Reasoning with Polarity in Categorial Type Logic

Redeneren met Polariteiten in Categoriale Typenlogica
(met een samenvatting in het Nederlands)

Uso delle Polarità nella Grammatica Categoriale
(con un riassunto in italiano)

Proefschrift

ter verkrijging van de graad van doctor aan de
Universiteit Utrecht,
op gezag van de Rector Magnificus
prof.dr. W.H. Gispen
ingevolge het besluit van het College voor Promoties
in het openbaar te verdedigen
op woensdag 19 juni 2002 des middags te 14.30 uur

door

Raffaella Anna Bernardi

geboren op 9 augustus 1971 te Pescara, Italië.

Promotor: Prof.dr. M.J. Moortgat
Utrecht Institute of Lingusitics OTS
Utrecht University

Printed and bound by Print Partners Ipskamp.

ISBN: 90-393-3070-0

Contents

Acknowledgments

Coming to The Netherlands has been a great event in my life. The five year immersion in Dutch culture represents a turning point in my personal and academic development. During these years, almost all the dreams I had as a teenager came true. My biggest desires were to get to know people and different cultures, to write a book and to get a lot of diamonds. The Dutch University is certainly a good place to find all these things. It is extremely international, the quality and quantity of publications is very high, and the diamonds are all over —though, in my dream the latter were somehow different. I would like to thank Claudia Casadio for her suggestion to come here, and for introducing me to Michael Moortgat.

Michael has been the perfect supervisor. Through all these years, he had always time to meet, to answer tons of my emails, to listen to my craziest ideas, to spot possible lines of research in my mess of notes. His supervision has been really inspiring on all sorts of subjects. His suggestions went from how to write a good paper to how to cook a good Italian dish; from how to give a good talk to where to go on vacation.

I am grateful to the members of the committee, Michele Abrusci, Johan van Benthem, Jan van Eijck, Henriëtte de Swart and Albert Visser for finding the time to read my thesis.

I would like to thank Jim Lambek for his willingness to be present at my defense, and Dick Oehrle for his constant support of my on-going research through all these years.

I am deeply in debt to Paolo Fregulia for his long-distance supervision on the decisions I had to take in my academic carrier and for introducing me to the logic group in Siena before my Dutch period started. I thank Fabio Bellisima, Duccio Luchi, Franco Montagna, Paolo Pagli and Andrea Sorbi, for the three month visit at their Department. They were supportive and friendly, and they showed me the beauty of being in a research group.

Finally, among the Italian scholars I am thankful to Corrado Bologna for pushing me to study abroad, and to Mario Piazza for keeping me in contact with the Italian University.

Special thanks goes to Carlos Areces for his friendship. The five years without the Argentinian warmth would have been much harder. One of the great things Carlos does with his friends is writing papers, reading their abstracts, notes and articles. Well, he

did all of this with me and for me. I was very lucky to meet him.

During these years, I had the privilege of having two offices, one in Utrecht (at the OTS) and one in Amsterdam (at the ILLC). This doubled the number of colleagues, but even better the number of friends. I am grateful to Dick de Jongh, Ria Rettob, Maarten de Rijke and Marjan Veldhuisen for this.

Dick also played a crucial role in the achievement of my desire of filling my life with diamonds. I am grateful to him for the personalized intensional course he gave me, and for the time he spent correcting my exercises.

Maarten has brought light in a year of darkness. He allowed me into his stimulating research group where everybody would feel at home. He pushed me to broad my mind and to have an overview of the field —I have not succeeded in this yet, but I will keep on trying! Maarten taught me how to be a public relations officer by supervising my work for FoLLI. He had always time to squeeze me into his agenda and he never gave up pushing me to think of my future and showing the paths I could follow.

Herman Hendriks and Yoad Winter deserve special thanks for helping me solve a puzzle which kept me awake at night. It was quite amazing to see how fast they can spot problems and solve them.

I also thank Anastasia Giannakidou, Rajeev Goré, Greg Restall, Victor Sánchez-Valencia and Ton van der Wouden for always being ready to answer my questions about their works and give comments about mine.

The list of people who put efforts in making this thesis readable is extremely long. Greg Kobele, Breanndán Ó Nualláin, and Alistair Butler cleansed the thesis of my strong Italian accent, and gave valuable comments on its content. Balder ten Cate, Evangelia Vlachou, Rosella Gennari, Marco Hollenberg, Gerhard Jäger, Paola Monachesi, Richard Moot, Øystein Nilsen, Willemijn Vermaat and Marco Vervoort read several times the several drafts I sent around. I am thankful to them for helping me in writing this book.

Giving talks and receiving feedback from the audience always turned out to be stimulating and helpful for my research. I am grateful to INRIA and the Universities of Amsterdam, Bologna, Dublin, Roma III and the Saarland for giving me the possibility of presenting my work to their students and researchers.

I acknowledge the University of Chieti for the two year grant which helped me start this all; the COFIN 1999-2000 Project directed by Marica De Vincenzi for the interest in my research and financially contribute to it; The Utrecht Institute of Linguistics for providing extra financial support which was not planned originally; the SOCRATES exchange programme and the Department of Philosophy (University of Chieti) for financially supporting my seminars at the University of Chieti, and the students there for making me feeling good learning and teaching with them.

The joy I received from working on this thesis would have been meaningless without my relatives and friends. I will thank them personally. Here, instead, I thank Antonio Meucci for inventing the telephone which allowed me to be present in my family's every day life, and the inventor of the email, Ray Tomlinson, for let me stay in contact with my friends.

Finally, last but not least, I am grateful to the Dutch weather for showing me the beauty of working hard!

Raffaella Bernardi

Abstract

The research presented in this thesis 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 the possibilities it offers for analyzing natural language structures and their interpretation.

The thesis is divided into three parts. Each of them has an introductory chapter. In Chapter 1, we introduce the background assumptions of the categorial approach in linguistics, and we sketch the developments that have led to the introduction of CTL. We discuss the motivation for using logical methods in linguistic analysis. In Chapter 3, we propose our view on the use of unary modalities as 'logical features'. In Chapter 5, we set up a general notion of grammatical composition taking into account the form and the meaning dimensions of linguistic expressions. We develop a logical theory of *licensing* and *antilicensing* relations that cross-cuts the form and meaning dimensions.

Throughout the thesis we focus attention on *polarity*. This term refers both to the polarity of the logical operators of CTL and to the polarity items one finds in natural language, which, furthermore, are closely connected to natural reasoning. Therefore, the title of this thesis *Reasoning with Polarity in Categorial Type Logic* is intended to express three meanings.

Firstly, we reason with the *polarity of the logical operators* of CTL and study their derivability patterns. In Chapter 2, we explore the algebraic principles that govern the behavior of the type-forming operations of the Lambek calculus. We extend the categorial vocabulary with downward entailing unary operations obtaining the full toolkit that we use in the rest of the thesis. We employ unary operators to encode and compute monotonicity information (Chapter 4), to account for the different ways of scope taking of generalized quantifiers (Chapter 6), and to model licensing and antilicensing relations (Chapter 7).

Secondly, in Chapter 4, we model *natural reasoning* inferences drawn from structures suitable for negative polarity item occurrences. In particular, we describe a system of inference based on CTL. By decorating functional types with unary operators we encode the semantic distinction between upward and downward monotone functions. Moreover, we study the advantages of this encoding by exploring the contribution of

monotone functions to the study of natural reasoning and to the analysis of the syntactic distribution of negative polarity items.

Thirdly, in Chapter 7, we study the distribution of *polarity-sensitive* expressions. We show how our theory of licensing and antilicensing relations successfully differentiates between negative polarity items, which are 'attracted' by their triggers, and positive polarity items, which are 'repelled' by them. We investigate these compatibility and incompatibility relations from a cross-linguistic perspective, and show how we reduce distributional differences between polarity-sensitive items in Dutch, Greek and Italian to differences in the lexical type assignments of these languages.

Part I

Categorial Type Logics

In this thesis, we use the tools of Categorial Type Logic (CTL) [Moo97] to study natural language syntax and semantics, and the reasoning patterns that arise from the process of composing grammatical structure. The primary explanatory devices in categorial grammar are the type-forming operations, which build structured complex categories out of a small set of basic types. 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. An important aspect of the CTL theory of type-forming operations is the distinction between the mathematical core meaning of these operations, as given by the laws of opposites just described, and structural extensions of this core meaning. The mathematical core captures *invariants* of the composition of natural language form and meaning. The combination of the core with its structural extensions makes it possible to study *variation* in the way natural languages express the relation between patterns of form and meaning.

In Chapter 1, we introduce the background assumptions of the categorial approach in linguistics, and we sketch the developments that have led to the introduction of CTL. We discuss the motivation for using logical methods in linguistic analysis.

In Chapter 2, we study the algebraic principles that govern the behavior of the type-forming operations. The principle of *residuation* provides the basic law of opposites for the unary and binary type-forming operations that have been studied so far. We show that the mathematical structure of the CTL base component naturally accommodates a closely related algebraic principle that gives rise to *Galois connected* pairs of type-logical operations. We extend the categorial vocabulary with these operations, thus obtaining the full tool-kit that will be used in the rest of the thesis.

Chapter 1

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. Readers familiar with the categorial approach to natural language syntax and semantics can skip this chapter and go directly to Chapter 2.

The present chapter is organized as follows. We start by introducing classical and combinatory categorial grammars, two formalisms closely related to CTL (Section 1.1). Then, by highlighting the differences between these frameworks and the logical approach assumed in this thesis, we introduce the main aspects of CTL (Section 1.2). Moreover, we discuss the proof theoretical perspective on form-meaning assembly of linguistic expressions.

1.1 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 the 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.

1.1.1 Classical Categorial Grammar

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

DEFINITION 1.1. [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 Jim Lambek (Section 1.2), 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 1.2. [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:

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

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 1.3. 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 1.3. [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 1.4. [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 $\in s$? $\quad\rightsquigarrow\quad$ $np, (np\backslash s)/np, np/n, n/n, n \Rightarrow s$?

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

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

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

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.

1.1.2 Combinatory Categorial Grammar

For an excellent exposition of Combinatory Categorial Grammar (CCG), we refer the reader to [Ste00][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 1.5. [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 1.4 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 rely on the combinator [$\mathtt{B}_\times$] as illustrated below.

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

EXAMPLE 1.6. [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]. We present the derivations of (5-b) in (i) and (ii). In the discussion of meaning assembly in Section 1.3, 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? These questions will be addressed in the next two sections.

1.2 A Logic of Types: Lambek 1958

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 1.1. 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.

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

Table 1.1: Rules-based approach vs. logical approach.

1.2.1 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 1.7. [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 1.8. [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 1.9. [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 1.10. [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 1.11. [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.

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

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 Mati Pentus [Pen93] that L grammars are context free provides the formal underpinnings for this claim.

1.2.2 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, because it is obtained from L by dropping associativity.

Definition 1.12. [The Lambek Family]. **Logical rules** for the base logic NL:

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

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

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

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.

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

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, 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 $(\backslash_i, /_i)$ and $\circ_i$, 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 $\Gamma[\Delta \circ_2 \Delta'] \vdash A$;

ii. Interaction structural rules which mix distinct modes.

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 $\circ_a$.

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

Example 1.13. [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.

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

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 $[\mathsf{B}_\times]$. 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. We will come back to this point in Chapter 3 after we have explored the algebraic structure of NL.

The example above illustrate 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 1.14. [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).

The logical and structural modules of CTL have been used to account for the constants of grammatical reasoning and the structural variations, respectively. In Chapter 2, we show how NL is interpreted by a *universal* algebraic structure which can be restricted so to capture the variations expressed by the other CTLs obtained from NL by adding structural rules.

In this thesis, attention is focused on the logical module. To this end, in Chapter 2 we investigate the algebraic structure of NL and highlight other logical properties which have not been investigated so far. When we make use of structural rules (Chapter 4), we apply them to carry semantic information which are universally shared. On the other hand, when we assume a crosslinguistic perspective, (Chapter 7), the differences across languages are reduced to different lexical type assignments exploiting the expressivity of the logical module.

1.2.3 Structural Constraints

It will be clear from the above that 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 1.15. [Non-Expanding Structural Rules] Given an antecedent configuration Σ, the length of Σ is defined as follows:

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

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

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

1.3 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 1.16. [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 $m(f(a_1, \ldots, a_n)) = m'(f)(m(a_1), \ldots, m(a_n))$.

1.3.1 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 1.17. [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 1.18. [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 1.19. [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 1.20. [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 1.21. [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 1.22. [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 1.23. [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 1.24. [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 1.25. 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 $(\mathtt{type}(B), \mathtt{type}(A))$ and are interpreted in the domain $Dom_{(b,a)}$.

1.3.2 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 $\mathtt{lori}$. 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 1.26. [Term Labelled Lexicon] Given a set of basic expressions of a natural language Σ, a term labeled categorial lexicon is a relation,

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

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 1.27. [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 1.28. [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.[4]

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 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 1.24 this term is denoted by a function g such that for each entity $d \in Dom_e$, gives $g(d) = 1$ iff $[\![\texttt{walk}(x)]\!]_{\mathcal{M}}^{f[x:=d]} = 1$, 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.

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

Example 1.29. [Logical Constants] By evaluating the lambda expressions in Example 1.27 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 6.

Taking advantage of the fact that the denotation of all natural language expressions can be reduced to sets, we can extend our model with a partial order recursively defined again by means of types [GS84, Ben86].

Definition 1.30. [Partially Ordered Domains] Let $\mathcal{M} = \langle D, \leq, I \rangle$ be our model, where $\leq$ is defined recursively as follows

$$\begin{array}{lllll} \text{If } \beta, \gamma \in Dom_e, \text{ then} & [\![\beta]\!] \leq_e [\![\gamma]\!] & \text{iff} & [\![\beta]\!] = [\![\gamma]\!] \\ \text{If } \beta, \gamma \in Dom_t, \text{ then} & [\![\beta]\!] \leq_t [\![\gamma]\!] & \text{iff} & [\![\beta]\!] = 0 \text{ or } [\![\gamma]\!] = 1 \\ \text{If } \beta, \gamma \in Dom_{(a,b)}, \text{ then} & [\![\beta]\!] \leq_{(a,b)} [\![\gamma]\!] & \text{iff} & \forall \alpha \in Dom_a, [\![\beta(\alpha)]\!] \leq_b [\![\gamma(\alpha)]\!]. \end{array}$$

Let us look at our toy-model again and check the order relations holding among its expressions. Establishing such an order is quite immediate when working with sets, and only appears more complex when using the recursive Definition 1.30.

Example 1.31. [Order Relations] The set denoting the expression *tall student*, obtained by taking all elements which are in both the sets [![tall]!] and [![student]!], *viz.* [![tall student]!] = {lori}, is clearly a subset of the set denoting *student*, *viz.* [![student]!] = {lori, ale, sara}. Using functional denotation the proof of [![tall student]!] $\leq_{(e,t)}$ [![student]!] is as follows.

$$\begin{array}{ll} [\![\text{tall student}]\!] \leq_{(e,t)} [\![\text{student}]\!] & \text{iff } \forall \alpha \in D_e \\ [\![\text{tall student}(\alpha)]\!] \leq_t [\![\text{student}(\alpha)]\!] & \text{iff} \\ [\![\text{tall student}]\!]([\![\alpha]\!]) \leq_t [\![\text{student}]\!]([\![\alpha]\!]) & \text{iff} \\ [\![\text{tall student}]\!]([\![\alpha]\!]) = 0 \text{ or } [\![\text{student}]\!]([\![\alpha]\!]) = 1. & \end{array}$$

Assume this is not true, *viz.* [![tall student]!]([![α]!]) = 1 and [![student]!]([![α]!]) = 0. Then $\exists d \in Dom_e$ s.t. $d \in$ [![tall student]!], but $d \notin$ [![student]!], which is obviously impossible. Note, that the inclusion relation is due to the presence of the intersective predicate *tall*.

Finally, having a formal definition of the domains of interpretation and a partial order over them, one can distinguish expressions interpreted in the same domain but which differ with respect to the partial order. For instance, we can distinguish upward ($\uparrow$Mon) and downward ($\downarrow$Mon) monotone functions, where the former preserve and the latter reverse the partial order. We illustrate this concept by means of the example below.

Example 1.32. [Monotonicity in Natural Language] Let W be our domain of interpretation. Consider the generalized quantifier *every N*, which has a syntactic category $s/(np\backslash s)$. It is interpreted as a function in $D_{(e,t)} \rightarrow D_t$, defined by

$$[\![\texttt{every\_N}(X)]\!] = 1 \text{ iff } \mathrm{card}([\![N]\!] - [\![X]\!]) = 0.$$

To prove that *every_N* is an ↑Mon function, we have to show that whenever $[\![X]\!] \leq [\![Y]\!]$ then $[\![\texttt{every\_N}(X)]\!] \leq [\![\texttt{every\_N}(Y)]\!]$. Assume $[\![\texttt{every\_N}(X)]\!] = 1$, then it holds that $\mathrm{card}([\![N]\!] - [\![X]\!]) = 0$ by definition. This implies that for every superset Y of X, $\mathrm{card}([\![N]\!] - [\![Y]\!]) = 0$. Hence $[\![X]\!] \leq [\![Y]\!]$ and $[\![\texttt{every\_N}(X)]\!] = 1$ implies $[\![\texttt{every\_N}(Y)]\!] = 1$ and we are done.

In a similar way, one can prove that, for instance, *nobody* is a downward monotone function.

The connection between syntactic categories and semantic types seems to be lost when looking at the current research in CTL and in the Montagovian school. The categorial grammarians are mostly interested in the grammaticality of linguistic structures, whereas the Montagovians are focused on their interpretation and entailment relations. The system presented in Chapter 4 combines again the two traditions by exploiting their logical connection. In particular, we encode the different monotonicity properties of GQs in the logical types of a CTL and make use of Definition 1.30 to achieve a proof theoretical account of natural reasoning.

1.4 Putting Things Together

In this section we explain how the syntactic derivations of the formal grammars discussed in Sections 1.1 and 1.2 are associated with instructions for meaning assembly.

1.4.1 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	$A/B : g \quad A\backslash C : f \Rightarrow C/B : \lambda x.f(gx)$	[$\mathrm{B}_\times$]

Example 1.33. [Meaning Assembly in CCG] Given the lexical assignments of the labelled lexicon above, CCG builds the meaning of *which Sara wrote* as follows.

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

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

EXAMPLE 1.34. [Ambiguous Sentences] Let `some, some'` and `every` abbreviate the lambda terms from our labelled lexicon $\lambda x.\exists z\texttt{student}(z) \wedge x(z)$, $\lambda xu.\exists z\texttt{student}(z) \wedge x(u,z)$, and $\lambda x.\forall z\texttt{student}(z) \rightarrow x(z)$, respectively. CCG builds the meaning of *every student knows some book* as following.

(i)

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

(ii)

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

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

1.4.2 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 1.35. [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.

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

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

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

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 1.42). A formal definition of the lambda calculus fragment corresponding to LP is given below[5].

Definition 1.36. [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 [Ben87a, Bus87, Wan92].

Theorem 1.37. 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 1.38. [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 1.39. 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.

[5]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 1.40. [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 1.41 and 1.42, respectively.

EXAMPLE 1.41. [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 1.42. [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.

1.5 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. The correspondence between proofs and natural language models, via the lambda terms, properly accounts for the natural language syntax semantics interface.

Chapter 2

The Mathematical Structure of CTL

In this chapter, we take a closer look at the mathematical structure of the grammatical base logic. 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 [Gor98b], 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].

We start by introducing the algebraic principle of residuation which lies at the heart of NL and NL($\Diamond$). We then present the Gentzen-style sequent calculus and the Kripke interpretation for these systems (Section 2.1). In Section 2.2, by means of Display Calculi, we clarify how the logical rules of NL and NL($\Diamond$) actually encode the definition of residuated operators. The same method is applied to explore the related Galois connected operators (Section 2.3). Again, we start by introducing the algebraic principle, and show its relation with the one of residuation. 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($\Diamond$,$\cdot^{\mathbf{0}}$) (Section 2.4).

The results presented here draw on work done in collaboration with Carlos Areces and Michael Moortgat [AB01, ABM01].

2.1 Capturing Residuation

The property of residuation arises in the study of order-preserving mappings [Fuc63, BJ72]. Let us look at the definition of the principle. The notion of residuated functions can be generally introduced for maps of arbitrary arity, however, here, we restrict our attention to unary and binary functions.

Definition 2.1. [Residuation] Let $\mathcal{A} = (A, \sqsubseteq_A)$, $\mathcal{B} = (B, \sqsubseteq_B)$ and $\mathcal{C} = (C, \sqsubseteq_C)$ be three

partially ordered sets. A pair of functions (f, g) such that $f : A \to B$ and $g : B \to A$ forms a *residuated pair* if [RES$_1$] holds.

$$[\text{RES}_1] \qquad \forall x \in A, y \in B \left(\begin{array}{rcll} fx & \sqsubseteq_B & y & \text{iff} \\ x & \sqsubseteq_A & gy & \end{array} \right).$$

A triple of functions (f, g, h) such that $f : A \times B \to C$, $g : A \times C \to B$, $h : C \times B \to A$ forms a *residuated triple* if [RES$_2$] holds.

$$[\text{RES}_2] \qquad \forall x \in A, y \in B, z \in C \left(\begin{array}{rcll} f(x,y) & \sqsubseteq_C & z & \text{iff} \\ y & \sqsubseteq_B & g(x,z) & \text{iff} \\ x & \sqsubseteq_A & h(z,y) & \end{array} \right).$$

In both cases the function f is said to be the *head* of the residuated pair or triple.

Remark 2.2. An equivalent characterization is obtained in terms of the monotonicity properties of the functions. Saying that (f, g) is a residuated pair is equivalent to the conditions *i*) and *ii*) below, where we write f is a $[\uparrow]$-function (f is a $[\downarrow]$-function) meaning that f is upward (downward) monotone in its argument. An upward (downward) monotone function is a function which preserves (reverses) the order holding among its arguments.

i. f and g are $[\uparrow]$-functions.
ii. $\forall y \in B(fgy \sqsubseteq_B y)$ and $\forall x \in A(x \sqsubseteq_A gfx)$.

Similarly, saying that (f, g, h) is a residuated triple is equivalent to requiring

i. f is a $[\uparrow, \uparrow]$-function, g is an $[\downarrow, \uparrow]$-function and h is an $[\uparrow, \downarrow]$-function.
ii. $\forall x \in A, y \in B, z \in C((f(x, g(x, z)) \sqsubseteq_C z)$ & $(y \sqsubseteq_B g(x, f(x, y)))$
& $(f(h(z, y), y) \sqsubseteq_C z)$ & $(x \sqsubseteq_A h(f(x, y), y)))$.

In what follows we focus on type forming operations $O_i : \mathsf{FORM}^i \to \mathsf{FORM}$ —where i marks the arity and FORM is the set of formulas— and we will investigate their behavior with respect to the poset $\langle \mathsf{FORM}, \longrightarrow \rangle$ where $\longrightarrow$ is the derivability relation among types.

2.1.1 The Logic of Residuation NL

As we anticipated in Chapter 1, the base system for 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 2.3. [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 2.4. [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 2.1. 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 presentation of NL. To establish the equivalence between the two formats, define the translation $\cdot^t : \mathsf{STRUCT} \to \mathsf{FORM}$ as

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

PROPOSITION 2.5. [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.

[1] [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 2.1: Gentzen sequent calculus for NL.

The system presented in Figure 2.1 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 2.6. [Cut-elimination and Decidability] The cut-rule is admissible in NL, and the system is decidable.

2.1.2 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 2.7. [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.

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

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

Its axiomatic presentation is obtained by simply adding to the axioms in Definition 2.4 the one below.

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

which defines the pair $(\Diamond, \Box^{\downarrow})$ as a residuated pair of operators (Definition 2.1).

As in the case of $[\text{RES}_2]$, 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 2.2.

As in the case of the base logic of binary operators, NL($\Diamond$) can be extended with structural rules obtaining a family of multimodal categorial type logics. The structural constraints given in Definition 1.15 are extended straightforwardly to the multimodal systems. The extended multimodal systems are proved to be in PSPACE in [Moo02].

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

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

Figure 2.2: Logical rules for residuated unary operators of $\mathsf{NL}(\Diamond)$.

2.1.3 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 thesis we will use Kripke models to reason with linguistic resources.

Definition 2.8. [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 $V : \mathsf{ATOM} \rightarrow \mathcal{P}(W)$. 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].

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

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 2.9. [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

[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.

$A \longrightarrow B$. 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 2.10. [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 2.10 $\mathcal{M}^c, A \nVdash B$. As $\mathcal{M}^c, A \Vdash A$, we have $\mathcal{M}^c \not\models A \longrightarrow B$ and hence $\not\models A \longrightarrow B$.

In order to maintain completeness in the presence of structural rules, one has to impose restrictions on the interpretation of the accessibility relations $R_\bullet$ and $R_\Diamond$. 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 2.11. [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 2.12. [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.

2.2 Displaying Residuation

In this section we clarify the residuation principle hidden in the logical rules of $\mathsf{NL}(\Diamond)$ by means of Display Calculi (DCs). This will give us a general method to compile algebraic principles into Gentzen sequents.

Display calculus, introduced by Belnap in [Bel82], is a general Gentzen style proof theoretical framework designed to capture many different logics in one uniform setting. DCs generalize Gentzen's notion of structures, by using multiple, complex, structural

connectives. One of the main characteristics of DCs is a general cut-elimination theorem, which applies whenever the rules of the display calculus obey certain, easily checked, conditions.

We will base our presentation on the system introduced by Goré in [Gor98b]. The main innovation of Goré's system over Belnap's concerns the use of additional structural connectives to capture the inherent duality of every logic, by means of dual sets of display postulates. Building on these features, DCs obtain the fundamental property which gives them their name, '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.

2.2.1 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 2.13. [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'.

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

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 2.3).

To obtain the still missing rules we have to work only slightly harder. As we pointed out in Remark 2.2, 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 2.3.

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

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

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

Figure 2.3: DC logical rules for residuated binary operators.

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.

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

And in a similar way we can prove the "composition property" we mentioned in Remark 2.2.

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 2.1 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

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

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

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

2.2.2 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 $\mathsf{NL}(\diamond)$. Let us start by defining the proper logical and structural languages.

Definition 2.14. [Logical and Structural Languages for a DC Presentation of $\mathsf{NL}(\diamond)$)] 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 $\mathsf{NL}(\diamond)$ 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 $(\circ, \bullet)$,

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

And we obtain the rules for the logical operators by projection and monotonicity behavior. The full set of rules is given in Figure 2.4.

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

$$\frac{\bullet A \Rightarrow \Delta}{\diamond A \Rightarrow \Delta}\ [\diamond\text{L'}] \qquad \frac{\Delta \Rightarrow A}{\bullet\Delta \Rightarrow \diamond A}\ [\diamond\text{R'}]$$

Figure 2.4: DC logical rules for residuated unary operators.

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

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

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

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

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

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

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

2.3 Galois Connected Operations

The algebraic structure of the base logic can also accommodate a pair of order-reversing *Galois connected* operators, which in this thesis we will write as ${}^{\mathbf{0}}\cdot, \cdot^{\mathbf{0}}$ following Goré's notation. To understand the relation between these two concepts, it may be useful to situate them in their natural algebraic context. Residuated and Galois connected pairs of mappings were studied in the work of Birkhoff [Bir67] and Ore [Ore44], among others. The relevance of this early work for current research on substructural logics has been emphasized by Michael Dunn, from whose [Dun91] we draw the following definitions. For the ease of exposition we repeat the definitions of residuated pairs as well.

Definition 2.15. [Residuated and Galois connected pairs] Consider two posets $\mathcal{A} = (A, \sqsubseteq_A)$ and $\mathcal{B} = (B, \sqsubseteq_B)$, and functions $f : A \to B,\ g : B \to A$. The pair (f, g) is said to be *residuated* iff

$$[\text{RES}_1] \qquad fx \sqsubseteq_B y \quad \text{iff} \quad x \sqsubseteq_A gy.$$

The pair (f, g) is said to be *Galois connected* iff

$$[\text{GC}] \qquad y \sqsubseteq_B fx \quad \text{iff} \quad x \sqsubseteq_A gy.$$

Keeping the posets $\mathcal{A}$ and $\mathcal{B}$ distinct helps understanding the connection between residuated and Galois connected pairs of mappings. Let us introduce a third poset $\mathcal{B}' = (B, (\sqsubseteq_B)^{-1})$ where $(\sqsubseteq_B)^{-1} = \{(b, a) \mid a \sqsubseteq_B b\}$, and consider a function $h : B \to A$. Following [RES_1] the functions f, h form a residuated pair iff the biconditional $fa\ (\sqsubseteq_B)^{-1}\ b$ iff $a \sqsubseteq_A hb$ holds. But now, replacing $(\sqsubseteq_B)^{-1}$ by $\sqsubseteq_B$, we obtain that $b \sqsubseteq_B fa$ iff $a \sqsubseteq_A hb$, *i.e.* the pair f, h is Galois connected with respect to the orders $\sqsubseteq_B$ and $\sqsubseteq_A$. As Dunn [Dun91] puts it, the Galois connected pair is obtained by turning around the inequality $\sqsubseteq_B$ in the characterization of residuation.

Finally, notice that the definition of Galois connected operators given in Definition 2.15 can be extended further considering their duals.

DEFINITION 2.16. [Dual Galois Connections] Let $\mathcal{A} = (A, \sqsubseteq_A)$ and $\mathcal{B} = (B, \sqsubseteq_B)$ be two partially ordered sets. Consider a pair of functions $f : A \to B$ and $g : B \to A$. The pair (f, g) is called a *dual Galois connection* if [DGC] below holds.

$$\text{[DGC]} \qquad \forall x \in A, y \in B \left(\begin{array}{lll} fx & \sqsubseteq_B & y \quad \text{iff} \\ gy & \sqsubseteq_A & x \end{array} \right).$$

As with residuation, there is an equivalent formulation of these properties in terms of their monotonicity behavior and a composition rule. [GC], for example, is equivalent to require that f and g are both $[\downarrow]$-functions, and that for all x, $x \sqsubseteq fgx$, and $x \sqsubseteq gfx$ (here again, we just consider f and g as functions defined on the same poset). Recall that, when we cast this algebraic discussion in terms of categorial type logics the objects we will be considering are types ordered by their derivability relation.

Galois connected operators have been also studied in the context of Linear Logic, see [Lam93, Abr91, Gor98b, 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 thesis, 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).

2.3.1 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].

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

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

[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.

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

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

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

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

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 2.17. [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

$$\dfrac{\dfrac{\vdash A \longrightarrow {}^{\mathbf{0}}B \quad \dfrac{\vdash D \longrightarrow B}{\vdash {}^{\mathbf{0}}B \longrightarrow {}^{\mathbf{0}}D}\,[R2]}{\vdash A \longrightarrow {}^{\mathbf{0}}D}\,[\text{TRANS}]}{\vdash D \longrightarrow A^{\mathbf{0}}}\,[\text{GC}] \tag{2.1}$$

QED

THEOREM 2.18. [Completeness] Given $A \longrightarrow B$, then $\models A \longrightarrow B$ implies $\vdash A \longrightarrow B$.

PROOF. Suppose $\nvdash A \longrightarrow B$. Then by Lemma 2.17 $\mathcal{M}^c, A \nVdash B$. 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 2.17 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 2.19. 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.

2.3.2 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 $(\natural, \flat)$.

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

We now project this behavior into the logical operators $({}^{\mathbf{0}}\cdot, \cdot^{\mathbf{0}})$ as it is shown in Figure 2.5.

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

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

Figure 2.5: 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 $[{}^{\mathbf{0}}\cdot\mathrm{L}]$ and $[\cdot^{\mathbf{0}}\mathrm{L}]$ as they are. To obtain $[{}^{\mathbf{0}}\cdot\mathrm{R}]$ and $[\cdot^{\mathbf{0}}\mathrm{R}]$ we need to apply [gc].

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

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

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

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

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

Figure 2.6: Compiled logical rules for Galois connected unary operators.

The full set of rules obtained is shown in Figure 2.6. 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,

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

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

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

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.

2.3.3 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 (2.2) we give the four instances of the cut-rule we have to consider.

$$\frac{\Delta \Rightarrow A \quad \Gamma[A] \Rightarrow \Delta'}{\Gamma[\Delta] \Rightarrow \Delta'}\,[\mathrm{cut}_1] \qquad \frac{\Delta' \Rightarrow \Gamma[A] \quad A \Rightarrow \Delta}{\Delta' \Rightarrow \Gamma[\Delta]}\,[\mathrm{cut}_2]$$
$$\frac{\Delta \Rightarrow A \quad \Delta' \Rightarrow \Gamma\{A\}}{\Delta' \Rightarrow \Gamma\{\Delta\}}\,[\mathrm{cut}_3] \qquad \frac{\Gamma\{A\} \Rightarrow \Delta' \quad A \Rightarrow \Delta}{\Gamma\{\Delta\} \Rightarrow \Delta'}\,[\mathrm{cut}_4] \tag{2.2}$$

The logical rules we have obtained in the previous section (Figure 2.6) 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 2.7. 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 2.20. [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 (2.3) and (2.4), 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.)

$$\dfrac{\dfrac{\dfrac{A \Rightarrow \flat\Delta}{\Delta \Rightarrow A^{\mathbf{0}}}\,[\cdot^{\mathbf{0}}\mathrm{R}] \quad \dfrac{\dfrac{\Gamma\{A\} \Rightarrow \Delta'}{\Gamma\{\flat(A^{\mathbf{0}})\} \Rightarrow \Delta'}\,[\cdot^{\mathbf{0}}\mathrm{L}^{+}]}{\Gamma[A^{\mathbf{0}}] \Rightarrow \Delta'}}{\Gamma[\Delta] \Rightarrow \Delta'}\,[\mathrm{cut}_1]}{\Gamma\{\flat\Delta\} \Rightarrow \Delta'} \quad \rightsquigarrow \quad \dfrac{\Gamma\{A\} \Rightarrow \Delta' \quad A \Rightarrow \flat\Delta}{\Gamma\{\flat\Delta\} \Rightarrow \Delta'}\,[\mathrm{cut}_4] \tag{2.3}$$

$$\dfrac{\dfrac{\dfrac{A \Rightarrow \flat\Delta}{\Delta \Rightarrow A^{\mathbf{0}}}\,[\cdot^{\mathbf{0}}\mathrm{R}] \quad \dfrac{\dfrac{\Delta' \Rightarrow \Gamma[A]}{\Delta' \Rightarrow \Gamma[\flat(A^{\mathbf{0}})]}\,[\cdot^{\mathbf{0}}\mathrm{R}^{-}]}{\Delta' \Rightarrow \Gamma\{A^{\mathbf{0}}\}}}{\Delta' \Rightarrow \Gamma\{\Delta\}}\,[\mathrm{cut}_3]}{\Delta' \Rightarrow \Gamma[\flat\Delta]} \quad \rightsquigarrow \quad \dfrac{\Delta' \Rightarrow \Gamma[A] \quad A \Rightarrow \flat\Delta}{\Delta' \Rightarrow \Gamma[\flat\Delta]}\,[\mathrm{cut}_2] \tag{2.4}$$

In (2.5) and (2.6), antitone cuts $[\mathrm{cut}_4]$, $[\mathrm{cut}_2]$ are replaced by isotone cuts $[\mathrm{cut}_1]$, $[\mathrm{cut}_3]$.

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

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

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

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

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

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

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

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

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

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

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

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

Figure 2.7: Logical rules of $\mathsf{NL}(\Diamond,\cdot^{\mathbf{0}})$.

$$\dfrac{\dfrac{\Gamma\{\natural A\} \Rightarrow \Delta'}{\Gamma\{A^{\mathbf{0}}\} \Rightarrow \Delta'}\,(\cdot^{\mathbf{0}}\mathrm{L}^{-}) \quad \dfrac{\Delta \Rightarrow A}{A^{\mathbf{0}} \Rightarrow \natural\Delta}\,[\cdot^{\mathbf{0}}\mathrm{L}]}{\Gamma\{\natural\Delta\} \Rightarrow \Delta'}\,[\mathrm{cut}_4] \quad \rightsquigarrow \quad \dfrac{\Delta \Rightarrow A \quad \Gamma\{\natural A\} \Rightarrow \Delta'}{\Gamma\{\natural\Delta\} \Rightarrow \Delta'}\,[\mathrm{cut}_1] \tag{2.5}$$

$$\dfrac{\dfrac{\Delta' \Rightarrow \Gamma[\natural A]}{\Delta' \Rightarrow \Gamma[A^{\mathbf{0}}]}\,[\cdot^{\mathbf{0}}\mathrm{R}^{+}] \quad \dfrac{\Delta \Rightarrow A}{A^{\mathbf{0}} \Rightarrow \natural\Delta}\,[\cdot^{\mathbf{0}}\mathrm{L}]}{\Delta' \Rightarrow \Gamma[\natural\Delta]}\,[\mathrm{cut}_2] \quad \rightsquigarrow \quad \dfrac{\Delta \Rightarrow A \quad \Delta' \Rightarrow \Gamma[\natural A]}{\Delta' \Rightarrow \Gamma[\natural\Delta]}\,[\mathrm{cut}_3] \tag{2.6}$$

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

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

We start by proving the following.

Proposition 2.21. 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 2.22. 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 2.23. [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 2.21 proves soundness of the cut-rules. For rules $[\mathbf{0}{\cdot}\mathrm{R}^{+}]$, $[\cdot^{\mathbf{0}}\mathrm{R}^{+}]$ it is trivial, $[\mathbf{0}{\cdot}\mathrm{L}]$ For rules $[\mathbf{0}{\cdot}\mathrm{R}]$, $[\cdot^{\mathbf{0}}\mathrm{R}]$, $[\mathbf{0}{\cdot}\mathrm{L}]$ and $[\cdot^{\mathbf{0}}\mathrm{L}]$ use the fact that the [GC] rule is sound. For rules $[\mathbf{0}{\cdot}\mathrm{R}^{-}]$, $[\cdot^{\mathbf{0}}\mathrm{R}^{-}]$, $[\mathbf{0}{\cdot}\mathrm{L}^{+}]$ and $[\cdot^{\mathbf{0}}\mathrm{L}^{+}]$ use Proposition 2.21 plus the fact that axioms [A1] and [A2] are valid. QED

Theorem 2.24. [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 2.7) 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. (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 this thesis 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 the next chapters.

2.4 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 2.25 below.

Definition 2.25. [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 2.1), *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,

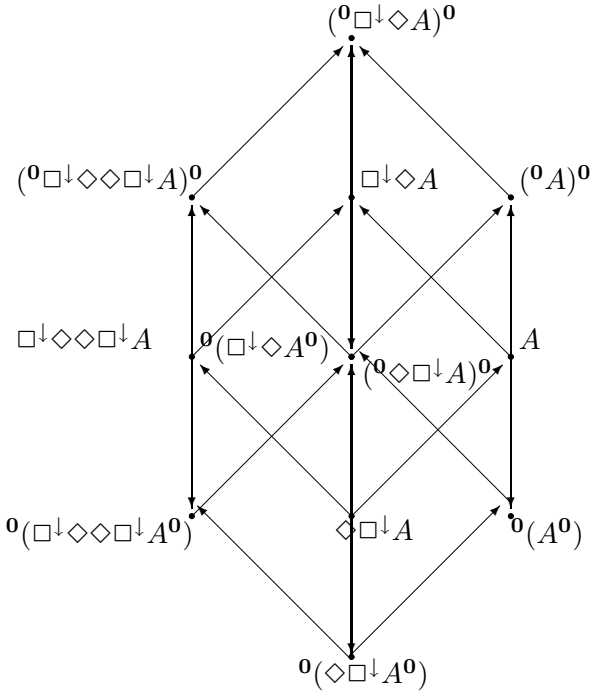
$$\Box^{\downarrow}\Diamond\Box^{\downarrow}\Diamond A \longrightarrow \Box^{\downarrow}\Diamond A \text{ and } ({}^{\mathbf{0}}(({}^{\mathbf{0}}A)^{\mathbf{0}}))^{\mathbf{0}} \longrightarrow ({}^{\mathbf{0}}A)^{\mathbf{0}}.$$

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.



2.5 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.

4. By means of Display Calculi we have described a method to explore a landscape of other kinds of algebraic properties, assuming that they can be encoded in terms of structural rules.

Part II
Reasoning with Modalities

In the first part of this thesis, we have looked at the mathematical principles that govern residuated and Galois connected pairs of unary connectives, and the relation between these unary operations and the more familiar binary connectives of categorial grammar. In this part of the thesis, we put our extended type language to use in linguistic analysis.

In Chapter 3, we review the main linguistic applications for unary modalities proposed so far in the literature. We then identify other possible uses of the unary operators which have not been considered yet. We present these possibilities on a theoretical level and look at unary operators as 'logical features'. In particular, we propose to employ them to capture distinctions within the domains of interpretation of linguistic signs.

In Chapter 4, we move to a concrete applications focusing on the role of monotone functions in natural language reasoning. By decorating functional types with unary operators we encode the semantic distinction between upward and downward monotone functions. Furthermore, we study the advantages of this encoding by exploring their contribution to the study of natural reasoning and of the syntactic distribution of negative polarity items.

Chapter 3

Modalities for Structural Control

In this chapter we investigate the linguistic applications of the logical tools that come with the move from unimodal to multimodal logics. We present the different ways in which unary modalities have been used in the literature. In particular, we focus attention on their use to control structural reasoning in the derivation of long distance dependencies and to enforce morphological agreement (Section 3.2). In Section 3.3, by abstracting away from the actual implementations, we summarize the logical properties used so far and we identify the potentialities of the logical system which have not been exploited yet.

3.1 Multimodal Systems

The mathematical structure of the Lambek systems studied in Chapter 2 provides new logical tools which can be employed in the modelling of linguistic composition. In particular, it provides us with structure-building operations of different arities and it makes it possible to distinguish different modes of composition for these operators. Multimodal systems have found several applications in different fields [BRV01]. In computer science they have been widely used; the accessibility relations, in this case, are seen as transitions among states of a computation and the labels stand for programs. In the case of Categorial Type Logic (CTL), the accessibility relation for the binary operators is used to model linguistic composition; the modes stand for the different ways linguistic signs compose. In other words, the new systems can accomodate different composition relations as living together, which is a desirable property for a logical system employed to model linguistic phenomena. Moreover, the accessibility relation of the unary operators can been seen as a feature checking relation which is a widely used concept in linguistic theories. Finally, the interplay between the logical and the structural module allows the use of modes and unary modalities to control grammatical resource management. The full picture can be summarized by saying that the logical rules of the binary residuated operators offer a logical analysis of the merging of linguistic signs; the packages of structural rules give a logical perspective on structural variation; and modes and unary operators enable fine-grained control on the application of structural reasoning. In this chapter, we illustrate the linguistic applications of the unary oper-

ators introduced in [KM95, Kur95, Moo96b] by reviewing how they have been used in the literature so far. Moreover, we introduce the tasks they will carry out in this thesis.

The natural deduction logical rules of the residuated unary operators $(\Diamond, \Box^{\downarrow})$ are as below where the t, u and v labelling the rules stand for the meaning representation. In this chapter we will concentrate of the form dimension of linguistic composition, whereas in Chapter 4 we will take into consideration the meaning dimension as well.

$$\frac{\Gamma \vdash t : \Box^{\downarrow} A}{\langle \Gamma \rangle \vdash {}^{\vee} t : A}\ [\Box^{\downarrow}\mathrm{E}] \qquad \frac{\langle \Gamma \rangle \vdash t : A}{\Gamma \vdash {}^{\wedge} t : \Box^{\downarrow} A}\ [\Box^{\downarrow}\mathrm{I}]$$

$$\frac{\Delta \vdash u : \Diamond A \quad \Gamma[\langle v : A \rangle] \vdash t : B}{\Gamma[\Delta] \vdash t[{}^{\cup} u / v] : B}\ [\Diamond\mathrm{E}] \qquad \frac{\Gamma \vdash t : A}{\langle \Gamma \rangle \vdash {}^{\cap} t : \Diamond A}\ [\Diamond\mathrm{I}]$$

Let us look at their linguistic applications.

3.2 Controlling Structural Reasoning

The main property of the unary operators which has been exploited so far is their ability to lexically control structural reasoning. The trade-off between logical types and the structural configuration of a sequent allows one to model dependencies involving precedence, dominance and agreement relations. We briefly discuss two illustrations of this use of modalities: the treatment of wh-dependencies by means of structural control modalities (Subsection 3.2.1) and morphological agreement via modal inclusion postulates (Subsection 3.2.2).

3.2.1 Movement in CTL

Relative clause formation gives rise to a dependency between the relative pronoun which introduces the clause and a hypothetical noun phrase resource (a 'gap') somewhere within the relative clause body. We have seen in Chapter 1 that this kind of dependency can be established by a higher-order type assignment to the relative pronoun, such as $(n\backslash n)/(np\backslash s)$ or $(n\backslash n)/(s/np)$. The challenge, for a type logical account, is to adequately characterize which structural positions are 'accessible' for extraction. Recall that in English extraction can be performed from the main subject position (1-a), or another noun phrase position, which can be right-peripheral (1-b), or non-peripheral (1-c). The subject case (1-a) can be derived in the base logic, using $(n\backslash n)/(np\backslash s)$ as relative pronoun type. The derivations of (1-b) and (1-c) require a *controlled* form of structural reasoning, involving both rebracketing and reordering. Obviously, *global* form of associativity or commutativity would wildly overgenerate.

(1) a. who knows Lori.
b. which Sara wrote [...].
c. which Sara wrote [...] there.

It is shown in [Moo99] that the structural rules in 3.1 perform the required task, together with a lexical type assignment $(n\backslash n)/(s/\Diamond\Box^{\downarrow}np)$ generalizing over the (1-b) and (1-c) cases.

$$\frac{\Gamma[\Delta_1 \circ (\Delta_2 \circ \langle\Delta_3\rangle)] \vdash C}{\Gamma[(\Delta_1 \circ \Delta_2) \circ \langle\Delta_3\rangle] \vdash C}\,[\mathrm{ass}_\Diamond] \qquad \frac{\Gamma[(\Delta_1 \circ \langle\Delta_3\rangle) \circ \Delta_2] \vdash C}{\Gamma[(\Delta_1 \circ \Delta_2) \circ \langle\Delta_3\rangle] \vdash C}\,[\mathrm{diss}_\Diamond] \tag{3.1}$$

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 (1-b) where the extraction is performed from a peripheral position.

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

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 (1-c). To account for right-branch extraction from a non-peripheral position one needs the interaction structural rule $[\mathrm{diss}_\Diamond]$ as well.

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

$$\mathrm{which} \circ (\mathrm{Sara} \circ (\mathrm{wrote} \circ \mathrm{there})) \vdash n\backslash n$$

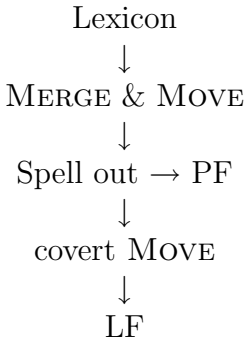
This derivation can be read in a similar way than the previous one. The only difference is the application of [diss$_\Diamond$] which brings the hypothesis in a peripheral position.

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.

The categorial account of structural control is surprisingly close to 'feature-driven' structural reasoning in generative grammar, especially within the minimalism program [Cho95] formalized in [Sta97]. We briefly discuss this correspondence, using Stabler's algebraic version of minimalism and Vermaat's type logical translation of Stabler-style grammars [Ver99], as our point of reference. First we sketch the main ideas of the minimalist program for readers not familiar with this framework.

The minimalist program

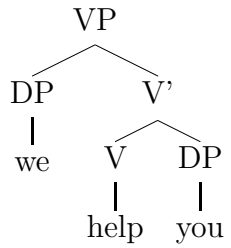
Minimalism is a recent development of generative grammar (See Radford [Rad97] and Haegeman [Hae94] for an introduction to its background assumptions). As in many other linguistic theories, in the minimalist program the way sentences are structured out of words and phrases is determined by the specific categories assigned to such constituents. The two basic structure-building operations, Merge and Move, are both driven by feature checking. The system generates structures which will be given an overt form (the phonetic form PF) when the required conditions for grammaticality are met. Covert movement can still be applied to interpret the syntactic structure once built, yielding the logical form (LF).



Categories can be either *lexical categories*, e.g. nouns, adjectives, adverbs, and prepositions (the words belonging to these categories have lexical/descriptive content), or *functional categories*, e.g. determiners, pronouns and complementizers (the words belonging to these categories have essentially a grammatical function). Moreover, besides these overt constituents, syntactic structures may contain *empty* (covert, or null) categories as well, *i.e.* categories which have no overt phonetic form, and hence which are inaudible or silent —empty categories are used for example in gapping. Furthermore,

a distinction is made among grammatical features which are *interpretable* (at LF) by virtue of having semantic content (e.g. person, gender, tense, etc.), and *uninterpretable* (at LF) by virtue of having no semantic content (e.g. wh, case, etc.). LF-representation may contain only (semantically) interpretable features. Finally, categories can have 'strength' features which determine whether they trigger over (*strong*) or cover (*weak*) movement.

Phrases are formed by *merging* (combining) two expressions together provided their categories match. Every phrase has a *head* word which determines the nature of the overall phrase and *selects* for the constituents it can combine with. For instance, an expression such as *help you* is a verb phrase (VP), because its head word *help* is a verb (V). The verb phrase headed by *help* with complement *you* and with specifier *we* is represented as the following tree.



The different words in a sentence carry grammatical features which are guaranteed to be compatible with those of other words in the same sentence by means of a *feature checking mechanism* [Cho95]. If a derivation (the sequence of steps of MERGE and MOVE used to build a complex expression from lexical expressions) gives rise to an LF-representation containing only (semantically) interpretable features, the relevant derivation is said to *converge* (at LF); if it gives rise to an LF-representation containing one or more (semantically) uninterpretable features, the derivation is said to *crash* (at LF), and the corresponding sentence is ill-formed. Grammatical features are checked in the course of a derivation, and uninterpretable features are erased once checked.

Long distance dependency in the minimalist grammar

By focusing attention on MERGE and MOVE, Stabler has captured the basic ideas of the minimalist syntax in a derivational system known as Minimalist Grammar (MG) [Sta97]. Briefly, he shows that MERGE corresponds to functional application accounting for predicate-argument relations, whereas MOVE deals with the restructuring of built structures. The clear distinction between these two operations is made possible by the use of an algebraic grammar format where different sorts of features are clearly defined.

First of all, the lexicon of MG is a finite set of trees lexical head seen as a sequence of syntactic features. The latter are defined as below.

DEFINITION 3.1. [Syntactic Features of MG] The language of MG is built over the set of base features BASE, given by the union of syntactic categories (SynCate) and functional categories (FunCate) as below, where the complex features operators can be interpreted as in Table 3.1.

$$\begin{aligned} \mathsf{BASE} ::= &\quad \mathsf{SynCate} \cup \mathsf{FunCate}. \\ \mathsf{Feature} ::= &\quad \mathsf{BASE} \mid {=}\mathsf{BASE} \mid {+}\,\mathsf{BASE} \mid {-}\,\mathsf{BASE}. \end{aligned}$$

Feature	Interpretation
$= \mathtt{x}$	selection requirement
$+\mathtt{x}$	assignment of x to specifier (licensor)
$-\mathtt{x}$	requirement of x (licensee)

Table 3.1: Features in MG.

The two structure building operators of MG are called *merge* and *move* since they are intended to capture the operations of the minimalist program. The former is a binary operation over sequences of features $merge(S_1, S_2) = S$, where S_1 has an accessible feature =x and S_2 has an accessible feature x; = x and x are cancelled out in S. On the other hand, *move* is a unary operation on sequences with an accessible feature +f and an accessible link with root labelled −f. $S = move(S_1)$ is the tree obtained by moving the subtree with the accessible feature −f to the specifier position of S_1, and cancelling −f and +f. Finally, the information concerning the head/complement/specifier positions is marked by < and > labelling the nodes in the tree. The two labels point to the head of the projection, and consequently they mark the specifier and complement which are at the left and right of the head, respectively.

We illustrate how the grammar works by looking at a derivation of the structure involving a wh-dependency. Let the set of syntactic categories be SynCate := {n, v} which stand for 'noun' and 'verb phrase' and the set of functional categories FunCate := {ip, cp, d} which stand for 'inflection phrase', 'complementizer phrase' and 'determiner phrase'. The lexical entries used to derive *which book the student write* are given below where ϵ stands for the empty phrases[1].

which::=	[=n d −wh]	ϵ::=	[=v ip]
book::=	[n]	ϵ ::=	[=ip +wh cp]
student::=	[n]		
the::=	[=n d]		
write::=	[=d =d v]		

For instance, the sequence of features assigned to the relative pronoun *which* expresses that (a) it combines with a noun =n to yield a determiner phrase d, and (b) the composed phrase has the feature −wh to be checked against a functional category providing +wh.

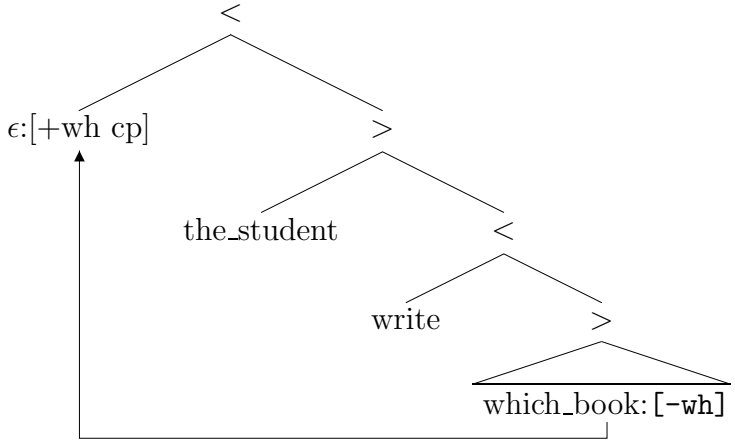
The connection between MG and CTL is spelled out in [Ver99]. In particular, it is shown how the elimination rules [/E] and [\E] of CTL play the role of *merge*. On the other hand, the features driving the *move* operation are encoded by means of modes carried by the unary operators, *viz.* $(\Diamond_s, \Box_s^{\downarrow})$, with $s \in \{wh, nom, acc, fut, past, \ldots\}$. For instance,

[1]Note that this example taken from [Sta01] is a pseudo-English phrase since it ignores the tense.

the lexical assignment of *which* is decorated with $\Box^{\downarrow}_{wh}$ requiring 'to be checked' by $\Diamond_{wh}$. The merging of the lexical entries *which* and *book* is obtained by applying [/E].

$$\frac{\text{which} \vdash \Box^{\downarrow}_{wh} d/n \quad \text{book} \vdash n}{\text{which} \circ \text{book} \vdash \Box^{\downarrow}_{wh} d} \; [/\text{E}]$$

In MG the uninterpretable feature `-wh` carried by the relative pronoun forces the application of *move* as illustrated below. The wh-phrase moves from the position where it occurs at the surface structure to the complementizer position deriving for example *which book the student write*.



Alternatively, the tree above can be represented in a natural deduction format which facilitates the comparison with the CTL approach.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\text{which::=n d -wh} \quad \text{book::n}}{\text{(which book):d -wh}}\,merge_1}{\text{write::=d =d v} \quad \text{(which book):d -wh}}\,merge_2 \quad \dfrac{\text{the::=n d} \quad \text{student::n}}{\text{(the student):d}}\,merge_1}{\text{(write):=d v, (which book):-wh} \quad \text{(the student):d}}\,merge_3}{\epsilon\text{::=v ip} \quad \text{(the student write):v, (which book):-wh}}\,merge_3}{\epsilon\text{::=ip +wh cp} \quad \text{(the student write):ip, (which book):-wh}}\,merge_3}{\text{(the student write):+wh cp, (which book):-wh}}\,move}{\text{(which book the student write):cp}}$$

In the derivation we distinguish different *merge* operations: $merge_1$ is the simple function application explained above; $merge_2$ has as second argument an expression carrying a feature which requires to be checked, hence it must be left accessible for feature checking; $merge_3$ operates on sequences such that the second argument is composed of a chain of sequences. The final step is the *move* operation which can be applied since the feautre carried by the wh-phrase is licensed. See [Sta01] for a detailed discussion.

In the analysis proposed in [Ver99], this second part of the derivation is taken care of by the unary operators. The restructuring of the derived phrase performed by the *move* operation is accounted for by means of interaction structural rules. Without going into the details of this analysis, we would like to draw attention to the way 'requirements' vs. 'production' of a syntactic feature are expressed in the two approaches. First of all, note how in CTL the selection requirement expressed by the = is captured by the

logical properties of the /. Similarly, the different polarities + and − displayed on the features in MG are captured by the logical relation between the unary operators. Let us explain this by looking back at the derivation in CTL of *which book the student write* given in the previous section. The assumed *np* is decorated with $\Box^{\downarrow}$ requiring a feature to be checked. The $\Diamond$ carried by *which* provides the means to perform the required feature-checking. In other words, the structural reasoning place the role of the overt movement of the hypothesis 'gap', whereas the residuated relation between $\Diamond$ and $\Box^{\downarrow}$, and between $\bullet$ and / establish the wh-dependency driving the restructuring and checking the feature agreement. Finally, note how the MG and CTL are very similar in the recursive definition of their language, and in the distinction between the two sorts of rules to build trees and restructure them. However, they diverge essentially in two ways: (i) in MG the interplay between features is purely stipulated, whereas in CTL it derives from the logical properties of the operators; and (ii) since MG is based only on modus ponens, it lacks a rule performing the hypothetical reasoning.

In addition to overt movement the minimalist framework also provides one with *covert* movement, *viz.* movement which is not visible at the surface structure or at the phonological level, but which determines the scope distribution of the moved expression at LF. Quantifier phrases are an example of the class of phenomena analyzed in terms of covert movement. In the CTL framework they have been modelled by means of (interaction) structural rules [Mor94, Moo96a] exploiting the Curry-Howard correspondence (Section 1.3). Chapter 6 is dedicated to the analysis of these phenomena and it also contains a comparison of the categorial type logical approach with the analysis proposed within the minimalist program.

3.2.2 Morphological Agreement in CTL

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.

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

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

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

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.

3.3 Zooming in on the Semantic Domains

Generalizing over the analyses we have outlined, 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 \ and / at the level of meaning assembly, where the directionality information plays no role anymore[2].

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, both the plural and singular Italian articles are 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.

Let us now summarize the logical properties of CTL which have been exploited so far. In all the accounts we reviewed a crucial role has been played by the residuation law of both the binary and unary operators. Moreover, structural reasoning has been lexically anchored to account for linguistic composition of items sensitive to either word order or morphological features. Finally, lexical type distinctions and structural rule packages have been used to account for crosslinguistic differences regarding these sensitive properties.

In our investigation of the mathematical structure of CTL we discussed other logical properties. First of all, residuated operators could be identified also by their way of composing. Recall the patterns

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

[2]Technically, we assume that in the mapping from syntactic to semantic types, the unary operators are 'forgotten': $\mathtt{type}(\Diamond A) = \mathtt{type}(\Box^{\downarrow}A) = \mathtt{type}(A)$.

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 ${}^{\mathbf{0}}B \longrightarrow {}^{\mathbf{0}}A$.

In this thesis we will make use of these other logical properties of $\mathsf{NL}(\Diamond,\cdot^{\mathbf{0}})$ which have not been exploited yet. As in the previous accounts, we employ the residuated unary operators as well as the Galois connections to zoom in on the domains of interpretation encoding differences not visible otherwise. However, attention is focused on different sorts of features distinguishing the members of the same semantic type. 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. These properties are also crucial to identify modes of linguistic composition. However, these different composition relations are not sensitive to the internal structure of the constituents, but only to their distribution. We will show how $\mathsf{NL}(\Diamond,\cdot^{\mathbf{0}})$ can account for the sentences in our original checklist (Example 1.3) which the grammars discussed so far fail to recognize. We repeat those sentences below indicating their interprations by means of [X > Y], *viz.* X has scope over Y.

(2) 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]

(3) 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:

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

Briefly, we use the unary operators to carry semantic features distinguishing the types of, for instance, the upward and downward monotone quantifiers *some student* and *no student*, respectively, and study the advantages of expressing this distinction when modelling both linguistic composition and natural reasoning (Chapter 4). Moreover, 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 (Chapter 6). 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. In Chapter 7, 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 section 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.

3.4 Key Concepts

The relevant lesson to keep in mind about unary operators is that they can be thought of as logical tools which can be employed to:

1. Lexically anchor linguistic composition.
2. Distinguish items which belong to the same semantic category, but differ in some other aspects relevant for grammaticality.
3. Express subset relations within the domains of interpretation.
4. Mark the presence of sensitive items, and pass the information through the derivation.

Chapter 4

Reasoning with Monotone Functions

In this chapter we present a proof theoretical account of natural reasoning using linguistic structures as the vehicle of inference. Following [Sup79, Ben86], we refer to this system as a Natural Logic. In particular, we focus attention on monotone inference. The notion of monotonicity is closely related to that of negative polarity items (NPIs). We aim to develop a system which, while marking the linguistic structures with the information required to model natural reasoning inferences, accounts for NPI distribution.

We start by showing the link between NPIs and natural reasoning (Section 4.1). While discussing this connection, we give the background behind the natural logic proposed by van Benthem [Ben86] and further studied by Sánchez [SV91] (Section 4.2). By looking at an alternative system proposed by Dowty in [Dow94], we show that a logical account of NPIs requires the use of internalized polarity markers (Section 4.3). Finally, in Section 4.4 we describe a natural logic based on a multimodal categorial type logic (MCTL). We show that MCTL has the required expressivity to account for NPIs and produce marked structures ready for deriving natural reasoning inferences.

4.1 Parsing and Reasoning

The task of accounting for the role of language in drawing inferences is commonly considered to belong to the domain of formal semantics. Most of the literature in natural reasoning assumes a model theoretic perspective, and uses a formal language as an intermediate step into which natural language expressions are translated. In this translation information is lost about natural language structures which might be relevant when drawing inferences. It is an interesting question whether a system can be developed which is able to compute natural reasoning inferences taking into consideration the information human beings rely on when reasoning. In particular, we are referring to the syntactic and semantic information obviously involved in reasoning. This is the task that the construction of a natural logic is supposed to tackle as explained in [Ben86, Ben87b] inspired by [Sup79], where the name indicates that instead of employing logical forms as vehicles of inference, natural language expressions are used directly.

In this approach, the applicability of an inference pattern is read off from a derivation. The advantage gained by assuming such proof theoretical perspective is that one can

study how natural language structures contribute to natural reasoning. It is, of course, an ambitious project, given the high complexity of sentences, the richness of ambiguity which is a hallmark of any human language, and the variety of natural reasoning forms. In order to make it more tractable, we focus on specific inference patterns selected together with a restricted set of linguistic structures. Following [Ben86, SV91] we will look at monotonicity inferences. See [FWF00] for an interesting natural logic involving conservativity in addition to monotonicity.

Besides contributing to natural reasoning, monotonicity also plays an important role in establishing the grammaticality of some forms of linguistic composition. In natural language one finds expressions known as negative polarity items (NPIs), whose syntactic distribution is determined by the monotonicity property of the phrases they are in construction with. A system employed to model the composition of linguistic signs must be able to take monotonicity information into consideration when working with these expressions. In this chapter we show how the two aspects of parsing and reasoning can be accounted for within the categorial type logic framework. In particular, we present a system which can deal with the distribution of NPIs and produce marked parsed output from which monotone inferences can be drawn.

4.1.1 Negative Polarity Items and Monotonicity

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*. However, no explicit references to the existence of a phenomenon of negative polarity were made, yet. The move to a conceptualization of it and the introduction of the terminology now in use is due to Baker [Bak70]. In that work, the notion of licensing elements made an implicit appearance, since the distribution of polarity items is seen as a matter of being in the scope of a suitable element, though "scope" and "suitable element" are not properly defined yet. Fauconnier [Fau75] looked at NPIs as denoting extreme elements among a set of alternatives. In doing so, he introduced a semantic perspective on the issue. This work inspired Ladusaw [Lad79], who gave a precise semantic interpretation to the vague idea of affective licensors proposed by Klima, identifying them with downward monotone expressions. Since then, negative polarity items have been studied from various perspectives, some more syntactically oriented [Lin81, Pro88], others more semantic in nature [Zwa86, Wou94, Gia97] and yet others more focused on the pragmatic aspects [KL89, KL93, Chi02]. In Chapter 7, we analyze the distributional behavior of NPIs in detail, but for now it is sufficient to define them as follows.

Definition 4.1. [Negative Polarity Items] *Negative polarity items* are expressions which can appear felicitously only in the scope of monotone decreasing functions.

This definition calls for another one used to identify NPIs' licensors.

Definition 4.2. [Monotone Functions] Let $f : A \to B$ be a function and let $\leq_A$, $\leq_B$ be partial orders on A and B, respectively. Then,

a. f is *monotone increasing* ($\uparrow$Mon) iff $\forall x, y \in A$, $x \leq_A y$ implies $f(x) \leq_B f(y)$.
b. f is *monotone decreasing* ($\downarrow$Mon) iff $\forall x, y \in A$ $x \leq_A y$ implies $f(y) \leq_B f(x)$.

The $\uparrow$Mon functions are also referred to as upward monotone and the $\downarrow$Mon functions as downward monotone. Let us illustrate the meaning of these definitions when applied to linguistic data.

In the classical categorial grammar (CG) approach to natural language, the derivation of sentences is seen as a sequence of function applications starting from the categories assigned to the lexical items in the lexicon. The original references [Ajd35, BH53] deal mainly with syntax, and treat composition as type combination. However, the framework lends itself so naturally to the formal analysis of monotonicity phenomena, that the categorial grammar literature about the application of the monotonicity calculus to natural language is quite broad. A classic reference is [Zwa86] where these kinds of semantic properties are investigated from an algebraic perspective. This methodology permits very sharp semantic analyses of monotonicity phenomena, such as generalized quantifiers. Let us first recall the definition of partially ordered domains we discussed in Section 1.3, which can be used to check the monotonicity property of linguistic signs.

Definition 4.3. [Partially Ordered Domain] Let Dom_a be a domain of type a, where $a \in \mathsf{TYPE}$ and TYPE is built over the set $\{e, t\}$.

If $\beta, \gamma \in Dom_e$, then	$[\![\beta]\!] \leq_e [\![\gamma]\!]$	iff	$[\![\beta]\!] = [\![\gamma]\!]$.
If $\beta, \gamma \in Dom_t$, then	$[\![\beta]\!] \leq_t [\![\gamma]\!]$	iff	$[\![\beta]\!] = 0$ or $[\![\gamma]\!] = 1$.
If $\beta, \gamma \in Dom_{(a,b)}$, then	$[\![\beta]\!] \leq_{(a,b)} [\![\gamma]\!]$	iff	$\forall \alpha \in Dom_a, [\![\beta(\alpha)]\!] \leq_b [\![\gamma(\alpha)]\!]$.

Once we know the monotonicity of linguistic expressions, we also know how to deal with NPIs. For instance, Definition 4.1 correctly predicts the data below where the NPI *yet* and *anybody* are licensed by the $\downarrow$Mon function *nobody* (1-a) and *doubt* (1-c) and are ungrammatical in the scope of the $\uparrow$Mon function *everybody* (1-b) and *think* (1-d).

(1) a. *Nobody* left yet.
b. *Everybody left yet.
c. John *doubts* that anybody left.
d. *John thinks that anybody left.

As soon as we move to consider more complex sentences, an important information about monotone functions, is the way they compose. For example, in (2-a) *anybody* is ungrammatical since *didn't* and *doubt* compose yielding an upward monotone function [Tov96]. Similarly, in (2-b) the two downward monotone functions *not* and *all* compose together and therefore cannot license *anything* [Hoe86].

(2) a. *John *didn't doubt* that anybody left.
b. **Not all* students who know anything about logic know Modus Ponens.

Formally, monotone functions compose as follows.

Proposition 4.4. [Monotone Functions Composition] It is straightforward to prove that the composition of monotone functions follows a *sign rule*. Let $A^+ \to B$ ($A^- \to B$) stand

for an upward (resp. downward) monotone function $f : A \to B$. For $g : A^x \to B$ and $f : B^y \to C$, $f \circ g : A^{sg(x,y)} \to C$, where $sg(x, y) = +$ for $x = y$, and $-$ otherwise. Expressed as a table:

$f \circ g$	$=$	h	$h : A^z \to C$
$\uparrow$Mon $\circ$ $\uparrow$Mon	$=$	$\uparrow$Mon	$h : A^{sg(+,+)} \to C$
$\downarrow$Mon $\circ$ $\downarrow$Mon	$=$	$\uparrow$Mon	$h : A^{sg(-,-)} \to C$
$\uparrow$Mon $\circ$ $\downarrow$Mon	$=$	$\downarrow$Mon	$h : A^{sg(+,-)} \to C$
$\downarrow$Mon $\circ$ $\uparrow$Mon	$=$	$\downarrow$Mon	$h : A^{sg(-,+)} \to C$

Finally, Linebarger [Lin81] shows that negative polarity items must occur in the *immediate scope* of their licensor with no logical elements intervening, where logical elements are expressions able to entering into scope ambiguities. For instance, in (3-a) the reading with *every* as an intervener between the negation and the negative polarity item is ungrammatical. However, there also exist harmless interveners like the bridge predicate *think* which does not block the link between the NPI and its licensor (3-c). Moreover, multiple NPIs can be licensed by the same trigger as in (3-d) taken from [Lad92]. Let [X > Y] mean 'X has scope over Y', and let % mark awkward sentences.

(3) a. Mary *didn't* wear <u>any</u> earrings to every party. [Neg > Any > Every].
b. %Mary *didn't* shout that John had <u>any</u> problems.
c. Mary *didn't* think that John had <u>any</u> problems.
d. *Nobody* said <u>anything</u> to <u>anybody</u>.

The contrast between (3-b) and (3-c) has been explained by assuming that non-bridge verbs like *shout* are essentially quotational and hence embed a structure that contains an illocutionary operator [Kri95]. These examples show that in addition to the logical composition of monotone functions, what matters is the way in which linguistic signs are actually assembled. Therefore, a deductive account of negative polarity items must allow for controls on the way functions compose. We will come back to this point in Sections 4.3 and 4.4. First, let us look at the role monotone functions play in drawing inferences.

4.1.2 Monotonicity in Natural Reasoning

Looking at the definition of monotone functions, it can be seen that monotonicity is closely tied to natural reasoning, since the partial order $\leq$ (Definition 4.3) can be interpreted as the entailment relation. Let us clarify this connection by means of an example.

Example 4.5. [Natural Reasoning Patterns] Consider these intuitively correct inferences:

$$\frac{\text{Everybody (left } \textit{something expensive})}{\text{Everybody (left } \textit{something})}\ (\text{A1}) \qquad \frac{\text{Nobody (}\textit{left}\text{ yet)}}{\text{Nobody (}\textit{left in a hurry}\text{ yet)}}\ (\text{B1})$$

$$\frac{\text{Some boy will } \textit{run fast}}{\text{Some boy will } \textit{run}}\ (\text{A2}) \qquad \frac{\text{No boy will } \textit{run}}{\text{No boy will } \textit{run fast}}\ (\text{B2})$$

$$\frac{\text{Not every } \textit{good logician} \text{ wanders}}{\text{Not every } \textit{logician} \text{ wanders}}\ (\text{A3}) \qquad \frac{\text{Every } \textit{logician} \text{ wanders}}{\text{Every } \textit{good logician} \text{ wanders}}\ (\text{B3})$$

These inferences are valid only because *something* is "more general" than *something expensive*; *left* is "more general" than *left in a hurry*; and *run* and *logician* are more general than *run fast* and *good logician*, respectively. Formally, they involve replacing one expression by another, the denotation of which is a superset (as in A) or a subset (as in B) of the denotation of the original expression.

We aim to derive the above inferences proof theoretically from parsed marked sentences. In order to achieve this goal, first of all we need to have a clearer picture of the kind of inferences we need to model. A substitution of an expression with something more or less general could in fact be done in any position in a sentence. Therefore, the described behavior can be expressed in more general terms by the following inference schemas. Let P and Q stand for linguistic expressions, and let $\leq$ be a partial order between their denotations, respecting their degree of "generality" (see Example 1.31). Then, if $[\![P]\!] \leq [\![Q]\!]$,

$$\frac{N[P]}{N[Q]}\ (\text{A}) \quad \text{or} \qquad \frac{N[Q]}{N[P]}\ (\text{B})$$

The examples (A1), (A2) and (A3) instantiate (A), while (B1), (B2) and (B3) exemplify (B). The question which arises at this point is whether these schemas can always be applied, and how we can decide which of the two produces a valid inference. An answer to this question is given by the monotonicity calculus, and its connection to the computation of the polarity of the position in which the expression occurs.

Before looking at the relation between monotonicity and polarity in formal languages, we want to draw attention on the fact that there exist also environments which do not allow inferences in any direction [Hoe86]. These contexts are created by nonmonotone functions like *exactly three boys*. For example, from *exactly three boys were skating*, one cannot derive either *exactly three boys were moving* nor *exactly three boys were skating fast*. There are two sorts of nonmonotone functions: those which block any kind of substitution, also known as opaque or intentional contexts like *believe*, and those which still allow substitution of equivalent expressions. In the remainder of the chapter we will not take the nonmonotone functions into consideration, but the systems described here could be extended to include them as well.

4.1.3 Monotonicity and Polarity

Monotonicity and polarity are well-known notions in mathematics [Lyn59]. We have already anticipated the definition of monotone functions, what we need to make explicit is the flow of information from the function to its argument.

Definition 4.6. [Monotone Argument Positions] The argument of an ↑Mon function $f : A \to B$ is said to be in an *increasing monotone position*, whereas the argument of a ↓Mon function is in a *decreasing monotone position*.

When interpreting linguistic expressions as functions and the construction of sentences as function composition, the monotonicity calculus gives us the required information to decide which one of the inference schemas given above can be applied for drawing correct inferences: an expression in an increasing (resp. decreasing) monotone position is substituted with a more (resp. less) general one following (A) (resp. (B)). Let us go back to Example 4.5 and consider again the inference in (B2), trying to apply the intuitions we just discussed.

Example 4.7. Suppose we take a functional perspective and consider the linguistic phrases *no boy* and *will* as functions represented as `no_boy` and `will` and the sentence *No boy will run* as the result of the functional application of the composed function `no_boy` ∘ `will` to `run`. Using the standard linguistic categories, the function representing *run* is of category *iv* (intransitive verb), and the whole sentence is *s* (sentence).

$$\begin{array}{r} \texttt{no\_boy} : iv^{-} \to s \\ \texttt{will} : iv^{+} \to iv \\ \texttt{no\_boy} \circ \texttt{will} : iv^{sg(+,-)} \to s \\ \texttt{no\_boy} \circ \texttt{will} : iv^{-} \to s \\ \texttt{no\_boy} \circ \texttt{will} \circ \texttt{run} : s \end{array}$$

(1) no boy will run^{-}

$$\frac{\text{no boy will } \textit{run}^{-}}{\text{no boy will } \textit{run fast}}$$

The monotonicity of the composed function `no_boy` ∘ `will` gives us the information required to derive the inference above. But suppose we want to derive *no boy or no girl will run* from (1). The monotonicity calculus does not help us there, as we are substituting in a function position, instead of in an argument position. We need to extend the monotonicity calculus, so that it will also yield marked positions for these cases, and this is not as straightforward as it seems, because we need a way to refer explicitly to the functions. We need to compute the polarity of the position where the function occurs.

In first order logic (FOL) polarity is defined in terms of the number of negations surrounding a subformula [Lyn59]. But FOL does not capture the compositional behavior needed for formalizing natural language functional application. In Section 1.3, we have seen that a way of properly speaking about functions is provided by the lambda calculus [Chu40]. By introducing it into the picture we have the final missing ingredient.

Briefly, the set of typed lambda terms is defined as follows. For any type a, let CON_a be a set of constants of type a and VAR_a be a set of variables of type a, then the set of typed lambda TERM_a expressions over CON_a and VAR_a is given by

$$\begin{array}{lcl} \mathsf{TERM}_a & := & c_a \mid x_a \mid (\mathsf{TERM}_{(b,a)}\ \mathsf{TERM}_b) \\ \mathsf{TERM}_{(b,a)} & := & \lambda x_b.\mathsf{TERM}_a. \end{array}$$

where $c_a \in \mathsf{CON}_a$ and $x_a \in \mathsf{VAR}_a$. The operator λ is said to *bind* x; unbounded variables are said to be *free*, we abbreviate with $FV(M)$ the set of free variables in M. For this language it is now possible to define both monotone and polarity positions. The definitions were originally given in [Ben86] and further explored in [SV91].

Definition 4.8. [Monotone Occurrence] Let N'_a be a lambda term like N_a except for containing an occurrence of M'_b where N_a contains M_b (*viz.* $N'_a = N_a[M'_b/M_b]$), where a, b stand for the type of the indexed terms.

i. N_a is *upward monotone in* M_b iff for all models $\mathcal{M}$ and assignments f: $[\![M]\!]^f_{\mathcal{M}} \leq_b [\![M']\!]^f_{\mathcal{M}}$ entails $[\![N]\!]^f_{\mathcal{M}} \leq_a [\![N']\!]^f_{\mathcal{M}}$;

ii. N_a is *downward monotone in* M_b iff for all models $\mathcal{M}$ and assignments f: $[\![M]\!]^f_{\mathcal{M}} \leq_b [\![M']\!]^f_{\mathcal{M}}$ entails $[\![N']\!]^f_{\mathcal{M}} \leq_a [\![N]\!]^f_{\mathcal{M}}$.

Definition 4.9. [Polarity of Occurrences] Given a lambda term N and a subterm M of N. A specified occurrence of M in N, is called *positive* (*negative*) according to the following clauses:

i. M is positive in M.

ii. M is positive (negative) in PQ if M is positive (negative) in P.

iii. M is positive (negative) in PQ if M is positive (negative) in Q, and P denotes an upward monotone function.

iv. M is negative (positive) in PQ if M is positive (negative) in Q, and P denotes a downward monotone function.

v. M is positive (negative) in $\lambda X.P$ if M is positive (negative) in P and $X \notin FV(M)$.

Having added the lambda notation we can complete Example 4.7. The sentence *no boy will run* can be represented by (`no_boy will`)`run`, where `no_boy`, `will` and `run` are constants: the first one is a downward monotone function and the others two are upward monotone. By Definition 4.9, the function (`no_boy`) receives a positive polarity, and given than *no boy or no girl* is more general that *no boy* we are allowed to monotonically conclude *no boy or no girl will run* from *no boy will run*.

The relation between monotonicity and polarity in lambda terms has been studied in [Ben86, Ben91, SV91] where it is proved that the polarity of the occurrences implies their monotonicity.

Proposition 4.10. If M_b is positive (resp. negative) in N_a, then N_a is upward (resp. downward) monotone in M_b.

Corollary 4.11. If X_a is positive (resp. negative) in N_b, then $\lambda X_a.N_b$ denotes an upward (resp. downward) monotone function.

The differences between monotonicity and polarity could be summarized in a few words by saying that monotonicity is a semantic property of functions which is dynamically passed to the argument positions while building a formula. On the other hand, polarity is a static syntactic notion which can be computed for all positions in a given formula. This connection between the semantic notion of monotonicity and the syntactic one of

polarity is what one needs to reach a proof theoretical account of natural reasoning and build a natural logic.

Categorial type logic (CTL) derivations can be interpreted as lambda terms due to the Curry-Howard correspondence [CF68, How80, Ben88] (see Section 1.4). This correspondence makes it possible to associate CTL derivations with the notion of polarity on lambda terms (Definition 4.9) and consequently, by Proposition 4.10, with their monotone positions [Ben86]. In other words, CTL derivations can supply the information required for deriving monotone inference from the parsed linguistic structures.

By linking up the two topics here introduced, we recall that our aim is twofold. We want to encode monotonicity information and compute the polarity positions within parsed structures so as to account for negative polarity distribution and produce structures readily available for deriving monotone inferences.

4.2 A Natural Logic based on LP

In [SV91] Sánchez works out the proposal of van Benthem [Ben86]. The syntactic issues of function composition is connected to the semantic features of monotonicity inference. This link is obtained extra-logically by means of an algorithm *working on* derivations of the associative and commutative Lambek calculus (LP) (Section 1.2), which is able to correctly mark the parsed strings determining the different polarity positions. Due to the use of an external algorithm to compute the polarity of the nodes in LP derivations, we will refer to this system as LP+EPol . For the ease of presentation here we use our notation while describing LP+EPol derivations. As a consequence we use two implicational operators $\backslash, /$ though in LP they collapse into one. The original presentation is given in Appendix A, where LP+EPol is embedded into a multimodal categorial type logic.

Given a derivation, the algorithm starts from marked leaves and propagates these markers through the proof by labelling the nodes of the logical rules. First, the logical types in each leave are enriched with monotonicity markers encoding the monotonicity property of the corresponding linguistic entry. Let A/B be the type assigned to an upward (resp. downward) monotone function, then A/B^+ (resp. A/B^-) is the marked type. Similarly for $B\backslash A$. Types corresponding to variables in the lambda terms are left unspecified A/B, $(A\backslash B)$. From this information, the algorithm proceeds by propagating the monotonicity markers $+, -$ from the leaves through the derivation. Finally, the polarity of the nodes in the derivation is computed: a negative marker flips the monotonicity of all nodes above it in the derivation, *viz.* $+$ becomes $-$ and *vice versa*, as required by Proposition 4.4 and Definition 4.9. An unmarked node breaks the polarity assignment leaving the nodes above it in the branch unspecified. The final result is a parsed output in which polarity positions are correctly displayed, and which can be used as a vehicle of natural reasoning inference. The formal definition of the algorithm consists of the two parts below.

Monotonicity Markers: Let $\mathcal{D}$ be a derivation of LP, and $\mathcal{D}'$ be the derivation $\mathcal{D}$ with the types of the lexical items marked according to their monotonicity properties. The monotonicity markers are copied from the functions to their arguments in the derivation

$$\frac{\Delta \vdash B/A \quad \Gamma \vdash A}{\Delta \circ \Gamma \vdash B}\,[/\mathrm{E}] \quad \text{rewrites to} \quad \frac{\Delta \vdash \overset{+}{B}/A \qquad \Gamma \vdash A}{\Delta \circ \Gamma \vdash B}\,[/\mathrm{E}]$$

$$\frac{\Delta \vdash B/A^{x} \quad \Gamma \vdash A}{\Delta \circ \Gamma \vdash B}\,[/\mathrm{E}] \quad \text{rewrites to} \quad \frac{\Delta \vdash \overset{+}{B}/A^{x} \qquad \Gamma \vdash \underset{x}{A}}{\Delta \circ \Gamma \vdash B}\,[/\mathrm{E}]$$

$$\frac{\begin{array}{c}[A \vdash A]^{i}\\ \vdots\\ \Gamma \circ A \vdash B\end{array}}{\Gamma \vdash B/A}\,[/\mathrm{I}]^{i} \quad \text{rewrites to} \quad \frac{\begin{array}{c}[A \vdash A]^{i}\\ \vdots\\ \Gamma \circ A \vdash \overset{+}{B}\end{array}}{\Gamma \vdash B/A^{y}}\,[/\mathrm{I}]^{i}$$

Figure 4.1: Monotonicity algorithm of LP+EPol.

by means of the rewrite rules in Figure 4.1 where $x \in \{+, -\}$. Similar rules hold for $\backslash$. The value of y in $[/\mathrm{I}]^{i}$ is determined as follows: y is $-$ (resp. $+$) if all the nodes in the path from $\Gamma \circ A \vdash B$ to $[A \vdash A]$ are marked, and the number of nodes marked with $-$ in the path is odd (resp. even). As a last step, the monotonicity algorithm marks the root of the derivation with $+$.

Polarity Markers: Given a marked derivation $\mathcal{D}$ a node is assigned a polarity marker $+$ or $-$ if all the nodes in the path from the node to the root are marked. The node is $-$ if the number of nodes marked with $-$ in the path is odd, and $+$ otherwise.

Note that unspecified polarity nodes will occur only in derivations involving abstraction. They are in fact produced when the formula corresponding to the node is taken as argument by variables, or in other words when it is the minor premise of a functional application [/E] or [\E] having a hypothesis (or a subformula of a hypothesis) in the major premise. Let us see this algorithm at work.

Example 4.12. [LP+EPol Functional Applications] Given the set of lexical entries $\{\text{not} \in s/s, \text{wanders} \in np\backslash s, \text{good\_logician} \in n, \text{every} \in (s/(np\backslash s))/n\}$, the sentence *not every good logician wanders* is proved to be of type s in LP as follows:

$$\dfrac{\text{not} \vdash s/s \qquad \dfrac{\dfrac{\text{every} \vdash (s/(np\backslash s))/n \quad \text{good\_logician} \vdash n}{\text{every} \circ \text{good\_logician} \vdash s/(np\backslash s)}\,[/\mathrm{E}] \qquad \text{wanders} \vdash np\backslash s}{(\text{every} \circ \text{good\_logician}) \circ \text{wanders} \vdash s}\,[/\mathrm{E}]}{\text{not} \circ ((\text{every} \circ \text{good\_logician}) \circ \text{wanders}) \vdash s}\,[/\mathrm{E}]$$

Now, we need to encode the monotonicity information for the lexical entries, namely the leaves of the derivation. From formal semantics, we know that *not* and *wanders* are downward and upward monotone functions, respectively; whereas *every* is downward monotone in its first argument and upward monotone in its second argument. The derivation is labelled accordingly by replacing the functional types at the leaves of the above derivation with the corresponding marked ones: $s/s^{-}, (s/(np\backslash s)^{+})/n^{-}$ and

$np^+\backslash s$. These monotonicity markers are then propagated through the derivation as illustrated below in (a). Finally, the polarity positions of the nodes are computed from the monotone ones in (b). We give below the skeleton of the marked derivations.

(a) **Monotonicity markers**

$$\dfrac{\dfrac{(s/(np\backslash s)^+)/n^-\quad n}{s/(np\backslash s)^+}\quad np^+\backslash s}{\dfrac{s/s^-\quad s}{s}}$$

(b) **Polarity markers**

$$\dfrac{\dfrac{(s/(np\backslash s))/n\quad n}{s/(np\backslash s)}\quad np\backslash s}{\dfrac{s/s\quad s}{s}}$$

Note how in (a) the negative marker carried by 'not', being a downward monotone function, is passed to the expression taken as argument, 'every good_logician wanders'. This reflects what was expressed in Definition 4.6. Moreover, in (b) the negative monotonicity marker assigned to the corresponding node flips (reverses) the markers assigned to the nodes above it, mirroring Definition 4.9-(iv). For each node in the polarity marked derivation, if a node is labelled with a + (−), the corresponding term in the final lambda term representing the whole sentence, `Not`((`Every good_logician`) `wanders`), is in a positive (negative) position. For instance, the node n is assigned a + and the lambda term corresponding to it, `good_logician`, is in a positive polarity position. Finally, note that the root of the derivation receives a positive polarity marker, resembling the first point of Definition 4.9, M is positive in M.

We can now display the polarity markers on the linguistic structures corresponding to each node obtaining a marked parsed string: $(\text{not}^+((\text{every}^-\ \text{good\_logician}^+)^-\ \text{wanders}^-)^-)^+$. Due to the link between polarity and monotonicity (Proposition 4.10 and Definition 4.8), this structure can be used to derive monotone natural reasoning inferences. For instance, the inference (A3) in Example 4.5 can be derived by replacing 'good_logician' with the more general term 'logician'. We use the thick inference line to distinguish this new inference from the logical and structural rules of the logical system.

$$\dfrac{(\text{not}^+((\text{every}^-\text{good\_logician}^+)^-\text{wanders}^-)^-)^+ \vdash s}{(\text{not}^+((\text{every}^-\text{logician}^+)^-\text{wanders}^-)^-)^+ \vdash s}$$

Note that the information regarding the partial order holding among the expression involved in the substitution is still based on formal semantics and it is computed on the corresponding lambda terms. See [FWF00] for a natural logic where the order relation is computed within the system.

Besides functional applications, LP derivations can contain abstraction rules. As an example of the latter, we look at the lifting of an np to a higher order type.

Example 4.13. [Abstraction in LP+EPol] Let mary $\in np$ be our lexicon entry. The lifted type $s/(np\backslash s)$ is obtained as follows:

$$\dfrac{\dfrac{\text{mary} \vdash np \quad [x \vdash np\backslash s]^1}{\text{mary} \circ x \vdash s}\,[\backslash\text{E}]}{\text{mary} \vdash s/(np\backslash s)}\,[/\text{I}]^1$$

Following the algorithm, since x is a variable, it is an unspecified monotone function and its type is left unmarked.

(a) **Monotonicity markers**

$$\dfrac{\dfrac{np \qquad [np\underset{+}{\backslash} s]^1}{s}\,[\backslash\text{E}]}{s/(np\backslash s)^{+}}\,[/\text{I}]^1$$

(b) **Polarity markers**

$$\dfrac{\dfrac{np \qquad [np\underset{+}{\backslash} s]^1}{\underset{+}{s}}\,[\backslash\text{E}]}{\underset{+}{s/(np\backslash s)^{+}}}\,[/\text{I}]^1$$

The point of interest in this case is the role played by the assumed function x. Since its monotonicity could be either positive or negative, the monotone argument position where 'mary' occurs is also unspecified. Furthermore, when the hypothesis is discharged an upward monotone function is built: the abstraction is over a variable in a positive polarity position (Corollary 4.11).

The role of the hypothetical reasoning in the assignment of polarity positions can be better illustrated by looking at an example involving coordination of an np with a higher order type, requiring the application of the lifting theorem above.

Example 4.14. [Coordination] The coordination *and* is an upward monotone function in both arguments. Its lambda term representation is: $\lambda RQZ.QZ \wedge RZ$. The lambda term of the coordinated phrase is built as follows.

$$\dfrac{\begin{array}{c}\text{m} \vdash \texttt{m} \\ \vdots \\ \text{mary} \vdash \lambda P.P\texttt{m}\end{array} \qquad \dfrac{\text{and} \vdash \lambda RQZ.QZ \wedge RZ \quad \text{every\_logician} \vdash \texttt{Every\_logician}}{(\text{and} \circ \text{every\_logician}) \vdash \lambda QZ.QZ \wedge \texttt{Every\_logician}\ Z}}{\text{mary} \circ (\text{and} \circ \text{every\_logician}) \vdash \lambda Z.Z\texttt{m} \wedge \texttt{Every\_logician}\ Z}$$

In the final lambda term the polarity of 'mary', represented by `m`, is still unspecified. It depends on the monotonicity of the verb phrase which will be taken as an argument and will replace the variable Z. For instance, if the coordinated structure is applied to the verb phrase *left* it yields *mary and every logician left* represented by: `Left m` $\wedge$ `Every_logician Left` where `m` is in a positive polarity position.

Monotonicity Marking

$$\dfrac{\begin{array}{c}\text{mary} \vdash np \\ \vdots \\ \text{mary} \vdash \underset{+}{s/(np\backslash s)^{+}}\end{array} \qquad \dfrac{\text{and} \vdash ((s/(np\backslash s))^{+}\backslash(\underset{+}{s/(np\backslash s)^{+}}))/(s/(np\backslash s))^{+} \quad \text{every\_logician} \vdash \underset{+}{s/(np\backslash s)^{+}}}{(\text{and} \circ \text{every\_logician}) \vdash \underset{+}{(s/(np\backslash s))^{+}\backslash(s/(np\backslash s)^{+})}}}{(\text{mary} \circ \text{and}) \circ \text{every\_logician} \vdash \underset{+}{s/(np\backslash s)^{+}}}$$

Sánchez' algorithm does not assign a polarity to 'mary', leaving it unspecified even after the application of the coordinated structure to a verb phrase, since the assignment of the polarity requires that *all* the nodes must be marked and the leaf 'mary $\vdash$ *np*' is not.

By extending the result in [Ben91] Sánchez proves in [SV91] the soundness of LP+EPol. By interpreting derivations as lambda terms, marked functional nodes will correctly correspond to monotone functions, and polarity markers will correctly correspond to polarity positions in the lambda terms[1]. Therefore, LP+EPol properly accounts for the tasks it was built for: it generates marked parsed outputs ready for deriving monotonicity inference. But this was not our original aim. Let us look back at what we wanted to achieve.

Our project was to build a system able to (i) encode monotonicity information, (ii) compute polarity positions, (iii) use this information while parsing (to verify correct use of NPIs) and (iv) produce marked output from which monotonicity inferences can be derived. Though LP+EPol can account for (i),(ii) and (iv), it does not account for (iii). This shortcoming is due to the fact that the monotonicity and polarity algorithms are external to the logic: they mark the derivation only *after* the grammaticality of the linguistic structure is established. Hence, polarity markers do not play an active role in the derivation and cannot be used to control the grammaticality of linguistic structures as required by the NPIs. A solution can be obtained by internalizing the marking algorithms.

4.3 Internalizing Polarity Marking in CG

Dowty [Dow94] proposes a natural logic based on classical categorial grammar (CG+Pol), in which the independent steps of monotonicity and polarity marking collapse into a syntactic derivation. In this approach the markers '+' and '–' are used to indicate the final polarity. The main characteristics of Dowty's system are:

a. Since one and the same word can appear with positive polarity in one derivation and with negative polarity in another, most lexical items —with the important exception of negative polarity items[2]— will have both a '+' and a '–' marked category, with the same interpretation.

b. ↑Mon functions are assigned categories of the forms A^+/B^+ and A^-/B^-, meaning that they *preserve* the polarity markers. We will mark them as A^x/B^x.

c. ↓Mon functions are assigned categories of the forms A^+/B^- and A^-/B^+, since they *reverse* the polarity markers. We will mark them as A^x/B^y.

For complex categories we use the convention: $(A/B)^x =_{def} (A^x/B)^x =_{def} (A^x/B)$, and similarly for $(A\backslash B)^x$)[3].

[1]The case discussed in the Example 4.14 does not constitute a problem for the soundness proof which only concerns marked nodes.

[2]Dowty considers also positive polarity items, as expressions required to occur in a positive polarity context.

[3]This definition is based on the fact that in a function-argument combination the function always has the same polarity as the combination as a whole.

Functional application respects the polarity markers in the following way. Let $x, y \in \{+, -\}$, then

$$\frac{A^x/B^y \quad B^y}{A^x}$$

where 'x' and 'y' coincide when the major premise A^x/B^y is an ↑Mon function, and differ when is a ↓Mon function.

The grammar thus defined generates sentences of category S^+ or S^-. The former is the category of independent (grammatical) sentences, the latter of sentences embedded inside a ↓Mon function. We will assume the lexical entries below. Let $x \in \{+, -\}$, let y be the "opposite" of x, and let $VP = NP\backslash S$,

Lexical Entries

walks $= VP^x$	**a** $= (S^x/VP^x)/CN^x$
reads $= VP^x/NP^x$	**no** $= (S^x/VP^y)/CN^y$
John $= S^x/VP^x$	**every** $= (S^x/VP^x)/CN^y$
doesn't $= VP^x/VP^y$	**any** $= (S^-/VP^-)/CN^-$

We present some examples of Dowty-style polarity marking.

Example 4.15. [Polarity Marking]

1. *John walks.*

$$\dfrac{\dfrac{\text{John}}{S^+/VP^+} \qquad \dfrac{\text{walks}}{VP^+}}{S^+}$$

2. *John doesn't walk.*

$$\dfrac{\dfrac{\text{John}}{S^+/VP^+} \qquad \dfrac{\dfrac{\text{doesn't}}{VP^+/VP^-} \qquad \dfrac{\text{walk}}{VP^-}}{VP^+}}{S^+}$$

Comparing these two derivations, note how the polarity of the VP is changed by the presence of the ↓Mon function 'does not'. Using a bottom-up reading, the inference says that if 'John doesn't walk' is a well-formed independent sentence S^+, then 'John', the last function applied, has to have S^+ as a value. Since 'John' is an ↑Mon function, this means its argument must be marked with a '+' as well. This requires 'doesn't walk' to be of category VP^+. Therefore, the value of the ↓Mon function 'doesn't' must be marked with the '+' as well, and consequently its argument is marked negatively.

To deal with more complex sentences in which a generalized quantifier occurs in the object position, Dowty includes in the lexicon for each determiner $(S^x/VP^y)/CN^z$ an object counterpart, of category $(TV^y\backslash VP^x)/CN^z$, where $TV^y = (NP^y\backslash S^y)/NP^y$. The object counterparts of the above given lexical entries 'a', 'no' and 'any' are as follows:

$$\textbf{a} = (TV^x \backslash VP^x)/CN^x \qquad \textbf{no} = (TV^x \backslash VP^y)/CN^x$$
$$\textbf{any} = (TV^- \backslash VP^-)/CN^- \qquad \textbf{every} = (TV^x \backslash VP^x)/CN^y$$

Example 4.16. [Negative Polarity Item]

1. *No boy reads any book.*

$$\dfrac{\dfrac{\dfrac{\text{no}}{(S^+/VP^-)/CN^-} \quad \dfrac{\text{boy}}{CN^-}}{S^+/VP^-} \quad \dfrac{\dfrac{\text{reads}}{TV^-} \quad \dfrac{\dfrac{\text{any}}{(TV^- \backslash VP^-)/CN^-} \quad \dfrac{\text{book}}{CN^-}}{TV^- \backslash VP^-}}{VP^-}}{S^+}$$

If we replace 'no' with a determiner which is upward monotone in its second argument, e.g. 'every', the derivation fails. In order to match the VP^- category of *reads any book*, 'every' has to be considered of category $(S^-/VP^-)/CN^+$ and the whole expression *every boy reads any book* would be proved to be of category S^- rather than S^+, where S^- is the category of the embedded sentence in the scope of the ↓Mon function.

This example brings us to look at NPIs in embedded sentences. Remember from Section 4.1 that NPIs in embedded sentences may be licensed by a downward monotone function in the matrix clause in two cases: either they are the object of a negative predicate (1-c), or there is a predicate bridging the NPI to its licensor (3-c). On the other hand, the composition of a negative predicate with another downward monotone function cancels their licensing property by yielding an upward monotone function (2-a), and an intervener blocks the licensing relation (3-a).

The polarity marking of CG+Pol correctly predicts (1-c) and (2-a) as shown by the example below, where the lexical entry for *doubt* is VP^y/S^x (as it is a ↓Mon function).

Example 4.17. [Embedded NPIs]

1. *John doubts anybody left.*

$$\dfrac{\dfrac{\text{John}}{S^+/VP^+} \quad \dfrac{\dfrac{\text{doubts}}{VP^+/S^-} \quad \dfrac{\dfrac{\text{anybody}}{S^-/VP^-} \quad \dfrac{\text{left}}{VP^-}}{S^-}}{VP^+}}{S^+}$$

2. **John didn't doubt anybody left.*

$$\dfrac{\dfrac{\text{John}}{S^-/VP^-} \quad \dfrac{\dfrac{\text{didn't}}{VP^-/VP^+} \quad \dfrac{\text{doubts anybody left}}{\vdots}}{VP^+}}{VP^-}}{S^-}$$

However, the marking does not correctly account for the sentence in (4) since *every boy reads any book* is assigned category S^- which can be taken as argument by *doubts* contrary to linguistic reality: the quantifier *every boy* works as an intervener blocking the licensing of the NPI.

(4) John doubts every boy reads any book. [*Doubt > Every > Any].

Furthermore, NPIs in the scope of two downward monotone functions do not always result in ungrammatical constructions, as exemplified by the sentences below [Hoe86].

(5) a. *If* he knows anything about logic, he will know modus ponens.
b. *If* he *doesn't* know anything about logic, he will not know modus ponens.

Again, CG+Pol makes wrong predictions in this case as well, blocking (5-b).

By internalizing the monotonicity and polarity information into the logical types, Dowty increases the expressivity of a CG so as to deal with NPIs and compute the polarity of the node required for the natural logic task. On the other hand, the flow of the markers from the argument to the functional types expressed by CG+Pol lexical type assignments is too strong to model linguistic facts. This causes the failure of CG+Pol in dealing with NPIs in embedded sentences and in controlling functional composition. Finally, note that although Dowty's system makes a first step towards a system with internalized monotonicity and polarity markers, it does not accomplish this change completely: the marking of a linguistic structure must still be done 'externally' by propagating the marker in the formula.

4.4 Internalizing Monotonicity and Polarity Markers in MCTL

In this section, we present a natural logic based on a multimodal categorial type logic with (polarity) structural rules (MCTL+Pol) where the latter are used simply as a tool for computing polarity position. Since the polarity of any position is positive unless modified by downward monotone functions, we leave the upward monotone functions unmarked and employ the unary operator $\diamondsuit$ to mark downward monotone functions[4]. Consequently, the corresponding unary structural connective $\langle\cdot\rangle^-$ marks a structure in a downward monotone argument position. Due to the logical relation holding between $\diamondsuit$ and $\boxdot^\downarrow$ (Section 2.1.2), and the link between monotonicity and polarity (Section 4.1.3), $\boxdot^\downarrow$ encodes (negative) polarity information, as we will show below.

The dynamic flow of information from the function to the argument (Section 4.1.3) is directly accounted for by the logical rules without the need of an external monotonicity marking algorithm. Similarly, structural rules allow us to internalize the polarity algorithm producing marked structures instead of marking the corresponding nodes. A first clear advantage of the move from LP to MCTL is that the marked structures are readily available for deriving monotone inference, improving on the natural logic based on LP+EPol where the polarity marker were 'externally' displayed on the structures once read off the nodes. Furthermore, NPI distribution can be controlled, since polarity information is encoded in the logical types.

For the sake of simplicity we consider a product-free logic, similar to the systems considered so far. We repeat below the logical and structural languages of MCTL. The

[4]Note that the mode index is simply used to emphasize its role as a downward monotone marker.

full system is presented in Chapter 2, and an introduction of the linguistic application of the binary and unary operators is given in Sections 1.2 and 3.1.

DEFINITION 4.18. [Languages] Logical Language: Given a set of basic categories ATOM, the set of categories FORM is built over $\backslash, /, \diamondsuit$ and $\boxdot^{\downarrow}$

$$\mathsf{FORM} := \mathsf{ATOM} \mid \mathsf{FORM}/\mathsf{FORM} \mid \mathsf{FORM}\backslash\mathsf{FORM} \mid \diamondsuit\mathsf{FORM} \mid \boxdot^{\downarrow}\mathsf{FORM}.$$

Structural Language: The set of structures STRUCT is built over the set of logical categories, by means of $\circ$, and $\langle\cdot\rangle^{-}$.

$$\mathsf{STRUCT} := \mathsf{FORM} \mid \langle\mathsf{STRUCT}\rangle^{-} \mid \mathsf{STRUCT} \circ \mathsf{STRUCT}.$$

These languages are used to encode monotonicity and polarity information as summarized in Table 4.1.

To express that:	type
A structure Γ is an $\uparrow$Mon function	$\Gamma \vdash B/A$ or $\Gamma \vdash A\backslash B$
A structure Γ is a $\downarrow$Mon function	$\Gamma \vdash B/\diamondsuit A$ or $\Gamma \vdash \diamondsuit A\backslash B$
A structure Γ has polarity $-$	$\langle\Gamma\rangle^{-}$
A structure Γ has polarity $+$	$\Gamma \neq \langle\Gamma'\rangle^{-}$
A structure Γ must have a polarity $-$	$\Gamma \vdash \boxdot^{\downarrow}A$

Table 4.1: Encoding of monotonicity and polarity information

To express this encoding it is enough to use a fragment of the logical language of MCTL. In the next section we will show that the use of types outside this fragment could lead to incorrect polarity assignments.

DEFINITION 4.19. [Safe Types] To account for NPI distribution and monotone inferences it is enough to mark only downward monotone functions and leave the upward monotone ones unmarked. Therefore, we need to work only with lexical type assignments from the set FORM_1 defined below,

$$\begin{aligned}\mathsf{FORM}_1 = \quad & \mathsf{ATOM} \mid \mathsf{FORM}_1/\diamondsuit\mathsf{FORM}_1 \mid \diamondsuit\mathsf{FORM}_1\backslash\mathsf{FORM}_1 \mid \\ & \mathsf{FORM}_1\backslash\mathsf{FORM}_1 \mid \mathsf{FORM}_1/\mathsf{FORM}_1 \mid \boxdot^{\downarrow}\diamondsuit\mathsf{FORM}_1 \mid \\ & \boxdot^{\downarrow}\diamondsuit\mathsf{FORM}_1/\boxdot^{\downarrow}\diamondsuit\mathsf{FORM}_1 \mid \boxdot^{\downarrow}\diamondsuit\mathsf{FORM}_1\backslash\boxdot^{\downarrow}\diamondsuit\mathsf{FORM}_1.\end{aligned}$$

where $\boxdot^{\downarrow}\diamondsuit\mathsf{FORM}_1/\boxdot^{\downarrow}\diamondsuit\mathsf{FORM}_1$ (and $\boxdot^{\downarrow}\diamondsuit\mathsf{FORM}_1\backslash\boxdot^{\downarrow}\diamondsuit\mathsf{FORM}_1$) will be used for lexical assignments of negative polarity items. The prefix $\boxdot^{\downarrow}\diamondsuit$ on the argument formula will allow multiple NPI occurrences, whereas on the value formula it will allow the marking of the context where the NPI occurs, the attraction of a licensor and the rejection of an upward monotone function. In Section 4.4.4 we will discuss how the use of this fragment of the logical language effects the possible derivations the system can make.

Based on the selection of this set of formulas we can identify a set of types which properly represent the polarity of the corresponding structure; we refer to them as *safe types*.

$$\mathsf{SAFE} = \mathsf{FORM}_1 \mid \Diamond \mathsf{FORM}_1.$$

Note that the set of safe types is the same as the set of all subformulas of the formulas in FORM_1.

LEMMA 4.20. Given a derivation $\mathcal{D}$ of $\Gamma \vdash A$, if $A \in \mathsf{FORM}_1$ and all the formulas in Γ are in SAFE, then all nodes in $\mathcal{D}$ are labeled by safe types.

PROOF. The lemma is a consequence of the subformula property of MCTL+Pol [Sza69, Moo97] and the definition of safe types. QED

In the remainder of the chapter we will consider only derivations with safe types, unless explicitly stated otherwise.

Sánchez' polarity marking algorithm is carried out by the structural rules in Figure 4.2. The [Pol$_-$] rule simply distributes markers into the substructures, and [Pol$_{--}$] cancels them out.

$$\frac{\Delta[\langle \Gamma_1 \circ \Gamma_2 \rangle^-] \vdash A}{\Delta[\langle \Gamma_1 \rangle^- \circ \langle \Gamma_2 \rangle^-] \vdash A}\ [\mathrm{Pol}_-] \qquad \frac{\Delta[\langle \langle \Gamma \rangle^- \rangle^-] \vdash A}{\Delta[\Gamma] \vdash A}\ [\mathrm{Pol}_{--}]$$

Figure 4.2: Structural Rules of MCTL+Pol.

The structural rule [Pol$_-$] is not in the class of those structural rules which guarantee the logical grammar to be in PSPACE, since it increases the length of the structure (Definition 1.15). However, in a normalized derivation [Pol$_{--}$] always precedes [Pol$_-$]. This is possible because the two rules are confluent. Therefore, when no other structural rules apply to mode -, the total maximum space for an MCTL+Pol sequent is linear with respect to the end-sequent, and hence MCTL+Pol is in PSPACE [Moo02][5].

PROPOSITION 4.21. [Complexity of MCTL+Pol] MCTL+Pol is decidable at most in deterministically polynomial space.

4.4.1 A Natural Logic based on MCTL+Pol

In Definition 4.23 we will formally define the intuitive encoding of polarity marked structures given in Table 4.1. First we need the definition below.

DEFINITION 4.22. [Normal Form for Sequents] A sequent is said to be in *normal form* if no polarity structural rules can be applied to it.

Note that any sequent $\Gamma \vdash A$ can be normalized by applications of polarity structural rules reaching a unique sequent $\Gamma' \vdash A$ in normal form.

[5]This complexity property of MCTL+Pol was pointed out to me by Richard Moot.

Definition 4.23. [Polarity of Structures] The polarity of a structure is defined in terms of $\langle\cdot\rangle^-$. Let $\Delta \vdash B$ be a normalized sequent and Γ a substructure of Δ,

(a) Let $\Gamma = \diamond\!\!\!\!-\, A$, where $A \in \mathsf{FORM}_1$. Γ is said to have *negative* polarity in Δ iff Γ is surrounded by no $\langle\cdot\rangle^-$, and positive polarity otherwise.

(b) Let $\Gamma = \langle A\rangle^-$, where $A \in \mathsf{FORM}_1$. Γ is said to have *positive* polarity in Δ iff A is of the form $\diamond\!\!\!\!-\, A'$, and negative polarity otherwise.

(c) Let $\Gamma = (\Gamma_1 \circ \Gamma_2)$. Γ is said to have *negative* polarity in Δ iff both Γ_1 and Γ_2 have negative polarity; positive polarity if both Γ_1 and Γ_2 have positive polarity, and undetermined polarity otherwise[6].

To understand the way polarity is carried out by the structural language, one has to keep in mind that $\langle\cdot\rangle^-$ is the structural connective corresponding to $\diamond\!\!\!\!-$, the operators which we used to denote decreasing monotone argument positions, and that polarity is linked to monotonicity (Definition 4.9).

Definition 4.24. [Monotonicity Rules] Given a derivation $\mathcal{D}$ of a sequent $\Delta \vdash N : C$ in normal form. Let M_A be a subterm of N corresponding to a substructure Γ in Δ such that $\Gamma \neq \langle\Gamma'\rangle^-$, *viz.* in $\mathcal{D}$ there is a subderivation $\mathcal{D}'$ of $\Gamma \vdash M : A$. Let M', M'' be two terms such that $[\![M'']\!] \leq_A [\![M]\!] \leq_A [\![M']\!]$, and let Γ_1, Γ_2 be two structures corresponding to M', M'', respectively. Then the following inference can be derived.

$$\text{(a)}\quad \frac{\Delta[\Gamma^+] \vdash N{:}C}{\Delta[\Gamma_1^+] \vdash N'{:}C} \qquad \text{(b)}\quad \frac{\Delta[\Gamma^-] \vdash N{:}C}{\Delta[\Gamma_2^-] \vdash N''{:}C}$$

where $N' = N[M'/M]$ and $N'' = N[M''/M]$, Γ_i^+ (resp. Γ_i^-) stand for a structure with positive (resp. negative) polarity.

Note how the formal definition of the monotonicity rules, intuitively described earlier in our discussion of LP+EPol, sheds light on the use of the lambda terms to compute the substitution of the marked structures. In the discussion of the examples we will skip the lambda terms and concentrate on the structures on which the inference is performed.

4.4.2 MCTL+Pol at Work

As in Sánchez' approach, to determine the polarity at the sentence level, we start by assigning monotonicity markers to the entries in the lexicon. From the semantic information we know the monotonicity property of a functional lexical entry. We distinguish ↓Mon functions by prefixing their argument with $\diamond\!\!\!\!-$. For instance, since *doubt* is ↓Mon in its first parameter, it is of type: $(np\backslash s)/\diamond\!\!\!\!-\, s$. Note that it can be represented by the lambda term $\lambda x_s.\lambda y_{np}.(\texttt{doubt}\ x)\ y$, where `doubt` is a downward monotone (functional)

[6] As observed while discussing Dowty's system, the polarity of a function-argument combination is equivalent to the polarity of the function. In a multimodal system, the information about the function-argument position could be encoded by means of modes on the binary operators [MM91]. In a system so extended it would be possible to define also the polarity of a structure composed out of two substructures with different polarity. We leave this out for the sake of simplicity.

constant in its first argument. Hence the lambda operator abstracts over a variable x which occurs in a negative position, and by Corollary 4.11 it correctly represents a downward monotone function. Similarly, since `doubt` is an $\uparrow$Mon function in its second argument, the y is in a positive position and the second abstraction yields an upward monotone function. Both facts are encoded in the logical type assignment. Let us see the rules at work step by step, focusing on the flow of information from the logical to the structural formulas.

- Application of a downward monotone function implies the propagation of the marker from the function to the argument:

$$\dfrac{\dfrac{\Delta \vdash B}{\langle\Delta\rangle^- \vdash \diamond B}\,[\diamond \mathrm{I}] \qquad \Gamma \vdash \diamond B \backslash A}{\langle\Delta\rangle^- \circ \Gamma \vdash A}\,[\backslash \mathrm{E}]$$

  This embodies (part of) Definition 4.9-*iv*: M is negative (positive) in PQ if M is positive (negative) in Q, and P denotes a downward monotone function. For the missing part (the computation of the polarity of M in Q) we need to compute the polarity of the structure corresponding to M in the structure corresponding to Q. This is done by means of the structural rules.

- Functions are built by applying $[\backslash \mathrm{I}], [/\mathrm{I}]$. Upward and downward monotone functions abstract over positive and negative positions, respectively, as shown below looking at $[/\mathrm{I}]$ by means of an example. Let $C \neq \diamond C'$,

$$\dfrac{\begin{array}{c} D \\ \vdots \\ \Gamma \circ C \vdash A \end{array}}{\Gamma \vdash A/C}\,[/\mathrm{I}] \qquad \dfrac{\begin{array}{c} D \\ \vdots \\ \Gamma \circ \diamond B \vdash A \end{array}}{\Gamma \vdash A/\diamond B}\,[/\mathrm{I}]$$

- The polarity information is passed from the structure to the logical type (a) and *vice versa* (b).

$$\text{(a)}\quad \dfrac{\langle\Delta\rangle^- \vdash A}{\Delta \vdash \boxminus^{\downarrow} A}\,[\boxminus^{\downarrow}\mathrm{I}] \qquad \text{(b)}\quad \dfrac{\Delta \vdash \boxminus^{\downarrow} A}{\langle\Delta\rangle^- \vdash A}\,[\boxminus^{\downarrow}\mathrm{E}]$$

- The substitution of a structure with negative polarity by another one with the same polarity is performed by $[\diamond \mathrm{E}]$. The soundness proof (Section 4.4.4) guarantees the correctness of this substitution.

$$\dfrac{\Delta \vdash \diamond A \quad \Gamma[\langle A\rangle^-] \vdash B}{\Gamma[\Delta] \vdash B}\,[\diamond \mathrm{E}]$$

We now look at some concrete examples. In Example 4.14, we consider the case of coordinated phrase for the system LP+EPol. Unlike there, we cannot leave (functional) variables unspecified for their monotonicity property, since in MCTL+Pol the monotonicity marking is not assigned 'externally'. Let us see what differences this will make.

Example 4.25. [Coordination] As we have seen, coordination of a noun phrase with a quantifier involves the lifting of the *np*. In MCTL+Pol the type *np* can be lifted to a higher order one, by assuming a unmarked type or a marked one.

(a)

$$\dfrac{\dfrac{\text{mary} \vdash np \quad [x \vdash np\backslash s]^1}{\text{mary} \circ x \vdash s}\,[\backslash\text{E}]}{\text{mary} \vdash s/(np\backslash s)}\,[/\text{I}]^1$$

(b)

$$\dfrac{\dfrac{\dfrac{\text{mary} \vdash np}{\langle \text{mary}\rangle^- \vdash \diamond np}\,[\diamond\,\text{I}] \quad [x \vdash \diamond np\backslash s]^1}{\text{mary} \circ x \vdash s}\,[\backslash\text{E}]}{\langle \text{mary}\rangle^- \vdash s/(\diamond np\backslash s)}\,[/\text{I}]^1$$

The lambda term corresponding to the lifted type is $\lambda P.P\ \mathtt{m}$, where P represents an upward monotone function in (a) and a downward monotone function in (b). Hence $\mathtt{m}$ is in a positive position in (a) and so is its corresponding structure, whereas in (b) it is in a negative position and it is marked by $\langle\cdot\rangle^-$. This lifted type can now be coordinated with a quantifier phrases. The conjunction *and* receives a polymorphic type $(X\backslash X)/X$ which respects the monotonicity properties of the coordinated phrases as illustrated by the derivation below.

$$\dfrac{\dfrac{\vdots}{\langle \text{mary}\rangle^- \vdash s/(\diamond np\backslash s)} \quad \dfrac{\text{and} \vdash ((s/(\diamond np\backslash s))\backslash(s/(\diamond np\backslash s)))/(s/(\diamond np\backslash s)) \quad \text{every\_boy} \vdash s/(\diamond np\backslash s)}{\text{and} \circ \text{every\_boy} \vdash (s/(\diamond np\backslash s))\backslash(s/(\diamond np\backslash s))}\,[/\text{E}]}{\langle \text{mary}\rangle^- \circ (\text{and} \circ \text{every\_boy}) \vdash s/(\diamond np\backslash s)}\,[\backslash\text{E}]$$

The argument of this function must be of type $\diamond np\backslash s$, hence it can only be a downward monotone function, let it be $\mathtt{R}$. The lambda term corresponding to the whole structure is $\lambda Z.((\lambda P.P\ \mathtt{m})\ Z \wedge \mathtt{Every\_boy}\ Z)\ \mathtt{R}$. By β-reduction the term reduces to $\mathtt{R}\ \mathtt{m} \wedge \mathtt{Every\_boy}\ \mathtt{R}$, where $\mathtt{m}$ is in a negative polarity position. Hence, the corresponding structure is properly marked. If *and* had been composed with the derivation in (a) above, the verb phrase could have been only ↑Mon and again the polarity would have been correctly assigned in the structure. In other words, the type (and therefore the monotonicity property) of the assumed variable depends on the monotonicity property of the expression it will be replaced with.

The example above is also interesting for a second reason. From a closer look at the 'lexical' type of *every boy* we see that if we represent the lexical semantics of the constant $\mathtt{every\_boy}$ as $\lambda Q.\forall x.\mathtt{Boy}(x) \to Q(x)$, the polarity of the second occurrence of x depends on the monotonicity of Q (or the term replacing it by β-conversion). In Example 4.25, Q is replaced by a downward monotone function $\mathtt{R}$, hence x is in a negative polarity position. The type matching forces Q to have the same monotonicity property. This is reflected by the logical type of the quantifier $s/(\diamond np\backslash s)$ —which would have been $s/(np\backslash s)$ in case $\mathtt{R}$ was an upward monotone function. In Chapter 7, we will discuss the behavior of NPIs with respect to coordination.

Finally, before looking at how MCTL+Pol can work with structure containing NPIs we want to clarify the relevance of using a selected fragment of the logical language. Note that lexical assignments from outside the set FORM_1 may generate unsafe types in the derivation. The example below shows that this can lead to an incorrect computation of polarity positions. This is why we restrict ourself to a selected fragment of the logical language.

Example 4.26. [Unsafe Types] In the derivations (a) and (b) below the types in the boxes are not in SAFE. In (a) the structure Δ is assigned a negative polarity though it is not in a decreasing argument position. In (b) the structure $\Gamma \circ \Delta$ is assigned a positive polarity, though it is the argument of a downward monotone function.

(a)

$$\dfrac{\Gamma \vdash A \qquad \dfrac{\boxed{\Delta \vdash \boxdot^{\downarrow}(A \backslash B)}}{\langle \Delta \rangle^{-} \vdash A \backslash B}\,[\boxdot^{\downarrow}\mathrm{E}]}{\Gamma \circ \langle \Delta \rangle^{-} \vdash B}\,[\backslash \mathrm{E}]$$

(b)

$$\dfrac{\dfrac{\Gamma \vdash A \qquad \boxed{\Delta \vdash A \backslash \diamondsuit B}}{\Gamma \circ \Delta \vdash \diamondsuit B}\,[\backslash \mathrm{E}] \qquad \Sigma \vdash \diamondsuit B \backslash C}{(\Gamma \circ \Delta) \circ \Sigma \vdash C}\,[\backslash \mathrm{E}]$$

4.4.3 Negative Polarity Items in MCTL+Pol

Because monotonicity information is available on-line during parsing, MCTL+Pol can control NPI distribution. The type assignment of an NPI must encode three pieces of information: (i) the structure where the item occurs must have negative polarity; (ii) the structure containing the NPI must be taken as an argument by a structure of type $A/\diamondsuit B$ or $\diamondsuit B \backslash A$ (as negative polarity is assigned by downward monotone functions); (iii) NPIs must be in the immediate scope of their licensor. Using the encoding in Table 4.1 this means that the the whole structure containing the NPI must be headed by $\boxdot^{\downarrow}\diamondsuit$ and the NPI must have wide scope in the corresponding lambda term. Let us look at the adverb *yet* by means of an example.

Example 4.27. [Negative Polarity Items] Let $\boxdot^{\downarrow}\diamondsuit iv \backslash \boxdot^{\downarrow} diaminusiv$ be the type assignment of *yet*. It denotes that it must occur in a negative polarity position which it passes to its argument.

1. *Nobody left yet.*

$$\dfrac{\mathrm{nobody} \vdash s/\diamondsuit iv \qquad \dfrac{\dfrac{\dfrac{\dfrac{\mathrm{left} \vdash iv}{\langle \mathrm{left} \rangle^{-} \vdash \diamondsuit iv}\,[\diamondsuit \mathrm{I}]}{\mathrm{left} \vdash \boxdot^{\downarrow}\diamondsuit iv}\,[\boxdot^{\downarrow}\mathrm{I}] \qquad \mathrm{yet} \vdash \boxdot^{\downarrow}\diamondsuit iv \backslash \boxdot^{\downarrow}\diamondsuit iv}{\mathrm{left} \circ \mathrm{yet} \vdash \boxdot^{\downarrow}\diamondsuit iv}\,[\backslash \mathrm{E}]}{\langle \mathrm{left} \circ \mathrm{yet} \rangle^{-} \vdash \diamondsuit iv}\,[\boxdot^{\downarrow}\mathrm{E}]}{\mathrm{nobody} \circ \langle \mathrm{left} \circ \mathrm{yet} \rangle^{-} \vdash s}\,[/\mathrm{E}]$$

If we replace the quantifier *nobody* with an $\uparrow$Mon quantifier, e.g. *everybody*, the derivation fails: *everybody* has type s/iv which does not provide the needed negative feature to match the type assigned to *left yet*.

Example 4.28. [NPIs in Embedded Sentences]

1. *John doubts anybody left.*

$$\dfrac{\text{John} \vdash np \qquad \dfrac{\text{doubts} \vdash (np\backslash s)/\Diamond s \qquad \dfrac{\dfrac{\text{anybody} \vdash \boxdot^{\downarrow}\Diamond s/\boxdot^{\downarrow}\Diamond iv \quad \text{left} \vdash \boxdot^{\downarrow}\Diamond iv}{\text{anybody} \circ \text{left} \vdash \boxdot^{\downarrow}\Diamond s}\,[/\mathrm{E}]}{\langle \text{anybody} \circ \text{left}\rangle^{-} \vdash \Diamond s}\,[\boxdot^{\downarrow}\mathrm{I}]}{\text{doubts} \circ \langle \text{anybody} \circ \text{left}\rangle^{-} \vdash np\backslash s}\,[/\mathrm{E}]}{\text{John} \circ (\text{doubts} \circ \langle \text{anybody} \circ \text{left}\rangle^{-}) \vdash s}\,[\backslash\mathrm{E}]$$

The negative auxiliary *didn't* receives a polymorphic type X/X. The different instantiation of the variable type, must encode the monotonicity property of the expression. Furthermore, by decorating the X differently with unary operators, we can control the ways *didn't* composes with the bridge and non-bridge predicates.

2. **John didn't doubt anybody left.*

Starting from the lambda term $\lambda Pxy.\neg(P\ x)\ y$ representing *didn't*, the phrase *didn't doubt* has term $\lambda xy.\neg(\texttt{doubt}\ x)\ y$ and hence P is $\downarrow$Mon. The polarity of x is positive since it is in the scope of two downward monotone constants $\neg$ and `doubt`. Therefore, the function $\lambda xy.\neg(\texttt{doubt}\ x)\ y$ is $\uparrow$Mon. This makes the whole sentence semantically ill-formed, which is correctly predicted by MCTL+Pol: By simply applying our encoding, the type for *didn't* is $(iv/s)/\Diamond(iv/\Diamond s)$ from which it follows that the phrase *didn't doubt* is of type (iv/s) which fails to compose with *anybody left*.

3. *John didn't think anybody left.*

Similarly to the example above, the monotonicity properties of *think* and *didn't* motivate the type assignments of the items involved here, and in particular of *didn't*. The term $\lambda xy.\neg(\texttt{think}\ x)\ y$ represents a downward monotone function, since `think` is an upward monotone function. Therefore, type matching requires the assignment to *didn't* of a term $\lambda Pxy.\neg(P\ x)\ y$ where P denotes an $\uparrow$Mon function, and consequently the type $(iv/\Diamond s)/\Diamond(iv/s)$. The composed function 'didn't think' of type $(iv/\Diamond s)$ licenses the occurrence of the NPI *anybody*.

Example 4.29. [Multiple Negative Polarity Occurrences] Multiple occurrences of negative polarity items can be licensed by the same licensor, as illustrated by *anybody* and *at all* and their trigger *doubts*.

1. *John doubts anybody came at all.*

$$\dfrac{\text{John} \vdash np \qquad \dfrac{\text{doubts} \vdash (np\backslash s)/\Diamond s \qquad \dfrac{\dfrac{\text{anybody} \circ \text{came} \vdash \boxdot^{\downarrow}\Diamond s \quad \text{at all} \vdash \boxdot^{\downarrow}\Diamond s\backslash\boxdot^{\downarrow}\Diamond s}{((\text{anybody} \circ \text{came}) \circ \text{at all}) \vdash \boxdot^{\downarrow}\Diamond s}\,[\backslash\mathrm{E}]}{\langle (\text{anybody} \circ \text{came}) \circ \text{at all}\rangle^{-} \vdash \Diamond s}\,[\boxdot^{\downarrow}\mathrm{E}]}{\text{doubts} \circ \langle (\text{anybody} \circ \text{came}) \circ \text{at all}\rangle^{-} \vdash np\backslash s}\,[/\mathrm{E}]}{\text{John} \circ (\text{doubts} \circ \langle (\text{anybody} \circ \text{came}) \circ \text{at all}\rangle^{-}) \vdash s}\,[\backslash\mathrm{E}]$$

We close this section with an example that is still problematic for our system as it stands —we return to it in Chapter 6. The problem relates to generalized quantifier expressions, which in natural language can have varying scopal possibilities. For instance, the sentence *Three good referees read few abstracts* can be assigned two different representations: either `(Three_good_referees (Few_abstract Read))` or `(Few abstracts (Three_good_referees Read))`. The different scope of the downward monotone function `Few_abstract` modifies the polarity of the subterm in the whole term. In the first case, the constant `Three_good_referees` has positive polarity; whereas in the second case, it is in negative positions. From this it follows that different inferences will be drawn from the sentence *Three good referees read few abstracts*, depending on the interpretation assigned to it.

(a)

$$\frac{(\text{Three good referees})^{+} \text{ read few abstracts} \vdash_s}{\text{Three referees read few abstracts} \vdash_s}$$

(b)

$$\frac{(\text{Three good referees})^{-} \text{ read few abstracts} \vdash_s}{\text{Three good dutch referees read few abstracts} \vdash_s}$$

where (a) would be logically correct in the interpretation (Three > Few) and (b) in the wide scope reading (Few > Three). However, while (a) is a correct natural reasoning inference, (b) is not. This example shows that natural reasoning, and in particular monotone inference, also uses other information than the semantic information discussed so far. Though monotonicity plays a crucial role in the derivation of inference, properties of different nature may also interact with it and affect natural reasoning inference. In Chapter 6, we will explore the properties affecting the interpretation of linguistic structures containing quantifier phrases (such as the one in the premises of the inferences above) and give an deductive analysis within the CTL framework.

4.4.4 Soundness

We have to guarantee that our natural logic does not derive inferences which are not valid model theoretically, *i.e.* we need to prove that the natural logic is sound.

We use $\phi \vdash_{\mathsf{MON}} \psi$ to denote that a marked parsed structure ψ is inferred from the marked structure ϕ by means of the monotonicity rules given in Definition 4.24. Formally, proving that the natural logic fragment is sound means to prove that for all models $\mathcal{M}$ and assignments f,

$$\text{if} \quad \phi \vdash_{\mathsf{MON}} \psi \quad \text{then} \quad [\![\phi]\!]^f_{\mathcal{M}} \leq [\![\psi]\!]^f_{\mathcal{M}}$$

where $[\![\phi]\!]^f_{\mathcal{M}}$ and $[\![\psi]\!]^f_{\mathcal{M}}$ stand for the denotations of the lambda terms corresponding to the structures ϕ and ψ in $\mathcal{M}$ under f, respectively.

Proving the above claim reduces to proving that monotonicity in lambda terms is tied up with the syntactic notion of polarity, and consequently to the markers assigned to the parsed structures. The link between monotonicity and polarity is given by Proposition 4.10 and its Corollary 4.11 as we have already commented. Our task is now to

prove that if a substructure in the conclusion of a (sub)derivation has positive (resp. negative) polarity, then the corresponding lambda term is in a positive (resp. negative) position in the term corresponding to the whole conclusion (see Definition 4.8).

PROPOSITION 4.30. [Soundness] Let $\mathcal{D}$ be a derivation of $\Gamma \vdash M : A$ with $A \in \mathsf{SAFE}$ and Γ built on SAFE. Let $(B/C)/\vec{D}$ denote $(((B/C)/D_1)\dots D_n)$ with $n \geq 0$. Then[7],

(a) If A is of the form $(B/C)/\vec{D}$ or $\diamondsuit((B/C)/\vec{D})$ (modulo occurrences of $\boxminus^{\downarrow}\diamondsuit$), then M is $\downarrow$Mon in the argument corresponding to C if C is of the form $\diamondsuit C'$, and $\uparrow$Mon otherwise.

(b) If Γ contains a logical formula $(B/C)/\vec{D}$ or $\diamondsuit((B/C)/\vec{D})$ (modulo occurrences of $\boxminus^{\downarrow}\diamondsuit$), then the lambda term in M corresponding to this substructure is $\downarrow$Mon in the argument corresponding to C if C is of the form $\diamondsuit C'$, and $\uparrow$Mon otherwise.

(c) If $A \in \mathsf{FORM}_1$, then for any substructure Γ' in Γ, if Γ' has positive (resp. negative) polarity in Γ then the corresponding lambda term is in a positive (resp. negative) position in M.

(d) If A is of the form $\diamondsuit A'$, then for any substructure Γ' in Γ, if Γ' has negative (resp. positive) polarity in Γ then the corresponding lambda term is in a positive (resp. negative) position in M.

PROOF. The proof goes by induction on the length of the derivation.

1. Base case. Assume $\Gamma \vdash M : A$ is a leaf. Then (a), (b), (c) and (d) hold trivially.

2. I.H. Let [R] be the last rule applied. Assume (a), (b), (c) and (d) hold for the premises of [R], we have to show that they hold also in its conclusion.

   (i) (i_1) [R] = [Pol$_-$]; (i_2) [R] = [Pol$_{--}$].
   Neither the polarity of the structures, nor the lambda term, nor the formula on the right side of the $\vdash$ changes. Hence, the proposition holds by I.H.

   (ii) (ii_1) [R] = [$\diamondsuit$I]; (ii_2) [R] = [$\boxminus^{\downarrow}$I]; (ii_3) [R] = [$\boxminus^{\downarrow}$E].
   (ii_1)

$$\dfrac{\begin{array}{c} D_1 \\ \vdots \\ \Gamma' \vdash M : A' \end{array}}{\langle\Gamma'\rangle^- \vdash {}^{\cap}M' : \diamondsuit A'}\ [\diamondsuit \mathrm{I}]$$

   (a) and (b) are preserved[8]. Since $A \in \mathsf{SAFE}$, we know that $A' \in \mathsf{FORM}_1$. By I.H. (c) holds in the premise, hence (d) holds in the conclusion of [$\diamondsuit$I].
   The same applies in case [R] = [$\boxminus^{\downarrow}$I], or [R] = [$\boxminus^{\downarrow}$E].

[7] We simplify the proof by considering only /. The directionality of the functional implication does not affect the proof.

[8] The terms M and ${}^{\cap}M$ have the same monotonicity property. The same holds for terms decorated by the other unary operators introduced by the logical rules of $\diamondsuit$ and $\boxminus^{\downarrow}$, ${}^{\cup}$, ${}^{\vee}$ and ${}^{\wedge}$.

(iii) [R] = [/I].

$$\dfrac{\begin{array}{c}D_1\\ \vdots\\ \Gamma \circ C \vdash M : B\end{array}}{\Gamma \vdash \lambda x.M : B/C}\ [/\mathrm{I}]$$

(a) Let $B', C', \vec{D}$ be such that $A = ((B'/C')/\vec{D})$. If $n \gneq 0$, then (a) follows directly from I.H. Otherwise $B' = B$ and $C' = C$. Since $B/C \in \mathsf{SAFE}$, $B \in \mathsf{FORM}_1$. Hence (c) applies to the premise of [/I]. Therefore, the lambda term corresponding to C is in a positive (resp. negative) position in M, if $C \neq \Diamond C''$ (resp. $C = \Diamond C''$). By Corollary 4.11, $\lambda x.M$ denotes an $\uparrow$Mon (resp. $\downarrow$Mon) function in x (the variable corresponding to C).

The points (b) and (c) follow directly by I.H., and (d) holds trivially.

(iv) [R] = [/E].

$$\dfrac{\begin{array}{c}D_1\\ \vdots\\ \Delta \vdash t : A/B\end{array} \quad \begin{array}{c}D_2\\ \vdots\\ \Gamma \vdash u : B\end{array}}{\Delta \circ \Gamma \vdash t(u) : A}\ [/\mathrm{E}]$$

The points (a) and (b) apply by I.H. and (d) holds trivially. We have to consider (c).

Let Γ' be in $\Delta \circ \Gamma$, we have to consider three cases: 1. Γ' is in Δ, 2. Γ' is in Γ, and 3. $\Gamma' = \Delta \circ \Gamma$.

1. If $\Gamma' \subseteq \Delta$, then (c) applied to Γ' follows directly from I.H. applied to the major premise.
2. If $\Gamma' \subseteq \Gamma$, then
   (2') Suppose Γ' has positive polarity in Γ, then it has positive polarity also in $\Delta \circ \Gamma$. If $B \in \mathsf{FORM}_1$, then by (a) t denotes an $\uparrow$Mon function, and by (c) applied to the minor premise of [/E], the lambda term corresponding to Γ' occurs in a positive position in u. By Definition 4.9, it is in a positive position in $t(u)$. If $B = \Diamond B'$, then by (a) t denotes a $\downarrow$Mon function, and by (d) applied to the minor premise of [/E], the lambda term corresponding to Γ' occurs in a negative position in u. By Definition 4.9, it is again in a positive position in $t(u)$.
   (2") Similarly for Γ' with negative polarity in Γ.
3. If $\Gamma' = \Delta \circ \Gamma$, then (c) holds trivially.

(v) [R] = [$\Diamond$ E].

$$\dfrac{\begin{array}{c}D_1\\ \vdots\\ \Delta \vdash u : \Diamond A\end{array} \quad \begin{array}{c}D_2\\ \vdots\\ \Gamma[\langle v : A\rangle^-] \vdash t : B\end{array}}{\Gamma[\Delta] \vdash t[^{\cup}u/v] : B}\ [\Diamond \mathrm{E}]$$

First of all, observe that by I.H. on (a) and (b), the monotonicity property of u and v are both completely determined by the form of A in exactly the same manner. Thus, substituting u for v in t does not affect any monotonicity related property. Let us refer to this as (*).

(a) The claim in (a) follows directly from (*) and the I.H. on the minor premise of $[\diamond\,\mathrm{E}]$.

(b) For Γ' in Γ (b) applied to Γ' follows from (*) and the I.H. on the minor premise. For Γ' in Δ (b) follows directly by I.H. on the major premise of $[\diamond\,\mathrm{E}]$.

(c) $B \in \mathsf{FORM}_1$. If Γ' in Γ or $\Gamma' = \Gamma''[\Delta]$, then (c) applied to Γ' follows from (*) and the I.H. on the minor premise of $[\diamond\,\mathrm{E}]$. For Γ' in Δ,

1. If Δ is surrounded by an odd number of $\langle\cdot\rangle^-$ in $\Gamma[\Delta]$, then A has positive polarity in the minor premise (it is surrounded by an even number of $\langle\cdot\rangle^-$). Then by I.H. on (c) v is in an positive position in t. Furthermore, if Γ' in Δ has positive polarity in Δ then it has negative polarity in $\Gamma[\Delta]$. By I.H. on (d) applied to Γ', the term z corresponding to Γ' in u is in a negative position, and by Definition 4.9, since v is in a positive position in t, z is in a negative position in $t[^{\cup}u/v]$. Similarly, for Γ' with negative polarity in Δ.
2. If Δ is surrounded by an even number of $\langle\cdot\rangle^-$ in $\Gamma[\Delta]$, then A has negative polarity. The proof is similar to the previous case, apart from the fact that v is in a negative position and A is surrounded by an odd number of $\langle\cdot\rangle^-$.

(d) $B = \diamond B$, the proof is similar to the one of case (c).

QED

4.4.5 Summary

We have already commented on some of the differences between the three systems LP+EPol, CG+Pol and MCTL+Pol, the most important being that the first system obtains polarity markers *extra-logically*, while in CG+Pol and MCTL+Pol the marking is obtained on-line as part of the logical derivations. Now let us analyze carefully the effects that these differences have on the final aim of the project: the design of a natural logic to account for negative polarity items.

To start with, all three formalisms should be able to determine the grammaticality of linguistic structures. This is a basic requirement, even if we are not interested in analyzing monotonicity phenomena. Here already, LP+EPol and CG+Pol are outperformed by MCTL+Pol, since LP does not take syntactic structure into account, while CG+Pol lacks hypothetical reasoning and needs multiple lexical entries to compensate for this deficiency (Chapter 1).

More interesting to the subject discussed in this chapter is the difference in dealing with polarity intra-logically and extra-logically. LP+EPol implements polarity marking as a rewriting algorithm which takes a proof tree as input and decorates it with markers.

It is crucial to note that this approach leaves out the possibility of polarity information actually taking active part in the derivation. In CG+Pol and MCTL+Pol, in contrast, the monotonicity information stored in the lexicon plays a fundamental role during the construction of the derivation. The presence of monotonicity markers enables or blocks the possibility of applying specific derivation rules. In other words, beside working as markers for making valid inferences at the natural reasoning level, they contribute to determine grammatical structures.

Moreover, the analysis of NPIs in embedded sentences shows that the way monotone functions compose in natural language can be effected by other constraints, and some control on functional application is necessary in order to properly model this linguistic phenomenon. Neither LP nor CG has the right expressivity to tackle this problem. On the other hand, MCTL provides us with the required logical tool kit to obtain fine-grained type assignments that lexically anchor the way structures are built.

Furthermore, there are also other things to be gained by analyzing polarity phenomena in MCTL. Negative polarity phenomena are observed in many languages. Being able to account for them in terms of the base logic opens the door to a comparative study of both NPIs and natural reasoning. Crosslinguistic variations could be accounted for in terms of different type assignments and structural rule packages. An interesting area for future research is how structural rules interact with the computation of the polarity positions and how they effect the distribution of negative polarity items. LP+EPol is too flexible, and CG+Pol is too strict to tackle this question.

Finally, a last remark concern the correctness of the marking mechanisms used by the three systems. While LP+EPol and MCTL+Pol are proven to be sound, no analogous result is known for CG+Pol. Note that the soundness of the marking algorithm does not fully guarantee the correct modelling of natural reasoning inference, since natural language may diverge from formal language in the combinatorial scope possibilities of its scoping elements. An important constraint the parser must satisfy is therefore that it produces as output only the available readings of the parsed linguistic structure. For the reasons discussed so far MCTL+Pol seem to be most promising in this respect.

4.5 Key Concepts

The analysis discussed in this chapter shows that:

1. Monotonicity and polarity information can be encoded into CTL syntactic type assignments by means of unary operators.

2. CTL can be employed to achieve a proof theoretical account of semantic issues.

3. A system in which semantic information relevant for grammaticality is internalized into the logical language presents several advantages with respect to a system based purely on functional types where semantic information is *post hoc* by and extra-logical marking mechanism. In particular, it can account for the distribution of items sensitive to such information.

4. MCTL seems to provide a natural devision of labor between the tasks of reasoning and parsing. The latter is handled in the logical part of a sequent, the former in the structural part.

Part III

Natural Language Typologies

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 thesis, we argue that the extended vocabulary of type-logical constants introduced in Chapter 2 provides the means to encode the required distinctions in lexical type assignment.

In Chapter 5, we introduce the analytical concepts we need to carry out this task. We set up a general notion of grammatical composition taking into account the form and the meaning dimensions of linguistic expressions. We develop a logical theory of *licensing* and *antilicensing* relations that cross-cuts the form and meaning dimensions.

In Chapters 6 and 7, we show how these relations can be type-logically expressed in concrete linguistic analyses. In Chapter 6, we investigate lexical differences in the scope construal possibilities for generalized quantifier expressions. We present the minimalist analysis of these phenomena which is based on structural notions of functional projections and feature checking. We then develop a type-logical account of scope construal which recasts the feature-checking mechanisms in purely deductive terms.

In Chapter 7, we study the distribution of *polarity-sensitive* expressions. We show how our theory of licensing and antilicensing relations successfully differentiates between negative polarity items, which are 'attracted' by their triggers, and positive polarity items, which are instead 'repelled' by them. Finally, we investigate these 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 5

Composition Relations

In the discussion of natural reasoning in Chapter 4, we have noticed that natural language quantifiers do not realize the full set of combinatorial possibilities for scope dependencies predicted by their semantic type assignment as sets of properties. As a result, certain monotonicity inference substitutions that would be logically valid are not available in natural reasoning. Furthermore, we have seen that the syntactic distribution of certain expressions depends on the semantic properties of other expressions in their syntactic environment, which act as licensors. In this part of the thesis, we investigate these phenomena in more detail switching the focus from natural reasoning inferences to the study of grammatical composition relations.

Linguistic composition is affected by several aspects of the constituents involved. In Chapter 6, we investigate logico-semantic properties of quantifier phrases, and how they influence their different scope behavior. In Chapter 7, we focus attention on the composition relations based on the sensitivity of an item with respect to a certain semantic property shared by other expressions called 'triggers'. Following [Gia97], we consider the relation between a sensitive item and the trigger to be either a licensing or an antilicensing relation. From this it follows that a structure can be ungrammatical either because the sensitive item is not provided with the required property, or because it occurs in a context supplying the property the item is allergic to. Categorial Type Logic (CTL) helps us clarify these differences among composition relations and the ways scope elements interact.

The behavior both of quantifier phrases and sensitive items provides the information required to reach a classification of such expressions. These classifications can be thought of as 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 syntactic licensing/antilicensing relations.

5.1 Two Sorts of Deviations

In her discussion of scopal possibilities [Sza97], Szabolcsi makes an important distinction between *coherent* and *incoherent* deviations, illustrated by the two examples below.

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

(2) a. *How didn't Fido behave?
b. Who didn't Fido see?

The difference between (1-a) and (1-b) shows that no incoherence results when *few_n* takes scope over *three_N*. This reading, though blocked in (1-a), is available in (1-b). The reason for this contrast is to be found on the syntax-semantic interface and has been described by saying that 'counting' quantifier phrases take scope locally [BS97], or in other words that they cannot have scope wider than where they occur overtly.

The sentences in (2) form a different case of deviation. The inability of the wh-phrase *how* to take scope over *didn't* (2-a) is traditionally thought of as the effect of a syntactic constraint: the so-called weak island constraint of Ross [Ros67]. Recently, weak islands have been explained in terms of algebraic semantic characterizations of scope interaction [SZ97], which would explain the incoherence of the interpretation needed in (2-a) and the availability of it in (2-b).

Szabolcsi and Zwarts [SZ97] consider wh-phrases as items *sensitive* to weak islands, or more specifically, sensitive to the property of the scope elements which form the island. For instance, *how* is said to be sensitive to the weak island formed by *didn't* and the extraction from it is blocked (2-a). Moreover, Szabolcsi and Zwarts show that different wh-phrases are sensitive to weak-islands of different strength.

In the Szabolcsi and Zwarts' account, for a wh-phrase to take wide scope over some scope element (SE) the definition/verification of the answer involves specific operations associated with the SE. For instance, *not* corresponds to taking the complement of a set ($\neg$), *universal* quantifiers are associated with intersection ($\cap$), and *existential* quantifiers with union ($\cup$). If the wh-phrase ranges over a semantic domain corresponding to an algebraic structure which is not closed under such an operator, it is unable to have scope over the SE. One could say that a wh-phrase is *allergic* to a property particular to the SE or more generally that it is *sensitive* to the weak-island formed by the SE. Briefly, the different distributional behavior of wh-phrases receives a semantic explanation: A classification of weak-islands and hence of the extractees can be given based on the properties of their domains of interpretation.

Let us illustrate this theory by looking at an example. From the fact that *how* ranges over manner adverbial which denote on an algebraic structure closed under $\cup$, namely semilattices (SL), it follows that *how* is sensitive to weak islands created by SEs involving $\cap$ and $\neg$ (e.g. it cannot have scope over universal quantifier and negation). Similarly, since *how many* ranges over numbers —lattices (LA) which are closed under $\cup$ and $\cap$— and *who* over individuals —boolean structures (BO) which are closed under $\cup, \cap$ and $\neg$ — it follows that *how many* is sensitive to SEs associated with $\neg$ (2-a), and *who* can extract from all weak islands (2-b). Note that since a set inclusion relation

holds among these three algebraic structures, $\mathsf{SL} \subseteq \mathsf{LA} \subseteq \mathsf{BO}$, a wh-phrase sensitive to a weak island corresponding to a certain structure will be sensitive to a richer one as well, where 'richness' is defined in terms of the operations which are defined in the structure. For instance, *how* is sensitive to the weak island built by the universal quantifier (which denotes over LA) and also fails to extract from a weak island built by negation (which denotes over BO).

(3) a. *How did no kid behave?
b. How did everyone behave? [Every > How] [*How > Every]

The behavior of wh-phrases with respect to weak-islands can be described in more general terms by considering licensing and antilicensing relations.

5.2 Licensing and Antilicensing Relations

In the discussion of negative polarity distribution in Section 4.1, we have seen that they are items in a *licensing* relation with downward monotone functions. One could say that they are *attracted* by the monotonicity property of their licensor and incompatible with functions which do not share this feature. On the other hand, assuming Szabolcsi and Zwarts' analysis of wh-phrases, *how* could be said to be *repelled* by the property of 'having the complement operation', shared by the scope elements interpreted over boolean structures. In other words, the wh-phrase can be said to be in an *antilicensing* relation with such property. The linguistic classification of Dutch *positive* polarity items given in [Wou94] could be interpreted in a similar way. We clarify these two composition relations by reviewing the analysis of Dutch negative and positive polarity items.

Recall from Section 4.1 that the polarity of a context is closely connected to the monotonicity of its constituents. The concept of monotonicity is linked to the concept of negation identified by the De Morgan's laws. 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 5.1 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: $\mathsf{AM} \subseteq \mathsf{AA} \subseteq \mathsf{DM}$[1].

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 5.1: Monotone functions classification.

[1]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].

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 following examples illustrate their different relations with such functions. The monotone functions are emphasized, whereas the polarity items are underlined. We take *weinig* (tr. few), *niemand* (tr. nobody) and *niet* (tr. not) as representative of the sets DM, AA and AM, respectively; the determiner *ook maar* (tr. any) and the idiomatic *mals* (tr. tender) are examples of negative polarity items (NPIs) whereas *allerminst* (tr. not-at-all) and *een beetje* (tr. a bit) exemplify their positive counterparts. We indicate with % mildly ungrammatical sentences.

(4) a. %*Weinig* monniken zullen ook maar iets bereiken. [%DM > ook maar].
Few monks will anything achieve.
tr. Few monks will achieve something.

b. *Niemand* zal ook maar iets bereiken. [AA > ook maar].
Noboy 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

d. *Van *weinig* monniken was de kritiek mals. [*DM > mals].
Of few monks was the criticism tender.
tr. The criticism of few monks was tender.

e. *De kritiek van vader abt was *nooit* mals. [*AA > mals].
The criticism of father abbot was never tender.
tr. The criticism of father abbot was never tender.

f. De kritiek zal *niet* mals zijn. [AM > mals].
The criticism will not tender be.
tr. The criticism will be harsh.

(5) a. **Weinig* monniken zijn allerminst gelukkig. [*DM > allerminst].
Few monks are not-at-all happy.
tr. Few monks are not-at-all happy.

b. *Weinig* monniken zijn een beetje gelukkig. [DM > een beetje].
Few monks are a bit happy.
tr. Few monks are a bit happy.

c. %*Niemand* is een beetje gelukkig. [%AA > een beetje].
Nobdy is a bit happy.
tr. Nobody is a bit happy.

d. *Niemand* wil nog Donne lezen. [AA > nog].
Nobody wants still Donne read.
tr. Nobody wants to read Donne anymore.

e. *Jan wil *niet* nog Donne lezen. [*AM > nog].
Jan wants not still Donne read.

tr. Jan does not want to read Donne anymore.

From a comparison of the sentences in (4) and (5), it follows that positive polarity items (PPIs) mirror the behavior of their negative relatives. The whole picture is summarized in Table 5.2 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 5.2: Polarity items distribution in Dutch.

Table 5.2 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 functions 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 we will work out in detail in the next chapters to reach a CTL analysis of licensing and antilicensing relations.

5.3 Calibrating Grammatical Composition Relations

Our aim in the next chapters is to obtain a deductive account of the linguistic classifications discussed above. In particular, we account for the scope deviations among quantifier phrases, and the licensing/antilicensing relations using modalities as 'logical features' controlling composition relations.

We claim that a type logical approach sheds light on the distinction between (a) an element sensitive to the function which can taken it as argument, and (b) a functional expression sensitive to its argument, e.g. NPIs. Thus, in a function-argument structure the function can be either (a) the trigger or (b) the sensitive item. Moreover, in each case the sensitive item and the trigger can be either in a licensing or in an antilicensing relation. These distinctions call for a general definition of the ways in which linguistic expressions containing sensitive items are composed.

Recall from Section 1.3 that linguistic signs are structured objects and their composition is driven by the way their components interact. In particular, we can think of an expression as a pair consisting of a form component α, and a meaning component α', represented as $[\alpha_A : \alpha'_a]$, where A and a are the syntactic and semantic types, respectively. Expressions with the same semantic type take their denotation in the same

domain, though their forms and therefore their syntactic types may be different. In particular, a sensitive item can be interpreted in the same domain of a non-sensitive item of the same type, but the two expressions show different distributional behavior. We illustrate this difference in the example below.

Example 5.1. Let us consider to sentences with the same structure, which differ only on the signs they are composed of.

(6) a. John didn't read anything.
b. $\texttt{Didn't}(\texttt{Anything}(\lambda y.((\texttt{read}\ y)x)))$

(7) a. John didn't read something.
b. $^{*}\texttt{Didn't}(\texttt{Something}(\lambda y.((\texttt{read}\ y)x)))$

The sign *anything* hase the same semantic type as *something*. However, the latter is ungrammatical in (7) with the meaning in (b) and similarly *anything* would be ungrammatical if we replace (6) *didn't* with *did*. The difference is due to the way *something* and *anything* are effected by the semantic property of *didn't*: *anything* is licensed by this property, whereas *something* is incompatible with it.

The example shows the importance of distinguishing the form and meaning components of an expression and hence their syntactic and semantic type. Moreover, by looking at the way the sensitive item *anything* is in construction with the trigger *didn't*, we can reach a general representation of the relation of *be in construction with*. Let us give a global definition of grammatical composition which abstract away from irrelevant details.

$$\mathbb{C}([\gamma : \gamma'], [\alpha : \alpha'], [\beta : \beta']) \text{ iff } \mathbb{R}(\gamma, \alpha, \beta) \wedge \mathbb{M}(\gamma', \alpha', \beta')$$

where $\mathbb{R}(\gamma, \alpha, \beta)$ stands for the syntactic composition of a structure γ out of α and β and possibly other constituents —it is intended to be a reminiscent of the grammatical composition relation $R_\bullet$ of the Kripke models (Definition 2.8)— and $\mathbb{M}(\gamma', \alpha', \beta')$ stands for the semantic composition of a term γ' out of α' and β' and possibly of other terms —it is meant to generalize the meaning assembly carried out by the operation of the semantic algebra (Definition 1.16). We say that $[\alpha : \alpha']$ is in construction with $[\beta : \beta']$ in $[\gamma : \gamma']$. Based on this assembly of forms and meanings, we can define the licensing and antilicensing relations holding between a sensitive item and the semantic property of its triggers.

Definition 5.2. [Composition Relations] The following composition relations can hold between two signs.

i. A sign $[\alpha : \alpha']$ is in a *compatibility* relation with a sign $[\beta : \beta']$, if the relation below holds. Notation: $[\![\beta']\!] \in P$ stands for $[\![\beta']\!]$ has the property P.

$$\text{If } [\![\beta']\!] \in P, \text{ then} \exists [\gamma : \gamma'] \text{ s.t. } \mathbb{C}([\gamma : \gamma'], [\alpha : \alpha'], [\beta : \beta']).$$

ii. A sign $[\alpha : \alpha']$ is in a *licensing* relation with a sign $[\beta : \beta']$, if

$$[\![\beta']\!] \in P \text{ iff } \exists [\gamma : \gamma'] \text{ s.t. } \mathbb{C}([\gamma : \gamma'], [\alpha : \alpha'], [\beta : \beta']).$$

iii. A sign $[\alpha : \alpha']$ is in a *incompatiblity* relation with a sign $[\beta : \beta']$, if the relation below holds:

$$\text{If } [\![\beta']\!] \in P, \text{then} \neg\exists[\gamma : \gamma'] \text{ s.t. } \mathbb{C}([\gamma : \gamma'], [\alpha : \alpha'], [\beta : \beta']).$$

iv. A sign $[\alpha : \alpha']$ is in an *antilicensing* relation with a sign $[\beta : \beta']$, if

$$[\![\beta']\!] \in P \text{ iff } \neg\exists[\gamma : \gamma'] \text{ s.t. } \mathbb{C}([\gamma : \gamma'], [\alpha : \alpha'], [\beta_1 : \beta'_1]).$$

We will alternatively say that a sign is licensed by the property which is licensor must have. Finally, as commented above in a function-argument structure, we can distinguish two cases:

(a) $[\alpha : \alpha']$ is an element sensitive to the property of a function, then in the points above $\mathbb{M}$ is such that β' has immediate scope over α' in γ';

(b) $[\alpha : \alpha']$ is a function sensitive to the property of its argument, then in the points above $\mathbb{M}$ is such that α' has immediate scope over β' in γ'[2].

Remark 5.3. Some logical consequences derive from the definition above. In particular, if a sign $[\alpha : \alpha']$ is licensed by a sign $[\beta : \beta']$ that has a property P, it will be compatible (resp. incompatible) with any sign $[\beta_1 : \beta'_1]$ that has a property equal to or stronger (resp. weaker) than P. Similarly, if a sign $[\alpha : \alpha']$ is antilicensed by a sign $[\beta : \beta']$ that has a property P, it will be incompatible (compatible) with any sign $[\beta_1 : \beta'_1]$ that has a property equal to or stronger (resp. weaker) than P.

Intuitively, one could think of the composition of a sensitive item with a trigger as a relation which 'must' or 'must not' hold, and the grammatical and ungrammatical construction which follows as relations which 'can' or 'cannot' hold. Based on this definition we can identify the sensitive items and their triggers as below.

Definition 5.4. [Sensitive Items and their Triggers]

i. An expression $A := [\alpha : \alpha']$ is a *sensitive item* if it is in a licensing or antilicensing relation.

ii. A sign $B := [\beta : \beta']$ is a *direct trigger* of A sensitive to P, if $[\![\beta']\!] \in P$ and for any other stronger property P' $[\![\beta']\!] \notin P'$.

iii. A sign $B_1 := [\beta_1 : \beta'_1]$ is an *indirect trigger* of A if $[\![\beta'_1]\!] \in P$ and also $[\![\beta'_1]\!] \in P'$ for $P' \subseteq P$.

Let us now check how these definitions apply to the linguistic phenomena we have introduced in the previous section. In the case of Dutch negative polarity items, the relevant sets of licensors are identified by the properties of 'antimorphic', 'antiadditive'

[2]Note that the definition of antilicensing relation we propose differs from the definition given in [Gia97] where it is seen as the negation of a licensing relation. Her definition of antilicensing relation correponds to what we refer to as incompatiblity relation: it is a negative information from which no positive relation can be derived.

and 'downward monotone', which we have represented as the sets AM, AA, and DM. A negative polarity item, licensed by a certain property, will have as direct triggers the expressions displayed in Table 5.1 as representative of the corresponding set.

Consider the weak negative polarity item *ook maar* (tr. any) which is licensed by 'antiadditivity'. Let *ook maar* be represented by $A := [\alpha : \alpha']$. For any function $B := [\beta : \beta']$ which belongs to a set stronger than or equal to AA $\exists C := [\gamma : \gamma']$ $\mathbb{C}(C, A, B)$; whereas for any function B_1 which does not belong to AA such C does not exist. For instance, *niemand* (tr. nobody) and *niet* (tr. not) are in AA and *ook maar* is grammatical when in construction with them, whereas *weinig_n* (tr. few_n) does not belong to AA and *ook maar* is not grammatical in its (immediate) scope. Moreover, *niemand* is a direct trigger whereas *niet* is an indirect one.

The definition of antilicensing relation can be illustrated by looking at (a) Dutch positive polarity items and their relation with respect to monotone functions, and (b) the behavior of wh-phrases with respect to the scope elements forming weak-islands. The first case exemplifies Definition 5.2-(iva), whereas the second instantiates Definition 5.2-(ivb). A weak positive polarity item like *nog* (tr. still) is antilicensed by 'antimorphicity': it is incompatible with the characteristic function identifying the set AM, and is compatible with all the other functions building bigger sets. In other words, it is ungrammatical in construction with its triggers, but it is grammatical with the functions belonging to bigger sets and which are not in AM.

Similarly, in English the wh-phrase *how many* is antilicensed by the property of 'having the complement operation'. Consequently, its application to scope elements which take their denotation over domains of elements having this property is undefined. Again, *how many* is compatible with the characteristic function identifying bigger sets, *i.e.* it can be in construction with elements belonging to bigger sets[3].

Finally, an example of an expression in a compatibility relation with a semantic property is given by the adverb *almost* which can modify universal quantifiers (8-a), but not the existential ones (8-b).

(8) a. Almost every student came.
b. *Almost some student came.
c. He almost missed the train.

Almost is compatible with the property 'being universal' and it is incompatible with the one of 'being existential'. Note that the compatibility relation is weaker than the licensing one, since it does not require the item to be incompatible with all the expressions which do not have the property it is compatible with (8-c). Similarly, the incompatibility relation differs from the antilicensing one, since it does not say anything about how the item behaves with respect to other weaker properties[4].

[3]Note that, the subset relation holding among the algebraic structures is reversed when considering the sets of the expressions which denote over them. For example, the set of the expressions with the property of 'having the complement' (which denote over BO) is smaller than the set of the expressions with the property of 'having the intersection' (which denote over LA).

[4]In [Gia99] the English negative polarity item *any* is claimed to be incompatible (accordingly to our terminology) with veridicality. Therefore, it may be grammatical with nonveridical functions, but not necessarily with all of them.

5.4 Key Concepts

In this chapter we have prepared the ground for this part of the thesis. 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.

4. We have calibrated the definition of *composition relations* distinguishing the ways a sensitive item relates with a certain semantic property.

Chapter 6

Quantifier Scope

In this chapter, we study the scopal behavior of quantifier phrases using the unary modalities of $\mathsf{NL}(\Diamond)$, we show how we can lexically enforce different scopal possibilities for subclasses of the general class of quantifier phrases.

In Section 6.1, we present the empirical data we have to account for. In Sections 6.2 and 6.3 we look at earlier analyses within the type logical tradition and within generative grammar. The core of our proposal is in Section 6.4 where we present modally decorated type assignments making the right empirical prediction. Finally, we suggest a refinement of the minimalist analysis of the feature checking suggested by the comparison with the logical approach (Section 6.5)[1].

6.1 Quantifier Scope. The Problem

Quantified noun phrases (QPs), e.g. *nobody, a N, every N*, offer interesting challenges for the treatment of the syntax/semantic interface. First of all, they can take scope wider than where they occur overtly as illustrated by the examples below.

(1) a. John *wants* to marry *a Canadian princess*.
b. *Every boy* read *two books*.
c. John *didn't* marry *a Canadian princess*.

In all these sentences the QPs can have either narrow or wide scope with respect to the other emphasized expressions giving rise to scope ambiguities. The sentence in (1-a) is a case of existence *presupposition*: if the existentially quantified indefinite *a Canadian princess* falls under the scope of *want*, then the speaker needs not be committed to the existence of any Canadian princess (*de dicto* reading); on the other hand, if the QP scopes over it, then the speaker understands that such an individual exists (*de re* reading). Sentence (1-b) regards the *distributivity* property of the two QPs involved. If the indefinite QP *two books* outscope *every boy*, then the sentence is interpreted as referring to two specific books which every boy read. On the other hand, if *two*

[1]The results presented in this chapter are partially based on joint work with Richard Moot [BM00].

books falls under the scope of the distributive quantifier *every boy*, the total number of books involved is potentially much greater than two. In the latter case, the QP having wide scope is called the *distributor* and the narrow-scope indefinite the *distributed share* [Cho87]. In (1-c) the ambiguity is given by the scopal interaction between *logical operators*: the negation and existential quantifier. In one reading ($\neg\exists$), it is not the case that John married a Canadian princess (maybe he married a Dutch one); in the second case ($\exists\neg$), there is a specific person, namely a Canadian princess, who John did not marry.

The second challenge has to do with quantifiers is the lexical differences with respect to the ways of scope taking [BS97]. We use [X > Y] to mean 'X has scope over Y'.

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

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

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

In (2) the preferred reading is for negation to scope over the existential QP in (2-a) and over the universal QP in (2-b). However, while the existential QP is free to scope over negation (2-a), the universally quantified object can scope over negation only if focussed. The contrast in (3) shows that while *every* has the distributivity property, *all* lacks it as emphasized by the presence of *different*. Finally, (4) illustrates the inability of *few abstracts* to take scope over a QP preceding it in the surface structure (s-structure).

6.2 QPs in Type Logic

In [Tho74] Montague gives a solution to the first problem presented by quantifier scope, namely the ability of QPs to take scope wider than where they occur at s-structure. His proposal was to use syntactic rules of quantification. We look at the simplest case which helps reaching a more formal description of the problem at hand.

Definition 6.1. [Quantifying-In] If $\alpha \in P_{(e,t),t}$ and $\phi \in P_t$, then $F_n(\alpha, \phi) \in P_t$, and $F_n(\alpha, \phi) = \phi'$, where ϕ' is the result of the following substitution in ϕ:

i. If α is not a syntactic variable he_k, then replace the first occurrence of he_n or him_n with α, and the other occurrences of he_n or him_n with appropriate anaphoric pronouns;

ii. if $\alpha = he_k$, then replace every occurrence of he_n with he_k and of him_n with him_k.

This approach has been criticized in [Coo83], because it creates unnecessary (derivational) syntactic ambiguities. As an alternative, Cooper proposes a storage mechanism aiming to eliminate the quantifying-in operation from the syntactic level, thus isolating the effects of quantification in the semantics. In [Hen93] Hendriks proposes a *flexible* type assignment strategy which leads to a more adequate division of labor between the

syntactic and semantic components, making both quantifying-in and storage mechanism superfluous.

The starting point of Hendriks' approach is loosen the traditional functional correspondence between syntactic categories and semantic types (see Section 1.3) and assume a flexible type assignment. The system so obtained shares Cooper's idea of using a set of interpretations. However, differently from the interpretation used in the storing mechanism, the sets are obtained by means of general type-shifting rules. Consequently, contrary to Montague's approach of generalizing to the worst case[2] Hendriks' flexible type assignment can start from the best case, *viz.* the minimal type adequate for that expression. The recursive nature of the type-shifting rules, in fact, excludes the existence of a 'worst case'. The type-shifting rules required to account for the full sets of scope possibilities of a QP are the ones given below.

DEFINITION 6.2. [Type-Shifting Rules] Let $(\vec{a}, b)$ stand for $(a_1, (\ldots (a_n, t)\ldots))$, where $a_1 \ldots a_n$ are arbitrary types and $n \geq 0$. The type-shifting rules are: Value Raising (VR), Argument Lowering (AL), and Argument Raising (AR)[3].

VR $(\vec{a}, b) \longrightarrow_{\text{VR}} (\vec{a}, ((b, d), d))$;
AL $(\vec{a}, (((b, d), d), (\vec{c}, d))) \longrightarrow_{\text{AL}} (\vec{a}, (b, (\vec{c}, d)))$;
AR $(\vec{a}, (b, (\vec{c}, d))) \longrightarrow_{\text{AR}} (\vec{a}, ((((b, d), d), b), (\vec{c}, d)))$.

On closer inspection, it turns out that with the exception of AR, the directional versions of Hendriks' type-shifting schemata are in fact derivable within the categorial base logic NL. The specific instances of AR (in their directional version) where the sequence $\vec{c}$ is empty are valid in NL too. But the general version of AR needed for scope construal requires a structural extension of the base logic.

Moortgat [Moo91] provides such an extension in the form of a three-place binding type constructor q. The intuitive interpretation of the subtypes is the following: syntactically, an expression of type $q(A, B, C)$ occupies the position of an expression of type A within a structural context of type B; using the q connective turns the B domain into an expression of type 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 N.D. rule below encapsules this combined syntactic/semantic behavior. With a typing $q(np, s, s)$ for generalized quantifier expressions, one derives the Montagovian Quantifying-In rules, or the instances of Hendriks' type shifting rules, as special cases.

$$\frac{\Gamma \vdash \alpha : q(A, B, C) \quad \Delta[x : A] \vdash \beta : B}{\Delta[\Gamma] \vdash \alpha(\lambda x.\beta) : C}\ [q\text{E}]$$

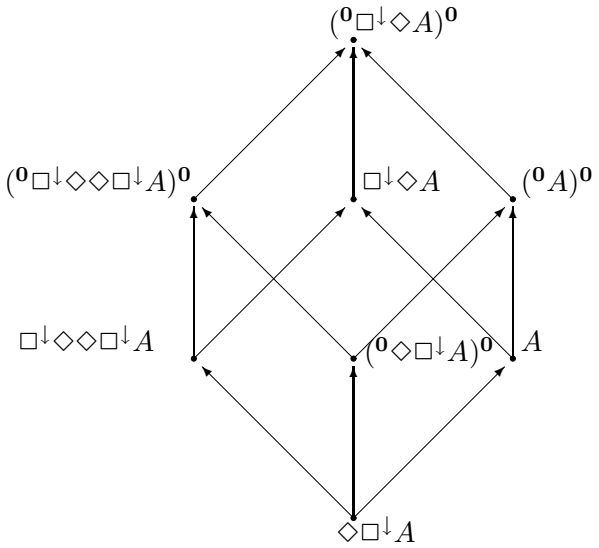
In the multimodal setting of Chapter 2, the q connective of course cannot be a *primitive* connective: the challenge is to show how it can be synthesized in terms of logical and

[2]Worst case type approach: the highest type required by some expression in a certain syntactic category is uniformly assigned to all members of that category.

[3]For expository purposes, we give a simplified version which does not include the intensional type s associated with the set of possible worlds considered in [Hen93].

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]. The exact details of the decomposition are not directly relevant to the issues dealt with in this thesis. In what follows, we will use the simple format of [qE] as a derived rule of inference.

In multiple quantifier environments, the type shifting rules of Hendriks as well as the account in terms of the q binding operation deliver the full set of combinatorially possible scope relations. The accounts, in other words, solve the problem of non-local scope discussed in Section 6.1, but they do not address the problem of non-uniform scope possibilities for generalized quantifier expressions. The vocabulary of the extended type language we have introduced in Chapter 2 provides logical tools to replace the type assignment $q(np, s, s)$ by refined versions where the subtypes are decorated with unary modalities. The modal prefixes studied in Chapter 2 and the derivability patterns among them are repeated in the figure below.



Let us take a closer look at the linguistic data by presenting the analysis they have received in the minimalist program.

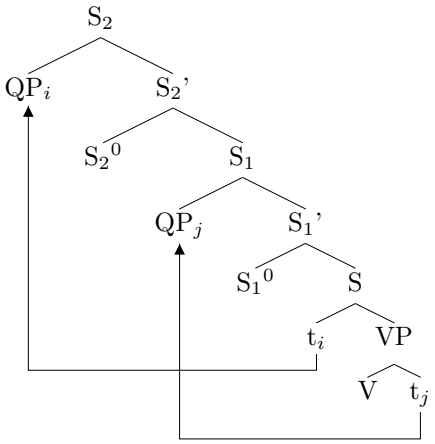
6.3 QPs in Generative Grammar

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)[4]; (ii) QPs are assigned scope by undergoing *movement* to their scope positions in the derivation of the LF representations. Let us illustrate these two claims by means of an example.

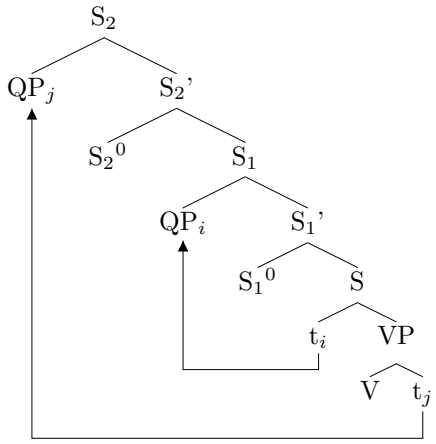
A generative grammar can produce different LF-structures for the same (ambiguous) well-formed clause. Due to the different c-command relations at the level of LF the surface structure (s-structure) receives different meanings. For example, a structure with two quantifiers, *every boy read two books*, would be interpreted by means of the following LF structures, where QP_i and QP_j abbreviate *every boy* and *two books*, respectively.

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

1) $\mathrm{QP}_i > \mathrm{QP}_j$



2) $\mathrm{QP}_j > \mathrm{QP}_i$



Quantifiers are raised from their surface positions (i.e. where the traces t_i, t_j are) to their scope position adjoining to S. For expository reasons we have used indexes on the sentential categories but they are not meant to mark any difference between the functional category S_1 and S_2. Therefore, the two QPs can move to the specifier SPEC of either of the two S-nodes, resulting in two different interpretations. Movement takes a s-structure and returns LF representations that are unambiguous with respect to quantifier scope relations: the c-command relations between the adjoined quantifiers determine which is in the scope of which. For instance, in 1) since QP_i c-commands QP_j, the reading assigned to the original surface structure is the *direct* one with QP_i having wide scope over QP_j. The *inverse* scope reading, with *two books* having wide scope, is obtained by moving the quantifiers in the opposite ways as illustrated in 2).

The movement operation marked by the arrows introduces a variable (a trace) which is bound by the moved constituent (the quantifier). Recalling what we said when introducing the $q(np, s, s)$ type —the np is the bound expression in the sentential domain, and the whole operation returns a sentence— it becomes clear that the $[q\mathrm{E}]$ rule of CTL achieves the effects of the movement operation spelled out above. In Section 6.4, we will work this out in more detail.

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 SPEC of the node dominating the selcted 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 corresponds 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 first look at the analysis they propose.

6.3.1 QP Classification

In [BS97] it is shown that (i) scope interaction 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*. If they do, then the discourse referent must be bound/checked by some logical operator hosted in the HEAD of a functional projection (FP) (see Section 3.2), therefore the QP must move to the SPEC of such a FP. If they do not, movement is not required and the QP takes scope locally in its case-position. 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 definites 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 (5-a), while members of the first group can (5-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 (5-c).

(5) 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 (6-a) is awkward, and (7-a) does not have a generic reading. On the other hand, the set of variables introduced by *every* can be bound by negative (6-b) and generic operators (7-b) as well as by the distributive operator (5-b). This contrast gives rise to different scope possibilities.

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

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

Recall 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 (8-a,b). Therefore, it is considered to be part of GQPs group since it behaves like them with respect to negation (9-a,b), though it differs from the other members for not being able to work as distributed share of DQPs (5-b) and (9-c).

(8) 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].

(9) 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*.

6.3.2 Feature Checking Theory for QP Scope

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 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 6.1 where the head-positions are compiled in for the ease of presentation.

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) (10-a.), but can have scope on QPs

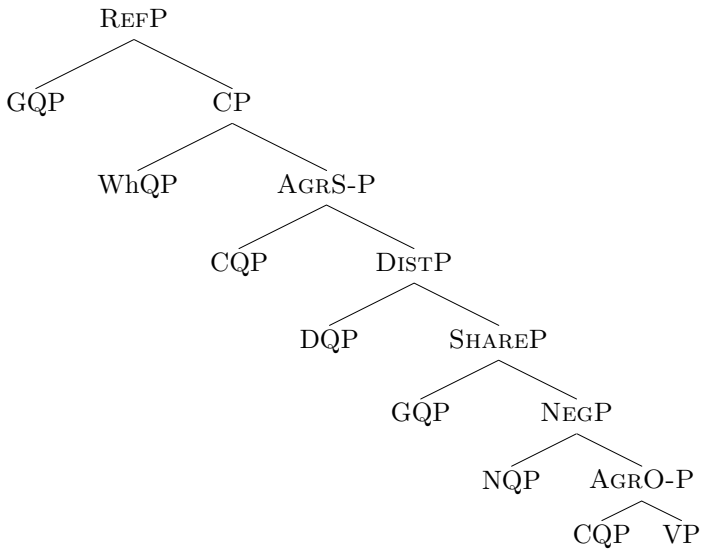


Figure 6.1: Phrase structure for QPs.

which are on a lower level, e.g. *few books* which stay in its case position, Spec-AgrO-P (10-b).

(10) 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 (4-a) and (4-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[5]. 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.

6.4 Controlling Scope Distribution in CTL

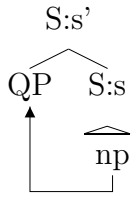
The aim of this section is to obtain the scope constraints discussed above deductively by means of modally refined type assignments. We start by comparing the movement operation used within the minimalist framework with the $[q\text{E}]$ rule used in CTL. Via this comparison we learn how to decorate the type assignment $q(np, s, s)$ with unary modalities as to control the scope distribution of the different QPs. Once the method

[5]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.

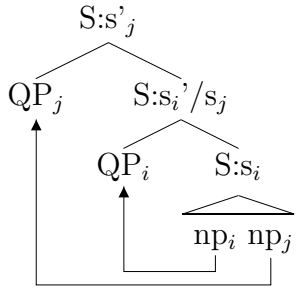
has been explained, we look back at the possible types at our disposal and start exploring the landscape of natural language quantifiers.

6.4.1 Modalities for Feature Checking

In Section 6.2, we have seen that the type $q(np, s, s)$ assigned to QPs in [Moo91] 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. 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 represent 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 (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 Section 6.2, we have presented the full scale of derivability patterns at our disposal. The simplest cases are repeated in Figure 6.2.

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

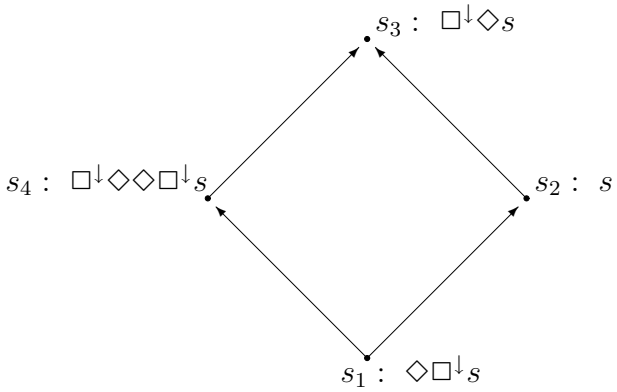


Figure 6.2: Basic sentential levels.

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.

6.4.2 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 the 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.

(11) 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.

(12) 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 6.1. We refer to the types using the corresponding abbreviations s_i given in Figure 6.2. Recalling what said in Section 6.4.1 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 others 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 6.1: 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 (11) follow from the lexical assignments in Table 6.1 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 6.3, 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 1.3) 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 6.3 builds the lambda terms as shown in Figure 6.4, where the Q is a variable to be replaced by the actual term representing the quantifier in the structure.

The first derivation in Figures 6.3 and 6.4 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 6.1 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

[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 6.3: Wide and narrow scope negation.

[Not > QP]

$$\dfrac{\dfrac{\mathtt{j} : np \qquad \dfrac{\lambda P.\neg P : (np\backslash s_2)/(np\backslash s_2) \qquad \dfrac{\dfrac{\boxed{Q : q(np, s_x, s'_y)} \qquad (\mathtt{Read}\ x)\ y : s_x}{Q(\lambda x.(\mathtt{Read}\ x)\ y) : \boxed{s'_y}}\,[q\text{E}]^1}{Q(\lambda x.(\mathtt{Read}\ x)\ y) : \boxed{s_2}}\,[D_y]}{\lambda y.Q(\lambda x.(\mathtt{Read}\ x)\ y) : np\backslash s_2}\,[\backslash\text{I}]^2}{\lambda z.\neg Q(\lambda x.(\mathtt{Read}\ x)\ z) : np\backslash s_2}\,[/\text{E}]}{\neg Q(\lambda.x(\mathtt{Read}\ x)\mathtt{j}) : s_2}\,[\backslash\text{E}]}{\neg Q(\lambda x.(\mathtt{Read}\ x)\mathtt{j}) : s_3}\,[D_2]$$

[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 6.4: Meaning assembly.

$[\mathrm{QP}_{\mathrm{sub}} > \mathrm{QP}_{\mathrm{obj}}]$

$$\dfrac{\boxed{\mathrm{QP} \vdash q(np, s_x, s'_y)} \qquad \dfrac{\dfrac{\dfrac{\boxed{\mathrm{QP} \vdash q(np, s_u, s'_v)} \qquad \dfrac{\dfrac{[x \vdash np]^2 \quad [y \vdash np]^1}{\vdots}}{x \circ (\mathrm{TV} \circ y) \vdash s_1}\,[D_1]}{x \circ (\mathrm{TV} \circ y) \vdash s_u}\,[qE]^1}{x \circ (\mathrm{TV} \circ \mathrm{QP}) \vdash \boxed{s'_v}}}{x \circ (\mathrm{TV} \circ \mathrm{QP}) \vdash \boxed{s_x}}\,[D_v]}{\mathrm{QP} \circ (\mathrm{TV} \circ \mathrm{QP}) \vdash s'_y}\,[qE]^2$$

$$\dfrac{\mathrm{QP} \circ (\mathrm{TV} \circ \mathrm{QP}) \vdash s'_y}{\mathrm{QP} \circ (\mathrm{TV} \circ \mathrm{QP}) \vdash s_3}\,[D_y]$$

$[\mathrm{QP}_{\mathrm{obj}} > \mathrm{QP}_{\mathrm{sub}}]$

$$\dfrac{\boxed{\mathrm{QP} \vdash q(np, s_u, s'_v)} \qquad \dfrac{\dfrac{\dfrac{\boxed{\mathrm{QP} \vdash q(np, s_x, s'_y)} \qquad \dfrac{\dfrac{[x \vdash np]^2 \quad [y \vdash np]^1}{\vdots}}{x \circ (\mathrm{TV} \circ y) \vdash s_1}\,[D_1]}{x \circ (\mathrm{TV} \circ y) \vdash s_x}\,[qE]^2}{\mathrm{QP} \circ (\mathrm{TV} \circ y) \vdash \boxed{s'_y}}}{\mathrm{QP} \circ (\mathrm{TV} \circ y) \vdash \boxed{s_u}}\,[D_y]}{\mathrm{QP} \circ (\mathrm{TV} \circ \mathrm{QP}) \vdash s'_v}\,[qE]^1$$

$$\dfrac{\mathrm{QP} \circ (\mathrm{TV} \circ \mathrm{QP}) \vdash s'_v}{\mathrm{QP} \circ (\mathrm{TV} \circ \mathrm{QP}) \vdash s_3}\,[D_v]$$

Figure 6.5: Structures with multiple QPs.

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 6.3 correctly account for the data in (11), and we can now move to consider multiple quantifiers contexts sharing the structure [QP [TV QP]].

Comparing the derivations in Figure 6.5 with the tree given in Section 6.4.1 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 (5) and (8-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 6.1.

(13) a. Some student visited few girls. [Some > Few].
b. Every student visited few girls. [Every > Few].

(14) 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 (13-a) to mean that for few girls it is the case that some student visited her. On the other hand, the structures in (14) 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 below.

(15) 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 others 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.

(16) a. Few students read each book. [Few > Each].
b. Few students didn't go to the party. [Few > Not].
c. Few students read the books. [*Few > The], [The > Few].
d. A student read few books. [A > Few], [*Few > A].
e. 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.

6.4.3 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 6.2 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 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.

(17) 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 (18) between *few_n* and *exactly five_n* [SZ97] seems to suggest that CQPs should be further subcategorized.

(18) 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, as we have briefly discussed in Section 5.1, wh-phrases exhibit different behavior with respect to weak islands. In particular, while *who* can escape islands formed by negation (19-a), *how* cannot (19-b). We repeat the data below [SZ97].

(19) 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 will come back to this point in the next chapter. We now move on to consider the type classification obtained by extending

the derivability patterns used so far. The whole picture given in Section 6.2 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*. Recall from Section 4.1 that *anybody* 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 Chapter 7, 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, as we will show in the next chapter, it is predicted that NPIs cannot occur in any of the two constituents of the conjunction whereas the other QPs can (20) and (21). On the other hand, the standard c-command analysis fails to predict these data as discussed in [Pro00, Hoe00].

(20) 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.

(21) 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.

6.4.4 A Problem of the Minimalist Analysis

An interesting problem encountered when using Beghelli and Stowell's phrase structure is the analysis of sentences like the one below, where two quantifiers of the same class interact with a third one[6].

(22) Every student gave some teacher every book (he had).

a. Every > Some > Every,
b. Some > Every > Every,
c. Every > Every > Some.

The problem comes from the fact that quantifiers of the same class must land in the same SPEC position. Therefore, when these two quantifiers co-occur in a sentence together with a third one, the only readings which can be derived are the ones with the two quantifiers of the same class having scope one immediately over the other. This means

[6]This failure of the Beghelli and Stowell's analysis has been noted by Ø. Nilsen (p.c.).

that the reading (a) cannot be derived. These cases are not problematic for the CTL account.

The example above can be connected with the one given in (12) and repeated in (23-a), where a GQP intervenes at LF between negation and the distributive quantifiers, allowing *each* to outscope negation. Similarly, as observed in [Sza01], negation can outscope the positive polarity item *someone* if there is an intervener (24-b).

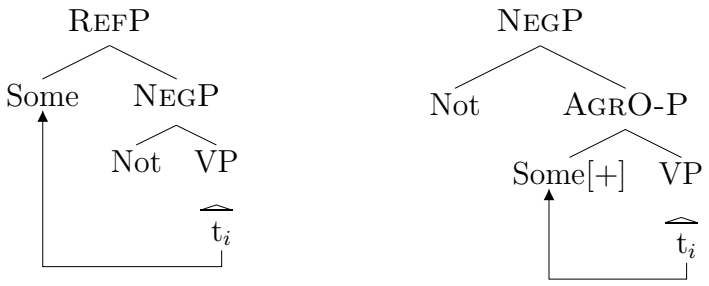
(23) a. One boy didn't read each book. [Each > One > Not].
b. %John didn't read each book.

(24) a. John didn't meet someone. [Some>Not], [*Not>Some].
b. John didn't always meet someone. [Some>Not>Alw.], [Not>Alw.>Some].

In all these cases, the presence of an intervener modifies the scope possibilities of the elements involved. This cannot be easily accommodated into a sequence of functional projections since extended projection lines are fixed and a constituent, e.g. a quantifier, must move to the suitable SPEC position regardless of interveners[7]. Let us look at the way the sentence (24) can be interpreted in the minimalist framework.

In order to block the negation to have wide scope over *someone* (24-a), the latter must land to a node higher than NegP. Following the minimalist analysis we have presented above, we can consider REFP to be this node. The positive polarity item must carry a feature [+] which will force it to move higher than NEGP and the HEAD-REFP must have this feature attracting the positive polarity item. This analysis correctly predicts the interpretation of (24-a) as schematically illustrated by the trees below.

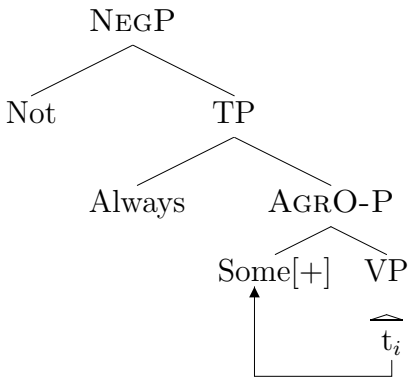
(a_1) (a_2)



The derivation in (a_1) converges at LF since no features are left to be checked, and the structure *John didn't meet someone* is interpreted with *some* outscoping negation. Instead, the derivation in (a_2) crashes since it still has a feature to be checked. In other words, the reading with *not* having wide scope cannot be given, since *someone* cannot take scope locally in AGRO-P. This same feature of the positive polarity item unables *not* to outscope it, even though an intervener occurs, whereas linguistic data show that the presence of an intervener modifies the scope possibilities (24-b). The failure of

[7] An account of PPIs into functional projection terms has been given in [Pro01]. There, however, the case of a scope element intervening between negation and PPI is not considered. See [Sza01] for an alternative solution.

this analysis to predict the linguistic data is exemplified by the derivation below which cannot be interpreted since it contains the feature [+].



In the CTL framework scope possibilities are determined *dynamically* in the step by step construction of the derivation: what matters is the derivability relation holding between the type of a constituent and the one of the element in its *immediate* scope. As a consequence interveners change the ways QPs take scope, and (24-b) above are correctly predicted as illustrated by the derivation below. Recall that the type of positive polarity quantifiers like *someone* is $q(np, s_3, s_4)$.

[Not > Alw. > Some]

$$\dfrac{\text{didn't} \vdash (np\backslash s_2)/(np\backslash s_2) \qquad \dfrac{\text{always} \vdash (np\backslash s_1)/(np\backslash s_3) \qquad \dfrac{\dfrac{\vdots}{\text{meet} \circ \text{someone} \vdash np\backslash s_4}\,[\mathrm{D}_4]}{\text{meet} \circ \text{someone} \vdash np\backslash s_3}}{(\text{always} \circ (\text{meet} \circ \text{someone})) \vdash np\backslash s_1}\,[/\mathrm{E}]}{(\text{always} \circ (\text{meet} \circ \text{someone})) \vdash np\backslash s_2}\,[\mathrm{D}_1]}{\text{didn't} \circ (\text{always} \circ (\text{meet} \circ \text{someone})) \vdash np\backslash s_2}\,[/\mathrm{E}]$$

6.5 Internalizing Feature Checking

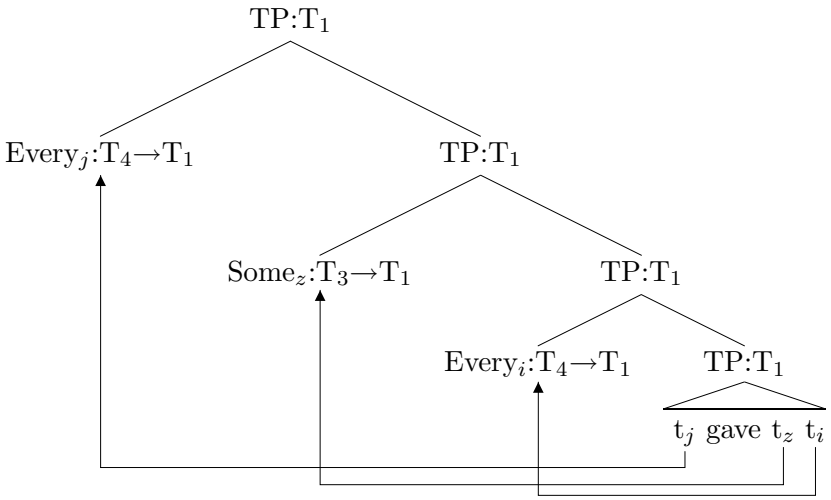
The preceding discussion suggests a way to refine feature checking by rethinking functional projections in terms of a deductive calculus of syntactic features. As suggested above, the reason why the functional projection hierarchy fails to deal with interveners is the fact that the features are checked after the tree has already been derived. The operation building the phrase structure, Merge, and the one taking care of feature agreement Agree are separated. Making things more explicit one could say that phrases carry two sorts of features: (i) features determining the selected node and (ii) features driving movement. For instance, the functional projection DistP selects for ShareP and attracts a quantifiers specified for distributivity. Schematically, this can be expressed by assigning to DistP the features [ShareP $\rightarrow$ DistP] and [$\forall$] in charge of (i) and (ii), respectively. A solution to the problems pointed out in the previous section can be found by *internalizing* the Agree operation into Merge and work with only one sort of feature carrying both types of information.

Feature checking can be seen as an asymmetric process [BN02] where the two expressions involved play different roles. The checker, has two features α_1, α_2 whereas

the CHECKED expression, has one feature β. Checking can apply when β AGREEs with α_1 [Cho99]. If this holds, the two categories can MERGE, yielding an expression with the feature α_2 (feature VALUATION, cf. [Cho99]). If $\alpha_1 \neq \alpha_2$, it is crucial to know which of them enters into AGREE with β. By considering α_1, α_2 as the antecedent and consequent of the logical implication $\rightarrow$, we have used to express the 'selecting' features, we achieve this. In this way, the feature of the checker becomes $\alpha_1 \rightarrow \alpha_2$ and feature checking together with MERGE reduces to an application of modus ponens. In order for modus ponens to apply to $\alpha_1 \rightarrow \alpha_2$ and β, β must derive (AGREE) with α_1. In other words, AGREE is reduced to a derivability relation in a deductive calculus of syntactic features. Let us look at an example to clarify matters.

Suppose the tense phrase TP has the feature T_1 which derives three other tense levels T_2, T_3, T_4, where T_2 and T_4 derives T_3. We can now control the order with which elements merge to TP by attributing different features to the QPs. The contrast in (23) is predicted by assigning $[T_2 \rightarrow T_2]$ to *didn't*, $[T_4 \rightarrow T_4]$ to *each* and $[T_3 \rightarrow T_1]$ to *one*.

The reading in (23-a) is derived as follows: the negation merges with TP, yielding $[_{TP}$ not...] with the feature $[T_2]$. Since T_2 agrees with T_3, *one* can merge with this phrase yielding $[_{TP}$ one $[_{TP}$ not ...]] with the feature $[T_1]$ which can be merged with *each* since $[T_1]$ agrees with its antecedent feature $[T_4]$. On the other hand, if one tries to merge *each* directly with $[_{TP}$ not...] the derivation fails, since the feature of this expression $[T_2]$ does not agree with the antecedent features of *each* $[T_4]$. Similarly, the other cases are predicted by this deductive feature mechanism which dynamically builds the phrase structure of the minimalist program. The reading (22-a) is obtained by the phrase structure below where the features carried by the HEAD-TP are displayed on the TP nodes.



The phrase structure results to be correctly built because SPEC-HEAD agreement is satisfied.

We conclude this section by noticing that the comparison we have given between the CTL and generative grammar approach to QPs has shown that modelling linguistic phenomena often requires a way of encoding (semantic) differences between items of the same syntactic category. In the generative grammar tradition this is done by means of

features, whereas in CTL one can exploit the logical properties of the unary operators. An important difference between the two solutions consists in the nature of the tools employed: in the first case the features assigned to QPs do not have any intrinsic logical properties, whereas in the second case they are logical constants of the system for which we have well defined rules of inference.

Furthermore, the discussion of quantifier scopes shows how CTL can profit from the studies of the empirical data done in the generative grammar tradition. On the other hand, the change of perspective, from the generative framework to the type logical one, sheds light on some problems of the minimalist analysis otherwise hidden into the system. Finally, it helps gain a better understanding of the linguistic phenomena and reach a refinement of the theory.

6.6 Key Concepts

In this chapter, we have given a type logical account of the different scope distribution of quantifier phrases. We have

1. Surveyed the classification of QPs presented by Beghelli and Stowell [BS97].
2. Given modally decorated types to account for the different ways QPs take scope.
3. Shed light on new classes of QPs not considered in the linguistic theory.
4. Shown how the comparison between a logical framework and a more linguistically oriented one can be advantageous for both systems.

Chapter 7

Licensing and Antilicensing Relations

The aim of this chapter is to investigate the licensing and antilicensing relations holding between a sensitive item and a trigger. By using categorial type logic we shed light on the effects of the inclusion relations holding among a trigger and the other elements of the same semantic domain.

We start by presenting a type logical analysis of the licensing relation (Section 7.1). We then present a cross linguistic comparison of this composition relation by looking at Dutch and Greek negative polarity items (Section 7.2). In Section 7.3, we formally characterize the family of contexts where these items must occur. The subset relations holding among these contexts are encoded by means of the derivability relation among type assignments which lexically anchor the distribution of Dutch and Greek negative polarity items. The analysis is then extended to Italian (Section 7.4). Finally, the same logical approach is applied to model antilicensing relations by looking in particular at Dutch positive polarity items (Section 7.5).

7.1 Licensing Relations in CTL

Linguistic signs of a same semantic type may differ in their distribution behavior. This is expressed by differentiating the composition relation which governs the assembly of linguistic expressions. In particular, some items are compatible or incompatible with a certain semantic property. Thus, they are grammatical or ungrammatical when in construction with expressions having such a property. Moreover, there exist items which are in a licensing or antilicensing relation with a certain property. Therefore, they 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. In a slogan an item 'can' or 'cannot', or 'must' or 'must not' be in construction with a trigger having a certain property.

In this chapter, we show how these forms of composition can be expressed by categorial types. To express this demands of the items we need something more than simple function application and abstraction. By way of illustration, we look at negative and positive polarity items as a case of expressions in a licensing and antilicensing relation

with their triggers, respectively. In both cases, we make use of the unary operators of CTL.

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 others 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 derivability relations among their types and the inheritance relations among the direct trigger and its relatives is determined deductively. More specifically, due to the logical property of the functional connectives a structure of type A/B (or $B\backslash A$) composes with a structure of any type C such that $C \longrightarrow B$. This will be the main property we exploit in our analysis of licensing and antilicensing relations.

A second interesting fact about these phenomena is that these compositional relations differ across languages. For instance, while *possibly* does not license *any*, their Greek counterparts are in a licensing relation. CTL helps clearly grasp these differences simply by means of lexical assignments. In this chapter, we will look at Greek and Dutch negative polarity items and build a lexicon for the two fragments of natural languages. Furthermore, based on this analysis, we look at some Italian data and suggest a first classification of the negative polarity items taken into consideration.

7.1.1 Licensing Relations as Features Exchanges

In Chapter 4, we already encountered negative polarity items and gave a first analysis of them in connection with monotone functions. There we focused on natural reasoning and the account of polarity items was a side effect of the feature marking carried by the downward monotone functions' types and of the computation of the polarity positions exerted by the structural language. In this chapter, we want to shed light on the licensing relation between the NPIs and their triggers —which traditionally are considered to be downward monotone functions [Lad79]. Therefore, we can leave out the computation of the polarity carried by the structural marking and modify the types accordingly.

Recall that the downward monotone quantifier *nobody* and the negative polarity adverb *yet* were assigned the type $s/\Diamond(np\backslash s)$ and $\Box^{\downarrow}\Diamond(np\backslash s)\backslash\Box^{\downarrow}\Diamond(np\backslash s)$, respectively. The $\Box^{\downarrow}\Diamond$ in the argument position of the *yet*-type was employed to allow multiple occurrences of NPIs, whereas the $\Box^{\downarrow}$ on the value formula was intended to mark the structure where the item occurs before being taken as argument by the licensor. For the reason just explained, we do not need this marking process here, and we can replace the type of the negative quantifier with $s/\Box^{\downarrow}\Diamond(np\backslash s)$, so that no flow of polarity information from the logical to the structural level is performed as illustrated by the derivation below.

$$\dfrac{\text{nobody} \vdash s/\Box^{\downarrow}\Diamond(np\backslash s) \qquad \dfrac{\dfrac{\dfrac{\text{left} \vdash (np\backslash s)}{\langle\text{left}\rangle \vdash \Diamond(np\backslash s)}\,[\Diamond\text{I}]}{\text{left} \vdash \Box^{\downarrow}\Diamond(np\backslash s)}\,[\Box^{\downarrow}\text{I}] \qquad \text{yet} \vdash \Box^{\downarrow}\Diamond(np\backslash s)\backslash\Box^{\downarrow}\Diamond(np\backslash s)}{\text{left} \circ \text{yet} \vdash \Box^{\downarrow}\Diamond(np\backslash s)}\,[\backslash\text{E}]}{\text{nobody} \circ (\text{left} \circ \text{yet}) \vdash s}\,[/\text{E}] \tag{7.1}$$

Recall also that upward monotone functions were left unmarked, e.g. everybody: $s/(np\backslash s)$. Therefore, the type assigned by *yet* to the structure *left yet* would not match the one required by *everybody*.

In [Fry99], an account of negative polarity items is given within the framework of Lexical Functional Grammar (LFG) where multiplicative linear logic (MLL) is used as the semantic 'glue' language. See [Ber00] for a comparison between the approach described here and the one proposed in [Fry99]. The metaphor used there can help us explaining our type logical account of licensing relations. Polarity can be considered as a "resource" required by the NPI and produced by the licensor. In MLL this resource represented as a proposition ℓ carried by the type assignments. Let B/A and $C\backslash D$ be the standard logical types of the licensor and the NPI, respectively, the resource is added to them in the following way:

$$\text{Licensor: } B/((A \bullet \ell)/\ell) \qquad\qquad \text{NPI: } (C \bullet \ell)\backslash(D \bullet \ell)$$

The licensor's type introduces the resource hypothetically and can "clean it up" type-internally if it is not required elsewhere. The NPI's type, on the other hand, has the resource in both its antecedent and its succedent, therefore it simply "borrows" without really consuming the resource. This correctly predicts the behavior of NPIs. For example, the simple structure *Nobody left yet* is derived as shown below using CTL notation.

$$\dfrac{\text{nobody} \vdash s/((iv \bullet \ell)/\ell) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\text{left} \vdash iv \quad [x \vdash \ell]^1}{\text{left} \circ x \vdash iv \bullet \ell}\,[\bullet\text{I}] \qquad \text{yet} \vdash (iv \bullet \ell)\backslash(iv \bullet \ell)}{(\text{left} \circ x) \circ \text{yet} \vdash iv \bullet \ell}\,[\backslash E]}{\text{left} \circ (x \circ \text{yet}) \vdash iv \bullet \ell}\,[\text{Ass}]}{\text{left} \circ (\text{yet} \circ x) \vdash iv \bullet \ell}\,[\text{Per}]}{(\text{left} \circ \text{yet}) \circ x \vdash iv \bullet \ell}\,[\text{Ass}]}{\text{left} \circ \text{yet} \vdash (iv \bullet \ell)/\ell}\,[/\text{I}]^1}{\text{nobody} \circ (\text{left} \circ \text{yet}) \vdash s}\,[/\ \text{E}] \tag{7.2}$$

Comparing the derivation in 7.2 with the one given by CTL (7.1) we see a first desirable consequence brought by the switch from the use of binary to unary operators to carry special features: no structural rules are needed to prove the grammaticality of simple structures as the one above. By using unary operators in the type assignments of the licensor and the NPI, there is no need of a "phantom resources" as ℓ: The property of the

licensor of being a downward monotone function is not represented using a propositional formula as if it is comparable with the linguistic categories. Having the unary operators makes it possible to treat some items as having a specific property not shared with other items of the same linguistic class.

7.1.2 Negative Polarity Quantifiers

Negative polarity items can belong to different syntactic categories. In particular, in addition to the adverb *yet* there are also negative polarity quantifiers like *anybody*. Quantifier phrases (QPs) exhibit a particular scope behavior. We have largely studied them in Chapter 6 explaining how CTL accounts for the fact that QPs take wide scope over a structure while locally behaving as an *np*. In particular, we have discussed the application of the rule of use below

$$\frac{\Gamma \vdash \alpha : q(A,B,C) \quad \Delta[x:A] \vdash \beta : B}{\Delta[\Gamma] \vdash \alpha(\lambda x.\beta) : C}\ [q\mathrm{E}]$$

Given QPs of type $q(np, s, s)$, the rule above performs the replacement of the hypothetical *np* with the quantifier, while the lambda term accounts for the QP taking wide scope on the semantic level. Therefore, the type $q(np, s, s)$ assigned to QPs means that a quantifier is a binder of an *np* type variable within a sentential domain s, returning a sentence s. Finally, we have shown that QPs differ in their scope possibilities requiring more fine-grained type assignments obtained by differentiating their sentential types.

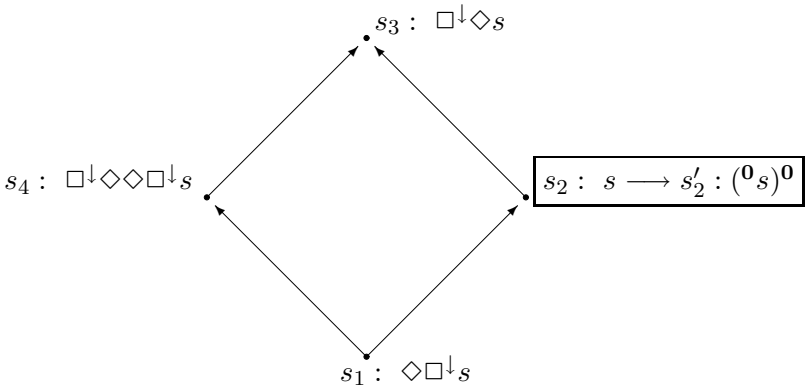
The scope distribution of negative polarity quantifiers differs from the one of the other quantifiers for being governed by the licensing relation we have discussed above. Therefore, their logical type assignment must encode this relation. Applying the general method proposed for differentiating QP scope distribution, we consider *any_n* to be of type $q(np, ({}^{\mathbf{0}}s)^{\mathbf{0}}, ({}^{\mathbf{0}}s)^{\mathbf{0}})$. This type allows us to extend the lexicon described in Section 6.4 maintaining the types for the other QPs as discussed there. The only types which must be modified are the ones assigned to the NPIs' licensors. The type assigned to *didn't*, $(np\backslash s)/(np\backslash s)$, is replaced with $(np\backslash s)/(np\backslash({}^{\mathbf{0}}s)^{\mathbf{0}})$ and similarly the type of *nobody* is now $q(np, ({}^{\mathbf{0}}s)^{\mathbf{0}}, s)$. This change does not effect our previous analysis since $s \longrightarrow ({}^{\mathbf{0}}s)^{\mathbf{0}}$, and therefore the new types derive the old ones. The type of the adverb *yet* considered above changes consequently as in Figure 7.1.

any_n	$q(np, s_2', s_2')$	*nobody*	$q(np, s_2', s_2)$;
each_n	$q(np, s_4, s_4)$	*every_n*	$q(np, s_4, s_1)$;
all_n	$q(np, s_3, s_2)$	*the_n*	$q(np, s_3, s_3)$;
a_n	$q(np, s_3, s_1)$	*left*	$np\backslash s_1$;
didn't	$(np\backslash s_2)/(np\backslash s_2')$	*yet*	$(np\backslash s_2')\backslash(np\backslash s_2')$.

Figure 7.1: Lexicon.

From now onwards, we will use s_2' to abbreviate $({}^{\mathbf{0}}s)^{\mathbf{0}}$, and the other labels are used as in the previous chapter (Figure 6.2), and as repeated here. The box in the figure

emphasizes the new derivability relation we exploit in this chapter.



Let us now look at some data. Recall from Section 4.1 that NPIs can be either in the same clause of their trigger, or in an embedded sentence while the trigger is in the matrix sentence (1-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 (1-f) [ES73].

(1) 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].

From the types in Figure 7.1 the ungrammaticality of sentences (1-a) and the grammaticality of (1-b) follow immediately.

We give the derivation in Example 7.1 where the relevant steps are highlighted with boxes. The [D$_i$] abbreviates a derivation from a type s_i to the type in the conclusion, while [D*] marks where the derivation fails.

Example 7.1. [Licensing of Negative Polarity Items]

1. **Anybody left.*

$$\dfrac{\dfrac{\boxed{\text{anybody} \vdash q(np, s_2', s_2')} \qquad \dfrac{\dfrac{[x \vdash np]^1 \quad \text{left} \vdash np\backslash s_1}{x \circ \text{left} \vdash s_1}\,[\backslash\text{E}]}{x \circ \text{left} \vdash s_2'}\,[\text{D}_1]}{\text{anybody} \circ \text{left} \vdash \boxed{s_2'}}\,[q\text{E}]^1}{\text{anybody} \circ \text{left} \vdash \boxed{s_3}}\,[\text{D}^*]$$

2. *John didn't read anything.*

[Not > Any]

$$\dfrac{\text{John} \vdash np \qquad \dfrac{\boxed{\text{didn't} \vdash (np\backslash s_2)/(np\backslash s'_2)} \qquad \dfrac{\dfrac{\boxed{\text{anything} \vdash q(np, s'_2, s'_2)} \qquad \dfrac{\dfrac{[y \vdash np]^2 \quad [x \vdash np]^1}{\vdots}}{\dfrac{y \circ (\text{read} \circ x) \vdash s_1}{y \circ (\text{read} \circ x) \vdash s'_2}\,[\text{D}_1]}}{y \circ (\text{read} \circ \text{anything}) \vdash s'_2}\,[q\text{E}]^1}{\text{read} \circ \text{anything} \vdash np\backslash s'_2}\,[\backslash\text{I}]^2}{\text{didn't} \circ (\text{read} \circ \text{anything}) \vdash np\backslash s_2}\,[/\text{E}]}{\text{John} \circ (\text{didn't} \circ (\text{read} \circ \text{anything})) \vdash s_2}\,[\backslash\text{E}]}{\text{John} \circ (\text{didn't} \circ (\text{read} \circ \text{anything})) \vdash s_3}\,[\text{D}_2]$$

[*Any > Not]

$$\dfrac{\dfrac{\boxed{\text{anything} \vdash q(np, s'_2, s'_2)} \qquad \dfrac{\dfrac{[x \vdash np]^1}{\vdots}}{\dfrac{\text{John} \circ (\text{didn't} \circ (\text{read} \circ x)) \vdash s_2}{\text{John} \circ (\text{didn't} \circ (\text{read} \circ x)) \vdash s'_2}\,[\text{D}_2]}}{\text{John} \circ (\text{didn't} \circ (\text{read} \circ \text{anything})) \vdash \boxed{s'_2}}\,[q\text{E}]^1}{\text{John} \circ (\text{didn't} \circ (\text{read} \circ \text{anything})) \vdash \boxed{s_3}}\,[\text{D}^*]$$

The case of embedded sentences is more involved. It requires a distinction of the lexical assignments of the predicates. In particular, one has to distinguish negative predicates like *doubts*, from the positive ones. Moreover, among the latter the bridge predicates like *think* must be distinguished from the non-bridge ones like *shout*. Following the argumentation discussed in Chapter 4 while studying monotone functions composition, we consider the negative auxiliary *didn't* to have a polymorphic type and exploit again the derivability relations among sentential types to differentiate the predicates. The instantiation required here for the type of *didn't* is $((np\backslash s_2)/s'_2)/((np\backslash s_2)/s_1)$.

negative predicates	$(np\backslash s_3)/s'_2$	e.g. *doubts*;
bridge predicates	$(np\backslash s_2)/s_1$	e.g. *think*;
positive predicates	$(np\backslash s_3)/s_1$	e.g. *shout*.

These lexicon assignments predict the data in (1) with the NPI in the embedded sentence as illustrated in Figure 7.2. The first derivation is easily readable. To check the derivability of the second derivation one has to replace the VP-type with the appropriate ones. The derivation abbreviated by $[\text{D}_{x,y}]$ will fail in case VP is replaced with *doubts* or *shout*, since s_y is instantiated with s_3 and $s_3 \not\longrightarrow s_2$. On the other hand, the derivation succeeds in the case of *think* since the types already match. Finally, the

third derivation fails in [D_2] when replacing the VP with *shout* and *think*: since they are not downward monotone function they do not provide the NPI with the required feature ($s_2' : ({}^{\mathbf{0}}s)^{\mathbf{0}} \not\longrightarrow s_1 : \Diamond\Box^{\downarrow}s$). On the other hand, *doubts* carries this feature licensing *anybody*, but the composed structure cannot be taken as argument by *didn't* ($s_3 : \Box^{\downarrow}\Diamond s \not\longrightarrow s_2' : ({}^{\mathbf{0}}s)^{\mathbf{0}}$), as required by the linguistic data.

$$\dfrac{\text{John} \vdash np \qquad \dfrac{\text{doubts} \vdash (np\backslash s_3)/s_2' \quad \dfrac{\vdots}{\text{anybody} \circ \text{left} \vdash s_2'}}{\text{doubts} \circ (\text{anybody} \circ \text{left}) \vdash np\backslash s_3}\,[/\text{E}]}{\text{John} \circ (\text{doubts} \circ (\text{anybody} \circ \text{left})) \vdash s_3}\,[\backslash\text{E}]$$

$$\dfrac{\dfrac{\dfrac{\text{didn't} \vdash ((np\backslash s_2)/s_2')/((np\backslash s_2)/s_1) \quad \dfrac{\text{VP} \vdash (np\backslash s_y)/s_x}{\text{VP} \vdash (np\backslash s_2)/s_1}\,[\text{D}_{x,y}]}{\text{didn't} \circ \text{VP} \vdash (np\backslash s_2)/s_2'}\,[/\text{E}] \qquad \dfrac{\vdots}{\text{anybody} \circ \text{left} \vdash s_2'}}{(\text{didn't} \circ \text{VP}) \circ (\text{anybody} \circ \text{left}) \vdash np\backslash s_2}\,[/\text{E}]}{\text{John} \circ ((\text{didn't} \circ \text{VP}) \circ (\text{anybody} \circ \text{left})) \vdash s_2}\,[\backslash\text{I}]$$

$$\dfrac{\dfrac{\text{John} \circ ((\text{didn't} \circ \text{VP}) \circ (\text{anybody} \circ \text{left})) \vdash s_2}{\text{John} \circ ((\text{didn't} \circ \text{VP}) \circ (\text{anybody} \circ \text{left})) \vdash s_3}\,[\text{D}_2]$$

$$\dfrac{\dfrac{\text{didn't} \vdash (np\backslash s_2)/(np\backslash s_2') \qquad \dfrac{\text{VP} \vdash (np\backslash s_y)/s_x \quad \dfrac{\text{anybody} \circ \text{left} \vdash s_2'}{\text{anybody} \circ \text{left} \vdash s_x}\,[\text{D}_2]}{\text{VP} \circ (\text{anybody} \circ \text{left}) \vdash np\backslash s_y}\,[/\text{E}]}{\text{VP} \circ (\text{anybody} \circ \text{left}) \vdash np\backslash s_2'}\,[\text{D}_y]}{\text{didn't} \circ (\text{VP} \circ (\text{anybody} \circ \text{left})) \vdash np\backslash s_2}\,[/\text{E}]$$

Figure 7.2: NPIs in embedded sentences.

Finally, recall from Section 6.4.3 that a negative quantifier in the constituent of conjunction cannot license a NPI occurring in the other constituent. This is predicted by our types. Again conjunction is assigned a polymorphic type; when used to coordinate quantifiers phrases the required instantiation is *and*: $((s_3/(np\backslash s_1))\backslash q(np, s_1, s_3))/(s_3/(np\backslash s_1))$. It will fail to coordinate a quantifier NPI with any of the other QPs, whereas will succeed with all the other QP-types. For the sake of simplicity, we abbreviate the arguments of the conjunction with X.

$$\dfrac{\dfrac{\text{QP}_1 \vdash q(np, s_x, s_y)}{\text{QP}_1 \vdash s_3/(np\backslash s_1)}\,[\text{D}_{x,y}] \quad \dfrac{\text{and} \vdash (X\backslash q(np, s_1, s_3))/X \quad \dfrac{\text{QP}_2 \vdash q(np, s_u, s_v)}{\text{QP}_2 \vdash s_3/(np\backslash s_1)}\,[\text{D}_{u,v}]}{\text{and} \circ \text{QP}_2 \vdash X\backslash q(np, s_1, s_3)}\,[/\text{E}]}{\text{QP}_1 \circ (\text{and} \circ \text{QP}_2) \vdash q(np, s_1, s_3)}\,[/\text{E}]$$

As the reader can check $q(np, s_2', s_2') \not\longrightarrow s_3/(np\backslash s_1)$, while for all the other QP-types in our lexicon the derivation holds.

7.2 Crosslinguistic Comparison

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 and Greek by way of example.

7.2.1 Dutch Negative Polarity Items

As anticipated in Section 5.2, 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[1], *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 7.1.

(2) 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.

(3) 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.

[1]The laws of De Morgan are: 1. $f(X \cup Y) = f(X) \cap f(Y)$, 2. $f(X \cap Y) = f(X) \cup f(Y)$.

(4) 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 7.1: Negative polarity items distribution in Dutch.

Giannakidou observes that the (medium) negative polarity item *ook maar iets* (tr. anything) is grammatical also in questions (5-a) and in *as soon as* clause (5-b) when interpreted as habitual [SWZ93, Gia97]. Based on these observations and on her study of Greek data, she proposes to enlarge the scale of the licensors' classification to the broader concept of nonveridicality. We do not take a position here on which of the two analyses would be the correct one for Dutch NPIs. If such a switch to nonveridicality should be taken, it would be necessary to give a more detailed analysis accounting for the differences shown by the wide range of Dutch data studied in [Wou94].

(5) a. Heb je ook maar iets gezien?
have you anything seen
tr. Did you see anything?

b. De kinderen vertrokken *zodra* zij ook maar iets ontdekten.
the children left.3pl as soon as they anything discovered.3spl
tr. The children used to leave as soon as they saw anything.

7.2.2 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][2], 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.

(6) 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?

(7) 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?

[2] *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.

(8) 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 7.2.

	NPI	API	FCI	Min
Veridical	–	–	–	–
Antiveridical	+	+	–	+
Opaque	–	+	+	–
Nonveridical	–	+	–	–

Table 7.2: 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 (9-a) and (9-b).

(9) 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.

7.3 (Non)veridical Contexts

As emphasized by Giannakidou the two analyses described in the previous section about Dutch and Greek data are not incompatible. First of all, polarity items might show different sensitivity properties across languages. Moreover, for the two languages at hand, the properties which have been investigated are logically related. In this section, we explore this relation in more formal terms. Let us start by introducing the definition of nonveridical functions given in [Zwa95].

DEFINITION 7.2. [(Non)veridical Operators (I)] Let O be a truth-conditional operator,

i. O is *veridical* iff $O(p) \Rightarrow p$ is logically valid. Otherwise O is *nonveridical*;

ii. A nonveridical operator O is *antiveridical* iff $O(p) \Rightarrow \neg p$ is logically valid[3].

In other words, the set of nonveridical functions contains as subset the one formed by antiveridical expressions (AV) which are negation-like operators. We will refer to these two sets as NV and AV, respectively, and use NV and AV to refer to their members. Giannakidou extends Definition 7.2 to other operators, namely to those denoting determiners [Gia99], and propositional verbs [Gia98][4]. The extended definitions are based on a set theoretical interpretation of natural language expressions given in terms of relations. Following them, we now introduce a more general definition of nonveridicality which holds for n-ary functions. The new perspective could shed light on further investigations of nonveridical contexts in natural language.

Notice that since the veridicality property of a function is defined in terms of the truth-values, only boolean functions can be defined to be (non)veridical. Moreover, they can be (non)veridical only in their boolean arguments[5]. This means that, for example, functions of the type (e, t) are not in the class we are interested in since their argument is not boolean.

DEFINITION 7.3. [(Non)veridical functions (II)] Let $(\vec{a}, t)$ stand for a boolean type $(a_1, (\ldots (a_n, t) \ldots))$ where $a_1, \ldots, a_n$ are arbitrary types and $0 \leq n$. Let $f_{(\vec{a},t)}$ be a constant.

1. The expression represented by f is *veridical* in its i-argument, if a_i is a boolean type, *i.e.* $a_i = (\vec{b}, t)$, and $\forall \mathcal{M}, g$

$$[\![f(x_{a_1}, \ldots, x_{a_{i-1}}, x_{(\vec{b},t)}, x_{a_{i+1}}, \ldots, x_{a_n})]\!]^g_{\mathcal{M}} = 1 \text{ entails } [\![\exists \vec{y}_{\vec{b}} .x_{(\vec{b},t)}(\vec{y}_{\vec{b}})]\!]^g_{\mathcal{M}} = 1.$$

   Otherwise f is nonveridical.

2. A nonveridical function represented by $f_{(\vec{a},t)}$ is *antiveridical* in its i-argument, if $a_i = (\vec{b}, t)$ and $\forall \mathcal{M}, g$

$$[\![f(x_{a_1}, \ldots, x_{a_{i-1}}, x_{(\vec{b},t)}, x_{a_{i+1}}, \ldots, x_{a_n})]\!]^g_{\mathcal{M}} = 1 \text{ entails } [\![\neg\exists \vec{y}_{\vec{b}} .x_{(\vec{b},t)}(\vec{y}_{\vec{b}})]\!]^g_{\mathcal{M}} = 1.$$

Notice that the base case of $a_i = t$ is obtained by taking $\vec{y}$ empty.

[3]In the original definition Zwarts used the term *averdical*, which has been replaced with *antiveridical* in [Gia99].

[4]For instance, the definition of nonveridical determiners says that: A determiner is *veridical* wrt its N argument iff it holds that $[\![\text{DET N VP}]\!] = 1 \rightarrow [\![\text{N}]\!] \neq \{\}$; otherwise DET is nonverdical.

[5]The set of boolean type is is built from the set of types TYPE as following: TYPE := ATOM, (TYPE, TYPE) and BTYPE := t, (TYPE, BTYPE) [KF85].

In [Zwa95], it is shown that the concept of nonveridicality is connected to the one of monotonicity. For truth-functional operators it holds that the set of the downward monotone ones is a proper subset of the one formed by nonveridical operators.

FACT 7.4. [Downward Monotonicity entails Nonveridicality (I)] Given a unary or binary truth conditional operator O, if O is downward monotone, then O is a nonveridical operator.

Given our definition of nonveridical functions, we can extend this fact to n-ary boolean functions.

FACT 7.5. [Downward Monotonicity entails Nonveridicality (II)] Given a n-ary function f, if f is downward monotone in its i-argument, then f is nonveridical.

PROOF. The proof goes by contradiction. Assume f is downward monotone and veridical, to prove falsum: For all boolean types $(\vec{b}, t)$, write $\bot_{(\vec{b},t)}$ for the function h such that for all $\mathcal{M}, g$ $[\![\exists\ \vec{y}\ .h(\vec{y})]\!] = 0$. Then observe that (i) if f is downward monotone then for all x $[\![f(x_1, \ldots, x_i, \ldots, x_n)]\!] \leq_t [\![f(x_1, \ldots, \bot, \ldots, x_n)]\!]$; (ii) if f is veridical then $[\![f(x_1, \ldots, \bot, \ldots, x_n)]\!] = 1$ entails $\exists\ \vec{y}\ .\bot(\vec{y}) = 1$. Contradiction. QED

For the analysis of Greek data discussed in the previous section it is relevant to emphase that the set NV contains AV as subset. Moreover, nonveridical contexts can be opaque as well; we refer to this second subset of NV, as NVI. One can prove that antiveridical operators are extensional, hence $\mathsf{AV} \not\subseteq \mathsf{NVI}$[6].

A further study of the inclusion relation among the sets identified by nonveridicality, intensionality and monotonicity could help reach a deeper understanding of negative polarity phenomena, pointing out the precise properties required by the single items and formally combine the analysis of Dutch and Greek data[7]. In Table 7.3, we give some example of nonveridical contexts underlining their monotone and intensional properties. Examples of veridical functions are *and*, the determiner *a* in its both arguments, *without* in its first argument, and *yesterday*. We now look at some example illustrating the definition given above.

An interesting case to look at are determiners. It can be surprising to note that no antiveridical determiners have been found. At first sight a possible candidate could be the determiner *no*. The fact that this is not the case can be illustrated by means of the following example taken from [Gia99].

[6]In [Gia97], Giannakidou gives a more fine-grained definition of nonveridical contexts relativizing them to individuals' models. This definition justify the fact that APIs are excluded from the complements of, for instance, weak intensional verbs like *believe*. The idea is that sentences are not true or false in isolation, but they are true or false with respect to an individual's epistemic state.

[7]Note that as they are defined the sets of antiveridical and antimorphic have a non empty intersection, but are not in a subset relation. For instance $(\neg p \vee \neg q)$ is antimorphic in p (and q) but is not antiveridical in them (it is nonveridical) [Zwa95]. If the definition of antiveridicality is considered as an iff condition than the set of these functions would be included in AM.

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 7.3: (Non)veridical contexts.

EXAMPLE 7.6. [Determiner] *No* denotes a two-argument function $f \in \mathsf{TERM}_{((e,t),((e,t),t)}$. Let us look at the first argument. We want to check whether f is antiveridical in it. Its type is (e,t). Assume f is antiverdical, then by Definition 7.3 $[\![f(x^1_{(e,t)}x^2_{(e,t)})]\!]^g_{\mathcal{M}} = 1$ entails $\neg\exists d \in Dom_e [\![x^1_{(e,t)} z_e]\!]^{g[z/d]}_{\mathcal{M}} = 1$. But this is not true as shown by the fact that a cross sentential anaphoric link can be established in the sentence below:

No students came to the meeting. They preferred to stay at home.

The pronoun *they* can refer to the discourse referent introduced by *no student*. Hence there is a model and an assignment for which $\exists d \in Dom_e [\![\mathtt{student}_{(\mathtt{e,t})}\ z_e]\!]^{g[z/d]}_{\mathcal{M}} = 1$. However, since we cannot entail that such z exists for all models and assignments, the function is nonveridical.

Having assumed the functional perspective in defining (non)veridical contexts has already a first positive effect: it extends the case of linguistic items to be considered. For example, we can consider the case of coordination of quantifier phrases.

EXAMPLE 7.7. [Connectives] The different behavior of the connectives *and* vs. *or* with respect to the veridicality of their arguments is easily checked applying Definition 7.3. The two connective are represented by $\lambda PQR.P\ R\ \wedge\ Q\ R$ and $\lambda PQR.P\ R\ \vee\ Q\ R$, respectively. It is easy to see that *and* is veridical in its both arguments P and Q, whereas *or* is not: $[\![((((\lambda PQR.(P\ R) \wedge (Q\ R))\ x_1)\ x_2)\ x_3)]\!]^g_{\mathcal{M}} = 1$ then $\exists d \in D_{(e,t)} [\![(x_1\ z)]\!]^{g[z/d]}_{\mathcal{M}} = 1$ namely $g(z) = g(x_3)$, and the same holds for Q. Whereas this is not the case for *or* in neither of the two arguments. Let us consider the sentence below.

1. Some student and every professor came to the party.
2. Some student or every professor came to the party.

Assume *some student and every professor came to the party* is true, than *some student came to the party* is true as well. On the other hand, in the case of *or* such conclusion cannot be drawn: the expression *some student or every professor came to the party* is valid also in case no student came to the party but every professor did.

Finally, notice that the veridicality of *and* versus the nonveridicality of *or* predicts the distributional behavior of negative polarity items occuring in one of their constituents [Hoe00]. Recall that these facts are instead problematic for a syntactic approach in terms of c-command relations (see Section 6.4.3). The example in (11) illustrates how the semantic approach instead correctly analyzes the behavior of Greek APIs in the constituent of disjunction[8].

(10) a. **No* student and any professor came to the party.

(11) a. *I* bike kanenas mesa *i* afisame to fos anameno.
or entered.3sg anyone in or left.1pl the light lit
tr. Either somebody broke into the house or we left the light on.

b. Bike kanenas mesa *ke* afisame to fos anameno.
entered.3sg anyone in and left.1pl the light lit
tr. Somebody broke into the house and we left the light on.

7.4 Classifications of Negative Polarity Items 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. Abstracting away from the different classifications of the licensors of Dutch and Greek NPIs, the two analyses can be summarized as below. Recall from Section 5.3 that the relation between a sensitive item and a trigger is accounted for by considering a definition of a composition relation more general than the one expressed by $R_\bullet$ and function application. We defined the relation as $\mathbb{C}([\gamma : \gamma'], [\alpha : \alpha'], [\beta : \beta'])$, which is equivalent to the combination of (i) the syntactic assembly of the forms α and β, and possibly other forms, yielding γ, and (ii) the meaning assembly of the terms α', β', and possibly other terms, yielding γ'. Moreover, we said that in the case of items sensitive to the semantic property of the function they can be in the scope of, the term of the trigger β' has immediate scope over the term α' of the sensitive item. NPIs are in this class of sensitive items.

To express this 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) $^*\mathrm{Li}_2 \circ \Delta\lceil NPI_1\rceil$

[8] In Greek as well as in Italian, FCI can occur in the constituent of disjunction [Gia01], though the letter does not create an opaque context. This is justified by the fact that in those cases the disjunction is modalized (*i.e.* it expresses *as far as* meaning) (p.c.).

Since the polarity item has scope over the structure it occurs in, it determines the type assigned to it (see the examples in Section 7.1.2). 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.

In Section 7.3, we have seen that *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 as just explained. In Chapter 6, we have seen that unary operators give us the right expressivity to account for such distinctions. Following the type assignments obtained in Section 7.1, we can consider *nodoby*: $q(np, s_2', s_2)$, and *a student*: $q(np, s_3, s_1)$. 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* (12-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* (13-a-b).

(12) a. **A student* saw anybody.
b. **A student* said a word.

(13) a. *At most three women* said anything.
b. **At most three women* said a word.

The examples above can be taken to illustrate the comment we made while introducing licensing and antilicensing relations (Remark 5.3). 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 (13-a) it follows that *Nobody said anything* is derivable as well. Schematically, given that $AA : q(np, s'_2, s_2) \longrightarrow DM : q(np, s'_1, s_2)$,

$$\frac{\Delta[q(np, s'_1, s_2)] \vdash C}{\Delta[q(np, s'_2, s_2)] \vdash C.}$$

The cube of derivability relations given in Section 2.4 and repeated in Figure 7.3 offers a much richer hierarchy of types which allow us to make more fine-grained distinctions among negative polarity items. We now move to investigate their applications to the crosslinguistic data presented in Section 7.2.

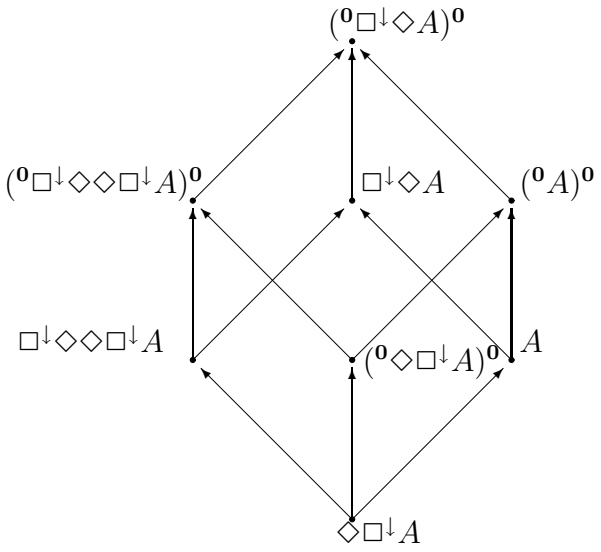


Figure 7.3: Some derivability patterns in $\mathsf{NL}(\Diamond, \cdot^{\mathbf{0}})$.

7.4.1 Types for Dutch Negative Polarity Items

The analysis of Dutch data showed 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).

Example 7.8. [Derived Negative Polarity Items Distribution] We look at *ook maar_n* (tr. any_n) by means of example. Since it is a medium NPI its direct licensor is an antiadditive function (14-a) and therefore it is also grammatical in construction with the antimorphic ones (14-b), whereas it refuses to compose with downward monotone functions (14-c).

(14) a. *Niemand* zal ook maar iets bereiken.
tr. Nobody will achieve anything.

b. Ik denk *niet* dat er ook maar iemand zal komen.
tr. I don't think that anybody will come

c. **Weinig* monniken zullen ook maar iets bereiken.
tr. Few monks will achieve something.

$$\dfrac{\boxed{\mathrm{QP} \vdash q(np, s_x, s_2)} \qquad \dfrac{\dfrac{}{y \vdash np}^{2} \quad \dfrac{}{x \vdash np}^{1}}{\vdots}}{\dfrac{\mathrm{ook\ maar\ iets} \vdash q(np, s'_2, s'_2) \quad y \circ \mathrm{zullen/zal} \circ x \circ \mathrm{bereiken} \vdash s'_2}{y \circ \mathrm{zullen/zal} \circ \mathrm{ook\ maar\ iets} \circ \mathrm{bereiken} \vdash \boxed{s'_2}}\,[q\mathrm{E}]^{1}}{\dfrac{}{\mathrm{zullen/zal} \circ \mathrm{ook\ maar\ iets} \circ \mathrm{bereiken} \vdash \boxed{s_x}}\,[\mathrm{D}^{*}]}}{\dfrac{}{\mathrm{QP} \circ (\mathrm{zullen/zal} \circ \mathrm{ook\ maar\ iets} \circ \mathrm{bereiken}) \vdash s_2}\,[q\mathrm{E}]^{2}}$$

$$\dfrac{\mathrm{niet} \vdash (np\backslash s_2)/(np\backslash s'_3) \qquad \dfrac{\dfrac{}{\mathrm{dat\_er} \circ \mathrm{ook\ maar\_iemand} \circ \mathrm{zal\_komen} \vdash np\backslash s'_1}\,[\mathrm{D}_1]}{\vdots}}{\mathrm{niet} \circ \mathrm{dat\_er} \circ \mathrm{ook\ maar\_iemand} \circ \mathrm{zal\_komen} \vdash np\backslash s_2}\,[/\mathrm{E}]$$

The first derivation fails when replacing the QP with the downward monotone function *weinig_n*: s_x is instantiated by s'_1 which is not derivable from the sentential type carried by the medium negative polarity item *ook maar_n*: $s'_2 : ({}^{\mathbf{0}}s)^{\mathbf{0}} \not\longrightarrow s'_1 : ({}^{\mathbf{0}}\Diamond\Box^{\downarrow}s)^{\mathbf{0}}$. On the other hand, the antiadditive quantifier *niemand* provides the right property required by *ook maar_n*, as reflected on the types: s_x is replaced by s'_2 and the derivation goes through. Similarly, one can derive the other licensing relations discussed while presenting the Dutch data.

Finally, notice that the derivability patterns have room for a type $s'_4 : ({}^{\mathbf{0}}\Box^{\downarrow}\Diamond\Diamond\Box^{\downarrow}s)^{\mathbf{0}}$ derivable from $s'_1 : ({}^{\mathbf{0}}\Diamond\Box^{\downarrow}s)^{\mathbf{0}}$ and deriving the type $s'_3 : ({}^{\mathbf{0}}\Box^{\downarrow}\Diamond s)^{\mathbf{0}}$. In other words, the types could express a subset of DM whose intersection with AA is empty. This

is the set of antimultiplicative functions (Am), like *niet altijd* (tr. not always), which license *even* (tr. equally). As mentioned in Section 5.2, the antimultiplicative functions seem to have a restricted linguistic application. Notice, however, that their behavior is correctly predicted by our types since their licensee can occur also with antimorphic functions (15-a) but not with the antiaddive ones (15-b). Moreover, weak negative polarity items can occur also in their scope (16).

(15) a. We zijn *niet* allemaal even gelukkig met dit voorstel. [AM].
We are not all equally happy with this proposal.
tr. We are not all that happy with this proposal.

b. **Niemand* is even gelukkig met dit voorstel. [AA].
Nobody is equally happy with this proposal.
tr. Nobody is that happy with this proposal

c. Ik heb *niet altijd* even veel geluk in de loterij. [Am].
I have not always equally much luck in the lottery.
tr. I am not always that happy in the lottery.

d. **Weinig* mensen zijn even gelukkig met dit voorstel. [DM].
Few people are equally happy with this proposal.
tr. Few people are that happy with this proposal.

(16) a. Hij hoeft *niet altijd* te roepen. [AM > WNPI].
He needs not always to shout
He doesn't need always to shout.

Summing up, 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 7.4.

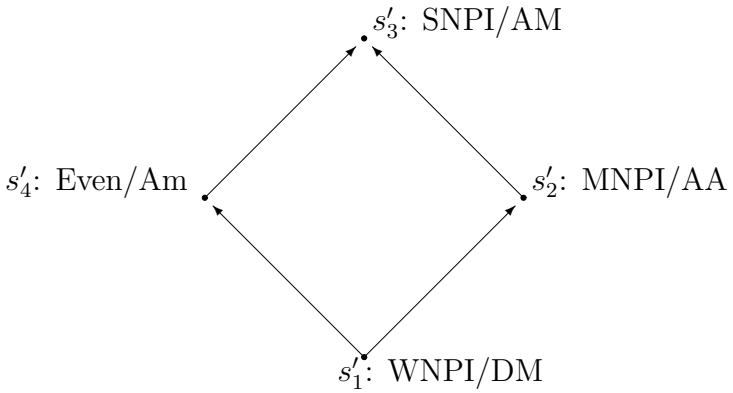


Figure 7.4: Types for Dutch NPIs.

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.

7.4.2 Types for Greek Negative Polarity Items

In contrast to the situation in Dutch, the licensors of negative polarity items in Greek are not in a linear order relation. Recall in fact that 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 7.3). 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)

Example 7.9. [Derived Distribution of Greek NPIs] We look at the different distribution exhibited by the affective polarity item *kanena* (tr. anybody) and the free choice item *opjondhipote* (tr. anybody-FCI). Since the latter requires to be in opaque contexts it is not grammatical in construction with antiveridical operator like *dhen* (tr. not).

(17) a. *Dhen* idha <u>kanenan</u>.
tr. I didn't see anybody.

b. **Dhen* idha <u>opjondhipote</u>.
tr. I didn't see FCI-anybody.

$$\dfrac{\mathrm{Dhen} \vdash s_2/(np\backslash s'_2) \qquad \dfrac{\dfrac{\mathrm{QP} \vdash q(np, s_z, s_y) \qquad \dfrac{[y \vdash np]^2 \quad [x \vdash np]^1}{\vdots}\; y \circ \mathrm{idha} \circ x \vdash s_z}{y \circ \mathrm{idha} \circ \mathrm{QP} \vdash s_y}\,[q\mathrm{E}]^1}{y \circ \mathrm{idha} \circ \mathrm{QP} \vdash s'_2}\,[\mathrm{D}_y]}{\mathrm{idha} \circ \mathrm{QP} \vdash np\backslash s'_2}\,[\backslash\mathrm{I}]}{\mathrm{Dhen} \circ \mathrm{idha} \circ \mathrm{QP} \vdash s_2}\,[/\mathrm{E}]$$

Applying the method that will be familiar by now, one can check that the data in (17) are correctly predicted. In particular, the second derivation fails when replacing the QP with the free choice item *opjondhipote*, whereas it is derivable in the case of *kanenan*.

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].

(18) 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 7.5.

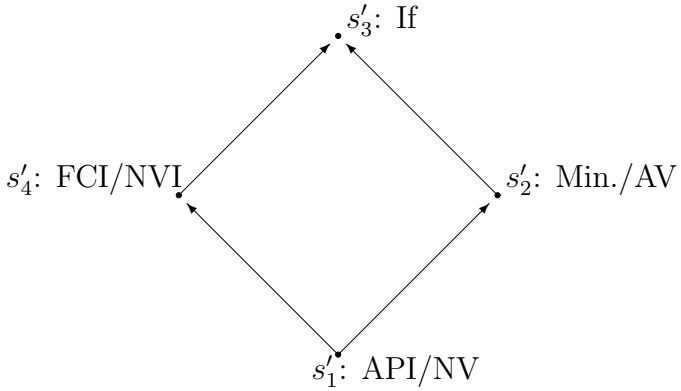


Figure 7.5: Types for Greek NPIs.

7.4.3 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.

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.

(19) 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.

(20) 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.

(21) 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.

(22) 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.

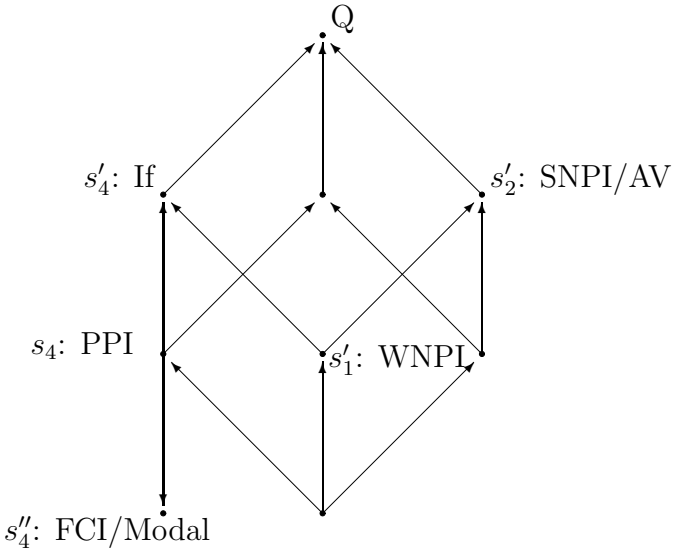
(23) a. **Non* ho visto qualcuno. [*AV > PPI].
Not have seen.sg1 somebody
I didn't see anybody.

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*

7.5 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. In Section 5.3, we have shown how positive information derives from this incompatibility relations. 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 $\{\cdot\}$ stand for a negative contexts (Section 2.3). The distribution of English wh-phrases 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.

7.5.1 Positive Polarity Items in Dutch

As anticipated in Section 5.2, Dutch positive polarity items are in an antilicensing relation with downward monotone functions. The following data from [Wou94] illustrate this statement and Table 7.4 summarizes the PPIs' distribution. The triggers are emphasized, whereas the PPIs are underlined.

(24) 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.

(25) 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.

(26) 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 7.4: Positive polarity items distribution in Dutch.

From Section 7.4, 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 (Remark 5.3). 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.

7.6 Key Concepts

In this chapter we have presented a logical approach to licensing and antilicensing relations holding between a sensitive item and a trigger.

1. The compositional relation is shown to be inherited by expressions in subset or superset relation to the (direct) trigger.

2. 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.

3. The inclusion relation among the sets has been captured in terms of derivability relation among the type assignments achieving a deductive account of licensing relations.

4. The study of licensing and antilicensing relation has been carried out on a crosslinguistic level, looking at Dutch negative and positive polarity items; and at Greek and Italian negative polarity items.

Part IV

Summing up

We started this thesis by observing the intuitive connections between the pair of opposites one finds in arithmetics and the binary operators of the logical grammar used throughout this thesis. Moreover, we have explored the mathematical structure of such a logic and shown that it has room for upward and downward monotonic unary operators as well.

In the second and third part of the thesis, we have investigated the expressivity of the extended logic to model linguistic composition. In particular, we have used the unary operators as 'logical features' to encode distinctions among items associated with the same domain of interpretation. Furthermore, we have exploited the logical properties of the unary operators to account for subset relations holding among these items within their domain. The subset relations have been encoded as derivability relations among the corresponding types. In other words, we have proposed a deductive feature checking mechanism to account for the distribution of quantifier phrases as well as polarity items.

In this last part, we draw some general conclusions, comment on some problems we left open and point out some directions for further research suggested by our studies.

Chapter 8

Conclusions and Further Research

8.1 Conclusions

In this thesis we have assumed a logical perspective on linguistic analysis and developed a 'logical grammar' to reason about linguistic resources. More specifically, our thesis offers a proof theoretical perspective on the tasks of *reasoning* with linguistic signs (Chapter 1).

Our aim has been to investigate the expressivity of the core system of Categorial Type Logics (CTLs), namely the logic characterized by pure algebraic principles with no addition of non-logical axioms. To this end, we explored the mathematical structure of CTLs. In particular, in Chapter 2 we studied the expressivity of a multimodal logical grammar based on the two simple algebraic principles characterizing *residuated* operators and *Galois* connected operators. The idea of using the pure residuated operators to model linguistic phenomena traces back to Lambek [Lam58], who proposed to interpret the functional connectives of Classical Categorial Grammar as logical constants, emphasizing the importance of having both elimination and introduction rules to compose and decompose structures. On the other hand, the idea of using residuated *unary* operators was first introduced by Moortgat and Kurtonina [KM95]. The addition of Galois connections was inspired by the works of Dunn [Dun91], and Goré [Gor98b]. These works show that the algebraic structure of the base logic has room for Galois connected operators which reverse the derivability relation among the formulas. By considering both residuated and Galois connected operators, we focused on the role of the tonicity of the logical operators in the derivability relation among types.

After introducing the multimodal logic framework, in Chapter 3, we distinguished the role played by the binary and unary operators in the task of modelling linguistic phenomena. The linguistic application of the binary residuated operators is widely known (Chapter 1). Their accessibility relation $R_\bullet$ in the Kripke models can be thought of as (syntactic) composition of linguistic signs. The operators $\backslash$ and / build syntactic categories corresponding to functional semantic types. The latter identify the functional domains of interpretation where the corresponding signs denote. Therefore, the logical rules of the binary operators account for the form-meaning assembly of natural language. Syntactic types are systematically associated with the semantic types. This mapping is

not an isomorphism, but a weaker (homomorphic) correspondence: signs of the syntactic type A/B and $A\backslash B$ have the same semantic type and hence are interpreted in the same domain. The directionality information carried by the $\backslash, /$ connectives is relevant for the syntactic assembly of expressions; at the meaning level, it is enough to know that they are functions. More generally, signs of the same semantic type can differ in their syntactic distribution. The work presented in this thesis focuses on this aspect of natural language.

We looked at some semantic differences which effect syntactic composition, but which are no longer relevant in the meaning assembly. Moreover, we zoomed in on the domains of interpretation and highlighted subset relations holding among sets of objects enjoying properties of different strength. We accounted for these difference by means of unary operators used as 'logical features'. Finally, we exploited the derivability of the composition of unary operators to capture the subset relations among members of the same semantic type. The final logical grammar consists of a logical mechanism to merge linguistic forms while building their meanings, and a deductive feature-checking mechanism which controls the merge operation and expresses the fine grained distinctions required by linguistic composition. We implemented this general idea to tackle different tasks and account for several linguistic phenomena.

In Chapter 4, we considered the differences within functional domains between upward and downward monotone functions. The role of monotonicity in natural language is twofold: it effects both the reasoning and the parsing process. On the one hand, the monotonicity properties of the constituents of a linguistic structure determine the inferences one can draw from it. On the other hand, they influence the syntactic distribution of negative polarity items. Summing up, we exploited the logical properties of the residuated pair of operators to develop a (fragment of a) natural logic: a system which uses parsed linguistic structures to draw inference [Ben86]. In particular, we took advantage of the modular architecture of the categorial type logical framework: the logical language propagates the monotonicity information from the functions to the arguments, and the structural language stores and computes the polarity positions relevant to account for natural reasoning. Moreover, the composition of the unary operators models the behavior of negative polarity items by marking the structure the negative polarity item occurs in, and by requiring to compose it with a downward monotone function.

In Chapter 6, we investigated the distributional behavior of quantifier phrases. In particular, we discussed their classification as proposed by Beghelli and Stowell [BS97]. We used the composition of unary operators to identify different sentential levels where the quantifiers can or cannot take scope. The logical perspective unveils new classes of quantifiers which were not considered in the original classifications. Moreover, we compared our categorial type logical analysis of quantifier scope distribution with the one proposed within the minimalist framework and pointed out some problems of the latter. Finally, by looking at the comparison between the two frameworks, we proposed some refinements of the minimalist analysis which overcome its limitations.

In Chapter 7, we enlarged the scale of the monotone function classification and identified other subsets. Following Zwarts [Zwa83], we identified the sets of antimorphic and antiadditive functions which are in a subset relation with the set of downward monotone functions. In [Wou94], this subset relation is shown to be relevant for the distribution of

negative polarity items in Dutch. Similarly, the identification of the sets of antiveridical expressions as a subset of the nonveridical ones gives the distinction required to account for the distributional behavior of Greek negative polarity items [Zwa95, Gia97]. We assumed a logical perspective on these linguistic analyses and proposed a categorial type logical account of the licensing relation holding between a negative polarity item and its triggers.

The logical perspective enabled us to discover new dependencies between linguistic phenomena. In particular, it helped carry out crosslinguistic comparisons and illustrate the possibilities of identifying linguistic typologies based on the licensing relation between the sensitive items and their triggers. Finally, it helped define an antilicensing relation holding between an item and a semantic property, where the former is 'allergic' to the latter. The definition of the licensing and antilicensing relations and their categorial type logical account showed the correspondence between the behavior of negative and positive polarity items in Dutch where the latter mirror the former. Moreover, they highlighted the similarity between the relation holding between positive polarity items and downward monotonicity, on the one hand, and wh-phrases and the properties of the scope elements forming weak-islands, on the other hand.

Briefly, the licensing and antilicensing relations hold between an item and a semantic property. Thus, the sensitive item can or cannot compose with all those expressions having the property they require or they repel, respectively. The definition we gave of antilicensing relation entails some positive information regarding the expressions the sensitive item can compose with, namely all those expressions which do not have the property the item repels. In particular, the sensitive item can compose with an expression of any weaker property than the one it is allergic to. Thus the composition of items in an antilicensing relation mirrors the one of the items in a licensing relation with certain triggers. The categorial modelling of these mirror-effects requires the reversal of the order relations of the type assigned to the triggers. In other words, it requires the use of the Galois connected operators.

8.2 Further Research

This thesis provides further evidence for the need of using unary modalities in a categorial account of linguistic phenomena. Besides answering questions, it also raises new ones.

First of all, we have used the unary operators purely as syntactic control devices: modally decorated types are assigned the same interpretation domains as their undecorated versions. We could have set up the syntactic-semantics interpretation in a different way by differentiating the domains of interpretation of the items decorated with the unary operators and by looking for a proper interpretation of the corresponding enriched lambda terms. It is plausible to think that an answer to this question could be found by looking for different interpretation of the functional implications, and by bringing in the issue of directionality for the unary residuated pair as well, namely by considering also the second pair of residuated operators.

A similar question can be raised about the Galois connected operators. Moreover, for these operators no study has been carried out yet about their Curry-Howard in-

terpretation and their logical rules in natural deduction format. Finally, the linguistic applications we have looked at seem to suggest that there are opportunities for interaction between the binary residuated operators and the Galois unary connections. Thus, it would be interesting to study the structural properties which could be added to the system, while preserving its soundness and completeness with respect to Kripke models with the appropriate frame constraints.

In addition to these logically oriented questions, the work presented in this thesis raises also some questions of a more linguistic nature. First of all, the crosslinguistic study of negative polarity items sheds light on the possibility of identifying typologies of languages characterized by different licensing relations. It could be interesting to check whether there are any reasons for the different interactions of syntax and semantics exhibited across languages. We believe that the categorial grammar approach proposed here can help answering this question. Moreover, the antilicensing analysis, though is intuitively correct and formally defined, has not been implemented in its full details. The answers to the logical questions raised above about the Galois connections may help find a solution to this problem. Finally, the categorial type logical analysis of quantifier scope and the one of the antilicensing relation should be integrated. In particular, an interesting point of interaction is provided by wh-phrases which are included in the classification of quantifier phrases we looked at, and can be considered to be in an antilicensing relation with the scope elements forming weak islands of different strength.

Appendix A

Appendix

In Chapter 4 we have presented a natural logic based on NL($\Diamond$) extended with polarity structural rules. The described system has been obtained by internalizing the monotonicity and polarity algorithm developed in [Ben86, SV91]. In other words, by replacing LP+EPol with NL($\Diamond$) and polarity structural rules (henc. MCTL+Pol). In this appendix, we shed light on the differences between the two systems by embedding LP+EPol into an extended version of MCTL+Pol. This translation might help the reader familiar with LP+EPol to gain a better understanding of MCTL+Pol.

We start in Section A.1 by formally introducing the systems, and analyzing their similarities and differences. We will prove that LP+EPol can be embedded into an extended version of MCTL+Pol (EMCTL+Pol) by means of an intermediate system which we introduce in Section A.2[1].

A.1 Introducing the Frameworks

We briefly introduce the system presented in [SV91] to obtain polarity markings on parsed strings, and the extended version of the system MCTL+Pol described in Chapter 4.

A.1.1 The System LP+EPol

The approach proposed by Sánchez separates the logical derivation of a type for a sentence, from the assignment of polarity markers to the different parts. Types are assigned by using LP, Lambek calculus with permutation [Ben88]. The logical system is very simple, with just the Modus Ponens [MP] rule for functional application and an Abstraction Rule [Ab] for functional abstraction. Polarity is dealt with extra-logically, by means of an algorithm that when runs over a proof of LP, first computes monotonicity values for nodes in the proof tree, and then transforms them into polarity markers.

Definition A.1. [The System LP] Let ATOM be the set $\{e, t, p\}$. Define the well formed formulas of LP as

[1]The work presented in this appendix has been carried out in collaboration with Carlos Areces.

$$\mathsf{FORM} := \mathsf{ATOM} \mid (\mathsf{FORM} \rightarrow \mathsf{FORM}).$$

ATOM is chosen to represent the atomic linguistic categories: e for the category of noun phrases, p for nouns and t for sentences. Complex categories are built recursively by the function operator $\rightarrow$. The inference rules of LP are

$$\frac{\alpha \rightarrow \beta \quad \alpha}{\beta}\ [\text{MP}] \qquad\qquad \frac{\begin{array}{c}[\alpha]\\ \vdots \\ \beta\end{array}}{\alpha \rightarrow \beta}\ [\text{Ab}].$$

The formulas in the leaves of a proof tree are called the *assumptions* of the proof. Those assumptions marked by the [Ab] rule are said to be *discharged*, those assumptions which have not been discharged are called *open*. An important condition is that an assumption can only be discharged once. Given a proof tree T, we say that a node α in T is *closed* if the subproof ending in α has no open assumptions.

Let $\Gamma \cup B \cup \{\alpha\} \subseteq \mathsf{FORM}$, then $\Gamma \vdash \alpha$ iff there is a proof of LP ending in α with $\Gamma \cup B$ as set of assumptions and all formulas in B and only those have been discharged.

This logical formalism needs to be tied up to natural language expressions. To do so, a well formed formula of LP (its type) is assigned to each word in the lexicon.

Definition A.2. Let $w_1 w_2 \cdots w_n$ be a non empty sequence of words from the lexicon, and let $T = \{w_1 \in \alpha_1, \cdots, w_n \in \alpha_n\}$ be an assignment of types (i.e. $\{\alpha_1, \cdots, \alpha_n\} \subseteq \mathsf{FORM}$) and let $A = \{\alpha_1, \cdots, \alpha_n\}$. We say that $w_1 w_2 \cdots w_n$ *has type* α iff $\{\alpha_1, \cdots, \alpha_n\} \vdash \alpha$. Furthermore, any proof D of $\{\alpha_1, \cdots, \alpha_n\} \vdash \alpha$ is called an *analysis* of $w_1 w_2 \cdots w_n$. A node in an analysis of $w_1 w_2 \cdots w_n$, is *analysis closed* if its set of open assumptions is a subset of A.

To find the polarity markers of a sentence Sánchez' algorithm works over an analysis of the sequence of words. Each different analysis represents a possible reading of the sentence, and polarity is determined on their account. Recall from Chapter 4 that in LP+EPol hypothesis are left unmarked. This is not relevant for the porpouse of this appendix, therefore in order to simplify the embedding we consider them as upward monotone functions and mark their logical types accordingly. The traslation we give could be easily extended to conver the case of unmarked hypothesis.

Monotonicity Markers: Let D be an analysis of a sentence $w_1 \ldots w_n$ with $T = \{w_1 \in \alpha_1, \cdots, w_n \in \alpha_n\}$ the type of $w_1 \ldots w_n$. Let B be the set of discharged assumptions used in D and $A = \{\alpha_1, \cdots \alpha_n\}$. The set A^m is obtained by decorating functional types in A with $\{+, -\}$, according to their monotonicity behavior, while B^m is obtained marking all functional types as denoting upward monotone functions.

Starting from D', which is D with B^m and A^m instead of the original leaves, we obtain D^m by means of the following rewriting rules:

$$\frac{\alpha^x \to \beta \quad \alpha}{\beta}\,[\text{MP}] \quad \text{rewrites to} \quad \frac{\overset{\alpha^x \to \beta}{+} \qquad \overset{\alpha}{x}}{\beta}\,[\text{MP}] \quad \text{where } x \in \{+,-\}, \text{ and}$$

$$\frac{\begin{array}{c}[\alpha]\\ \vdots \\ \beta\end{array}}{\alpha \to \beta}\,[\text{Ab}] \quad \text{rewrites to} \quad \frac{\begin{array}{c}[\alpha]\\ \vdots \\ \overset{\beta}{+}\end{array}}{\alpha^y \to \beta}\,[\text{Ab}]$$

where the value of y is determined as follows. Let [MP$^-$] be an application of [MP] with the minor premise marked with a $-$, then y is $-$ if the number of [MP$^-$] in the branch from β to α is odd, and y is $+$ otherwise. As a last step, the monotonicity algorithm marks the root of the derivation with a $+$.

Polarity Markers: By concentrating on monotone functions by construction all nodes in D^m are marked as $+$ or $-$. For analysis closed nodes in D^m we define polarity as follows: the node is $-$ if the number of nodes marked with $-$ in the path from the node to the root is odd, and $+$ otherwise[2].

We now illustrate the algorithms by repeating the examples discussed in Chapter 4, but using the above notation.

Example A.3. Let `Not every logician wanders` be the sequence of words of the lexicon we want to parse. Let $B = \{\text{not} \in t \to t, \text{every} \in p \to ((e \to t) \to t), \text{good\_logician} \in p, \text{wanders} \in e \to t\}$. We obtain the following analysis proving that the string is of type t.

$$\dfrac{\text{not} \in t \to t \qquad \dfrac{\dfrac{\text{every} \in p \to ((e \to t) \to t) \quad \text{good\_logician} \in p}{(e \to t) \to t}\,[\text{MP}] \qquad \text{wanders} \in e \to t}{t}\,[\text{MP}]}{t}\,[\text{MP}]$$

Following the algorithm we obtain:

Monotonicity markers

$$\dfrac{\dfrac{\overset{p^- \to ((e \to t)^+ \to t)}{+} \qquad \overset{p}{-}}{\overset{(e \to t)^+ \to t}{+}} \qquad \overset{e \to t}{+}}{\dfrac{\overset{t^- \to t}{+} \qquad \overset{t}{-}}{t}}$$

[2]Note that in the original presentation the polarity marker algorithm included the constraint of having all the nodes in the branch marked. However, since we also mark the hypotheses all derivations have all nodes marked.

Polarity markers

$$\dfrac{\dfrac{\dfrac{p \to ((e \to t) \to t) \quad p}{(e \to t) \to t} \qquad e \to t}{t}}{t}$$

Summing up the parsed string will be polarity marked as: $(\text{not}^{+}((\text{every}^{-}\ \text{good\_logician}^{+})^{-}\ \text{wanders}^{-})^{-})^{+}$.

As an example of how the algorithm treats an [Ab] rule we look at the lifting of e to $(e \to t) \to t$.

EXAMPLE A.4. Let `mary` be our lexicon entry. Let $A = \{\text{mary} \in e\}$ and let the $B = \{\texttt{P} \in e \to t\}$ be the set of discharged assumptions. We obtain the following analysis proving that the string is of type $(e \to t) \to t$.

$$\dfrac{\dfrac{\texttt{mary} \in e \quad [\texttt{P} \in e \to t]^1}{t}\,[\text{MP}]}{(e \to t) \to t}\,[\text{Ab}]^1$$

Following the algorithm:

Monotonicity markers

$$\dfrac{\dfrac{e \quad [e^{+} \to t]^1}{t}}{(e \to t)^{+} \to t}$$

Polarity markers

$$\dfrac{\dfrac{e \quad [e \to t]^1}{t}}{(e \to t) \to t}$$

A.1.2 Extended MCTL+Pol

In order to mimic the monotonicity and polarity marking of LP+EPol into a CTL, we need an extended version of the system MCTL+Pol described in Chapter 4.

First of all, note that since LP is the associative and commutative version of NL, the embedding of LP+EPol requires to take these two properties into consideration. We will do it by adding the corresponding structural rules. Moreover, in order to mimic the polarity marking algorithm, all the strucutures at each node in the derivation must be marked. This is done by heading with $\boxplus^{\downarrow}$ all the atomic and functional logical types. This generates information which is superflus for the final goal of building a natural logic and accounting for negative polarity items distribution, but it is relevant for obtaining a

proper embedding of LP+EPol. Similarly, [/I] and [Pol_] are modified so to add $\langle\cdot\rangle^+$ in their conclusion, we refer to the modified versions as [/I$^+$] and [Pol$^+_-$]. Finally, the $\langle\cdot\rangle^+$ surrounding the conclusion of [Pol$^+_-$] requires the application of [Esc] rule which enables the application of [/I].

With this comment in mind we can now introduce the extended version of MCTL+Pol on which we can map the algorithms described above for marking LP derivations.

Definition A.5. [Logical and Structural Languages] Given $\mathsf{ATOM} = \{np, n, s\}$, the well formed formulas of EMCTL+Pol are

$$\mathsf{FORM} \;:=\; \mathsf{ATOM} \mid \mathsf{FORM}/\mathsf{FORM} \mid \diamondsuit \mathsf{FORM} \mid \blacklozenge \mathsf{FORM} \mid \boxminus^{\downarrow}\mathsf{FORM} \mid \boxplus^{\downarrow}\mathsf{FORM}.$$

$$\mathsf{STRUCT} := \mathsf{FORM} \mid \langle \mathsf{STRUCT} \rangle^- \mid \langle \mathsf{STRUCT} \rangle^+ \mid \mathsf{STRUCT} \circ \mathsf{STRUCT}.$$

The calculus is presented as a set of sequence rules, where a sequent is a pair $(\Gamma, A) \in \mathsf{STRUCT} \times \mathsf{FORM}$, which we will note as $\Gamma \vdash A$. The rules of the calculus are given in Figure A.1.

Again we illustrate EMCTL+Pol by repeating the examples discussed in Chapter 4.

Example A.6. For the sake of simplicity we use linguistic items in the place of the corresponding structural formulas.

$$\dfrac{\dfrac{[Q \vdash \blacklozenge \boxplus^{\downarrow}(\boxplus^{\downarrow}s/ \blacklozenge \boxplus^{\downarrow}np)]^2 \qquad \dfrac{\dfrac{[P \vdash \boxplus^{\downarrow}(\boxplus^{\downarrow}s/ \blacklozenge \boxplus^{\downarrow}np)]^1}{\langle P\rangle^+ \vdash \boxplus^{\downarrow}s/ \blacklozenge \boxplus^{\downarrow}np}[\boxplus^{\downarrow}\mathrm{E}] \quad \dfrac{\mathrm{mary} \vdash \boxplus^{\downarrow}np}{\langle \mathrm{mary}\rangle^+ \vdash \blacklozenge \boxplus^{\downarrow}np}[\blacklozenge \mathrm{I}]}[/\mathrm{E}]}{\langle P\rangle^+ \circ \langle \mathrm{mary}\rangle^+ \vdash \boxplus^{\downarrow}s}}{Q \circ \langle \mathrm{mary}\rangle^+ \vdash \boxplus^{\downarrow}s}[\blacklozenge \mathrm{E}]^1}{\langle\langle \mathrm{mary}\rangle^+\rangle^+ \vdash \boxplus^{\downarrow}s/ \blacklozenge \boxplus^{\downarrow}(\boxplus^{\downarrow}s/ \blacklozenge \boxplus^{\downarrow}np)}[/\mathrm{I}^+]^2}{\langle \mathrm{mary}\rangle^+ \vdash \boxplus^{\downarrow}(\boxplus^{\downarrow}s/ \blacklozenge \boxplus^{\downarrow}(\boxplus^{\downarrow}s/ \blacklozenge \boxplus^{\downarrow}np))}[\boxplus^{\downarrow}\mathrm{E}]$$

Example A.7.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathrm{every} \vdash \boxplus^{\downarrow}(\boxplus^{\downarrow}(\boxplus^{\downarrow}s/ \blacklozenge \boxplus^{\downarrow}(\boxplus^{\downarrow}s/ \blacklozenge \boxplus^{\downarrow}np))/ \diamondsuit \boxplus^{\downarrow}n)}{\langle \mathrm{every}\rangle^+ \vdash \boxplus^{\downarrow}(\boxplus^{\downarrow}s/ \blacklozenge \boxplus^{\downarrow}(\boxplus^{\downarrow}s/ \blacklozenge \boxplus^{\downarrow}np))/ \diamondsuit \boxplus^{\downarrow}n}[\boxplus^{\downarrow}\mathrm{E}] \quad \dfrac{\mathrm{good\_logician} \vdash \boxplus^{\downarrow}n}{\langle \mathrm{good\_logician}\rangle^- \vdash \diamondsuit \boxplus^{\downarrow}n}[\diamondsuit \mathrm{I}]}[/\mathrm{E}]}{\langle \mathrm{every}\rangle^+ \circ \langle \mathrm{good\_logician}\rangle^- \vdash \boxplus^{\downarrow}(\boxplus^{\downarrow}s/ \blacklozenge \boxplus^{\downarrow}(\boxplus^{\downarrow}s/ \blacklozenge \boxplus^{\downarrow}np))}[\boxplus^{\downarrow}\mathrm{E}] \quad \dfrac{\mathrm{wanders} \vdash \boxplus^{\downarrow}(\boxplus^{\downarrow}s/ \blacklozenge \boxplus^{\downarrow}np)}{\langle \mathrm{wanders}\rangle^+ \vdash \blacklozenge \boxplus^{\downarrow}(\boxplus^{\downarrow}s/ \blacklozenge \boxplus^{\downarrow}np)}[\blacklozenge \mathrm{I}]}{\langle\langle \mathrm{every}\rangle^+ \circ \langle \mathrm{good\_logician}\rangle^-\rangle^+ \vdash \boxplus^{\downarrow}s/ \blacklozenge \boxplus^{\downarrow}(\boxplus^{\downarrow}s/ \blacklozenge \boxplus^{\downarrow}np)}[/\mathrm{E}]}{\langle \mathrm{every}\rangle^+ \circ \langle \mathrm{good\_logician}\rangle^- \rangle^+ \circ \langle \mathrm{wanders}\rangle^+ \vdash \boxplus^{\downarrow}s}$$

$$\dfrac{\dfrac{\mathrm{not} \vdash \boxplus^{\downarrow}(\boxplus^{\downarrow}s/ \diamondsuit \boxplus^{\downarrow}s)}{\langle \mathrm{not}\rangle^+ \vdash \boxplus^{\downarrow}s/ \diamondsuit \boxplus^{\downarrow}s}[/\mathrm{E}] \qquad \dfrac{\langle\langle \mathrm{every}\rangle^+ \circ \langle \mathrm{good\_logician}\rangle^-\rangle^+ \circ \langle \mathrm{wanders}\rangle^+ \vdash \boxplus^{\downarrow}s}{\langle\langle\langle \mathrm{every}\rangle^+ \circ \langle \mathrm{good\_logician}\rangle^-\rangle^+ \circ \langle \mathrm{wanders}\rangle^+\rangle^- \vdash \diamondsuit \boxplus^{\downarrow}s}[\diamondsuit \mathrm{I}]}{\langle \mathrm{not}\rangle^+ \circ \langle\langle\langle \mathrm{every}\rangle^+ \circ \langle \mathrm{good\_logician}\rangle^-\rangle^+ \circ \langle \mathrm{wanders}\rangle^+\rangle^- \vdash \boxplus^{\downarrow}s}[/\mathrm{E}]$$

A.2 Bridging the Gap

In order to build a bridge between the two systems we will proceed in two steps. We first define a new system LP+SPol which remains close to LP+EPol but internalizes the meta-logical markers used by Sánchez. We then prove that the polarity markers obtained in the new system coincide with those computed by Sánchez. Finally, we define a functional embedding of LP+SPol into EMCTL+Pol.

Logical Rules for the Binary Operators	*Structural Rules*
$\dfrac{\Gamma \vdash A/B \quad \Delta \vdash B}{\Gamma \circ \Delta \vdash A}\ [/\mathrm{E}]$ $\dfrac{\begin{array}{c}[B \vdash B]^i \\ \vdots \\ \Gamma \circ B \vdash A\end{array}}{\langle\Gamma\rangle^+ \vdash A/B}\ [/\mathrm{I}^+]^i$	$\dfrac{\Gamma[\Delta_1 \circ (\Delta_2 \circ \Delta_3)] \vdash C}{\Gamma[(\Delta_1 \circ \Delta_2) \circ \Delta_3] \vdash C}\ [\mathrm{Ass}]$ $\dfrac{\Gamma[\Delta_2 \circ \Delta_1] \vdash C}{\Gamma[\Delta_1 \circ \Delta_2] \vdash C}\ [\mathrm{Perm}]$
Logical Rules for the Unary Operators	*Structural Polarity Rules*
$\dfrac{\Delta \vdash \diamondsuit A \quad \Gamma[\langle A\rangle^s] \vdash B}{\Gamma[\Delta] \vdash B}\ [\diamondsuit \mathrm{E}]^\dagger$ $\quad \dfrac{\Gamma \vdash A}{\langle\Gamma\rangle^s \vdash \diamondsuit A}\ [\diamondsuit \mathrm{I}]^\dagger$ $\dfrac{\Gamma \vdash \boxdot^{\downarrow}_s A}{\langle\Gamma\rangle^s \vdash A}\ [\boxdot^{\downarrow}_s \mathrm{E}]^\dagger$ $\quad \dfrac{\langle\Gamma\rangle^s \vdash A}{\Gamma \vdash \boxdot^{\downarrow}_s A}\ [\boxdot^{\downarrow}_s \mathrm{I}]^\dagger$ †*where* $s \in \{+,-\}$.	$\dfrac{\Gamma[\langle\Delta_1 \circ \Delta_2\rangle^s] \vdash C}{\Gamma[\langle\langle\Delta_1\rangle^s \circ \langle\Delta_2\rangle^s\rangle^+] \vdash C}\ [\mathrm{Pol}^+_s]^\dagger$ $\dfrac{\Gamma[\langle\langle\Delta\rangle^{s_1}\rangle^{s_2}] \vdash C}{\Gamma[\langle\Delta\rangle^{sg(s_1,s_2)}] \vdash C}\ [\mathrm{Pol}_{s_1 s_2}]^\dagger$ $\dfrac{\Gamma[\langle\Delta_1 \circ \langle\Delta_2\rangle^s\rangle^+] \vdash C}{\Gamma[\langle\Delta_1\rangle^+ \circ \langle\Delta_2\rangle^s] \vdash C}\ [\mathrm{Esc}]$ †*where* $s, s_1, s_2 \in \{+,-\}$, *and* $sg(s_1,s_2) = +$ *if* $s_1 = s_2$, $sg(s_1,s_2) = -$ *otherwise.*

Figure A.1: Rules of EMCTL+Pol.

A.2.1 Internalizing Markers

In LP+SPol we provide an explicit encoding of the monotonicity and polarity marking algorithms used in LP+EPol. Also, looking forward to the embedding into EMCTL+Pol, we already take the first step to achieve a collapsing of the monotonicity and polarity marking algorithms. We do so by using both a logical language and a structural language in the rewriting algorithm.

Definition A.8. [The Languages of LP+SPol] Given $\mathsf{ATOM} = \{e, t, p\}$, define the well formed (marked) formulas of LP+SPol as

$$\mathsf{MFORM} := \mathsf{ATOM} \mid (\mathsf{MFORM})^+ \to \mathsf{MFORM} \mid (\mathsf{MFORM})^- \to \mathsf{MFORM}.$$

The structural language of LP+SPol is generated by the following grammar built over a set VAR of auxiliary variables and the set LEX of lexical entries

$$\mathsf{MSTRUC} := s(\mathsf{VAR}) \mid s(\mathsf{LEX}) \mid s(\mathsf{MSTRUC}) \mid s(\{\mathsf{MSTRUC}, \mathsf{MSTRUC}\}),$$

where $s \in \{+,-\}$.

We are now ready to define explicit algorithms that mimic over MSTRUC the external monotonicity and polarity algorithms of LP+EPol. We define two recursive functions *Coll* (for collapse) and *Comp* (for compute) which will produce polarity marked structures. *Coll* will be used on the premise of the [Ab] rule, before discharging a hypothesis, to collapse all the markers around it; whereas *Comp* will take the final marked structure of a proof as input and compute the markers for the single substructure of it.

Definition A.9. [*Coll* and *Comp* Functions] $Coll : \mathsf{MSTRUC} \times \mathsf{VAR} \to \mathsf{MSTRUC}$ and $Comp : \mathsf{MSTRUC} \to \mathsf{MSTRUC}$ are recursive functions defined as follows. Let $s_i \in \{+, -\}$ and sg be the sign function,

$$
\begin{array}{rcl}
Coll(s_1(\{\!|s_2(A), s_3(B)|\!\}), x) & = & \textbf{if } x \in s_2(A) \ \& \ x \in s_3(B) \\
& & \textbf{then } +(\{\!|Coll(sg(s_1, s_2)(A), x), Coll(sg(s_1, s_3)(B), x)|\!\}) \\
& & \textbf{elseif } x \in s_2(A) \ \& \ x \notin s_3(B) \\
& & \textbf{then } +(\{\!|Coll(sg(s_1, s_2)(A), x), s_1(s_3(B))|\!\}) \\
& & \textbf{elseif } x \notin s_2(A) \ \& \ x \in s_3(B) \\
& & \textbf{then } +(\{\!|s_1(s_2(A)), Coll(sg(s_1, s_3)(B), x)|\!\}) \\
& & \textbf{else } s_1(\{\!|s_2(A), s_3(B)|\!\}); \\
Coll(s_1(s_2(A)), x) & = & \textbf{if } x \in s_1(s_2(A)) \\
& & \textbf{then } Coll(sg(s_1, s_2)(A), x) \\
& & \textbf{else } s_1(s_2(A)); \\
Coll(s_1(A), x) & = & s_1(A), \text{ for } A \in \mathsf{VAR} \cup \mathsf{LEX}.
\end{array}
$$

$$
\begin{array}{rcl}
Comp(s_1(\{\!|s_2(A), s_3(B)|\!\})) & = & s_1(\{\!|Comp(sg(s_1, s_2)(A)), Comp(sg(s_1, s_3)(B))|\!\}); \\
Comp(s_1(s_2(A))) & = & Comp(sg(s_1, s_2)(A)); \\
Comp(s_1(A)) & = & s_1(A), \text{ for } A \in \mathsf{VAR} \cup \mathsf{LEX}.
\end{array}
$$

Some remarks concerning the *Coll* and *Comp* functions are in order. First, notice that neither of the two functions modifies the bracketing $\{\!|\cdot, \cdot|\!\}$ in a structure, only the $+$ and $-$ signs are altered. The role of *Coll* and *Comp* is similar to the role of the [Pol] rules in EMCTL+Pol, and to the counting of [MP$^-$] rules in a branch of a proof in LP+EPol. The main difference between *Coll* and *Comp* is that the first computes markers only in the structures surrounding a given variable x. Because *Coll* can be applied many times to the same structure, we need to transform signs into $+$ once the value has been computed. *Comp* instead will be applied only once, to the final structure at the root of the proof, and hence we can retain the information concerning polarity of the external structures.

Rewriting Algorithm: Let D be an analysis of a sentence $w_1 \dots w_n$ with $T = \{w_1 \in \alpha_1, \cdots, w_n \in \alpha_n\}$ are assignment of types to $w_1 \dots w_n$. Let B be the set of discharged assumptions used in D and $A = \{\alpha_1, \cdots, \alpha_n\}$. The sets B^m and A^m are obtained by decorating functional types in B and A with $\{+, -\}$, according to their monotonicity behavior. Starting from D', which is D with B^m and A^m instead of the original leaves, we obtain D^m by means of the following rewriting rules. For a leaf α_i, rewrite it as $+(S_i) \vdash \alpha_i$ if the leaf corresponds to one of the types in A, and $+(x) \vdash \alpha_i$ otherwise, where we choose a new variable x for each leaf.

$$\frac{S_1 \vdash \alpha^s \to \beta \quad S_2 \vdash \alpha}{\beta}\ [\text{MP}] \quad \text{rewrites to}$$

$$\frac{S_1 \vdash \alpha^s \to \beta \quad S_2 \vdash \alpha}{+(\{+(S_1), s(S_2)\}) \vdash \beta}\ [\text{MP}] \quad \text{where } s \in \{+,-\},$$

$$\frac{\begin{array}{c}[x \vdash \alpha] \\ \vdots \\ S_1[s_1(\{s_2(S_2), s_3(x)\})] \vdash \beta\end{array}}{\alpha \to \beta}\ [\text{Ab}] \quad \text{rewrites to}$$

$$\frac{\begin{array}{c}[x \vdash \alpha] \\ \vdots \\ \mathit{Coll}(+(S_1), x)[+(\{s'_2(S_2), s'_3(x)\})] \vdash \beta\end{array}}{\mathit{Coll}(+(S_1), x)[s'_2(S_2)] \vdash \alpha^{s'_3} \to \beta}\ [\text{Ab}]$$

where s'_2, s'_3 are the markers assigned to S_2 and x after the application of *Coll* to $(+(S_1), x)$. Finally, apply *Comp* to the final structure of the root of the proof.

Again, some comments will help understand how the rewriting algorithm works. Notice that the [Ab] rule only withdraws the assumption which was recorded in the structure S_1 by the variable x associated to α. Furthermore, previous rewritings by the *Coll* do not alter this information, as *Coll* only modifies + and − signs. Finally, abstraction can be performed only once on a leaf. Hence we can uniquely identify the substructure $s_1(\{s_2(S_2), s_3(x)\})$ containing x in S_1. Applying *Coll* on the pair $(+(S_1), x)$ will compute new signs for S_2 and x, and change the s_1 sign to a +. Again, by properties of *Coll* we can uniquely identify the new substructure $+(\{s'_2(S_2), s'_3(x)\})$ surrounding x in $\mathit{Coll}(+(S_1), x)$ and copy the sign assigned to x into the logical type. Finally, we eliminate x (and perform in this way the abstraction in the structure) by replacing $+(\{s'_2(S_2), s'_3(x)\})$ in $\mathit{Coll}(+(S_1), x)$ by $s'_2(S_2)$.

In the proof of Theorem A.12 we will need the following simple but important properties of the rewriting algorithm.

Proposition A.10.

i. Let β be an analysis closed node in a proof of $\Gamma \vdash \alpha$ in LP with L its set of undischarged assumptions. Then the structure assigned to β by the rewriting algorithm does not contain variables and is built over L without repetitions.

ii. Let S be any substructure of the structures assigned to the premises of a rewritten [MP] rule, and let S' be the structure assigned to the conclusion of [MP]. Then S is a substructure of S'.

iii. Let S be any substructure of the structure assigned to the premise of a rewritten [Ab] rule not containing variables, and let S' be the structure assigned to the conclusion of [Ab]. Then S is a substructure of S'.

The new rewriting algorithm above computes polarity markers by means of a recursive function in a way which is similar to the algorithm used in MCTL+Pol. The step we should take now is to prove that in the relevant cases, *viz.* the closed nodes, the polarity values obtained in LP+SPol are the same as those computed in LP+EPol. To start with, we prove that the marked logical types assigned by LP+EPol and LP+SPol coincide.

PROPOSITION A.11. Let D be a proof in LP ending in α, with set of assumptions A (including discharged assumptions). And let D_1^m ending in α_1, and D_2^m ending in $S \vdash \alpha_2$ be the marked proofs obtained by running the monotonicity marking algorithm of LP+EPol and the rewriting algorithm of LP+SPol on D with marked assumptions A^m, respectively. Then $\alpha_1 = \alpha_2$.

PROOF. The proof is by induction on the complexity of D.

For the base case, suppose that no inference rule has been applied in D. Hence D is simply α, and α is in A. By definition D_1^m is α^m, and D_2^m is $+(w_i) \vdash \alpha^m$, where α^m is α marked with monotonicity information.

For the inductive case, we assume that the proposition holds for any proof E where less than n inference rules have been applied. We consider different cases according to which is the last inference rule applied in D,

Case [MP]: Let [MP] be the last rule applied in D

$$\dfrac{\begin{matrix}E_1 & E_2\\ \alpha \to \beta & \alpha\end{matrix}}{\beta}\ [\mathrm{MP}].$$

By induction hypothesis the markers assigned to $\alpha \to \beta$ as the root of E_1 by LP+EPol and LP+SPol coincide, and in particular the markers assigned to β, which is the root of D.

Case [Ab]: Let [Ab] be the last rule applied in D

$$\dfrac{\begin{matrix}[\alpha]\\ \vdots\\ \beta\end{matrix}}{\alpha \to \beta}\ [\mathrm{Ab}]$$

Let E be the subproof of D ending in β, and E_1^m and E_2^m the marking of E by LP+EPol and LP+SPol, respectively. By induction, the markings of α^m and β^m are equivalent in LP+EPol and LP+SPol. When rewriting the [Ab] rule by any of the two algorithms the only change in the marking of α^m and β^m will be on choosing $s \in \{+, -\}$ to obtain $(\alpha^m)^s \to \beta^m$ and $(\alpha^m)^s \to \beta^m$, respectively.

Now, α^m will be marked as $(\alpha^m)^-$ by LP+EPol, if the number of [MP$^-$] in the branch from β^m to α^m is odd, and as $(\alpha^m)^+$ otherwise. We will prove that LP+SPol assigns the same marker. Formally, for structures S_1 and S_2, and variable x, $S_1[s_1(\{\!|s_2(S_2), s_3(x)|\!\})] \vdash \beta^m$ is the root of E_2^m, by induction. We will prove that the sign that x receives in $Coll(+(S_1), x)$, which is the sign passed to α^m during the rewriting in LP+SPol, coincides with the marker assigned by LP+EPol. This is implied by the following observations.

Notice from the rewriting algorithm, that the sign of the simple components of the structure is modified only when rewriting an [MP$^-$] rule, as this is the only rule that introduces a negative sign in the structure. In addition, all [MP$^-$] rules (and only those) in the branch from β^m to α^m, modify the sign of x, as they add a $-$ sign to a structure containing x. From this, a simple counting argument forces the sign of x to be $-$ in $Coll(+(S_1), x)$ if the number of [MP$^-$] is odd, and $+$ otherwise, as required. QED

THEOREM A.12. [Embedding LP+EPol into LP+SPol] Let $T = \{w_1 \in \alpha_1, \ldots, w_n \in \alpha_n\}$ be a type of $w = w_1, \ldots, w_n$, let $A = \{\alpha_1, \ldots, \alpha_n\}$ and let D be a proof of $A \vdash \alpha$ in LP, with set of discharged assumptions B.

Let D_1^p be the polarity marked proof obtained by running the monotonicity and polarity algorithms of LP+EPol with initial markings A^m and B^m. And let D_2^p ending in $S \vdash \alpha'$ be the proof marked by the algorithm in LP+SPol.

Let β in D_1^p be an analysis closed node with set L of undischarged assumptions. Then there is a substructure S' of $Comp(S)$ such that

i. S' is built over L (without repetitions) and does not contain variables.

ii. The sign of S' coincides with the marker assigned to β.

PROOF. Let β be a node as in the hypothesis. Let $S_\beta \vdash \beta$ be the corresponding node in D_2^p. Because β is analysis closed, S_β does not contain variables and is built over L without repetitions. Furthermore, as the rewriting algorithm preserves substructures without variables, S_β will be a subterm of S. *Comp* might modify the signs in S_β producing a substructure S', but S' will still satisfy condition i).

What remains to be proved ii) is that the sign of S' coincides with the polarity marker assigned to β. Now, the sign of S' in $Comp(S)$ is determined by the original sign of S_β together with the number of $-$ in substructures of S properly containing S_β. Let D^m be the polarity marked proof obtained by running the monotonicity algorithm of LP+EPol with initial markings A^m and B^m.

We claim that:

1. The sign of S_β coincides with the monotonicity marker assigned to β in D_1^p.

2. The number of $-$ in substructures of S properly containing S_β coincides with the number of nodes marked as $-$ in the path from the root to β.

The proof of 1) is simple. First of all, notice that because S_β does not contain variables, its sign will not be affected by the application of a *Coll* function occurring further down in the proof of $A \vdash \alpha$. Hence, we only need to verify that the monotonicity signs assigned to nodes by the algorithm of LP+EPol coincide with those assigned to structures by LP+SPol. This can be seen immediately for the [Ab] rule which always assigns a $+$ symbol. For the [MP] rule use Proposition A.11 to verify that signs of logical types in D_1^p and D_2^p coincide.

The proof of 2) uses a similar argument. Consider the negative nodes in the path from the root of D_1^p to β. As we see from the monotonicity marking algorithm of LP+EPol, the only nodes that can receive a negative marker are the minor premises of an [MP] inference where the major premise is of type $(\gamma)^- \to \delta$. Again, by Proposition A.11, all logical types in D_2^p coincide with the marked types of D_1^p. Now, the rewriting rule of [MP] in LP+SPol will copy this sign around S_β, all other signs around S_β being positive.

QED

Half of the job is done by now. We only need to proceed with the second half of the embedding by mimicking LP+SPol into EMCTL+Pol. We define the embedding through translation functions from the logical and structural languages of LP+SPol into those of EMCTL+Pol.

DEFINITION A.13. [Translation Functions] We define the translation functions $ltr : \mathsf{MFORM} \to \mathsf{FORM}$ and $str : \mathsf{MSTRUC} \to \mathsf{STRUCT}$ by recursion as follows:

$$\begin{array}{l|l} ltr((\psi_1)^s \to \psi_2) = \boxplus^{\downarrow}(ltr(\psi_2) / \diamond\!\!\!\!\cdot\, ltr(\psi_1)); & str(s(\{S_1, S_2\})) = \langle str(S_1) \circ str(S_2)\rangle^s; \\ ltr(e) = \boxplus^{\downarrow} np; & str(s(S)) \quad = \langle str(S)\rangle^s; \\ ltr(p) = \boxplus^{\downarrow} n; & str(A) \quad = ltr(\alpha), \text{for} A \in \mathsf{VAR} \cup \mathsf{LEX} \\ ltr(t) = \boxplus^{\downarrow} s; & \quad\quad \text{and } A \vdash \alpha \text{ in } \mathsf{LP{+}SPol}. \end{array}$$

PROPOSITION A.14. Let $T = \{w_1 \in \alpha_1, \ldots, w_n \in \alpha_n\}$ be the types of $w = w_1, \ldots, w_n$, let $A = \{\alpha_1, \ldots, \alpha_n\}$ and let D be a proof of $A \vdash \alpha$ in LP, with set of discharged assumptions B.

Let D^p ending in $+(S) \vdash \alpha^m$ be the polarity marked proof obtained by running the algorithm of $\mathsf{LP{+}SPol}$ with initial markings A^m and B^m. Then there is a derivation D' in $\mathsf{EMCTL{+}Pol}$ which given $str(w_i) \vdash ltr(\alpha_i)$ as axioms derives $str(S) \vdash ltr(\alpha^m)$.

PROOF. The proof is by induction on the complexity of D

For the base case, suppose that no inference rule has been applied in D. Hence D is simply $+(w_i) \vdash \alpha_i^m$ and $str(w_i) \vdash ltr(\alpha_i^m)$. Notice that the (positive) polarity of the node can be displayed simply by applying the $[\boxplus^{\downarrow}\text{E}]$ rule.

For the inductive case, we assume that the proposition holds for any proof where less than n rules have been applied. We consider different cases according to which is the last inference rule applied in D.

Case [MP]: Let [MP] be the last rule applied in D, then the corresponding node in D^p is

$$\dfrac{\begin{array}{cc}\vdots & \vdots \\ S_1 \vdash (\alpha)^s \to \beta & S_2 \vdash \alpha\end{array}}{+(\{+(S_1), s(S_2)\}) \vdash \beta}\ [\text{MP}].$$

By induction we have: $str(S_1) \vdash ltr((\alpha)^s \to \beta)$ and $str(S_2) \vdash ltr(\alpha)$. The proof in $\mathsf{EMCTL{+}Pol}$ continues as follows

$$\dfrac{\dfrac{str(S_1) \vdash \boxplus^{\downarrow}(ltr(\beta) / \diamond\!\!\!\!\cdot\, ltr(\alpha))}{\langle str(S_1)\rangle^+ \vdash ltr(\beta) / \diamond\!\!\!\!\cdot\, ltr(\alpha)}\ [\boxplus^{\downarrow}\text{E}] \quad \dfrac{str(S_2) \vdash ltr(\alpha)}{\langle str(S_2)\rangle^s \vdash \diamond\!\!\!\!\cdot\, ltr(\alpha)}\ [\diamond\!\!\!\!\cdot\, \text{I}]}{\langle str(S_1)\rangle^+ \circ \langle str(S_2)\rangle^s \vdash ltr(\beta)}\ [/\text{E}].$$

which is $str((\{+(S_1), s(S_2)\})) \vdash ltr(\beta)$. Again the final + will be displayed by means of $[\boxplus^{\downarrow}\ \text{E}]$.

Case [Ab]: Let [Ab] be the last rule applied in D, then the corresponding node in D^p is

$$\dfrac{\begin{array}{c}[x \vdash \alpha] \\ \vdots \\ Coll(+(S_1), x)[s_1(\{s_2(S_2), s_3(x)\})] \vdash \beta\end{array}}{Coll(+(S_1), x)[s_2(S_2)] \vdash \alpha^{s_3} \to \beta}\ [\text{Ab}].$$

By induction we have $str(+(S_1)) \vdash ltr(\beta)$. We have to show that we can imitate both (1) the effect of the *Coll* function, and (2) the abstraction rule in EMCTL+Pol.

To prove (1), we need only to analyze the cases in the definition of the *Coll* function. We describe the more complex case, $S_1 = s_1(\{\!|s_2(S_2'), s_3(S_3)|\!\})$ and $x \in s_2(S_2')$, then

$$\dfrac{\dfrac{\dfrac{\langle\langle str(S_2')\rangle^{s_2} \circ \langle str(S_3)\rangle^{s_3}\rangle^{s_1} \vdash ltr(\beta)}{\langle\langle\langle str(S_2')\rangle^{s_2}\rangle^{s_1} \circ \langle\langle str(S_3)\rangle^{s_3}\rangle^{s_1}\rangle^{+} \vdash ltr(\beta)}\,[\mathrm{Pol}^{+}_{s_1}]}{\langle\langle str(S_2)\rangle^{sg(s_2,s_1)} \circ \langle\langle str(S_3)\rangle^{s_3}\rangle^{s_1}\rangle^{+} \vdash ltr(\beta)}\,[\mathrm{Pol}_{s_2 s_1}]}.$$

These rules can be applied on the S_2 structure till arriving to compute the final mark assigned to the variable. This result will be equivalent to one obtained by applying $Coll(S_1, x)$. Notice that the induction argument goes through because the structural rules $[\mathrm{Pol}^{+}_{s_1}]$ and $[\mathrm{Pol}_{s_2 s_1}]$ can be applied in any context.

We turn now to prove (2). If we observe the rewriting condition for the abstraction rule in LP+SPol, we see that it records information stored in the structural component into the logical type. In EMCTL+Pol we have to mimic this behavior, and in addition, perform the logical abstraction to create the functional type. For this reason, in EMCTL+Pol we will apply first the $[\diamondsuit \mathrm{E}]$ rule to record the computed polarity marking, followed by an application of $[/\mathrm{I}^{+}]$ to perform the abstraction. The complete proof will be as below. To facilitate the comparison we use x and y for assumed logical formulas.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[str(x) \vdash ltr(\alpha)]^1}{\vdots \longleftarrow (a)}}{str(S_1) \vdash ltr(\beta)}}{\vdots \longleftarrow (b)}}{S_c[\langle x\rangle^s] \vdash ltr(\beta)}}{\vdots \longleftarrow (c)}}{\dfrac{[y \vdash \diamondsuit_{\downarrow} ltr(\alpha)]^2 \quad S_c' \circ \langle x\rangle^s \vdash ltr(\beta)}{S_c' \circ y \vdash ltr(\beta)}\,[\diamondsuit \mathrm{E}]^1}}{\dfrac{\langle S_c'\rangle^{+} \vdash ltr(\beta)/\diamondsuit_{\downarrow} ltr(\alpha)}{S_c' \vdash \boxplus^{\downarrow}(ltr(\beta)/\diamondsuit_{\downarrow} ltr(\alpha))}\,[\boxplus^{\downarrow}\mathrm{I}]}\,[/\mathrm{I}^{+}]^2}$$

(a) is the original proof obtained by induction, with $str(S_1) \vdash ltr(\beta)$ the inference corresponding to the node before the abstraction. In (b) we use the polarity structural rules as we explained in (1) to simulate $Coll(+(S_1), x)$. In (c) we use the [Esc] rule which together with the rules of permutation and associativity let us push variables to the outside of the structure. Notice how the [Esc] rule let us perform the substitution of $str(Coll(+(S_1), x)[+(\{\!|s_2(S_2), s_3(x)|\!\})])$ by $str(Coll(+(S_1), x)[s_2(S_2)])$.

From then onwards, the proof is straightforward. The application of $[\diamondsuit \mathrm{E}]$ calls for an assumption with the appropriate polarity marker in the type ($\diamondsuit_{\downarrow} ltr(\alpha)$), which will be introduced in the argument position of the type by the $[/\mathrm{I}^{+}]$ rule. Notice that since in LP+EPol the associative rule is implicit in the system, open nodes might differ from the ones in LP+SPol and EMCTL+Pol where the markers on the variable have to 'collapse', before the abstraction can be executed. QED

The last step in our embedding of LP+SPol into EMCTL+Pol is to show that the latter system can account also for the *Comp* function. This last fact together with Theorem A.12, shows that the polarity markers obtained by LP+EPol for analysis closed nodes can be obtained in a purely logical formalism. The proof is similar to the mimicking of the *Coll* function we performed in Proposition A.14. We only need to take care of the fact that EMCTL+Pol is more "economic" than LP+EPol: it computes polarities on-demand. Formally,

PROPOSITION A.15. Let $S \vdash \alpha$ be a derivation in LP+SPol, and let D, ending in $str(S) \vdash ltr(\alpha)$ be the corresponding proof in EMCTL+Pol which exists by Proposition A.14. Let $z(S_1)$ be a substructure of $Comp(S)$, then either $str(z(S_1))$ is already in $str(S)$ or it can be obtained from $str(S) \vdash ltr(\alpha)$ by application of structural polarity rules.

The proof of Proposition A.15 relies on the fact that the polarity of a substructure S_1 in the structure S in EMCTL+Pol is determined when the structural rules have eliminated all the unary structural operators surrounding S_1 but one. This can always be achieved by the application of the structural polarity rules, following the operation of *Comp*, as we did with *Coll* in Proposition A.14.

We are now ready to put all the pieces together and define the final embedding of LP+EPol into EMCTL+Pol. Given a structure $S \in \mathsf{STRUCT}$ and S' a substructure of S, we say that S' *has polarity* s in S if $S' = \langle S_1 \rangle^s$, S_1 is in FORM or is of the form $S_2 \circ S_3$ and there is no structure $S_4 = \langle S_5 \rangle^-$ in S containing S'.

THEOREM A.16. [Embedding LP+EPol into EMCTL+Pol] Let $T = \{w_1 \in \alpha_1, \ldots, w_n \in \alpha_n\}$ be a type of $w = w_1, \ldots, w_n$, let $A = \{\alpha_1, \ldots, \alpha_n\}$ and let D be a proof of $A \vdash \alpha$ in LP, with set of discharged assumptions B.

Let D^p ending in α^m be the polarity marked proof obtained by running the monotonicity and polarity algorithms of LP+EPol with initial markings A^m and B^m. And let β in D^p be an analysis closed node, with set L of undischarged assumptions, receiving polarity s.

Then there exists a structure S such that $S \vdash ltr(\alpha^m)$ can be derived in EMCTL+Pol, and there is a substructure S' of S, with polarity s, built over L without repetitions or variables.

PROOF. Let $+(S_1) \vdash \alpha'$ be the last node obtained by running the algorithm in LP+SPol. By Proposition A.11 we know that $\alpha' = \alpha^m$. Then, by Theorem A.12 we know that there is a substructure S_2 of $Comp(+(S_1))$ such that S_2 is built over L without repetitions and variables, and the sign of S_2 coincides with the marker assigned to β, i.e., $S_2 = s(S_3)$. By Proposition A.14 there is a proof in EMCTL+Pol ending in $str(+(S_1)) \vdash ltr(\alpha^m)$. By Proposition A.15, either $str(s(S_3))$ is already in $str(+(S_1))$ or it can be obtained from $str(S_1) \vdash ltr(\alpha^m)$ by the application of structural polarity rules (which only modify the structural part of the sequent). But $str(s(S_3))$ is $\langle str(S_3) \rangle^s$, which has polarity s and is built over L without repetitions or variables. Hence, $\langle str(S_3) \rangle^s$ is the polarity marked structure corresponding to β. QED

The proof of Theorem A.16 shows that EMCTL+Pol can mimic any proof of LP while assigning to closed nodes in the proof the same polarity obtained by means of the monotonicity and polarity rewriting rules of LP+EPol. The embedding of LP+EPol into EMCTL+Pol is proved using LP+SPol as a bridge. In fact, on the one hand LP+SPol is closely connected to LP+EPol as regarding the logical types, and on the other it records polarity markers on the structures similarly to EMCTL+Pol.

We give two examples to help the reader understand how the three systems relate. In the first one only [MP] rules are involved, in the second we focus on [Ab].

EXAMPLE A.17. Let us consider again the proof D given in Example A.3. We show that for any closed node in D the polarity marker assigned to it by the marking algorithms of LP+EPol coincides with the one assigned by LP+SPol and EMCTL+Pol. To facilitate the comparison we repeat the final output obtained by LP+EPol after applying the polarity markers rewriting rules:

Polarity markers

$$\dfrac{\dfrac{\dfrac{\text{every} \in p \underset{-}{\rightarrow} ((e \rightarrow t) \rightarrow t) \qquad \text{good\_logician} \underset{+}{\in} p}{(e \rightarrow \underset{-}{t}) \rightarrow t} \qquad \text{wanders} \underset{-}{\in} e \rightarrow t}{\underset{-}{t}} \qquad \text{not} \underset{+}{\in} t \rightarrow t}{\underset{+}{t}}$$

Starting from the assignment $T = \{\text{not} \in (t \rightarrow t), \text{every} \in (p \rightarrow ((e \rightarrow t) \rightarrow t)), \text{good\_logician} \in p, \text{wanders} \in (e \rightarrow t)\}$, LP+SPol gives the following proof.

$$\dfrac{\text{not} \in t^{-} \rightarrow t \qquad \dfrac{\dfrac{\text{every} \in p^{-} \rightarrow ((e \rightarrow t)^{+} \rightarrow t) \quad \text{good\_logician} \in p}{{}_{+}(\{{}_{+}(\text{every}), {}_{-}(\text{good\_logician})\}) \vdash (e \rightarrow t)^{+} \rightarrow t}\ [\text{MP}] \qquad \text{wanders} \in e \rightarrow t}{{}_{+}(\{{}_{+}(\{{}_{+}(\text{every}), {}_{-}(\text{good\_logician})\}), {}_{+}(\text{wanders})\}) \vdash t}\ [\text{MP}]}{{}_{+}(\{{}_{+}(\text{not}), {}_{-}({}_{+}(\{{}_{+}(\{{}_{+}(\text{every}), {}_{-}(\text{good\_logician})\}), {}_{+}(\text{wanders})\}))\}) \vdash t}\ [\text{MP}]$$

Let us consider as an example the node $(e \rightarrow t) \rightarrow t$ negatively marked by LP+EPol. In LP+SPol the final polarity markers assigned to a node is computed by applying the *Comp* algorithm to the marked output of the proof. In the case we are considering the result is obtained as follow:

$$\begin{aligned}
&\mathit{Comp}({}_{+}(\{{}_{+}(\text{not}), {}_{-}({}_{+}(\{{}_{+}(\{{}_{+}(\text{every}), {}_{-}(\text{good_logician})\}), {}_{+}(\text{wanders})\}))\})) = \\
&{}_{+}(\{\mathit{Comp}(sg({}_{+}, {}_{+})(\text{not})), \mathit{Comp}(sg({}_{+}, {}_{-}))({}_{+}(\{{}_{+}(\{{}_{+}(\text{every}), {}_{-}(\text{good_logician})\}), {}_{+}(\text{wanders})\}))\}) = \\
&{}_{+}(\{{}_{+}(\text{not}), {}_{-}(\mathit{Comp}(sg({}_{-}, {}_{+})({}_{+}(\{{}_{+}(\text{every}), {}_{-}(\text{good_logician})\}))), \mathit{Comp}(sg({}_{-}, {}_{+})(\text{wanders})))\}) = \\
&{}_{+}(\{{}_{+}(\text{not}), {}_{-}({}_{-}(\{\mathit{Comp}(sg({}_{-}, {}_{+})({}_{+}(\text{every}), {}_{-}(\text{good_logician})))\}), {}_{-}(\text{wanders}))\}) = \\
&{}_{+}(\{{}_{+}(\text{not}), {}_{-}({}_{-}(\{\mathit{Comp}({}_{-}({}_{+}(\text{every}), {}_{-}(\text{good_logician})))\}), {}_{-}(\text{wanders}))\})
\end{aligned}$$

As the last line in the computation shows the final marker assigned to the substructure `every good_logician` which corresponds to the node $(e \rightarrow t) \rightarrow t$ we are considering,

is − as expected. The same result is obtained in EMCTL+Pol as shown by the Example A.7. In this system the polarity substructure is defined in terms on the unary structural operator and is computed by means of structural polarity rules. In case of the substructure we are interested in, the proof has to be completed as follows:

$$\dfrac{\dfrac{\dfrac{\vdots}{\langle\text{not}\rangle^{+}\circ\langle\langle\langle\text{every}\rangle^{+}\circ\langle\text{good\_logician}\rangle^{-}\rangle^{+}\circ\langle\text{wanders}\rangle^{+}\rangle^{-}\vdash\boxplus^{\downarrow}s}\,[\text{Pol}^{+}_{-}]}{\langle\text{not}\rangle^{+}\circ\langle\langle\langle\langle\text{every}\rangle^{+}\circ\langle\text{good\_logician}\rangle^{-}\rangle^{+}\rangle^{-}\circ\langle\langle\text{wanders}\rangle^{+}\rangle^{-}\rangle^{+}\vdash\boxplus^{\downarrow}s}\,[\text{Pol}_{-+}]}{\langle\text{not}\rangle^{+}\circ\langle\langle\langle\text{every}\rangle^{+}\circ\langle\text{good\_logician}\rangle^{-}\rangle^{-}\circ\langle\langle\text{wanders}\rangle^{+}\rangle^{-}\rangle^{+}\vdash\boxplus^{\downarrow}s}$$

In this example in LP+SPol there has been no need of applying the *Coll* algorithm, since no abstraction rule had been applied in the proof given in LP+EPol. For a better understanding of this algorithm and the way it interacts with the *Comp* one, we will show a second case where this rule is needed.

Example A.18. Let `No student attended all lectures` be the sequence of words from the lexicon we want to parse. Let $T = \{\text{no\_student} \in (e \to t) \to t, \text{attended} \in e \to (e \to t), \text{all\_lecture} \in (e \to t) \to t\}$ be its assignment of types. In LP+EPol the string is proved to be of type t by the following analysis:

$$\dfrac{\dfrac{\text{no\_student} \in (e \to t) \to t \qquad \dfrac{\dfrac{\text{attended} \in e \to (e \to t)[x \in e]^{1}}{e \to t}\,[\text{MP}]}{t}\,[\text{MP}]}{e \to t}\,[\text{Ab}]^{1} \qquad \text{all\_lecture} \in (e \to t) \to t}{t}\,[\text{MP}]$$

Following the algorithm:

Monotonicity markers $\Rightarrow$ **Polarity markers**

$$\dfrac{(e \to t)^{-}_{+} \to t \qquad \dfrac{e^{+} \to (e^{+} \to t) \qquad e^{+}}{e^{+} \to t}}{t}$$

$$\dfrac{\dfrac{e^{-} \to t}{t} \qquad (e \to t)^{+} \to t}{t}$$

$$\dfrac{(e \to t)^{-}_{+} \to t \qquad \dfrac{e^{+} \to (e^{+} \to t) \qquad e^{+}}{e^{+} \to t}}{t}$$

$$\dfrac{\dfrac{e^{-} \to t}{t} \qquad (e \to t)^{+} \to t}{t}$$

Applying the rewriting algorithm of LP+SPol we obtain the following polarity marker assignments. We focus our attention on the relevant part of the proof where the "counting" has to be done, namely where the abstraction is applied.

$$\dfrac{\text{no\_student} \vdash (e \to t)^{-} \to t \qquad \dfrac{\text{attended} \vdash e^{+} \to (e^{+} \to t) \quad [x \vdash e]^{1}}{{}_{+}(\{\!|{}_{+}(\text{attended}), {}_{+}(x)|\!\}) \vdash e^{+} \to t}\,[\text{MP}]}{{}_{+}(\{\!|{}_{+}(\text{no\_student}), {}_{-}({}_{+}(\{\!|{}_{+}(\text{attended}), {}_{+}(x)|\!\}))|\!\}) \vdash t}\,[\text{MP}]$$

Before applying [Ab] the algorithm *Coll* has to run over the reached conclusion so to collapse the markers surrounding the variable on which the abstraction takes place.

$$Coll({}_{+}(\{{}_{+}(\text{no\_student}), {}_{-}({}_{+}(\{{}_{+}(\text{attended}), {}_{+}(x)\}))\}), x) =$$
$${}_{+}(\{{}_{+}({}_{+}(\text{no\_student})), Coll(sg({}_{+}, {}_{-})({}_{+}(\{{}_{+}(\text{attended}), {}_{+}(x)\})), x)\}) =$$
$${}_{+}(\{{}_{+}({}_{+}(\text{no\_student})), Coll(sg({}_{-}, {}_{+})(\{{}_{+}(x), {}_{+}(\text{attended})\}), x)\}) =$$
$${}_{+}(\{{}_{+}({}_{+}(\text{no\_student})), {}_{+}(\{{}_{-}({}_{+}(\text{attended})), Coll(sg({}_{-}, {}_{+})(x), x)\})\}) =$$
$${}_{+}(\{{}_{+}({}_{+}(\text{no\_student})), {}_{+}(\{{}_{-}({}_{+}(\text{attended})), {}_{-}(x)\})\}) =$$

Now the proof can go on starting from the output of *Coll*

$$\frac{Coll(+(S_1), x)[+(\{-(+(\text{attended})), -(x)\})] \vdash t}{Coll(+(S_1), x)[-(+(\text{attended}))] \vdash e^{-} \rightarrow t}\ [\text{Ab}]^1$$

where S_1 stands for $(\{+(\text{no\_student}), -(+(\{+(\text{attended}), +(x)\}))\})$. Notice again that the marker assigned to the type $(e \rightarrow t)$ in the conclusion of [Ab] results from this computation and it coincides with the one obtained by the "manual" counting done in LP+EPol.

In EMCTL+Pol too some counting has to be done before discharging the hypothesis. This task is carried out by the polarity structural rules which assign to the assumed structure a negative polarity marker. However, the conclusion so obtained is not ready for applying the $[/\text{I}^{+}]$, yet. This rule can be applied only over structures not surrounded by any $(\cdot)$ or $\langle\cdot\rangle^{s}$. The first request is satisfied by the application of the [Ass] rule which re-brackets the structure of its premises, togheter with the [Esc] rule which distribute the $\langle\cdot\rangle^{+}$ making possible the final computation of the polarity of x. The second one is obtained by means of the $[\diamondsuit\,\text{E}]$ rule, which replaces on the the structural side the original hypothesis of type $\boxplus^{\downarrow}np$ with a new one of type $\diamondsuit\,\boxplus^{\downarrow}np$. This substitution is motivated by the correspondence between $\langle\cdot\rangle^{-}$ and the $\diamondsuit$ mentioned when introducing EMCTL+Pol (Chapter 2). At this point the abstraction can take place discharging the new assumption and giving a properly marked function, viz. a function with negatively marked argument. Again, we give only the part of the proof involving the abstraction and the computation.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[x \vdash \boxplus^{\downarrow}np]^1}{\vdots}}{\langle\langle\text{no\_student}\rangle^{+} \circ \langle\langle\langle\text{attended}\rangle^{+} \circ \langle x\rangle^{+}\rangle^{+}\rangle^{-}\rangle^{+} \vdash \boxplus^{\downarrow}s}\ [\text{Pol}_{+-}]}{\langle\langle\text{no\_student}\rangle^{+} \circ \langle\langle\text{attended}\rangle^{+} \circ \langle x\rangle^{+}\rangle^{-}\rangle^{+} \vdash \boxplus^{\downarrow}s}\ [\text{Pol}_{-}]}{\langle\langle\text{no\_student}\rangle^{+} \circ \langle\langle\langle\text{attended}\rangle^{+}\rangle^{-} \circ \langle\langle x\rangle^{+}\rangle^{-}\rangle^{+}\rangle^{+} \vdash \boxplus^{\downarrow}s}\ [\text{Pol}_{+-}]}{\langle\langle\text{no\_student}\rangle^{+} \circ \langle\langle\langle\text{attended}\rangle^{+}\rangle^{-} \circ \langle x\rangle^{-}\rangle^{+}\rangle^{+} \vdash \boxplus^{\downarrow}s}\ [\text{Esc}]}{\langle\langle\text{no\_student}\rangle^{+} \circ (\langle\langle\langle\text{attended}\rangle^{+}\rangle^{-}\rangle^{+} \circ \langle x\rangle^{-})\rangle^{+} \vdash \boxplus^{\downarrow}s}\ [\text{Ass}]}{\langle(\langle\text{no\_student}\rangle^{+} \circ \langle\langle\langle\text{attended}\rangle^{+}\rangle^{-}\rangle^{+}) \circ \langle x\rangle^{-}\rangle^{+} \vdash \boxplus^{\downarrow}s}\ [\text{Esc}]}{\langle(\langle\text{no\_student}\rangle^{+} \circ \langle\langle\langle\text{attended}\rangle^{+}\rangle^{-}\rangle^{+})\rangle^{+} \circ \langle x\rangle^{-} \vdash \boxplus^{\downarrow}s}\ [\diamondsuit\,\text{E}]^1}{\langle(\langle\langle\text{no\_student}\rangle^{+}\rangle^{+} \circ \langle\langle\text{attended}\rangle^{+}\rangle^{-})\rangle^{+} \circ y \vdash \boxplus^{\downarrow}s}\ [/\text{I}^{+}]^2}{\langle\langle\langle\langle\text{no\_student}\rangle^{+}\rangle^{+} \circ \langle\langle\text{attended}\rangle^{+}\rangle^{-}\rangle^{+}\rangle^{+} \vdash \boxplus^{\downarrow}s/\diamondsuit\,\boxplus^{\downarrow}np}\ [\boxplus^{\downarrow}\text{E}]}{\langle\langle\langle\text{no\_student}\rangle^{+}\rangle^{+} \circ \langle\langle\text{attended}\rangle^{+}\rangle^{-}\rangle^{+} \vdash \boxplus^{\downarrow}(\boxplus^{\downarrow}s/\diamondsuit\,\boxplus^{\downarrow}np)}$$

Bibliography

[AB01] C. Areces and R. Bernardi. Analyzing the core of categorial grammar. In P. Blackburn and M. Kohlhase, editors, *Proceedings of ICoS-3*, pages 7–28, April 2001.

[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).

[Bak70] C. Baker. Double negatives. *Linguistic Inquiry*, 1:169–186, 1970.

[Bel82] N. Belnap. Display logic. *Journal of Philosophical Logic*, 11(4):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.

[Ben86] J. van Benthem. *Essays in logical semantics*. Reidel Publishing Company, Dordrecht, 1986.

[Ben87a] J. van Benthem. Categorial grammar and lambda calculus. In D. Skordev, editor, *Mathematical Logic and its Applications*, pages 39–60. Plenum, New York, 1987.

[Ben87b] J. van Benthem. Meaning: interpretation and inference. *Synthese*, 73(3):451–470, 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.

[Ben91] J. van Benthem. *Language in Action. Categories, Lambdas and Dynamic Logic*. Studies in Logic. North-Holland, Amsterdam, 1991. (Student edition: (1995), MIT Press, Cambridge, MA.).

[Ber00] R. Bernardi. Polarity items in resource logics. A comparison. In *Proceedings ESSLLI Student Session*, 2000.

[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.

[Bir67] G. Birkhoff. *Lattice Theory*. American Mathematical Society, Providence, (1940, 1948, 1967).

[BJ72] T. S. Blyth and M. F. Janowitz. *Residuation Theory*. Pergamon Press, Oxford, 1972. International Series of Monographs in Pure and Applied Mathematics, Vol. 102.

[BM00] R. Bernardi and R. Moot. Scope ambiguities from a proof theoretical perspective. In J. Bos and M. Kohlhase, editors, *Proceedings of ICoS-2*, 2000. Revised version in *Journal of Language and Computation*. In printing.

[BN02] R. Bernardi and Ø. Nilsen. A type logical calculus of syntactic features. Scope and polarity phenomena. Manuscript, 2002.

[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.

[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.

[Chi02] G. Chierchia. Scalar implicatures, polarity phenomena, and syntax/pragmatics interface. Manuscript, 2002.

[Cho87] J. Choe. *Anti-quantifiers and a Theory of Distributivity*. PhD thesis, University of Massachussetts, Amherst, 1987.

[Cho95] N. Chomsky. *The Minimalist Program*. MIT Press, Cambridge Mass, 1995.

[Cho99] N. Chomsky. Derivation by phase. Technical Report 18, MIT, 1999.

[Chu40] A. Church. A formulation of a simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1940.

[Coo83] R. Cooper, editor. *Quantification and Syntatic Theory*. Reidel, Dordrecht, 1983.

[Dos92] K. Dosen. A brief survey of frames for the Lambek calculus. *Zeitschrift für mathematischen Logik und Grundlagen der Mathematik*, 38:179–187, 1992.

[Dow94] D. Dowty. The role of negative polarity and concord marking in natural language reasoning. In *Semantics and Linguistic Theory*, volume IV, pages 114–144. Cornell University, 1994.

[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.

[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.

[Fau75] G. Fauconnier. Pragmatic scales and logical structure. *Linguistic Inquiry*, 6:353–375, 1975.

[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.

[Fry99] J. Fry. Proof nets and negative polarity licensing. In M. Dalrymple, editor, *Semantics and Syntax in Lexical Functional Grammar. The Resource Logic Approach*, chapter 3, pages 91–116. MIT Press, 1999.

[Fuc63] L. Fuchs. *Partially Ordered Algebraic Systems*. Pergamon Press Inc., New York, 1963.

[FWF00] Y. Fyodorov, Y. Winter, and N. Francez. A natural logic inference system. In J. Bos and M. Kohlhase, editors, *Inference in Computational Semantics (ICoS-2)*, 2000. Revised version in *Journal of Language and Computation*. In printing.

[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.

[Gia98] A. Giannakidou. *Polarity sensitivity as (non)veridical dependency*. John Benjamins, Amsterdam, 1998.

[Gia99] A. Giannakidou. Affective dependencies. *Linguistics and Philosophy*, 236:367–421, 1999. Kluwer Academic Publishers.

[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.

[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.

[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. Substructural logics on display. *Logic Journal of the IGPL. Interest Group in Pure and Applied Logics*, 6(3):451–504, 1998.

[GS84] S. Groenendijk and M. Stokhof. *Studies on the Semantics of Questions and the Pragmatics of Answers*. PhD thesis, University of Amsterdam, 1984.

[Hae94] L. Haegeman. *Introduction to Government and Binding Theory*. Blackwell, Oxford, 2nd edition, 1994.

[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.

[Hoe86] J. Hoeksema. Monotonicity phenomena in natural language. *Linguistic Analysis*, 16(1-2):25–40, 1986.

[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.

[KL89] N. Kadmon and F. Landman. Polarity sensitive *any* and free choice *any*. In *Seventh Amsterdam Colloquium*, pages 227–251, 1989.

[KL93] N. Kadmon and F. Landman. Any. *Linguistics and Philosophy*, 16:353–422, 1993.

[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.

[Kri95] M. Krifka. The semantics and pragmatics of polarity items. *Linguistic Analysis*, 25:209–257, 1995.

[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.

[Lad92] W. Ladusaw. Expressing negation. In C. Baker and D. Dowty, editors, *Proceedings of SALT 2*, pages 237–260. Ohio State University Linguistics Department, Columbus, 1992.

[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.

[Lyn59] R. C. Lyndon. Properties preserved under homomorphism. *Pacific Journal of Mathematics*, 9:142–154, 1959.

[May77] R. May. *The Grammar of Quantification*. PhD thesis, MIT, 1977.

[MM91] M. Moortgat and G. Morrill. *Heads and phrases. Type Calculus for Dependency and Constituent Structure*. OTS, University of Utrecht, Utrecht, 1991.

[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.

[Mor94] G. Morrill. *Type Logical Grammar*. Kluwer, Dordrecht, 1994.

[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.

[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.

[Pro88] L. Progovac. *A Binding Approach to Polarity Sensitivity*. PhD thesis, University of Southern California, 1988.

[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.

[Pro01] L. Progovac. Negative and positive feature checking and the distribution of polarity items. In Brown and Przepiorkowski, editors, *Negation in Slavic*. Slavica Publishers, 2001. To appear.

[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.

[Rad97] A. Radford. *Syntax. A Minimalist Introduction*. Cambridge University Press, 1997.

[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.

[Ros67] J.R. Ross. *Constraints on Variables in Syntax*. PhD thesis, MIT, 1967.

[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).

[Sta01] E. Stabler. Recognizing head movement. 2001.

[Ste00] M. Steedman. *The Syntactic Process*. Cambridge, MA: MIT Press, 2000.

[Sup79] P. Suppes. Logical inference in English: A preliminary analysis. *Polska Akademia Nauk. Institut Filozofii i Socijologii. Studia Logica*, 38(4):375–391, 1979.

[SV91] V. Sánchez-Valencia. *Studies on Natural Logic and Categorial Grammar*. PhD thesis, University of Amsterdam, 1991.

[Swa98] H. de Swart. Negation, polarity and inverse scope. *Lingua*, 105(3):175–200, 1998.

[SWZ93] V. Sánchez-Valencia, T. van Wouden, and F. Zwarts. Polarity, veridicality and temporal connectives. In P. Dekker and M. Stokof, editors, *Proceedings of the 9th Amsterdam Colloquium*, pages 587–606. ILLC, University of Amsterdam, 1993.

[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.

[Sza01] A. Szabolcsi. Positive polarity - negative polarity. Manuscript, 2001.

[Tho74] R. Thomason, editor. *Formal Philosophy: Selected papers of Richard Montague*. Yale University Press, New Haven, 1974.

[Tov96] M. L. Tovena. *Studies on Polarity Sensitivity*. PhD thesis, University of Edinburgh, 1996.

[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. Kluwer, Dordrecht, 1983.

[Zwa86] F. Zwarts. *Categoriale Grammatica en Algebraische Semantiek*. PhD thesis, University of Groningen, 1986.

[Zwa95] F. Zwarts. Nonveridical contexts. *Linguistic Analysis*, 25:286–312, 1995.

Index

Samenvatting

Deze dissertatie volgt de *parsing as deduction* lijn van taalkundig onderzoek. We gebruiken het gereedschap van Categoriale Typenlogica (CTL) om de interface tussen de syntax en de semantiek van natuurlijke taal te onderzoeken. Ons doel is het onderzoeken van de mathematische structuur van CTL en het verkennen van de expressiviteit van CTL voor het analyseren van structuren in natuurlijke taal.

Deze dissertatie is onderverdeeld in drie delen met elk een inleidend hoofdstuk. In hoofdstuk 1 introduceren we de achtergronden van de categoriale visie op taalkunde en schetsen we de ontwikkelingen die hebben geleid tot de introductie van CTL. We motiveren ook het gebruik van logische methoden voor taalkundige analyse. In hoofdstuk 3 geven we onze visie op het gebruik van unaire modaliteiten als logische 'features'. In hoofdstuk 5 geven we een algemene notie van grammaticale compositie, die rekening houdt met zowel de vorm als de betekenis van taalkundige expressies. We ontwikkelen een logische theorie van *licensing* en *antilicensing* relaties, die de vorm- en betekenisdimensies doorkruist.

We zullen ons vooral bezighouden met *polariteit*. Deze term verwijst zowel naar de polariteit van de logische operatoren van CTL als naar de polariteitsgevoelige expressies die in natuurlijke taal voorkomen en die bovendien nauw samenhangen met natuurlijk redeneren. De titel van dit proefschrift, 'Redeneren met Polariteiten in Categoriale Typenlogica', heeft daarom drie interpretaties.

Ten eerste redeneren we met de *polariteit van de logische operatoren* en bestuderen we hun afleidbaarheidspatronen. In hoofdstuk 2 bestuderen we de algebraïsche principes die het gedrag van de operatoren van de Lambek calculus bepalen. We breiden het categoriale vocabulaire uit met neerwaarts implicerende unaire operaties om zo de volledige verzameling logische hulpmiddelen te verkrijgen die we de rest van dit proefschrift zullen gebruiken. We gebruiken unaire operatoren om monotoniciteitsinformatie te representeren en te berekenen (hoofdstuk 4), om een verklaring te geven voor verschillen in het bereik van gegeneraliseerde kwantoren (hoofdstuk 6), en om licensing en antilicensing relaties te modelleren (hoofdstuk 7).

Ten tweede modelleren we in hoofdstuk 4 inferenties van *natuurlijk redeneren* met behulp van structuren met 'negative polarity items'. Specifiek beschrijven we een redeneersysteem gebaseerd op CTL. Door functionele typen met unaire operatoren te verri-

jken, geven we een weergave van het semantische verschil tussen opwaarts en neerwaarts monotone functies. Verder bestuderen we de voordelen van deze codering door de contributies van monotone functies tot natuurlijk redeneren en de syntactische distributie van negative polarity items te onderzoeken.

In hoofdstuk 7, tenslotte bestuderen we de distributie van polariteitsgevoelige expressies. We laten zien hoe onze theorie van licensing en antilicensing relaties het juiste onderscheid maakt tussen negative polarity items die 'aangetrokken' worden door hun triggers, en positive polarity items die door hen worden 'afgestoten'. We onderzoeken deze compatibiliteits- en incompatibiliteitsrelaties vanuit een meertalig perspectief en laten zien hoe de verschillen in de distributies van polariteitsgevoelige woorden in het Nederlands, Grieks en Italiaans gereduceerd kunnen worden tot verschillen in de lexicale typetoekenningen in deze talen.

Riassunto

In questa tesi l'autore si avvale degli strumenti logici delle Grammatiche Categoriali (GC) per analizzare la sintassi e la semantica del linguaggio naturale. Obiettivo prinicipale che ha dato avvio alla ricerca, oggetto della tesi, è potenziare l'espressività delle GC.

La tesi è suddivisa in tre parti, ciascuna delle quali si apre con un'introduzione alle motivazioni e problematiche affrontate nei successivi capitoli. Nel Capitolo 1, l'autore introduce i concetti principali dell'approccio categoriale alla linguistica e le motivazioni che soggiaciono a tale approccio. Inoltre, presenta i limiti delle Grammatiche Categoriali classiche e come gli stessi vengono superati dalle logiche multimodali ottenute estendo il vocabolario logico delle GC tramite operatori unari. Nel Capitolo 3, l'autore propone un nuovo uso degli operatori unari ai fini dell'analisi di fenomeni linguistici. Infine, nel Capitolo 5, viene introdotto il metodo deduttivo sviluppato in dettaglio nei capitoli succesivi e si illustrano a livello teorico i vantaggi di tale approccio per la comprensione di classificazioni proposte in linguistica formale.

Nel corso della tesi l'attenzione è sempre rivolta al concetto di *polarità*. Il termine è usato in modo ambivalente: si riferisce sia alla polarità degli operatori logici, sia al fenomeno linguistico, noto come *negative polarity items* (NPI), esemplificato da esspressioni come *mai, ancora* ed *alcuno* che risultano grammaticali soltanto se utilizzate nel contesto corretto. Il titolo *Uso delle Polarità nella Grammatica Categoriale* esprime questo doppio significato.

Nel Capitolo 2, l'autore illustra il ruolo svolto dalle polarità degli operatori nella derivazione dei teoremi validi nella logica modale. Partendo dallo studio dei principi algebrici che governano il comportamento degli operatori della logica di base elaborata da Lambek [Lam58] e poi estesa da Moortgat e Kurtonina [KM95, Kur95, Moo96b], si mostra come la stessa struttura algebrica possa esprimere operatori unari con polarità inversa. Le proprietà logiche del sistema sono utilizzate per ottenere un sistema in grado di formalizzare le inferenze di un frammento del ragionamento naturale (Capitolo 4); inoltre le stesse proprietà sono impiegate per differenziare i quantificatori (Capitolo 6), e per rendere conto del comportamento sintattico dei NPI (Capitolo 7).

Nel Capitolo 4, l'attenzione è rivolta allo studio di inferenze basate su sostituzioni monotone. In particolare, l'autore descrive una logica naturale che si avvale delle analisi svolte dalla logica modale per dimostrare la grammaticalità delle strutture linguistiche

da cui derivare le inferenze.

Infine, nel Capitolo 7, l'autore studia il fenomeno linguistico delle polarità. Le caratteristiche che contraddistinguono i contesti che favoriscono la presenza di espressioni quali *mai, ancora* ed *alcuno*, da quelli che invece la bloccano, sono proprietà semantiche [Lad79, Gia97]. Un dato interessante che emerge da questi studi è che, sebbene tali proprietà semantiche siano constanti linguistiche (vale a dire non variano da lingua a lingua), i NPI mostrano diversi comportamenti sintattici, e.g. l'avverbio *possibilmente* in greco permette la presenza di NPI, a differenza del suo equivalente inglese. Nella tesi, l'autore riporta uno studio comparativo di questo fenomeno, prendendo in considerazione il greco, l'italiano e l'inglese.

Al fine di ottenere una chiara rappresentazione delle differenze tra i linguaggi naturali relativamente ai NPI si è assunta una prospettiva deduttiva, utilizzando la logica modale. Da questi studi è emerso che i comportamenti sintattici dei NPI possono essere correttamenti formalizzati e predetti assegnando appropiati tipi lessicali e applicando semplici regole logiche che stanno alla base delle Grammatiche Categoriali. Le differenze tra i linguaggi naturali sono pertando ridotte a pure differenze dei tipi lessicali.

Curriculum Vitae

Raffaella Bernardi was born on Augustus 9, 1971 in Pescara (Italy). She received her master degree in Philosophy in 1995 at the University of Chieti. In 1996 she awarded the Prize 'Soroptimist Club di Chieti Carla Martinotti di Baldassare', for her research activity at the University of Chieti. In 1997 she obtained a two year grant for studying abroad. In the years 1997 and 1998 she was a guest at the UiL OTS, Utrecht University. During these years she attended courses in Linguistics and Logic at Utrecht University and at the University of Amsterdam (ILLC). In 1999 she joint the International PhD Programme at the UiL OTS, where this dissertation was written. In addition to her work as a PhD student, she worked as public relation officer for the European Association for Logic, Language and Information (FoLLI). At the present she works for the 'Logic and Natural Language Processing' are of the European Net Work of Excellence in Computational Logic, (CologNet).