier Science B.V., 2001.

[Abr90] M. Abrusci. A comparison between Lambek syntactic calculus and intuitionistic linear propositional logic. *Z. Math. Logik Grundlag. Math.*, 36(1):11–15, 1990.

[Abr91] M. Abrusci. Phase semantics and sequent calculus for pure noncommutative classical linear propositional logic. *The Journal of Symbolic Logic*, 56(4):1403–1451, 1991.

[Acq92] P. Acquaviva. The representation of negative 'quantifiers'. *Rivista di Linguistica*, 4:319–381, 1992.

[Ajd35] K. Ajdukiewicz. Die syntaktische konnexität. *Studia Philosophica*, 1:1–27, 1935. (English translation in Storrs McCall (ed.) Polish Logic, 1920-1939. Oxford (1996), 207-231).

[Avr96] A Avron. The method of hypersequents in proof theory of propositional non-classical logics. In W. Hodges, M. Hyland, C. Steinhorn, and J. Truss, editors, *Logic: Foundations to Applications*, pages 1–32. Oxford Science Publications, 1996.

[Bal02] J. Baldridge. *Lexically Specified Derivational Control in Combinatory Categorial Grammar*. PhD thesis, ICCS, University of Edinburgh, 2002.

[Bel82] N D Belnap. Display logic. *Journal of Philosophical Logic*, 11:375–417, 1982.

[Ben84] J. van Benthem. Correspondence theory. In D. Gabbay and F. Günther, editors, *Handbook of Philosophical Logic*, volume II, pages 167–247. Reidel Publishing Company, Dordrecht, 1984.

[Ben87] J. van Benthem. Categorial grammar and lambda calculus. In D. Skordev, editor, *Mathematical Logic and its Applications*, pages 39–60. Plenum, New York, 1987.

[Ben88] J. van Benthem. The Lambek calculus. In E. Bach R. Oehrle and D. Wheeler, editors, *Categorial Grammars and Natural Language Structures*, pages 35–68. Reidel Publishing Company, Dordrecht, 1988.

[Ber02] R. Bernardi. *Reasoning with Polarity in Categorial Type Logic*. PhD thesis, UiL OTS, Utrecht University, 2002.

[BGS60] Y. Bar-Hillel, C. Gaifman, and E. Shamir. On categorial and phrase structure grammars. *The Bulletin of the Research Council of Israel*, 9:1–16, 1960. Reprinted in [BH64].

[BH53] Y. Bar-Hillel. A quasi-arithmetical notation for syntactic description. *Language*, 29:47–58, 1953.

[BH64] Y. Bar-Hillel. *Language and information. Selected essays on their theory and application*. Addison-Wesley Publishing Co., Inc., Reading, Mass.-London, 1964.

[BM03] R. Bernardi and R. Moot. Generalized quantifiers in declarative and interrogative sentences. *Logic Journal of IGPL*, 11(4), July 2003.

[BRV01] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge, 2001.

[BS97] F. Beghelli and T. Stowell. Distributivity and negation: The syntax of each and every. In A. Szabolcsi, editor, *Ways of Scope Taking*, chapter 3, pages 72–107. Kluwer, 1997.

[Bus87] W. Buszkowski. The logic of types. In J. T. Srzednicki, editor, *Initiatives in Logic*. Kijhoff, The Hague, 1987.

[Bus97] W. Buszowski. Mathematical linguistics and proof theory. In J. van Benthem and A. ter Meulen, editors, *Handbook of Logic and Language*, pages 685–736. The MIT Press, Cambridge and Massachusetts, 1997.

[Car99] B. Carpenter. The Turing-completeness of multimodal categorial grammars. In *Papers Presented to Johan van Benthem in honor of his 50th birthday*. 1999. European Summer School in Logic, Language and Information, Utrecht.

[Car03] B. Carpenter. Constraint-based processing. In L. Nadel, editor, *Encyclopedia of Cognitive Science*. Nature Publishing Group, 2003.

[CCCSS00] B. Carpenter, J. Chu-Carroll, R. Sproat, and C. Samuelsson. Computational linguistics. In M. Aronoff and J. Rees-Miller, editors, *Handbook of Linguistics*. Blackwell, 2000.

[CF68] H. B. Curry and R. Feys. *Combinatory Logic. Vol. I.* North-Holland Publishing Co., Amsterdam, 1968. With two selections by William Craig. Second printing. Studies in Logic and the Foundations of Mathematics.

[Cho95] N. Chomsky. *The Minimalist Program*. MIT Press, Cambridge Mass, 1995.

[Cro0X] Tristan Crolard. Subtractive logic. *Theoretical Computer Science*, ?:?–?, 200X. to appear.

[DG01] P. De Groote. Type raising, continuations, and classical logic. In R. van Rooy and M. Stokhof, editors, *Proceedings of the Amsterdam Colloquium 01*, pages 93–101. University van Amsterdam, 2001.

[DG02] Jeremy E. Dawson and Rajeev Goré. Formalised cut admissibility for display logic. In V. A Carreno, C. A. Munoz, and S. Tahar, editors, *TPHOLs02: Proceedings of the 15th International Conference on Theorem Proving in Higher Order Logics*, volume LNCS 2410, pages 131–147. Springer, 2002.

[DG03] Jeremy E. Dawson and Rajeev Goré. A new machine-checked proof of strong-normalisation for display logic. In J Harland, editor, *CATS03: Proceedings of the 8th Australasian Theory Symposium, Adelaide, February 2003*, volume ENTCS, pages ?–? Elsevier, to appear, ? 2003.

[Došš85] K Došen. Sequent-systems for modal logic. *Journal of Symbolic Logic*, 50(1):149–169, 1985.

[Dos92] K. Dosen. A brief survey of frames for the Lambek calculus. *Zeitschrift für mathematischen Logik und Grundlagen der Mathematik*, 38:179–187, 1992.

[Dun91] J. Dunn. Gaggle theory: an abstraction of Galois connections and residuation with applications to negation and various logical operations. In *JELIA 1990: Proceedings of the European Workshop on Logics in Artificial Intelligence*, volume LNCS 478. Springer, 1991.

[Dun93] J M Dunn. Gaggle theory applied to modal, intuitionistic, and relevance logics. In I Max and W Stelzner, editors, *Logik und Mathematik: Frege-Kolloquium Jena*, pages 335–368. de Gruyter, 1993.

[Eij85] J. van Eijck. *Aspects of Quantification in Natural Language*. PhD thesis, University of Groningen, 1985.

[ES73] N. Erteschik-Shir. *On the Nature of Island Constraints*. PhD thesis, MIT, 1973.

[Fre84] G. Frege. *Die Grundlagen der arithmetik. Eine logisch-mathematische Untersuchung über den Begriff der Zahl*. Köbner, Breslau, 1984. Reprint (1961) by Georg Olms, Hildesheim.

[Fre87] G. Frege. Begriffsschrift. In *Methods for research in logic (Russian)*, pages 83–151. "Metsniereba", Tbilisi, 1987.

[Gam91] Gamut. *Logic, Language and Meaning*, volume 2. The University of Chicago Press, Chicago and London, 1991.

[Gen38] G. Gentzen. Untersuchungen über das logische Schliessen. *Mathamtische Zeitschrift*, 39:176–210,305–431, 1938. English translation in [Sza69].

[Gia97] A. Giannakidou. *The Landscape of Polarity Items*. PhD thesis, University Groningen, 1997.

[Gia00] A. Giannakidou. Negative ... concord?*. *Natural Language and Linguistic Theory*, 18:457–523, 2000.

[Gia01] A. Giannakidou. The meaning of free choice. *Linguistics and Philosophy*, 2001. In printing.

[Gir87] J-Y Girard. *Proof Theory and Logical Complexity*. Bibliopolis, Naples, 1987.

[Gir93] J-Y Girard. On the unity of logic. *Annals of Pure and Applied Logic*, 59:201–217, 1993.

[GLT89] J-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*, volume 7 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1989.

[Gor97] R Goré. Cut-free display calculi for relation algebras. In D van Dalen and M Bezem, editors, *CSL96: Selected Papers of the Annual Conference of the European Association for Computer Science Logic*, volume LNCS 1258, pages 198–210. Springer, 1997. `http://arp.anu.edu.au/~rpg`.

[Gor98a] R. Goré. Gaggles, Gentzen and Galois: How to display your favorite substructural logic. *Logic Journal of the IGPL. Interest Group in Pure and Applied Logics*, 6(5):669–694, 1998.

[Gor98b] R Goré. Gaggles, Gentzen and Galois: How to display your favourite substructural logic. *Logic Journal of the Interest Group in Pure and Applied Logic*, 6(5):669–694, 1998. `http://arp.anu.edu.au/~rpg`.

[Gor98c] R. Goré. Substructural logics on display. *Logic Journal of the IGPL. Interest Group in Pure and Applied Logics*, 6(3):451–504, 1998.

[Gor98d] R Goré. Substructural logics on display. *Logic Journal of the Interest Group in Pure and Applied Logic*, 6(3):451–504, 1998. `http://arp.anu.edu.au/~rpg`.

[Hen93] H. Hendriks. *Studied Flexibility. Categories and Types in Syntax and Semantics*. PhD thesis, ILLC, University of Amsterdam, 1993.

[Hey99] D. Heylen. *Types and Sorts. Resource Logic for Feature Checking*. PhD thesis, UiL OTS, University of Utrecht, 1999.

[Hoe00] J. Hoeksema. Negative polarity items: tirggering, scope, and c-command. In L. R. Horn and Y. Kato, editors, *Negation and Polarity*, pages 115–144. Oxford University Press, 2000.

[How80] W. A. Howard. The formulae-as-types notion of construction. In *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 480–490. Academic Press, London, 1980.

[Jac77] R. Jackendoff. *X' Syntac: A study of Phrase Structure*. MIT Press, Cambridge, Mass., 1977.

[Jan97] T. Janssen. Compositionality. In J. van Benthem and A. ter Meulen, editors, *Handbook of Logic and Language*, pages 417–473. The MIT Press, Cambridge and Massachusetts, 1997.

[KF85] E. Keenan and L. Faltz. *Boolean Semantics for Natural Language*. Reidel, Dordrecht, 1985.

[Kli64] E. Klima. Negation in English. In J.A. Fodor and J.J. Katz, editors, *The Structure of Language*, pages 246–323. Prentice Hall, New Jersey, 1964.

[KM95] N. Kurtonina and M. Moortgat. Structural control. In P. Blackburn and M. de Rijke, editors, *Logic, Structures and Syntax*. Kluwer, Dordrecht, 1995.

[Kra96] M Kracht. Power and weakness of the modal display calculus. In H Wansing, editor, *Proof Theory of Modal Logics*, pages 92–121. Kluwer, 1996.

[Kur95] N. Kurtonina. *Frames and Labels. A Modal Analysis of Categorial Inference*. PhD thesis, OTS, University of Utrecht and ILLC, University of Amsterdam, 1995.

[Lad79] W. Ladusaw. *Polarity Sensitivity as Inherent Scope Relations*. PhD thesis, University of Texas at Austin, 1979.

[Lam58] J. Lambek. The mathematics of sentence structure. *American Mathematical Monthly*, 65:154–170, 1958.

[Lam61] J. Lambek. On the calculus of syntactic types. In R. Jakobson, editor, *Structure of Languages and its Mathematical Aspects*, pages 166–178. American Mathematical Society, 1961.

[Lam88] J. Lambek. Categorial and categorical grammar. In R. Oehrle, E. Bach, and D. Wheeler, editors, *Categorial Grammars and Natural Language Structures*, pages 297–317. Reidel Publishing Company, Dordrecht, 1988.

[Lam93] J. Lambek. From categorial to bilinear logic. In K. Došen P. and Schröder-Heister, editors, *Substructural Logics*. Oxford University Press, 1993.

[Lam99] J. Lambek. Type grammar revisited. In *Logical Aspects of Computational Linguistics*, pages 1–27. Springer, Berlin, 1999.

[Lam01] J. Lambek. Type grammars as pregroup. *Grammars*, 4, 2001.

[Les29] S. Lesniewski. Grundzüge eines neues Systems der Grundlagen der Mathematik. *Fundamenta Mathematicae*, 14, 1929.

[Lin81] M. C. Linebarger. *The Grammar of Negative Polarity*. PhD thesis, The Indiana University Linguistics Club, 1981.

[Lin87] M. Linebarger. Negative polarity and grammatical representation. *Linguistics and Philosophy*, 10:325–387, 1987.

[Mas92] A Masini. 2-sequent calculus: A proof theory of modalities. *Annals of Pure and Applied Logic*, 58:229–246, 1992.

[May77] R. May. *The Grammar of Quantification*. PhD thesis, MIT, 1977.

[Mit03] R. Mitkov, editor. *The Oxford Handbook of Computational Linguistics*. Oxford University Press, 2003.

[Moo91] M. Moortgat. Generalized quantification and discontinuous type constructors. In W. Sijtsma and A. van Horck, editors, *Discontinuous Constituency*. De Gruyter, 1991.

[Moo96a] M. Moortgat. In situ binding: A modal analysis. In P. Dekker and M. Stokhof, editors, *Proceedings of the 10th Amsterdam Colloqium*, pages 539–549, Amsterdam, 1996. ILLC.

[Moo96b] M. Moortgat. Multimodal linguistic inference. *Journal of Logic, Language and Information*, 5(3,4):349–385, 1996.

[Moo97] M. Moortgat. Categorial type logics. In J. van Benthem and A. ter Meulen, editors, *Handbook of Logic and Language*, pages 93–178. The MIT Press, Cambridge and Massachusetts, 1997.

[Moo99] M. Moortgat. Constants of grammatical reasoning. In *Proceedings KNAW Conference Interface Strategies*, 1999.

[Moo02] R. Moot. *Proof Nets for Linguistic Analysis*. PhD thesis, UiL OTS, University of Utrecht, 2002.

[Moo04] M. Moortgat. Dependency structures. In *Proceedings of "Categorial Grammars. An Efficient tool for NLP"*, pages 92–95, 2004.

[Mug90] I. Laka Mugarza. *Negation in Syntax: On the Nature of Functional Categories and Projections*. PhD thesis, Massachusetts Institute of Technology, 1990.

[Oeh99] R. Oehrle. Multimodal type logical grammar. In Borsley and Borjars, editors, *Non-Trasfornational Syntax*. Blackwell, 1999.

[Ore44] O. Ore. Galois connections. *Transactions of the American Mathematical Society*, 55:493–513, 1944.

[Pen93] M. Pentus. Lambek grammars are context free. In *Proceedings of the 8th Annual IEEE Symposium on Logic in Computer Science*, Los Alamitos, California, 1993. IEEE Computer Society Press.

[Pen95] M. Pentus. Models for the Lambek calculus. *Annals of Pure and Applied Logic*, 75(1–2):179–213, 1995.

[Per00] F. Pereira. Formal grammar and information theory: Together again? *Philosophical Transactions of the Royal Society*, 358(1769):1239–1253, April 2000.

[PMW90] B. Partee, A. ter Meulen, and R. Wallen. *Mathematical Methods in Linguistics*. Kluwer, Dordrecht, 1990.

[Pra65] D. Prawitz. *Natural Deduction: A Proof Theoretical Study*. Almqvist and Wiksell, Stockholm, 1965.

[Pri67] A. Prior. *Past, Present and Future*. Clarendon Press, Oxford, 1967.

[Pro94] L. Progovac. *Negative and Positive Polarity: A Binding Approach*. Cambridge University Press, Cambridge, 1994.

[Pro00] L. Progovac. Coordination, c-command, and 'logophoric' n-words. In L. R. Horn and Y. Kato, editors, *Negation and Polarity*, pages 88–114. Oxford University Press, 2000.

[Qui60] W. Quine. *Word and Object*. Cambridge University Press, Cambridge, 1960.

[Qui61] W. Quine. *From a Logical Point of View*. MIT Press, Cambridge, Mass., 1961.

[Rei97] T. Reinhart. Quantifier scope: How labor is divided between QR and choice functions. *Linguistic and Philosophy*, 20(4):335–397, 1997.

[Res00] G. Restall. *An Introduction to Substructural Logics*. Routledge, 2000.

[Riz82] L. Rizzi. *Issues in Italian Syntax*. Foris, Dordrecht, 1982.

[Rou97] W. Rounds. Feature logics. In J. van Benthem and A. ter Meulen, editors, *Handbook of Logic, Language and Information*. The MIT Press, 1997.

[Sta97] E. Stabler. Derivational minimalism. In C. Retoré, editor, *Logical Aspects of Computational Linguistics*, pages 68–95, NY, 1997. Springer-Verlag (Lectures Notes in Computer Science).

[Ste00] M. Steedman. *The Syntactic Process*. Cambridge, MA: MIT Press, 2000.

[Swa98] H. de Swart. Negation, polarity and inverse scope. *Lingua*, 105(3):175–200, 1998.

[SZ97] A. Szabolcsi and F. Zwarts. Weak islands and an algebraic semantics for scope taking. In A. Szabolsci, editor, *Ways of Scope Taking*, chapter 7, pages 217–262. Kluwer, 1997.

[Sza69] M. E. Szabo, editor. *The Collected Papers of Gerhard Gentzen*. North-Holland, 1969.

[Sza97] A. Szabolsci. Strategies for scope taking. In A. Szabolsci, editor, *Ways of Scope Taking*, chapter 4, pages 109–154. Kluwer, 1997.

[Tho74] R. Thomason, editor. *Formal Philosophy: Selected papers of Richard Montague*. Yale University Press, New Haven, 1974.

[TS96] A Troelstra and H Schwichtenberg. *Basic Proof Theory*. Number 43 in Cambridge Tracts In Theoretical Computer Science. Cambridge University Press, 1996.

[vBtM97] J. van Benthem and A. ter Meulen, editors. *Handbook of Logic, Language and Information*. The MIT Press, 1997.

[Ver99] W. Vermaat. The minimalist move operation in a deductive perspective. *Language and Computation*, 1999. To appear.

[VW90] K. Vijay-Shanker and D. Weir. Polynomial time parsing of CCG. In *Proceedings ACL*, pages 1–8, 1990.

[Wan92] H. Wansing. Formulas-as-types for a hierarchy of sublogics of intuitionistic propositional logic. In D. Pearce and H. Wansing, editors, *Non-Classical Logics and Information Processing*. Springer Lecture Notes in AI 619, Berlin, 1992.

[Wou94] T. van der Wouden. *Negative Contexts*. PhD thesis, University of Groningen, 1994.

[Zan91] R. Zanuttini. *Syntactic Properties of Sentential Negation: A Comparative Study of Romance Language*. PhD thesis, University of Pennsylvania, Philadelphia, 1991.

[Zwa83] F. Zwarts. Three types of polarity. In F. Hamm and E. Hinrichs, editors, *Plural Quantification*, pages 177–238. Klwuer, Dordrecht, 1983.

[Zwa95] F. Zwarts. Nonveridical contexts. *Linguistic Analysis*, 25:286–312, 1995.