Scope Ambiguities through the Mirror

Raffaella Bernardi

In this paper we look at the interpretation of Quantifier Phrases from the perspective of Symmetric Categorial Grammar. We show how the apparent mismatch between the syntactic and semantic behaviour of these expressions can be resolved in a typelogical system equipped with two Merge relations: one for syntactic units, and one for the evaluation contexts of the semantic values associated with these syntactic units.

Keywords: Symmetric Categorial Grammar, Terms and Contexts, Continuation Semantics, Quantifiers.

1 The Logic behind Natural Languages

"The Mathematics of Sentence Structures", the title of (Lambek, 1958), clearly summarizes the ambitious challenge Jim Lambek has launched to researchers interested on formal approaches to natural language. It could also be read as asking "which is the logic that captures natural language composition?", which can be rephrased as "which are the models of natural language composition?", "which are its formal properties?".

Since (Montague, 1974), the λ-calculus was used by linguists to build the meaning representation of sentences compositionaly. The high expectations on the line of research pioneered by Lambek were even increased by Johan van Benthem's results (van Benthem, 1987) on the correspondence between the identified logic (known as Lambek calculus) and the λ-calculus. However, though all the questions above, and even more, were answered pretty soon (see (Lambek, 1961; Moortgat, 1988) among others), the answers found hold only for a fragment of natural language. What remains outside of the defined logic (and hence model and proof system) seemed to be those structures containing non-local dependencies and scope operators. The proof of the lack of expressivity of the Lambek calculus was given by the results on the correspondence of the proposed logics with Context Free Language, since it is known that natural language is out side this class of formal languages. Through the years, many proposals have been made to overcome these well defined limitations, but these proposals departed in a way or another from the original line of research Lambek had launched, e.g. they strive for wide coverage of the parser rather than for logical completeness of the system (Steedman, 2000).

In this paper, we go back to 1961 and start from there again, but with the new knowledge put at our disposal by research on algebra, logic, programming languages and linguistics. We show that the time is ready to recover the original enthusiasm on the possibility to solve Lambek's original challenge. In particular, we focus on scope operators and the mismatch that they might create between syntactic and semantic structures. The original questions regarding the model, the proof systems, and the meaning representation of the structures recognized by the logic we will present are formally answered in (Kurtonina & Moortgat, 2007; Bernardi & Moortgat, 2007; Moortgat, 2007; Bernardi & Moortgat, 2009; Moortgat, 2009); here we give a more gentle introduction to the system we propose and focus on its contribution to the linguistic research on scope in natural language.

Lambek brought to the light the mathematical apparatus behind Categorial Grammar (CG) showing that CG was based on half of the residuation principle and illustrated the advantages that one would gain by considering the whole mathematical principle given below.

$$(\text{RES})\ \ A \subseteq B/C \quad \text{iff} \quad A \otimes C \subseteq B \quad \text{iff} \quad C \subseteq A \backslash B$$

In more familiar term, the Lambek calculus, or calculus of residuation, consists of function application and abstraction rules, which are used to prove the grammaticality of a given sentence. For instance, given

that *Alex* belongs to the category of determiner phrases (dp) and *left* to the one of intransitive verbs ($dp\backslash s$) (i.e. a function missing a dp on the left to yield a sentence s), the expression *Alex left* is proved to belong to the category s simply by function application:

$$\begin{array}{lllll} \text{If} & \textit{Alex} & \in & dp \\ & \textit{left} & \in & dp\backslash s \\ \text{Then} & \textit{Alex left} & \in & s \end{array}$$

which is the proof of the following theorem:

(1) $$(a)\ \ dp \otimes (dp\backslash s) \vdash s$$

While function application was already used in CG, Lambek showed that the residuation calculus includes also abstraction which is needed, for instance, to prove the statement below, i.e. the type lifting rule used by linguists.

(2) $$(b)\ \ dp \vdash s/(dp\backslash s)$$

We propose to extend the expressivity of the logical grammar by moving to a symmetric calculus, i.e. a calculus in which each rule has a dual. The concept of *dual rules* and *dual theorems* was introduced by Schröder (Schröder, 1980) who noticed that the rules of propositional logic could be formulated in pairs corresponding to De Morgan dual operators, e.g. the rules about $\wedge$ are dual to rules about $\vee$ and so are the theorems below involving them:

$$A \wedge (B \vee C) \vdash (A \wedge B) \vee (A \wedge C) \ \text{ dually } \ (C \vee A) \wedge (B \vee A) \vdash (C \wedge B) \vee A$$

The logic language of the Lambek calculus consists of residuated operators $((A\backslash B), (A \otimes B), (B/A))$. We extend it with their duals, viz. $((B \oslash A), (B \oplus A), (A \osl B))$ and claim that besides the residuation principle (RES) the mathematics of sentence structures includes its dual principle (DRES),

$$(\text{DRES})\ \ B \oslash A \subseteq C \quad \text{iff} \quad B \subseteq C \oplus A \quad \text{iff} \quad C \osl B \subseteq A$$

together with the properties governing the interaction between residuated and dual residuated operators. These interaction properties were first studied by V. N. Grishin (Grishin, 1983). We call the new system Lambek Grishin (LG) (See Section 3).

In the Lambek calculus the theorems $dp \otimes dp\backslash s \ \vdash \ s$ and $dp \ \vdash \ s/(dp\backslash s)$ are proved by function application and abstraction, in the extended system also their duals hold:

(3) $$(a')\ \ s \vdash (s \oslash dp) \oplus dp \quad (b')\ \ (s \oslash dp) \osl s \vdash dp$$

and are proved by co-application and co-abstraction, i.e. the dual rules of application and abstraction.

In this new framework, we can distinguish categories standing for linguistic expressions and categories standing for the contexts of those expressions: contexts of category A are structures with an A-hole. The $\otimes$-relation merges linguistic expressions and the $\oplus$-relation merges contexts where these expressions can be plugged in. For instance, $dp\backslash s$ stands for *left* and *knows mary*, whereas $s \oslash dp$ stands for the contexts of *left* and *knows mary*, viz. the tree with a hole for that expression; for atomic formulas, we should distinguish between e.g. dp, standing for *mary*, and dp^c standing for the context of a dp. We can avoid marking this difference at the level of atomic formulas since the merging relation used disambiguate them.

We briefly discuss the Quantifying In mechanism proposed in (Montague, 1974) and its variation, Quantifier Raising, proposed by linguists to describe the behavior of quantifiers (May, 1977; Reinhart, 1997) (Section 4). The logic obtained is in the Curry Howard Correspondence with the $\bar{\lambda}\mu\tilde{\mu}$-calculus. The meaning representation of the parsed structures are built thanks to this correspondence. For reason of space, in this paper we won't discuss this part; the reader interested in the issue is referred to (Bernardi & Moortgat, 2007, 2009); the only aspect to highlight here is that this extended lambda calculus has a way to represent and compose "context" too and to shift from one merging relation to the other.

In sum, the main novelty of our proposal is to introduce symmetry in categorial grammar by extending the Lambek calculus with the duals of its rules and with communication principles governing the interaction

between the Lambek calculus and its dual. The new framework sheds new lights that we believe could be of interest for other frameworks too:

1. besides having categories standing for words, there is the need of having categories standing for the words' contexts;
2. hence, besides the merging relation, that builds a new syntactic unit from adjacent syntactic objects, natural language structures need also its dual relation, viz. the relation that merges contexts. (See (44) and (49) in Section 3.) Therefore, we can distinguish syntactic trees and context trees, the two types of trees can interact but their structure should not be modified once built. (See (51) in Section 3.)
3. non-local scope operators, like quantifiers, consist of a syntactic and a semantic component tied to each other. Thanks to the interaction between the two merging relations, the syntactic component stays in the syntactic structure or constituent-domain (c-domain) while the semantic component can jump out of it to reach the semantic-domain (s-domain), viz. the structure on which the QP has scope. (See (38) in Section 2 and Section 4.)
4. the c-domain is governed by the merging relation, whereas the s-domain is governed by the relation governing contexts. (See (62) and (74) in Section 4.)
5. a syntactic structure containing scope operators can be built in different ways: the order in which scope operators are activated determine the readings of the structure. (See (67), (71) in Section 4.)

2 Preliminary Background

Logic is about entailment of a conclusion from some premises. We might want to know whether it holds that whenever it is true that *If Alex swims then Alex gets wet* ($p \rightarrow q$) and that *Alex swims* (p) then it is also true that *Alex gets wet* (q), which is formalized by the entailment below.

(4) $$\{p \rightarrow q, p\} \vdash q$$

But as the objects of the reasoning vary, so it may vary the logic one needs. Our objects are natural language structures; hence we are interested in (a) tailoring the logic of natural language syntactic composition and (b) capturing its correlation with the semantic representation. We want to grasp the fragment of logic that suits our needs, and use a logic as a grammar. This approach is known as the "parsing as deduction" approach.

Since the number and order of occurrence of formulas (words) and their structure are important when dealing with natural language the premise of our entailment is not a set as in propositional logic but a structure. In propositional logic, both $\{p \rightarrow q, p, p\} \vdash q$ and $\{p \rightarrow (p \rightarrow q), p\} \vdash q$ hold, whereas we need a logic sensitive to the number and the order of the formulas.

As for (a), employing a logic as a grammar to analyse natural language expressions means to assign atomic or complex formulas to words and let their logical rules check that a given string of words is of a certain category and return its syntactic structure. Given the lexical entries $w_1 \dots w_n$, the system will take the corresponding syntactic categories $A_1 \dots A_n$ and prove which structures they can receive so as to derive a given category B, i.e., check whether a tree-structure of category B can be assigned to the given string of words.

(5) $$\underbrace{A_1}_{w_1} \cdots\cdots \underbrace{A_n}_{w_n} \vdash B$$

To be more precise, syntactic categories are sets of expressions: those expressions that belong to a given category, i.e. $dp\backslash s$ etc. are category labels: names for such sets. In short, we are dealing with e.g.:

$$\underbrace{\{\text{john, mary}\}}_{np} \underbrace{\times}_{\otimes} \underbrace{\{\text{left, knows lori}\}}_{np\backslash s} \underbrace{\subseteq}_{\vdash} \underbrace{\{\text{john left, mary left, john knows lori, mary knows lori}\}}_{s}$$

We take the liberty to use the term "category" to refer to a category label. Similarly, the structure built out of the syntactic categories is a name for the sets of phrases that belong to that structure.

Since the introduction of the *Chomsky Hierarchy*, the attention of formal linguists has focused on understanding where natural languages fit in the hierarchy and hence, which formal grammars have the proper generative power to handle natural language structures. Similarly, when assuming a logical perspective the main question to address is which logic has the right expressivity to reason on natural language structures avoiding both under-generation and over-generation problems. In this perspective, the first question to ask is "which operators do we need to reason on natural language resources?". Since Frege, the idea of interpreting words as functions and their composition as function application has been broadly accepted in Formal Semantics. Hence, implication ($\rightarrow$) is an essential connective to have in our logical language. Yet, it remains to be understood which properties it should have, which other operators we would need and how they relate to each other.

A well known approach to our second goal (b) is the assumption of a Logical Form (Chomsky, 1976): the surface (syntactic) structure is input to a further set of rules which derive the Logical Form structure of the sentence. This latter structure is then interpreted by a semantic mechanism to obtain the meaning representation. Another well known solution is Montague's (Montague, 1974) where syntax and semantics goes in parallel: for each syntactic rule there is a corresponding semantic rule that builds the meaning representation of the phrase analyzed syntactically. The correspondence is postulated for each rule. This step-by-step procedure in some intermediate steps, e.g. in the quantification step, may build meaning representations also for structures which are not syntactic constituents. Our approach is similar to Montague's in treating syntax and semantics in parallel, but it differs from it into two ways. A first important difference with Montague's approach is that the correspondence between syntactic and semantic rules is defined at a general level: following (van Benthem, 1988), we exploit the correspondence introduced by Curry and Howard (Howard, 1980) between logical rules and lambda calculus; the proof of the derivation $A_1, \ldots, A_n \vdash B$ can be read as an instruction for the assembly of proof-term M with input parameters $x_1, \ldots, x_n$. In the "parsing as deduction" approach this turns out to mean that *labelled derivations* provide also the schema to build the meaning representation of the structure under analysis:

$$x_1 : \underbrace{A_1}_{w_1} \cdot \cdots \cdot x_n : \underbrace{A_n}_{w_n} \vdash M : B \tag{6}$$

It's important to emphasize that the correspondence holds between syntactic structures and semantic labels (proof terms). Recall the observation above that syntactic structures are labels for the set of expressions that belong to that structure. Once we instantiate the structure with a particular expression, we can also replace the semantic labels with the meaning representation of the words in the chosen expression. A second difference concerns the labelling of nodes: in LG only the syntactic constituents receive a semantic label.

Finally, there can be more than one way to prove that an entailment holds: different (normalized) derivations correspond to different readings of the same string of words. A given string can receive different structures and different proof terms, but also different proof terms for the same structure. In this paper, we are interested in the latter case (see Section 4). For reason of space, we won't discuss how a proof term is built by labelled derivations; however, since the linguistic phenomenon, we are interested in, lays on the interface of syntax and semantics of natural language, we introduce those concepts of Formal Semantics that are going to play a role in our analysis of quantifier phrases. The interested reader is referred to (van Benthem, 1987) for an introduction to the use of labelled derivations for building meaning representation of parsed strings, and to (Bernardi & Moortgat, 2007, 2009) for the discussion of the Curry-Howard correspondence for the logic presented in this paper with $\bar{\lambda}\mu\tilde{\mu}$-calculus and its translation into λ-terms.

2.1 *Syntax-Semantics interface in Natural Language*

In order to illustrate how words are interpreted in Formal Semantics, let us consider the scenario below. We are in a school and the following people are present: *Lori, Alex, Sara* and *Pim*. Assume that the first three are students whereas *Pim* is a teacher. They all sing, *Alex* is the only one who trusts himself, *Lori* knows every student, and he left. This is easily expressed set theoretically. Our domain of entities is

$E = \{\texttt{lori}, \texttt{alex}, \texttt{sara}, \texttt{pim}\}$; let $[\![\texttt{w}]\!]$ indicate the interpretation of *w*.

(7)
$$\begin{array}{lcl}
[\![\texttt{lori}]\!] & = & \text{lori;} \\
[\![\texttt{alex}]\!] & = & \text{alex;} \\
[\![\texttt{sara}]\!] & = & \text{sara;} \\
[\![\texttt{pim}]\!] & = & \text{pim;} \\
[\![\texttt{student}]\!] & = & \{\text{alex, sara, lori}\}; \\
[\![\texttt{teacher}]\!] & = & \{\text{pim}\}; \\
[\![\texttt{sing}]\!] & = & \{\text{alex, sara, lori, pim}\}; \\
[\![\texttt{trust}]\!] & = & \{\langle\text{alex, alex}\rangle\}; \\
[\![\texttt{know}]\!] & = & \{\langle\text{lori, alex}\rangle, \langle\text{lori, sara}\rangle\langle\text{lori, lori}\rangle\}; \\
[\![\texttt{left}]\!] & = & \{\text{lori}\}; \\
[\![\texttt{every student}]\!] & = & \{Y \subseteq E \mid [\![\texttt{student}]\!] \subseteq Y\}.
\end{array}$$

This amounts to saying 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. The case of *every student* is more interesting: it denotes the set of properties that every student has.

(8)
$$\begin{array}{lcl}
[\![\texttt{every student}]\!] = \{Y \subseteq E \mid [\![\texttt{student}]\!] \subseteq Y\} & = & \{\{\text{alex, sara, lori}\}, \{\text{alex, sara, lori, pim}\}\} \\
 & = & \{[\![\texttt{student}]\!], [\![\texttt{sing}]\!]\}
\end{array}$$

Alternatively, the words above can be seen as functions from the elements of their set to truth values. For instance, an intransitive verb like *left* can be seen as a function that, given an entity e as argument, will return a truth value t (`left(lori)` = true, `left(alex)` = false). The standard way of representing functions is by means of lambda terms: each word can be assigned a lambda term representing its meaning. For instance, a transitive verb like, *knows* or *trust*, is represented as a function with two arguments (or a function taking the ordered pair of the element in the sets above). To see expressions as functions allows to consider natural language expressions as built by means of function application. Richard Montague introduced the idea that the step that put together the syntactic structure are associated with the ones that give instructions for assembling meaning representations: the syntactic and semantic rules in his grammar work in parallel. Here, we introduce those concepts of Montague Grammar that are essential for the understanding of our proposal, namely Categorial Grammar and the connection between syntax and semantics.

Syntax-Semantics In Categorial Grammar (CG) syntactic categories can be atomic or complex. The latter are assigned to those expressions that are interpreted as functions and are designed to distinguish whether the argument A must occur to the left ($A\backslash B$) or the right (B/A) of the function. Hence, function application comes in two versions, a Forward Application (FA) and a Backward Application (BA):

(9)
$$\text{(FA)}\quad \begin{array}{c} B \\ \hline B/A \quad A \end{array} \qquad\qquad \text{(BA)}\quad \begin{array}{c} B \\ \hline A \quad A\backslash B \end{array}$$

The possibility to express whether the argument must occur on the left or the right of the function is important, for instance, in the case of the two arguments taken by the transitive verb: In English the object occurs on its right and the subject on its left. Having in mind the observation made above, viz. that a category is a name for the set of expressions that belong to it, we can look at function application in more precise terms. Given the categories A, B, C and D such that $C \subseteq A$ and $B \subseteq D$, i.e. every expression in C is also in A and every expression in B is also in D, then if a functor category combines with elements of A as an argument (i.e. it's either $A\backslash\cdot$ or $\cdot/A$), it also combines with elements of C; the result of the application is an expression that belongs to the set B and hence to any D of which B is a subset. We will come back to this observation when looking at the dual Lambek Calculus in Section 3.

(10)
$$\text{(FA)}\quad \begin{array}{c} B \subseteq D \\ \hline B/A \quad C \subseteq A \end{array} \qquad\qquad \text{(BA)}\quad \begin{array}{c} B \subseteq D \\ \hline C \subseteq A \quad A\backslash B \end{array}$$

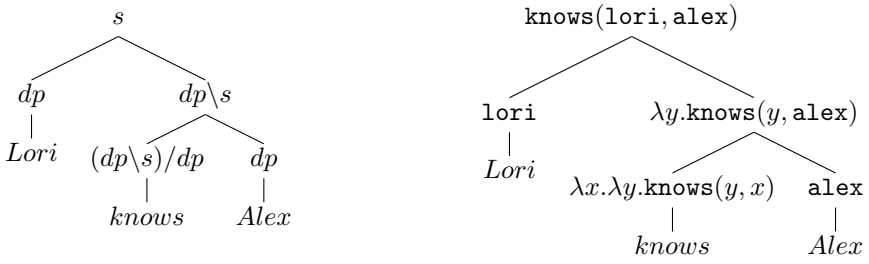
word	**category**	**type**	**λ-term**	**meaning**
alex	dp	e	$\texttt{alex}$	alex
left	$dp\backslash s$	$e \to t$	$\lambda x_e.(\textsf{left}\ x)_t$	{lori}
trusts	$(dp\backslash s)/dp$	$e \to (e \to t)$	$\lambda x_e.(\lambda y_e.((\textsf{trusts}\ x)\ y)_t)$	{⟨alex, alex⟩}
student	n	$e \to t$	$\lambda x_e.(\textsf{student}\ x)_t$	{alex, sara, lori}
every student	$(s/dp)\backslash s$	$(e \to t) \to t$	$\lambda Y.\forall\ \lambda x.((\textsf{student}\ x) \Rightarrow (Y\ x))$	$\{Y \subseteq E \mid [\![\texttt{student}]\!] \subseteq Y\}$

Table 1: Syntactic categories and Semantic types

The whole picture of our little lexicon is summarized in Table 1, where for each word is given the syntactic categorial category, its corresponding semantic type as well as the lambda term of that type that represents the set theoretical meaning of the word given in the last column of the table. We indicate only one representative for each category.[1], [2]

We can now look at an example highlighting the syntax-semantics link captured by CG and its connection with the λ-calculus. The syntactic tree (tree on the left) is built by (FA) and (BA) and each step in it has a one-to-one correspondence with the λ-rules applied to assemble the semantic representation (the tree on the right).

(11)



2.2 *The Mathematics of Sentence Structure (Lambek '58 and '61)*

Lambek (Lambek, 1958) showed that the deduction theorem, that is the backbone of Logic, is also at the heart of linguistic structure composition. With this application in mind, the theorem says that if a linguistic structure Γ composed with another constituent of syntactic category A is of category B then Γ is of category B given A, or alternatively, it is a B missing a A

(12) $$\text{If}\quad A, \Gamma \vdash B\ \text{ then }\ \Gamma \vdash A \to B$$

For instance, if *Lori knows the student* is a sentence then *knows the student* is a sentence missing a determiner phrase –given that *Lori* is of the latter category (dp).

(13) $$\text{If}\quad \text{Lori}_{dp}, [\text{knows the student}] \vdash s\ \text{ then }\ [\text{knows the student}] \vdash dp \to s$$

The connection above between the comma and the implication is known in algebra as residuation principle and it's an "iff". This principle is also behind the more familiar mathematical operators of multiplication ($\times$) and division ($-$):

(14) $$x \times y \leq z\ \text{ iff }\ y \leq \tfrac{z}{x}$$

The reader can compare the similar behaviour shown by the $\times$ and $-$ with respect to the $\leq$ with the one of the , and $\to$ with the $\vdash$. To learn from this comparison, one can observe that the multiplication is characterized by some well known properties like commutativity, associativity and distributivity over the $+$:

(15) $$x \times y = y \times x \quad (x \times y) \times z = x \times (y \times z) \quad (x + y) \times z = (x \times z) + (x \times y)$$

Similarly, we are interested in knowing which are the properties of the "comma" above when the concatenated elements are words. In the approach we follow, the "comma" is taken to be not-commutative and not-associative to avoid over-generation problems. (For other choices, see (Steedman, 2000), among others).

As the comma's behavior varies, so does the conditional's. In particular, when rejecting commutativity of the comma the implication splits into left implication ($A\backslash B$, "if A on the left then B") and right implication (B/A, "B if A on the right"), since the order of occurrences of the formulas matters. Hence, the residuation principle becomes:

(16)

$$(a1)\ \ A \otimes C \vdash B \ \text{ iff } \ C \vdash A\backslash B$$
$$(a2)\ \ C \otimes A \vdash B \ \text{ iff } \ C \vdash B/A$$

The difference with the comma is emphasised by using the $\otimes$ operator in its place.

NL (Non-associative Lambek calculus), introduced in (Lambek, 1961), is the pure calculus of residuation since it is characterized by only these two rules (besides transitivity, viz., if $A \vdash B$ and $B \vdash C$, then $A \vdash C$, and reflexivity, viz. $A \vdash A$, of the $\vdash$).

As we have observed when introducing CG, distinguishing the left and right implication turns out to be relevant when formulas stand for syntactic categories of words. By applying the residuation principle we obtain the category of a verb phrase as well as the one of a transitive verb.

(17)

$$(a1)\ \ \text{Lori}_{dp} \otimes [\text{knows the student}] \vdash s \ \text{ iff } \ [\text{knows the student}] \vdash dp\backslash s$$
$$(a2)\ \ [\text{knows}]_{tv} \otimes [\text{the student}]_{dp} \vdash dp\backslash s \ \text{ iff } \ [\text{knows}]_{tv} \vdash (dp\backslash s)/dp$$

Furthermore, the object of our toy example is a dp consisting of a noun (*student*) that is assigned the atomic category n and an article (*the*). Its category is determined in the, by now, usual way:

$$[\text{the}] \otimes [\text{student}]_n \vdash dp \ \text{ iff } \ [\text{the}] \vdash dp/n$$

Forward and Backward function applications are theorems of this calculus:

(18) $$B/A \otimes A \vdash B \ \text{ and } \ A \otimes A\backslash B \vdash B$$

The Lambek Calculus is the logic of the *merging* relation represented by the $\otimes$ (See (Vermaat, 2005) for a detailed comparison of this relation with the Merge relation of the Minimalist Framework):

(19) If there are two expressions $e_1 \in B/A, e_2 \in A$ s.t. $\texttt{Merge}(e_1, e_2, e_3)$, i.e. e_3 is the result of merging e_1 and e_2, then by definition of $\otimes$, $e_3 \in B/A \otimes A$, and, hence $e_3 \in B$, since $B/A \otimes A \vdash B$. An expression e_1 belongs to a functor category B/A if *for all* expressions e_2, e_3 such that $\texttt{Merge}(e_1, e_2, e_3)$ and $e_2 \in A$, then $e_3 \in B$.

2.3 *"Parsing as Deduction": Sequent Calculus*

A logic can be presented in several proof-system styles. Natural Deduction is the most well known one among linguists familiar with Logic. Other systems are Gentzen Sequent Calculus, Tableaux, Proof nets. Here we will use Sequent Calculus since it is the most suitable for introducing the concept of dual proofs and symmetric calculus that are at the heart of our proposal. It is a "decision procedure" and we use it to determine whether a phrase is a well-formed expression of a given category (and to assign it a meaning representation in the process of checking this.)

Gentzen Sequents Sequent Calculus is a proof system introduced by Gentzen to reason on structures. A *sequent* has the form

(20) $$\Sigma \vdash \Gamma$$

where both Σ and Γ are sequences of logical formulas (i.e., both the number and the order of the occurring formula matter). The symbol $\vdash$ is usually referred to as turnstile and is often read, suggestively, as "yields" or "proves"; it is not a symbol in the language, rather it is a symbol in the metalanguage used to discuss proofs. In the sequent above, Σ is called the antecedent and Γ is said to be the succedent of the sequent. A sequent derivation starts from the sequent to be proved and by application of the inference rules reduces it into simpler sequents (roughly, a sequent with a lower number of operators). In other words, inference rules are precise instructions to build the derivation of a given sequent. The choice of which inference rule to apply is dictated by the main operators of the formulas in the sequent. As we will show later, in same cases more choices are possible. A derivation is concluded when axiom links are reached.

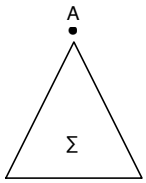
The properties of the calculus determines the actual inference rules which are written with a list of sequents above (premises) and below a line (conclusion). Each operator has an inference rule that eliminates it from the left of the turnstile and an inference rule that eliminates it from the right.[3] For instance, the logical rule that eliminates $\wedge$ of propositional logic from the left of the turnstile is represented as following

$$\frac{\Gamma[A] \vdash \Delta}{\Gamma[A \wedge B] \vdash \Delta} \ (\wedge L) \tag{21}$$

The rule says: the structure Γ containing the formula $A \wedge B$ yields the structure Δ, if Γ containing A can be proved to yield Δ, i.e. the problem of proving the sequent in the conclusion is reduced to a simpler problem by eliminating the $\wedge$.

Non-associative Lambek Calculus In the case of the Non-associative Lambek calculus (NL) the property encoded into the inference rules is the residuation principle discussed above. Here we do not go into the details of how this principle is captured by the inference rules of the Sequent proof system, and refer the reader interested in the issue to (Areces & Bernardi, 2004). Since in NL the sequence can be seen as tree-structure, to help understanding the austere sequent notation for each inference rule of the calculus we give a tree representation as well. In NL the succedent of sequents is not a structure but a formula:

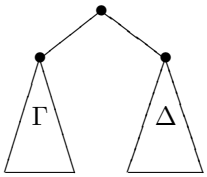
(22)



$\Sigma \vdash A$

This statement expresses the judgement that the tree structure Σ is of category A. Thinking of categories as labels for sets of expressions, the sequent can be read as a proof of the fact that every expression in Σ is also in A. We will thus take the liberty of saying that the sequent proves that the structure Σ is of category A.

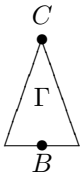
Recall, that trees are bracketed strings of formulas. The smallest tree is just a single formula A. Composite trees are formed by taking two trees Γ and Δ together as immediate daughters of a binary branching structure. In the sequent notation, we write (Γ, Δ). Graphically,



Axiom, Cut: How can we arrive at a judgement $\Gamma \vdash A$? In the case where Γ is just the single node tree consisting of the formula A, we are done. In the language of the sequent calculus, we have the axiom $A \vdash A$ (i.e. it simply expresses the reflexivity of the $\vdash$). We will graphically represent this case as

where we write the formula at the left-hand side of $\vdash$ below the node $\bullet$, and the formula at the right-hand side above it.

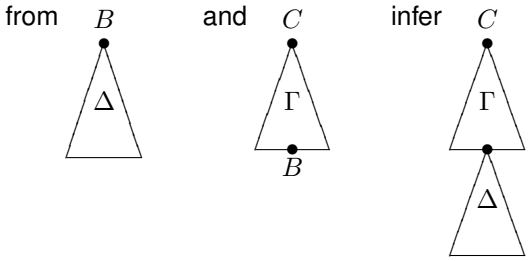
The axiom case deals with the simplest possible trees — trees consisting of one single formula. Next we need rules to reason about composite trees. Consider first the situation where we have a tree Γ which has among its leaves a formula B. Suppose we have arrived at the judgment that this tree is of category C. We can graphically depict this situation as



In the language of sequent calculus we write $\Gamma[B] \vdash C$. The square bracket notation $\Gamma[\Gamma']$ is used for picking out a distinguished sub-tree Γ' in a tree Γ. In this particular case, the sub-tree Γ' is the single-node tree B.

Suppose now that we also have a second tree Δ which we can show to be of category B. In that case, substituting the tree Δ for the node B in Γ will produce a tree that is still of category C. In other words: in a well-formed phrase of category C, we can replace any of its constituents by another constituent, as long as the replacement and the original constituent has the same category.

This grammaticality-preserving substitution yields the inference rule below, which is known as the Cut rule.

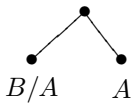


In the concise sequent notation, we write[4]

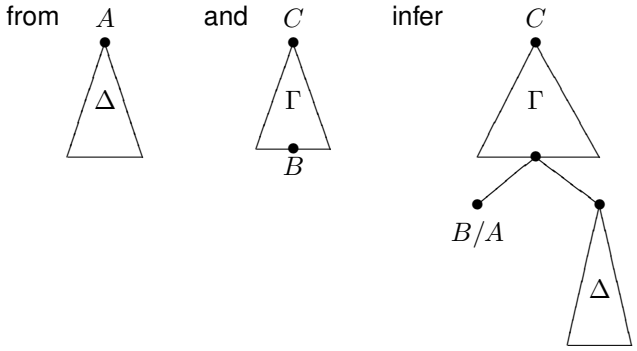
$$\frac{\Delta \vdash B \quad \Gamma[B] \vdash C}{\Gamma[\Delta] \vdash C}\ \text{(Cut)}$$

Logical rules: The Axiom and the Cut rule do not mention particular category-forming operations: they hold in full generality for any formula. Let us turn then to the inference rules for the category-forming operations. For each of the connectives of our grammar logic, $/, \otimes, \backslash$, we need a rule that tells us how to deal with it when we find it right of the derivability sign $\vdash$, or left of it. The first kind of rules are called *rules of proof*: they tell us how to arrive at a judgement $\Gamma \vdash B$ in the case where B is a complex formula: B/A, $A\backslash B$, or $A \otimes B$. The rules for the connectives left of $\vdash$ are called *rules of use*: they tell us how we can put the tree Γ together in accordance with the interpretation of the connectives $/, \otimes, \backslash$.

Recall the substitution rule (Cut) that allows us to replace any leaf formula B by a tree structure that can be shown to be of category B. One possibility of doing that, in a way that introduces a slash formula B/A, would be to replace the formula B by the tree below, putting together B/A with a formula A to its right — the familiar application schema.



More generally (and using our substitution reasoning again), we obtain a tree of category B by combining B/A with any sister tree Δ on its right, provided we can show Δ is of category A. Putting these steps together, we obtain the $(/L)$ inference rule, introducing $/$ in the antecedent tree structure.



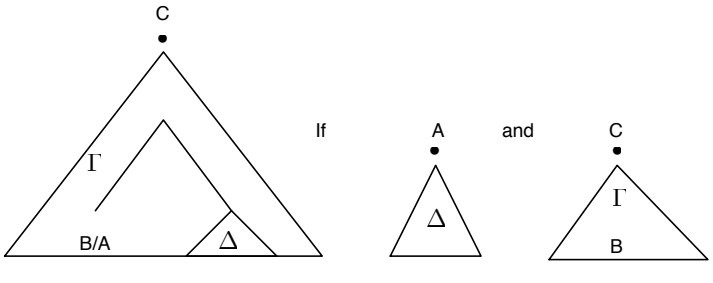
In sequent notation, we have:

$$\dfrac{\Delta \vdash A \qquad \dfrac{\dfrac{A \vdash A \quad B \vdash B}{B/A \otimes A \vdash B}\,(/\mathrm{L}) \qquad \Gamma[B] \vdash C}{\Gamma[B/A \otimes A] \vdash C}\,(\mathrm{Cut})}{\Gamma[B/A \otimes \Delta] \vdash C}\,(\mathrm{Cut})$$

In other words, the function application rule can be applied within a tree and its compact formulation is as below.

(23)

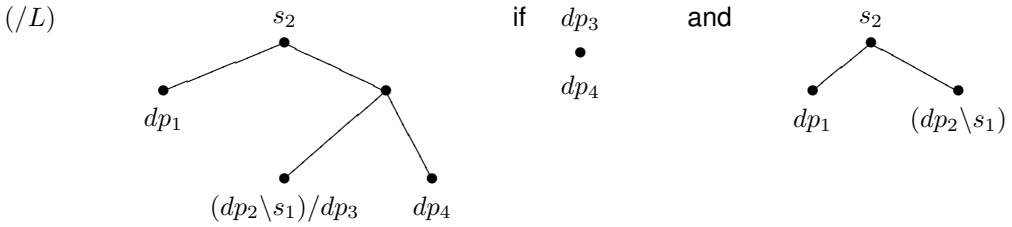
$$\dfrac{\Delta \vdash A \quad \Gamma[B] \vdash C}{\Gamma[B/A \otimes \Delta] \vdash C}\,(/L)$$



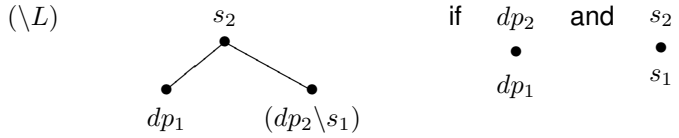
The elimination of the $/$ from the left of the turnstile, $(/L)$, is the application of a function of category B/A to a structure Δ, i.e. the sub-structure $B/A \otimes \Delta$ is proved to be of category B, if Δ can be proved to be of category A. The application happens within a bigger structure Γ. Therefore, the sequent is reduced into two simpler sequents. In particular, in the premise in the right branch the sub-structure $B/A \otimes \Delta$ is replaced by its category B. In other words, the sequent in the conclusion holds if it can be proved that Δ yields A and the structure Γ containing B yields C. Notice that if Γ is a structure consisting only of B/A and Δ, then the right branch is $B \vdash C$. Similarly for the $A\backslash B$ case.

Consider the sentence *Lori knows Alex*. We have seen that the words in this sentence can be assigned the category dp, $(dp\backslash s)/dp$ (transitive verb) and dp, respectively. We want to show that we obtain a tree of category s if we put together these words with the constituent structure '[Lori [knows Alex]]'. The inference steps that lead to that conclusion are given below. For ease of reference, we number the atomic formulas. The first step is to decompose our original problem into two smaller sub-problems by applying

the $(/L)$ rule. We check, in other words, whether the transitive verb $(dp_2\backslash s_1)/dp_3$ indeed has a tree of category dp as its right sister. The atomic tree dp_4 for the direct object *Alex* qualifies. What remains to be shown is that the subject verb-phrase combination $(dp_1, (dp_2\backslash s_1))$ is indeed a tree of category s.



This is shown by applying the $(\backslash L)$ rule.

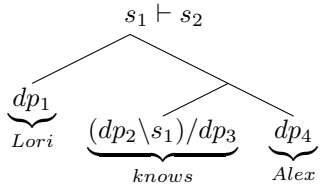


This rather laborious graphical account of the derivation can be written down more compactly in the sequent notation:

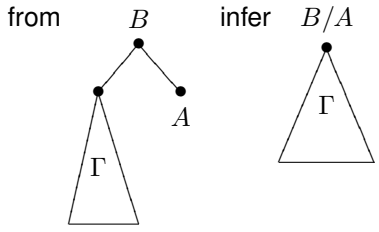
$$\dfrac{\dfrac{dp_4 \vdash dp_3 \quad \dfrac{dp_1 \vdash dp_2 \quad s_1 \vdash s_2}{(dp_1, (dp_2\backslash s_1)) \vdash s_2}\,(\backslash L)}{(\underbrace{dp_1}_{\text{[Lori}}, (\underbrace{((dp_2\backslash s_1)/dp_3)}_{\text{[knows}}, \underbrace{dp_4}_{\text{Alex]]}})) \vdash s_2}\,(/L)$$

The $(/L)$ and $(\backslash L)$ rules provide instructions on how to use formulas B/A, $A\backslash B$ to build tree structures. In sum, by application of inference rules, i.e. by structure simplification, we have reduced the whole structure $dp_1 \otimes ((dp_2\backslash s_1)/dp_3 \otimes dp_4)$ to s_1 which matches the top formula s_2 ($s_1 \vdash s_2$). In the tree we represent this match between the s-categories since it will play a role in the description of the QP analysis:

(24)



Let us turn then to the rules that tell us how we can decide that a tree structure Γ is of category B/A or $A\backslash B$. Given the interpretation of the slashes, this will be the case if Γ together with a sister node A to its right or left, can be shown to be a tree of category B. For formulas B/A, we obtain the $(/R)$ inference rule below.



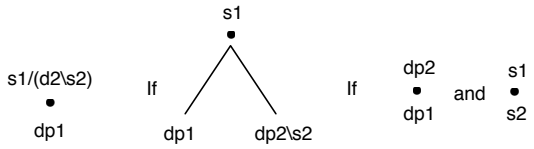
The sequent notation is given below, together with the $(\backslash R)$ rule.

$$\frac{(\Gamma, A) \vdash B}{\Gamma \vdash B/A}\ (/R) \qquad \frac{(A, \Gamma) \vdash B}{\Gamma \vdash A\backslash B}\ (\backslash R)$$

An interesting use of the right rules is the theorem $A \vdash C/(A\backslash C)$, an instance of which is the type lifting rule known to linguists, viz. from e to $(e \to t) \to t$. Any word of category dp is proved to be of category $s/(dp\backslash s)$. In other words, any expression that is in the category of determiner phrases is also in the category of quantifier phrases; any expression that is interpreted as an entity can be interpreted as the set of its properties (compare Table 1).

(25)

$$\frac{\dfrac{dp_1 \vdash dp_2 \quad s_2 \vdash s_1}{dp_1 \otimes dp_2\backslash s_2 \vdash s_1}\ (\backslash\mathrm{L})}{dp_1 \vdash s_1/(dp_2\backslash s_2)}\ (/\mathrm{R})$$



Since in the sequent in the conclusion there is only one main operator, $/$, and it occurs on the right of the turnstile, the only rule that can be applied is (/R), hence the sequent is reduced to a simpler sequent (a sequent with less implication operators), viz. $dp_1 \otimes dp_2\backslash s_2 \vdash s_1$. This sequent holds if $dp_1 \vdash dp_2$ and $s_2 \vdash s_1$, since both are axioms we are done. Notice that $s/(dp\backslash s) \nvdash dp$, as one would expected by thinking of the domains of interpretation of the corresponding functions, viz. it is not true that whatever can be interpreted as a set of properties can be interpreted as an entity.

This theorem is an essential part of the proofs of "Argument Lowering", $B/(C/(A\backslash C)) \vdash B/A$, and "Value Raising", $B/A \vdash (C/(B\backslash C))/A$, used by Hendriks (Hendriks, 1993) to account for scope ambiguities. We give the sequent derivation, the reader is invited to translate them into a tree notation.

(26)

$$\frac{\dfrac{\dfrac{\vdots}{A \vdash C/(A\backslash C) \quad B \vdash B}\ (/L)}{B/(C/(A\backslash C)) \otimes A \vdash B}\ (/R)}{B/(C/(A\backslash C)) \vdash B/A} \qquad \frac{\dfrac{\dfrac{\vdots}{A \vdash A \quad B \vdash C/(B\backslash C)}\ (/L)}{B/A \otimes A \vdash (C/(B\backslash C))}\ (/R)}{B/A \vdash (C/(B\backslash C))/A}$$

Hendriks uses also a third rule called "Argument Raising": this rule is not a theorem of the Logic, since as we have mentioned above $C/(A\backslash C)$ does not derive A and hence the proof fails to reach all axioms.[5] (See (Bastenhof, 2007) for a detailed discussion on Argument Raising.)

$$\frac{\dfrac{\dfrac{\text{FAILED}}{\vdots}}{C/(A\backslash C) \vdash A \quad A \vdash A}\ (/L)}{B/A \otimes C/(A\backslash C) \vdash B}\ (/R)}{B/A \vdash B/(C/(A\backslash C))}$$

Bottom-up vs. Top-down parsing strategies Let us now consider the sentence *The student who knows Sara left*. The relative pronoun *who* works as a bridge between a sentence missing a dp in subject position, *knows Sara*, and the noun *student* that the relative clause modifies, viz. *who* $\in (n\backslash n)/(dp\backslash s)$ as we can compute by means of the residuation principle:

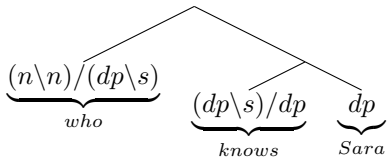
(27)

$$[\text{who}]_{rel} \otimes [\text{knows Sara}]_{dp\backslash s} \vdash n\backslash n \ \text{ iff } \ [\text{who}]_{rel} \vdash (n\backslash n)/(dp\backslash s)$$

This example will help us bringing in another important observation regarding parsing strategies. In the examples seen so far, at each step there was only one possible inference rule that could have been applied. However, it can happen that a sequent could be proved in more than one way. For instance, it contains more than one complex formula either in the antecedent or the succedent position. One can choose to follow either a (i) bottom-up or a (ii) top-down strategy as we will illustrate with the example below.

The problem to solve is to check whether the tree below is of category $(n\backslash n)$.

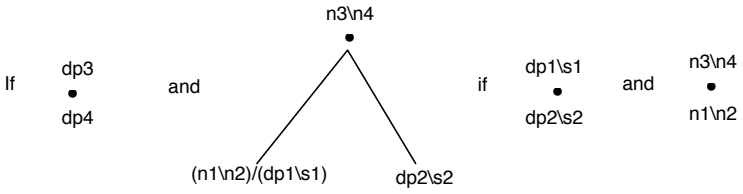
(28)



At each inference step, we follow a bottom-up strategies: We start from the bottom of the tree. In the leaves, there are two complex formulas that could be activated, hence there are two possible inference rules to apply as a first step: we could simplify the sequent either (i) by applying the function $(dp_2\backslash s_2)/dp_3$ to dp_4 or (ii) by applying the function $(n\backslash n)/(dp\backslash s)$ to the sub-structure $((dp_2\backslash s)/dp_3 \otimes dp_4)$. In a bottom-up approach the choice is guided by the atomic categories in the leaves, hence, in this case by dp_4. The function to activate first is, therefore, the one of the transitive verb whose argument matches the atomic leave. The other inference steps are applied using these same criteria. In the tree notation, we leave the last two trees unfolded. The reader can check by herself the last steps that lead to the axiom links indicated in the sequent.

(29) Bottom-up strategy:

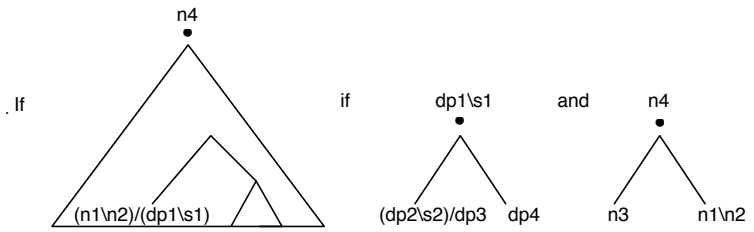
$$\dfrac{\dfrac{dp_4 \vdash dp_3 \qquad \dfrac{\dfrac{dp_1 \vdash dp_2 \quad s_2 \vdash s_1}{dp_1 \otimes dp_2\backslash s_2 \vdash s_1}\,(\backslash L)}{dp_2\backslash s_2 \vdash dp_1\backslash s_1}\,(\backslash R) \qquad \dfrac{\dfrac{n_3 \vdash n_1 \quad n_2 \vdash n_4}{n_3 \otimes n_1\backslash n_2 \vdash n_4}\,(\backslash R)}{n_1\backslash n_2 \vdash n_3\backslash n_4}\,(\backslash L)}{(n_1\backslash n_2)/(dp_1\backslash s_1) \otimes dp_2\backslash s_2 \vdash n_3\backslash n_4}\,(/L)}{\underbrace{(n_1\backslash n_2)/(dp_1\backslash s_1)}_{[who} \otimes (\underbrace{(dp_2\backslash s_2)/dp_3}_{[knows} \otimes \underbrace{dp_4}_{Sara]]}) \vdash n_3\backslash n_4}\,(/L)$$



The top-down strategy is illustrated below both in the sequent and in the tree notation. The starting point is the top-formula $n_3\backslash n_4$, that can be simplified by $(\backslash R)$. The next step is the inference rules whose result matches the new top formula, i.e. n_4. Hence, the function activated first is the category of *who*. Again we leave to the reader the unfolding of the last trees into the axiom links.

(30) Top-down strategy:

$$\dfrac{\dfrac{\dfrac{dp_4 \vdash dp_3 \qquad \dfrac{\dfrac{dp_1 \vdash dp_2 \quad s_2 \vdash s_1}{dp_1 \otimes dp_2\backslash s_2 \vdash s_1}\,(\backslash L)}{dp_2\backslash s_2 \vdash dp_1\backslash s_1}\,(\backslash R)}{(dp_2\backslash s_2)/dp_3 \otimes dp_4 \vdash dp_1\backslash s_1}\,(/\mathrm{L}) \qquad \dfrac{n_3 \vdash n_1 \quad n_2 \vdash n_4}{n_3 \otimes n_1\backslash n_2 \vdash n_4}\,(\backslash R)}{n_3 \otimes ((n_1\backslash n_2)/(dp_1\backslash s_1) \otimes ((dp_2\backslash s_2)/dp_3 \otimes dp_4)) \vdash n_4}\,(/\mathrm{L})}{\underbrace{(n_1\backslash n_2)/(dp_1\backslash s_1)}_{[who} \otimes (\underbrace{(dp_2\backslash s_2)/dp_3}_{[knows} \otimes \underbrace{dp_4}_{Sara]]}) \vdash n_3\backslash n_4}\,(\backslash R)$$



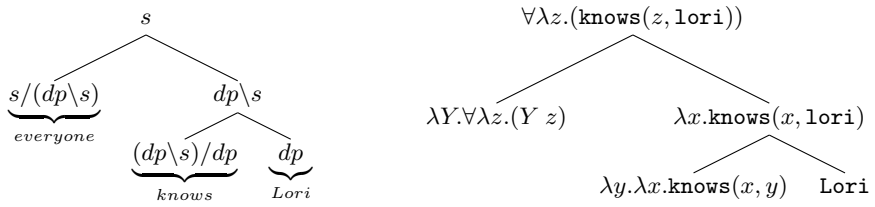
The first toy example shows that NL captures the composition principle involved in kernel sentences —sentences consisting of local dependencies only (dependencies among juxtaposed constituents). It also illustrates how the concepts of dependency and constituent, so important when looking at natural language, are encoded in the derivation. The dependency between the verb and its arguments are captured by the axiom links involving the determiner phrases, on the other hand the brackets put around the structure at each step in the derivation build the constituents out of the single words. The last example shows that NL also properly handles extraction when it happens from a peripheral (accessible) position.

2.4 *Lambek Calculus limitations*

Since in NL function application can happen only on juxtaposed structures, abstraction only from peripheral position, and there is no way of re-bracketing the tree once built. As a consequence, the system fails to recognize long distance dependencies and cross-dependencies, and does not capture the behavior of in-situ binders, like natural language quantifiers. Let's look at their case now.

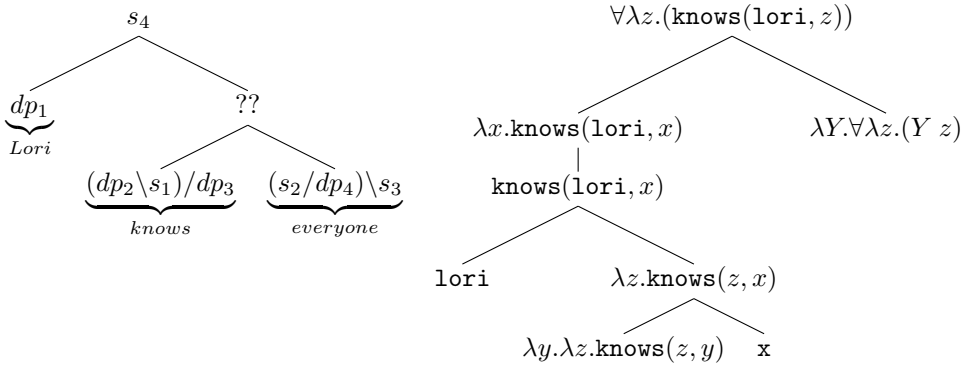
An important concept to introduce regards the notion of "scope" of quantifiers. Like in logic, the scope of a quantifier is the part of the expression on which it performs its action; the quantifier binds the variable x of category dp in its scope, e.g. the action of $\forall$ below is to check all the possible assignments of values to the variable x it binds.

(31)



The category that we have shown to be derivable from dp and that is mapped into the semantic type $(e \to t) \to t$ works properly in NL when in subject position. However, this category does not capture the behaviour of QP in general, e.g. it already causes a failure of the proof when assigned to a QP in object position, as marked by the question marks in the syntactic tree.

(32)



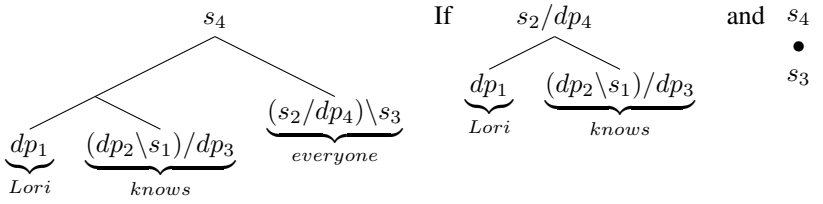
By comparing the relation between the syntax and the semantic trees of the first example with the one of this second example, one can see that quantifier may cause a mismatch between the two trees (in the syntactic tree the QP is in an embedded position, whereas in the semantic tree it is in a peripheral position.)

A solution would be to enrich CG with a way to reproduce in the syntactic tree the abstraction used in the semantic one. This solution, however, would cause the generation of a syntactic tree that does not reflect the syntactic constituents, i.e. we would obtain [[Lori knows] [every student]]. (See (Steedman, 2000) for interesting discussion and alternative view on the role played by syntactic constituents). This ability of natural language quantifier to take scope over the structure in which they occur, even if in an embedded position, challenges any formal account of the syntax-semantic interface of natural languages.

NL is not expressive enough to account for this behaviour of QP. Our goal is to understand which are the formal properties that characterize this behaviour.

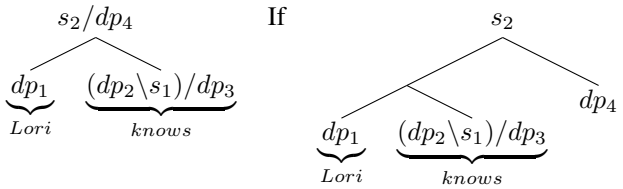
If one wants to mimic the semantic tree at the syntactic level, she could assign to QPs a second category, $(s/dp)\backslash s$ that is also mapped into $(e \to t) \to t$ and would need to be able to obtain the following tree on the left, which by function application is reduced to the two simpler trees on the right:

(33)



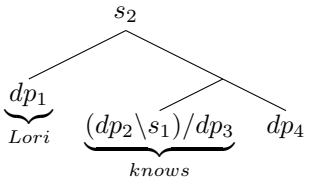
This can be proved by abstraction

(34)



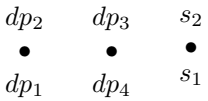
We have arrived to a tree that cannot be simplified further by means of the Lambek calculus' inference rules. One would first need to re-bracket it,

(35)



The tree obtained can then be simplified by function applications, arriving at the following matches of atomic formulas:

(36)



The re-bracketing corresponds to applying the associativity rule as shown in the sequent below where the corresponding inference step is marked by (Ass); the reader can check the correspondence of each inference rule with the above trees.

(37)

$$\dfrac{\dfrac{\dfrac{dp_1 \vdash dp_2 \quad dp_4 \vdash dp_3 \quad s_1 \vdash s_2}{\vdots\ D\ \vdots}}{\dfrac{dp_1 \otimes ((dp_2\backslash s_1)/dp_3 \otimes dp_4) \vdash s_2}{(dp_1 \otimes (dp_2\backslash s_1)/dp_3) \otimes dp_4 \vdash s_2}\ (\text{Ass})}\ (/\text{R})}{\dfrac{dp_1 \otimes (dp_2\backslash s_1)/dp_2 \vdash s_2/dp_4 \qquad s_3 \vdash s_4}{(\underbrace{dp_1}_{[\text{Lori}} \otimes (\underbrace{(dp_2\backslash s_1)/dp_2}_{[\text{knows}} \otimes \underbrace{(s_2/dp_4)\backslash s_3}_{\text{everyone}]])} \vdash s_4}\ (\backslash L)}$$

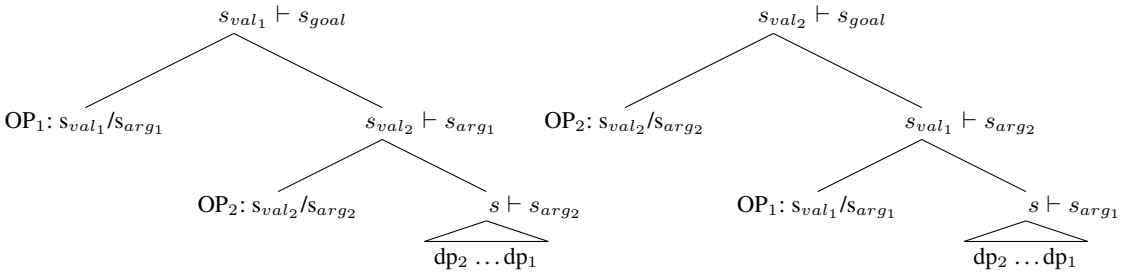
The derivation above proves that the structure [Lori [knows everyone]] belongs to s_4. However, adding associativity (Ass) to NL , i.e. going back to the Lambek's '58 paper, and using two different categories for quantifier in subject and object position is still not a general solution: the QP won't be able to jump out from any position, but only from those that can become peripheral by associativity, e.g [Lori [thinks [Alice read a book] while she was ill]] would receive only one reading.

Still it is interesting to reflect on the two examples with QP in subject and object position to grasp proof-theoretical insights on the syntax-semantic interface of natural language structures involving scope operators.

Syntax-Semantics seen from the axiom links The above example suggests that we can read the scope of the functions in the parsed structure out of the axiom links of the s-atomic formulas: the fact that *Everyone* has scope over the whole-structure (has wide scope) is captured by the axiom link $s_3 \vdash s_4$; the fact that the transitive verb is in the immediate scope of the quantifier is captured by the the axiom link $s_1 \vdash s_2$. Furthermore, the fact that the quantifier binds the variable taken as object by the transitive verb is captured by the axiom link $dp_4 \vdash dp_3$.

We can elaborate further what we have just observed by saying that a quantifier could intuitively be considered as consisting of two components tied to each other: a dp and a function that takes the sentential domain to produce a sentence. The syntactic component (dp) should stay in the embedded position where it works as argument of a verb, and the scope component should be free to travel through the tree so to reach its top level, without modifying the constituent structure. Below we focus on the scope component and represent it schematically as s_{val}/s_{arg} (compare $(s_{arg}/dp)\backslash s_{val}$ and $s_{val}/(dp\backslash s_{arg})$).[6] Generalizing, we consider a structure with two scope-operators (OP_1, OP_2) and give the two possible scoping order possibilities.

(38)



The trees are consistent with the standard idea that the later a quantifier is introduced in the semantic structure the wider its scope. In Section 4, we discuss the formal properties that characterize the function s_{val}/s_{arg} and its connection with the dp component as well as the composition relation holding between it and the structure over which it takes scope.

In short, the QP category that can be assigned using residuated operators gives raise to the following problems: (i) it requires multiple category assignments for the QP in subject and object position; it fails to produce both (ii) local and (iii) non-local scope ambiguity. We show that using dual residuated operators, QP can be assigned one category that overcomes these three problems. Reasoning with natural language structure requires some categories to *jump out* of the structure where they sit leaving, however, the structure of the other components unchanged. We show that this ability is captured by the calculus consisting of residuation, dual residuation and the distributivity principles.

3 "The Mathematics of Sentence Structure" Revised

In this section we introduce the concept of dual rules and dual proofs by looking at the dual of the residuation principle. We show how the jump required by the quantifiers is obtained by means of the distributivity principles.

The reader may recall the De Morgan's laws, i.e. rules relating pairs of dual logical operators in a systematic manner expressed in terms of negation. For instance,

(39) $$\neg(A \wedge B) = \neg A \vee \neg B$$

which in words says: Since it is false that two things together are true, at least one of them must be false. Saying that this equality holds for propositional logic means that $\neg(A \wedge B) \vdash \neg A \vee \neg B$ and $\neg A \vee \neg B \vdash \neg(A \wedge B)$ are theorems of the Logic.

In (Schröder, 1980) it is introduced the concept of dual rules, for instance, the two rules below are duals. Their duality is expressed in terms of the sequent-turnstile, the two rules are perfectly symmetric with respect to the turnstile:

(40)

$$\frac{\Gamma[A] \vdash \Delta}{\Gamma[A \wedge B] \vdash \Delta}\ (\wedge L) \qquad \text{dually} \qquad \frac{\Gamma \vdash \Delta[A]}{\Gamma \vdash \Delta[B \vee A]}\ (\vee R).$$

Similarly, one can think of dual theorems. For instance, one could express either the distributivity of the $\wedge$ over the $\vee$ or dually the one of the $\vee$ over the $\wedge$.

(41)

$A \wedge (B \vee C) \vdash (A \wedge B) \vee (A \wedge C)$ dually $(C \vee A) \wedge (B \vee A) \vdash (C \wedge B) \vee A$;

$(A \wedge B) \vee (A \wedge C) \vdash A \wedge (B \vee C)$ dually $(C \wedge B) \vee A \vdash (C \vee A) \wedge (B \vee A)$.

$\neg$ is one of the logical operators of propositional logic, whereas it is not part of the logical language of the Lambek calculus. Hence, we cannot define duality in De Morgan's style at the object-language level. However, we can still define it at the meta-level in terms of the behaviour of the operators w.r.t. the turnstile.

Residuation and Dual Residuation The logical operators of the Lambek Calculus are $(\backslash, \otimes, /)$; we introduce new operators $(\oslash, \oplus, \otimes)$ and define them as duals of the former in terms of their behaviour with respect to the $\vdash$. The dual residuation principle was studied by V. N. Grishin (Grishin, 1983) and further investigated by Lambek in (Lambek, 1993) and Rajeev Goré in (Goré, 1997).

We remind the residuation principle by copying it in (a1) and (a2). Their duals are in (b1) and (b2) respectively.

(42)

$$(a1)\ \ A \otimes C \vdash B \ \text{ iff } \ C \vdash A \backslash B \ \text{ and } \ (a2)\ \ C \otimes A \vdash B \ \text{ iff } \ C \vdash B / A$$

$$(b1)\ \ B \vdash C \oplus A \ \text{ iff } \ B \oslash A \vdash C \ \text{ and } \ (b2)\ \ B \vdash A \oplus C \ \text{ iff } \ A \otimes B \vdash C$$

In Section 2.2, while introducing the merge relation, $\otimes$, and the directional implications, $\backslash, /$, we saw that multiplication and fraction obey the residuation principle too. Similarly, there are familiar mathematical operations that obey the dual residuation principle, namely addition, $+$, and difference $-$.

$$y \leq z + x \ \text{ iff } \ y - x \leq z$$

We also discussed that while $\times$ enjoys associativity and commutativity, $\otimes$ does not and the lack of the latter causes having two functional implications $\backslash$ and $/$. Similarly, while $+$ has these two properties $\oplus$ does not and hence two "difference" operators exist $\oslash$ and $\otimes$.

Let us pause on the new type of functions we have introduced with the Dual calculus and the new relation among expressions. We have the dual function (or co-function) application:

(43)

$B/A \otimes C \vdash D$	if	$C \vdash A$	and	$B \vdash D$
dually		dually		dually
$D \vdash C \oplus (A \otimes B)$	if	$A \vdash C$	and	$D \vdash B$

The backward application and the dual backward application are in the same duality relation.

As we said, the Lambek calculus is the logic for *merging* two expressions into a new one; dually the Dual Lambek calculus is the logic for merging contexts into a new one. Hence, we will call the new relation $\texttt{Merge}^c$.

(44) If $c_3 \in B$, then $c_3 \in A \oplus (A \oslash B)$, since $B \vdash A \oplus (A \oslash B)$; then, by definition of $\oplus$, for all contexts c_1, c_2 if $\texttt{Merge}^{\texttt{c}}(c_1, c_2, c_3)$, viz. c_3 is the context resulting by the merge of the contexts c_1 and c_2, then $c_1 \in A$ or $c_2 \in A \oslash B$. A context c_2 belongs to a co-function category $A \oslash B$ if there are two contexts c_1 and c_3 such that $\texttt{Merge}^{\texttt{c}}(c_1, c_2, c_3)$, $c_1 \notin A$ and $c_3 \in B$.

Taking the dual perspective is as if we look at structures in a mirror and hence left and right are swapped around. For instance, the example used to explain residuation in Section 2.2, repeated below as (a), would be seen as in (b).

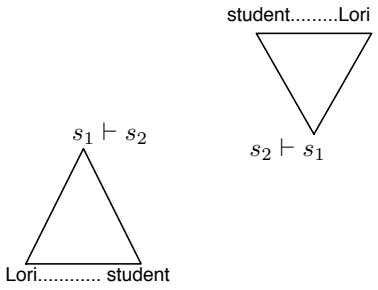
(45)

$$(a)\ \underbrace{dp}_{\text{[Lori}} \otimes \underbrace{((dp\backslash s_1)/dp \otimes (dp/n \otimes n))}_{\text{[knows [the student]]}} \vdash s_2 \text{ dually } (b) s_2 \vdash \underbrace{((n \oplus (n \oslash dp)) \oplus ((dp \oslash s_1) \oslash dp))}_{\text{[[student the] knows]}} \oplus \underbrace{dp}_{\text{Lori]}}$$

(a) by means of function applications the whole structure in the antecedent is reduced to s_1 which matches the top formula s_2. Dually (b) by means of dual function applications the whole structure on the succedent is reduced to s_1 that is matched by s_2. Notice that in this view, we are reading the entailment in (b) right-to-left, from the succedent to the antecedent. Another way to say this is that both in (a) and in (b) the "focus" is on s_2 that is built out of the other formulas. When necessary for the understanding of the explanation we will mark the left-to-right and right-to-left reading of the entailment as $\vdash_{\triangleright}$ and $\vdash_{\triangleleft}$, respectively, where the triangle points towards the focused formula.

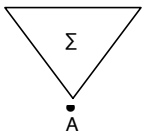
To emphasize the difference between the structure connected by $\otimes$ and the one connected by $\oplus$, in the tree notation we will represent the latter as upside down trees. The two trees could also be thought as the result of merging structures into a new one and the expansion of a structures into parts, respectively.

(46)



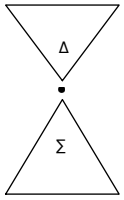
As shown by this example, the sequents of the Lambek calculus have a structure on the succedent, and a formula on the antecedent: $\Sigma \vdash A$; dually, the sequents of the Dual Lambek Calculus have the structure on the antecedent and a logical formula on the succedent of the turnstile.

(47)



$A \vdash \Sigma$

Lambek Grishin By taking together the Lambek Calculus and its Dual, we obtain sequents of the following shape:

(48) $\Sigma \vdash \Delta$ 

Since the structure in the antecedent, Σ, consists of formulas connected by $\otimes$ we call them $\otimes$-structure. Similarly, we call the structure in the succedent, Δ, $\oplus$-structure. The two logics communicate via the Cut rule that creates an unfocused entailment as illustrate by the example below. Observe that,

$$(B/A) \otimes A \vdash C \oplus (C \oslash B) \tag{49}$$

for every structure if it belongs to $(B/A) \otimes A$, then it belongs to $C \oplus (C \oslash B)$ too. Step-wise this can be explained via the communication established between the two merging relations by the Cut rule:

$$\frac{(B/A) \otimes A \vdash_{\triangleright} B \quad B \vdash_{\triangleleft} C \oplus (C \oslash B)}{(B/A) \otimes A \vdash C \oplus (C \oslash B)}\ (Cut)$$

The left premise is read left-to-right ($\triangleright$), it's governed by the relation merging two expressions into an expressions of category B whereas the right premise is read right-to-left ($\triangleleft$), it's governed by the relation that merges contexts into a new one of category B^c, hence we reach a tree of category B and a tree with an hole of category B, the former can hence be plugged into the latter reaching and unfocused entailment (marked by the absence of the triangle on the turnstile). In the sequel, we will omit the focus label since we won't go into any proof theoretical details.

The Sequent style function application and co-function application of LG are below. As the reader can see the only difference with respect to the one discussed in Section 2.2 is the presence of a structure in the succedent position, as we have just explained. The ($\oslash R$) rule is just the dual of ($/L$).

$$\frac{\Delta \vdash A \quad \Gamma[B] \vdash \Sigma}{\Gamma[B/A \otimes \Delta] \vdash \Sigma}\ (/L) \quad \text{dually} \quad \frac{A \vdash \Delta \quad \Sigma \vdash \Gamma[B]}{\Sigma \vdash \Gamma[\Delta \oplus (A \oslash B)]}\ (\oslash R) \tag{50}$$

Still, neither the Lambek calculus nor the dual Lambek calculus allows a category to jump out of an embedded position. What is still missing are the distributivity properties of the $\otimes$ and $\oplus$ over the co-implications $(\oslash, \oslash)$ and the implications $(\backslash, /)$, respectively. These properties were first studied by Grishin (Grishin, 1983). We call the system consisting of the residuation and dual residuation principles and the distributivity principles below, LG , Lambek Grishin. The distributivity rules are a form of mixed-associativity (MA) and mixed-commutativity (MC) since the former leaves the formulas in their position while changing the brackets, and the latter commutes the formulas while leaving the brackets unchanged.[7] The term Mix stands for the fact that each principle involves operators from the two families: the residuated triple $(\backslash, \otimes, /)$ and its dual $(\oslash, \oplus, \oslash)$. Below we give the principles for $\oslash$ and $/$, similar principles, modulo directionality, hold for their symmetric operators $\oslash$ and $\backslash$.

$$\begin{array}{llll} (\otimes \text{ MA}) & (B \oslash C) \otimes A \vdash B \oslash (C \otimes A) & \text{dually} & (\oplus \text{ MA}) \ (A \oplus C)/B \vdash A \oplus (C/B) \\ (\otimes \text{ MC}) & A \otimes (B \oslash C) \vdash B \oslash (A \otimes C) & \text{dually} & (\oplus \text{ MC}) \ (C \oplus A)/B \vdash (C/B) \oplus A \end{array} \tag{51}$$

These properties establish how the implication $\backslash$ and $/$ behave when occurring within a $\oplus$-structure, and similarly how the dual implications $\oslash$ and $\oslash$ behave when occurring within a $\otimes$-structure. They

are structure preserving, in the sense that they respect the non-commutativity and non-associativity of the operations they combine.

In the Sequent system, we compile the distributivity principles into the abstraction and co-abstraction inference rules.[8]

(52)

$$\frac{\Gamma \otimes A \vdash \Delta[B]}{\Gamma \vdash \Delta[B/A]}\ (/R) \quad \text{dually} \quad \frac{\Delta[B] \vdash A \oplus \Gamma}{\Delta[A \oslash B] \vdash \Gamma}\ (\oslash L)$$

Let's compare the new abstraction rules with the one given in Section 2.2: thanks to the compiled distributivity principles, abstraction can happen within a structure Δ, which is a $\oplus$-structure. Similarly, co-abstraction happens within a $\otimes$-structure, ie. the one where a quantifier is sitting in and needs to jump out from. It becomes clear already from the sequent rules how a category can now jump out from an embedded position, but we are going to look them at work in the next section.

The tree representations given in Section 2 need to be modified so to take the structure in the succedent position into account. We give the one of the abstraction rule $(/R)$ that is the most interesting to be considered:

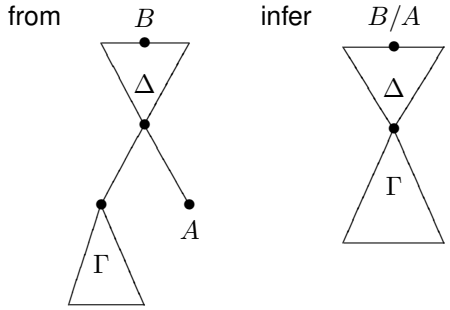


Figure 1 summarizes the Lambek Grishin calculus in the sequent notation which encodes the mathematical principles represented in Figure 2 that we claim characterize Natural Language Structure. In Figure 2, the symbol $^\sim$ marks that the principles of mix-associativity and mix-commutativity are the symmetric version of those without it, i.e. they are about $\backslash$ and $\oslash$ instead of $/$ and $\oslash$, respectively. As mentioned above, the axiom stands for the reflexivity (REF) of the $\vdash$. The cut rule encodes the transitivity (TRAN) of the $\vdash$ in the sequent notation by taking into account the fact that the formulas B can occur within structures (Γ' and Δ') and A and C can be structures, i.e. Δ and Γ, respectively. (See (Bernardi & Moortgat, 2009) for the precise sequent derivation, here we overlook some proof theoretical issues.) The abstractions $(/R, \backslash R)$ and co-abstractions $(\oslash L, \oslash L)$ encode one side of the residuation and dual residuation rule; the other side of residuation is compiled into the function and co-function applications, $(/L, \backslash L)$ and $(\oslash R, \oslash R)$. The reader is referred to (Areces & Bernardi, 2004) for a detailed explanation of the relation between the encoding of the mathematical principles into the sequent system.

4 Case study: Quantifier Phrases

As we have seen in (32) of Section 2, a quantificational expression semantically behaves as if it appeared in a different position than its actual position in the sentence: it exhibits "inverse scope effects". Because of this ability of jumping out from an embedded position, QPs cause scope ambiguity: the same syntactic structure can receive more than one interpretation. A classical example showing this effect is a sentence with quantifiers as subject and object; it may have two readings if either the subject has scope over the object (1a) or the other way around (1b), though the syntactic structure is shared (in both cases, *someone* is the subject and *everyone* is the object that together with the transitive verb forms the verb phrase). Similarly, scope ambiguity arises when a QP interacts with other scope operators like negation, intentional verbs, wh-phrases in questions, adverbs and coordination. This ability of the quantifiers to scope over the whole structure in which they are embedded can be unbounded, as illustrated by the case of the quantifiers

$$A \vdash A$$

$$\frac{\Delta \vdash \Gamma'[B] \quad B \vdash \Gamma}{\Delta \vdash \Gamma'[\Gamma]}\ (Cut1) \qquad \frac{\Delta \vdash B \quad \Delta'[B] \vdash \Gamma}{\Delta'[\Delta] \vdash \Gamma}\ (Cut2)$$

$$\frac{\Delta \vdash A \quad \Gamma[B] \vdash \Sigma}{\Gamma[B/A \otimes \Delta] \vdash \Sigma}\ (/L) \qquad \frac{A \vdash \Delta \quad \Sigma \vdash \Gamma[B]}{\Sigma \vdash \Gamma[\Delta \oplus A \oslash B]}\ (\oslash R)$$

$$\frac{\Gamma \otimes A \vdash \Delta[B]}{\Gamma \vdash \Delta[B/A]}\ (/R) \qquad \frac{\Delta[B] \vdash A \oplus \Gamma}{\Delta[A \oslash B] \vdash \Gamma}\ (\oslash L)$$

$$\frac{\Delta \vdash A \quad \Gamma[B] \vdash \Sigma}{\Gamma[\Delta \otimes A\backslash B] \vdash \Sigma}\ (\backslash L) \qquad \frac{A \vdash \Delta \quad \Sigma \vdash \Gamma[B]}{\Sigma \vdash \Gamma[(B \oslash A) \oplus \Delta]}\ (\oslash R)$$

$$\frac{A \otimes \Gamma \vdash \Delta[B]}{\Gamma \vdash \Delta[A\backslash B]}\ (\backslash R) \qquad \frac{\Delta[B] \vdash \Gamma \oplus A}{\Delta[(B \oslash A)] \vdash \Gamma}\ (\oslash L)$$

Figure 1: LG: Sequent Calculus

(REF) $A \vdash A$

(TRAN) if $A \vdash B$ and $B \vdash C$, then $A \vdash C$

(RES) $C \vdash A\backslash B$ iff $A \otimes C \vdash B$ iff $A \vdash B/C$

(DRES) $(B \oslash A) \vdash C$ iff $B \vdash C \oplus A$ iff $C \oslash B \vdash A$

($\otimes$ MA) $(B \oslash C) \otimes A \vdash B \oslash (C \otimes A)$ ($\oplus$ MA) $(A \oplus C)/B \vdash A \oplus (C/B)$

($\otimes$ MC) $A \otimes (B \oslash C) \vdash B \oslash (A \otimes C)$ ($\oplus$ MC) $(C \oplus A)/B \vdash (C/B) \oplus A$

($\otimes$ MA$^\sim$) $A \otimes (C \oslash B) \vdash (A \otimes C) \oslash B$ ($\oplus$ MA$^\sim$) $B\backslash(C \oplus A) \vdash (B\backslash C) \oplus A$

($\otimes$ MC$^\sim$) $(C \oslash B) \otimes A \vdash (C \otimes A) \oslash B$ ($\oplus$ MC$^\sim$) $B\backslash(A \oplus C \vdash A \oplus (B\backslash C)$

Figure 2: LG: properties

occurring in the complement sentence in (2) of (53). Again, the two interpretations differ in the scope relation ($>$) between *think* and *every man*, while the syntactic structure is shared.

Natural language offers also cases of bounded scope: structures that delimit the scope of the operators that cannot express their action outsides those boundaries, as illustrated by the scope possibility in (3a) that is not suitable for (3) Finally, it has been shown (Szabolcsi, 1997) that quantifiers differ in their scope behaviour as exemplified in (4).

(53)

(1) [Someone [knows everyone]$_{vp}$]$_s$

a. There exists someone that everyone knows [someone $>$ everyone]

b. For everyone there is someone that he knows [everyone $>$ someone]

(2) [Lori$_{dp}$ [thinks [every man is immortal]$_s$]$_{vp}$]$_s$

a. Lori thinks that being immortal is characteristic of men. [thinks $>$ every man]

b. Lori thinks of every actual man that he is immortal. [every man $>$ think]

(3) Two politicians spy on [someone from every city]

a. *every city $>$ two politicians $>$ someone

(4) Lori [didn't [read QP]]

a. Lori didn't read a book [Not $>$ A], [A $>$ Not]

b. Lori didn't read every book [Not $>$ Every], [*Every $>$ Not]

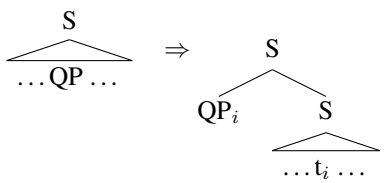
As clearly explained in (Szabolcsi, 2008), this mismatch between syntax and semantics exhibited by quantifiers might call for a distinction between syntactic domain and semantic scope, where the former has been defined in terms of c-command, maximal projection, feature inheritance, or other similar notions, while the latter is the part of the structure on which the operator performs its action, its scope. The hypothesis below, originally proposed in (Reinhart, 1979), has found many supporters among linguists. Basically, the claim is that syntactic structure "determines" semantic scope.

(54) "The scope of a linguistic operator coincides with its domain in some syntactic representation that the operator is part of."

The reader interested in an up-to-date overview of the QP problem and to the several solutions proposed in the literature is referred to (Szabolcsi, 2008) and (Ruys & Winter, 2009). Here we mention the two well known solutions of the quantifiers puzzle proposed in (May, 1977) and in (Hendriks, 1993), since they will help highlighting some important aspects of our proposal.

May's approach produces semantic constituent structures in abstract level. May proposes that syntax does not end with producing the surface string. Instead, a movement rule, called "Quantifier raising" (QR) continues to operate at an abstract level, called the "Logical Form" (LF), and attach each phrase containing a quantifier to its domain by splitting the node S of the surface structure into two nodes. The rule leaves a virtual trace t coindexed with the moved QP. Its earliest formulation operates as shown below.

(55)



QR may derive from one given surface structure several different LFs with different scope relations. The structures at the LF level are then interpreted obtaining the meaning representations.

In the solution we propose the syntactic tree is not re-written into a semantic tree, rather the QP category splits into two parts: the syntactic-component, dp, that actually stays in the syntactic structure – where in May's LF there is a virtual trace t – and it's only the scope-component that moves out and reaches the top S-node of the sentence where the QP performs its action.

Hendriks dissociates scope from pure syntax in that it allows one to maintain whatever constituent structure seems independently motivated and still delivers all imaginable scope relations: all the possible scope relations can be obtained by means of three type-change rules: "Argument Raising", "Value Raising", and "Argument Lowering". As we have seen in (26), the last two are already theorems of NL , whereas "Argument Raising" is not. A relevant instance of it is instead a theorem in LG : the argument of a transitive verb can be raised to the higher order category of quantifiers, as we show in (61). See (Bastenhof, 2007) for further details.

As emphasized in (Szabolcsi, 2008), an important task for researchers working on the syntax-semantic interface is "to determine whether the Reinhart's hypothesis is correct and if yes, exactly what kind of syntactic representation and notion of domain bears it out". In the logic perspective, the question reads as saying which are the logical operators that capture this connection between the two components of quantifiers, while allowing to scope over the top part of a tree and still remain in the original position as a dp. NL does not provide us with the required tools. It is the minimum logic to model local dependency and the assembly of direct scope constructors, but it is too weak to account for long-distance dependency, cross-dependencies and inverse scope effects. Hence, the need of a more expressive logic that goes behind these limitations. Below we show how LG properly captures the QP unbounded scope behaviour exhibited in the examples (1) and (2) above. We believe that the different scope distribution of QPs ((4) in (53)) can be accounted for by adding unary modalities to LG following (Bernardi, 2002; Bernardi & Szabolcsi, 2008). Following (Barker & Shan, 2006), we conjecture that the cases of bounded scope ((3) in (53)) are properly handled by delimited continuations which belong to the framework we have presented here. For reason of space we cannot go into the details of continuation semantics; in Section 5 we sketch its connection with LG and refer the reader to (Bernardi & Moortgat, 2009; Bastenhof, 2009a).

The "parsing as deduction" view on QP Let us use an example to guide our analysis of QPs. Take the sentence *Alex thinks everyone left*. The proof of its grammaticality corresponds to the following sequent where we leave the QP category undefined.

(56) $$\underbrace{dp}_{\text{alex}} \otimes (\underbrace{(dp\backslash s)/s}_{\text{[thinks}} \otimes (\underbrace{\quad QP \quad}_{\text{[everyone}} \otimes \underbrace{dp\backslash s}_{\text{left]]}})) \vdash s$$

Our task, now, is to understand which is the category to be assigned to QP. Recall the conclusion we reached while observing the quantifiers' semantic constituent tree ((38) in Section 2.4): a quantifier consists of a syntactic and a semantic component (the dp and a function from s to s), such that (a) the dp should stay in the embedded position where it works as an argument of a verb, and (b) the scope component should be free to travel through the tree so to reach its top level, without modifying the constituent structure. In other words, we start from a structure containing a QP, that is $\Gamma[QP]$, which is proved to be of category s, if the simpler structure $\Gamma[dp]$ is of category s and the semantic components of the QP reaches the top of the structure. In the following, we will explain what it means for the QP to "reach the top" of the structure and how it does it. This behaviour is captured by the co-abstraction rule which happens within an $\otimes$-structure Γ containing the QP that, for the moment, we represent as $B \oslash dp$ – where B is the category to be assigned to the QP semantic component and which still needs to be specified.

(57) $$\frac{\Gamma[dp] \vdash B \oplus s}{\Gamma[B \oslash dp] \vdash s} \ (\oslash L)$$

In our example, $\Gamma[dp]$ stands for $dp \otimes ((dp\backslash s)/s \otimes (dp \otimes dp\backslash s))$. By application of $(\backslash L)$ and $(/L)$ the sequent is reduced to axiom links and to the simpler sequent below.

(58) $$s \vdash B \oplus s$$

We want to know what could be the proper category to replace B. By dual residuation (DRES in Figure 2), we obtain that the semantic component of a QP is the co-functor category $(s \oslash s)$. We repeat in (59) the if-part of DRES we use below:

(59) $$\text{(DRES)}\ \ C \vdash B \oplus A \ \text{ iff }\ C \oslash A \vdash B$$

(60) $$s \vdash B \oplus s \ \text{ iff }\ (s \oslash s) \vdash B$$

Hence, the whole category of a quantifier should be $(s \oslash s) \oslash dp$, where the value-s and argument-s are the first and second one, respectively, viz. $(s_{arg} \oslash s_{val}) \oslash dp$.[9] Before going to look at some linguistic examples, it's worth underline the following two facts.

First of all, as the reader can check by herself, $(s \oslash s) \oslash dp \vdash s/(dp\backslash s)$: every expression that belongs to the non-local scope category, belongs to the local category too; whereas $s/(dp\backslash s) \not\vdash (s \oslash s) \oslash dp$. Moreover, whereas the "Argument Raising" is not a theorem of LG (see (26) in Section 2), the argument of a transitive verb can be raised to the QP category as shown below. In (Bastenhof, 2007), this theorem is used to handle the interaction of QPs with intentional verbs.

(61)
$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{dp \otimes ((dp\backslash s)/dp \otimes dp) \vdash (s \oslash s) \oplus s}}{dp \otimes ((dp\backslash s)/dp \otimes ((s \oslash s) \oslash dp)) \vdash s}\,(\oslash L)}{dp \otimes ((dp\backslash s)/dp \otimes (((s \oslash s) \oslash dp))) \vdash s}\,(\backslash R)}{(dp\backslash s)/dp \otimes ((s \oslash s) \oslash dp) \vdash dp\backslash s}\,(/R)}{(dp\backslash s)/dp \vdash (dp\backslash s)/(((s \oslash s) \oslash dp))}$$

Looking at the categories obtained in terms of the two merging relations, saying that $QP \in (s \oslash s) \oslash dp$ means that there are two parts y and x such that $\texttt{Merge}^{\texttt{c}}(y, QP, x)$, $y \notin (s \oslash s)$ and $x \in dp$. In other words, we obtain the syntactic component x by extracting the semantic component z from the QP. The syntactic component $x \in dp$ will stay in the $\otimes$-structure where it can merge with other expressions, whereas the semantic component will go to the $\oplus$-structure.

Let's check how this category behaves by looking at direct and inverse scope caused by a QP in subject (62) and object position (63), respectively. As the reader can check by herself, the application of the $(\backslash L)$ (and $(/L)$) in (62) (resp. (63)) builds a derivation that does not end with axiom links –hence it fails to prove that the sequent holds. The only other possible first step is the application of $(\oslash L)$.

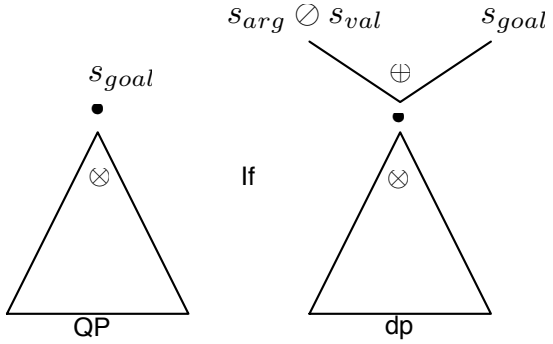
(62)
$$\dfrac{\dfrac{\dfrac{dp_1 \vdash dp_2 \quad \dfrac{s_3 \vdash s_1 \quad s_2 \vdash s_4}{s_3 \vdash (s_1 \oslash s_2) \oplus s_4}\,(\oslash R)}{dp_1 \otimes (dp_2\backslash s_3) \vdash (s_1 \oslash s_2) \oplus s_4}\,(\backslash L)}{\underbrace{((s_1 \oslash s_2) \oslash dp_1)}_{\text{everyone}} \otimes \underbrace{(dp_2\backslash s_3)}_{\text{left}} \vdash s_4}\,(\oslash L)$$

(63)
$$\dfrac{\dfrac{dp_1 \otimes ((dp_2\backslash s_1)/dp_3 \otimes dp_4)) \vdash (s_2 \oslash s_3) \oplus s_4}{\underbrace{dp_1}_{\text{[Alex}} \otimes (\underbrace{(dp_2\backslash s_1)/dp_3}_{\text{[knows}} \otimes \underbrace{((s_2 \oslash s_3) \oslash dp_4))}_{\text{everyone]]}} \vdash s_4}\,(\oslash L)}{\dfrac{\vdots}{s_1 \vdash (s_2 \oslash s_3) \oplus s_4}\,(\oslash R)}$$

With the first inference step, the co-abstraction $(\oslash L)$, the scope-component of the QP $(s_{arg} \oslash s_{val})$ is sent to the succedent of the $\vdash$ leaving on the antecedent a standard $\otimes$-structure, the syntactic constituent structure of the sentence, containing a dp in the place of the QP. By function application steps, this structure is reduced to the category s carried by its head, the verb, (s_3 in (62) and s_1 in (63)) – in (64), we will represent it with s_{vhd}. The last steps check the scope: the co-function application $(\oslash R)$ is applied and s_{vhd}

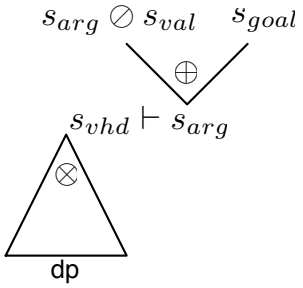
reaches s_{arg} ($s_3 \vdash s_1$ in (62), and $s_1 \vdash s_2$ in (63)) and s_{val} reaches s_{goal} ($s_2 \vdash s_4$ in (62) and $s_3 \vdash s_4$ in (63)).

As introduced in (46) of Section 3, we use a tree for the left side of the sequent and a mirror upside down tree for its right side; we obtain the division of labor below, where the upside down tree represents the semantic constituent structure (which plays the role of May's LF) and the usual tree the syntactic structure. The first inference step ($\oslash L$) in the two sequents corresponds to the schematic trees in (64) below.



(64)

To ease the comparison with the sequents above, we can represent the tree on the right in (64) as following:



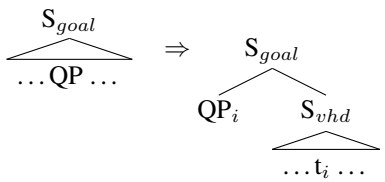
(65)

where the s_{vhd} stands for the s sub-formula of the category representing the head of the predicate argument structure, viz. the s value of the main verb.

The tree, containing the dp component of QP, is reduced to s_{vhd} which reaches ($\vdash$) the s_{arg} of the QP's scope component and the s_{val} reaches the s_{goal}.

It is interesting to compare (65) with the QR schema (55); we repeat it in (66) with the annotation of the S-nodes to help grasping the similarity between the two solutions: LG captures the mathematical principles behind QR.

(66)



The category assigned to QP built with dual-implications properly account both for direct and inverse scope. The examples above show that two of the problems encountered by the higher order categories consisting of the (Lambek) implications, viz. (i) multiple assignments and (ii) local scope, are solved. We now move to look at scope ambiguity and, in particular, at non-local scope.

The "parsing as deduction" view on Scope Ambiguity The simplest case with scope ambiguity is the one with QPs both in subject and object positions [Someone [knows everyone]]. The sequent to prove is:

(67) $$\underbrace{((s_1 \oslash s_2) \oslash dp_1)}_{\text{[someone}} \otimes (\underbrace{(dp_2 \backslash s_3)/dp_3}_{\text{[knows}} \otimes \underbrace{((s_4 \oslash s_5) \oslash dp_4)}_{\text{everyone]]}}) \vdash s_6$$

There are three main functional connectives (two $\oslash$ and one $/$) in the antecedent of the sequent that could be activated, hence three inference rules could be applied. As the reader can check by herself, the application of the $(/L)$ rule builds a derivation that does not end with axiom links –hence it fails to prove that the sequent holds. The other two possibilities are $(\oslash L)$ applied to the $\oslash$ of the subject or of the object. These two choices bring to the two different derivations schematized below.

(68)

[SOME $>$ EVERY]

$$\dfrac{\dfrac{\dfrac{\begin{array}{c} s_5 \vdash s_1 \quad s_2 \vdash s_6 \quad s_3 \vdash s_4 \\ \vdots \\ s_3 \vdash (s_4 \oslash s_5) \oplus ((s_1 \oslash s_2) \oplus s_6) \\ \vdots \\ dp_1 \otimes ((dp_2 \backslash s_3)/dp_3 \otimes dp_4) \vdash (s_4 \oslash s_5) \oplus ((s_1 \oslash s_2) \oplus s_6) \end{array}}{dp_1 \otimes ((dp_2 \backslash s_3)/dp_3 \otimes ((s_4 \oslash s_5) \oslash dp_4)) \vdash (s_1 \oslash s_2) \oplus s_6}\ (\oslash L)}{\underbrace{((s_1 \oslash s_2) \oslash dp_1)}_{\text{[someone}} \otimes (\underbrace{(dp_2 \backslash s_3)/dp_3}_{\text{[knows}} \otimes \underbrace{((s_4 \oslash s_5) \oslash dp_4)}_{\text{everyone]]}}) \vdash s_6}\ (\oslash L)$$

[EVERY $>$ SOME]

$$\dfrac{\dfrac{\dfrac{\begin{array}{c} s_5 \vdash s_6 \quad s_2 \vdash s_4 \quad s_3 \vdash s_1 \\ \vdots \\ s_3 \vdash (s_1 \oslash s_2) \oplus ((s_4 \oslash s_5) \oplus s_6) \\ \vdots \\ dp_1 \otimes ((dp_2 \backslash s_3)/dp_3 \otimes dp_4) \vdash (s_1 \oslash s_2) \oplus ((s_4 \oslash s_5) \oplus s_6) \end{array}}{(s_1 \oslash s_2) \oslash dp_1 \otimes ((dp_2 \backslash s_3)/dp_3 \otimes ((s_4 \oslash s_5) \oslash dp_4)) \vdash (s_4 \oslash s_5) \oplus s_6}\ (\oslash L)}{\underbrace{(s_1 \oslash s_2) \oslash dp_1}_{\text{[someone}} \otimes (\underbrace{(dp_2 \backslash s_3)/dp_3}_{\text{[knows}} \otimes \underbrace{(s_4 \oslash s_5) \oslash dp_4)}_{\text{everyone]]}}) \vdash s_6}\ (\oslash L)$$

Again, the structure $dp_1 \otimes ((dp_2 \backslash s_3)/dp_3 \otimes dp_4)$ reduces to s_3 by function application steps as explained in Section 2, the reader is invited to check the steps by herself. Instead, we spell out the second part that brings to the relevant axioms by means of co-function applications:[10]

(69)

[SOME $>$ EVERY]

$$\dfrac{\dfrac{s_5 \vdash s_1 \quad s_2 \vdash s_6}{s_5 \vdash (s_1 \oslash s_2) \oplus s_6}\ (\oslash L) \qquad s_3 \vdash s_4}{s_3 \vdash (s_4 \oslash s_5) \oplus ((s_1 \oslash s_2) \oplus s_6)}\ (\oslash L)$$

[EVERY $>$ SOME]

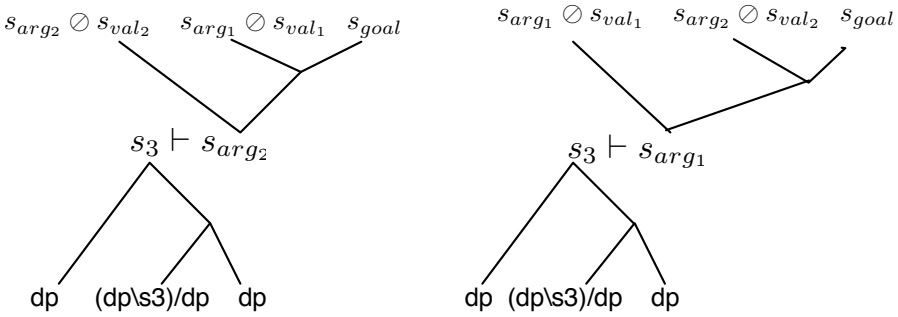
$$\dfrac{\dfrac{s_5 \vdash s_6 \quad s_2 \vdash s_4}{s_2 \vdash (s_4 \oslash s_5) \oplus s_6}\ (\oslash L) \qquad s_3 \vdash s_1}{s_3 \vdash (s_1 \oslash s_2) \oplus ((s_4 \oslash s_5) \oplus s_6)}\ (\oslash L)$$

Recall the observation made above regarding the s-value and s-argument of the QP, viz. $(s_{arg} \oslash s_{val}) \oslash dp$. As shown by the axiom links in the first derivation the s-value of *someone* matches the s-goal formula $(s_2 \vdash s_6)$, the s-value of *everyone* matches the s-argument of *someone*, and the s of the transitive verb matches the s-argument of *everyone*. These are the axiom links of the subject wide scope reading [SOME $>$ EVERY]. The axioms linking s-categories of the other derivation are different: they give the object wide scope reading [EVERY $>$ SOME]. Observe that the earliest a quantifier jumps out of the $\otimes$-structure, the widest scope it has (recall we are looking at scope through a mirror! –through the dual residuation calculus). Notice, that the axioms linking dp-categories are the same in the two derivations, i.e. $dp_1 \vdash dp_2$ and $dp_4 \vdash dp_3$: in both derivations, the variable taken as object by the verb is bound by the QP in object position and the variable taken as subject of the verb is bound by the QP in subject position.

In a tree notation the two derivations above are schematized as in (70). Again, to help thinking of s-value and s-argument, here we mark the s-categories with this information instead of numbers. $s_{arg_2} \oslash s_{val_2}$

is the semantic component of the quantifier in object position, *everyone*, i.e. $s_4 \oslash s_5$; whereas $s_{arg_1} \oslash s_{val_1}$ is the semantic component of the quantifier in subject position, *someone*, i.e. $s_1 \oslash s_2$.

(70)



The two readings are read out of the upside down trees: the tree on the left gives the subject wide scope reading, and the tree on the right gives the object wide scope reading. The syntactic constituents are read out of the usual tree that is the same in the two cases –it is the sequent that reduces to s_3 in the two derivations above. The cases with QP interacting with other scope operators will work out in a similar way. The reader is invited to try, for instance, to build the derivations for the "John doesn't read a book" (viz. prove that $dp \otimes ((dp\backslash s)/(dp\backslash s) \otimes ((dp\backslash s)/dp \otimes ((s \oslash s) \osl dp)) \vdash s$).

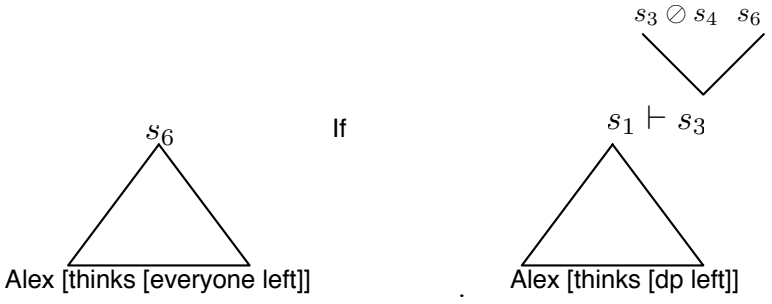
The case of unbounded scope works in the same way: given the structure corresponding to [Alex [thinks [everyone left]]], there are two main connectives $/$ and $\osl$, hence there are two possible inference rules to be applied as a first step: Either we activate first *everyone* (71) or *thinks* (73).

(71)

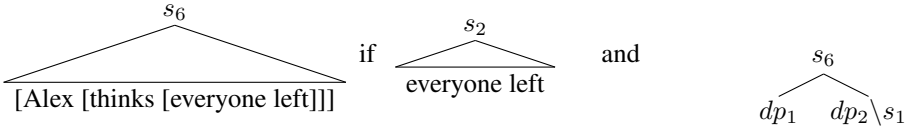
$$\text{[EVERY} > \text{THINK]} \quad \dfrac{\dfrac{s_4 \vdash s_6 \quad s_1 \vdash s_3}{s_1 \vdash (s_3 \oslash s_4) \oplus s_6}\ (\oslash R) \quad \vdots}{\underbrace{dp_1}_{\text{Alex}} \otimes (\underbrace{(dp_2\backslash s_1)/s_2}_{\text{[thinks}} \otimes (\underbrace{((s_3 \oslash s_4) \osl dp_3)}_{\text{[everyone}} \otimes \underbrace{dp\backslash s_5}_{\text{left]]}})) \vdash s_6}\ (\osl L)$$

Again, the full structure on the left of the $\vdash$, by means of function applications, reduces to s_1 (i.e., the value carried by the head of the structure, *thinks*); hence, as we did above, we can abbreviate the steps of the derivation and focus on the branch of the derivation containing the dual residuated operators and that account for the scope structure. The scope component of the quantifier has reached the wide position of the whole structure, and takes scope on its semantic domain on which it is applied by means of co-function application ($\oslash$R). The value-s (s_4) of the operator (*everyone*) having wide scope matches the s-goal formula (s_6), whereas its argument-s (s_3) is obtained from the value-s (s_1) of the narrow operator (*thinks*). We can represent the sequents' steps with the following tree. Now that the reader might be familiar with the procedure, we take the liberty of using words instead of the category and give only the schema of the trees.

(72)



In the second reading of the sentence, the syntactic and semantic domain of *thinks* coincides: it takes scope over the structure it c-commands/dominates, hence also on the quantifier. Proof theoretically this means that its s-value (s_1) is linked to the goal formula (s_6), and its s-argument (s_2) is reached by the quantifier s-argument (s_4). The application of the function *thinks* to the expression *everyone left* reduces the problem of verifying that the full structure is of category s_6 to two simpler problems: verifying that [everyone left] is s_2 and the subject dp_1 composed with the verb phrase category $dp_1 \backslash s_1$ is of category s_6. The first tree is checked as we have seen in (62), the second simply by function application.



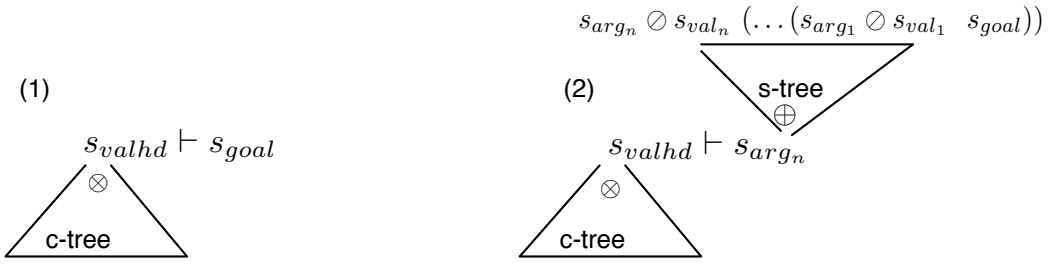
The derivation corresponding to this second reading is built by focusing first on *thinks* and then on *everyone*; as we have seen in the case of multiple quantifiers, the operator which is activated earlier receives wider scope.

(73)

$$\dfrac{\dfrac{\dfrac{dp_3 \vdash dp_4 \quad \dfrac{s_5 \vdash s_3 \quad s_4 \vdash s_2}{s_5 \vdash (s_3 \oslash s_4) \oplus s_2}\,(\oslash L)}{dp_3 \otimes dp_4 \backslash s_5 \vdash (s_3 \oslash s_4) \oplus s_2}\,(\backslash L)}{((s_3 \oslash s_4) \oslash dp_3) \otimes dp_4 \backslash s_5 \vdash s_2}\,(\otimes L) \quad \dfrac{dp_1 \vdash dp_2 \quad s_1 \vdash s_6}{dp_1 \otimes dp_2 \backslash s_1 \vdash s_6}\,(\backslash L)}{\underbrace{dp_1}_{\text{Alex}} \otimes (\underbrace{(dp_2 \backslash s_1)/s_2}_{\text{[thinks}} \otimes (\underbrace{((s_3 \oslash s_4) \oslash dp_3)}_{\text{[everyone}} \otimes \underbrace{dp_4 \backslash s_5}_{\text{left]]}})) \vdash s_6}\,(/L)$$

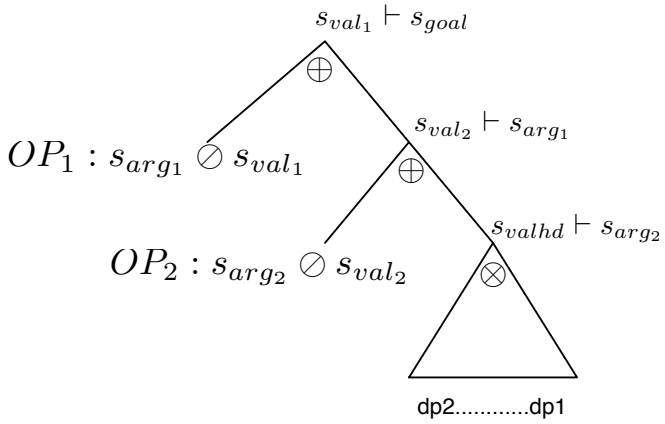
Summing up, with (62) and (63) we have shown that the category assigned to the QP using the extended language (i) avoids multiple assignments, allows for (ii) local scope; this last example shows that it also handles (iii) non-local scope. Moreover, the examples with multiple scope operators show that syntactic structures containing scope operators can be built by activating them in different orders and these different steps produce different readings of the same structure. Finally, in LG we exploit its two relations on trees to build the constituent tree (c-tree) and the scope commanded tree (s-tree), when the latter may diverge from the former: The c-tree is built out of $\otimes$ whereas the s-tree is built out of $\oplus$, as illustrated by the tree on the right in (74). A scope operator that acts only locally, like *thinks*, scopes on the structure it c-commands, a $\otimes$-structure: by means of function applications its s-value reaches the goal-s formula, as illustrated by the tree on the left in (74):

(74)



If we go back to the preliminary analysis done in Section 2.4, when we introduced the concept of a semantic and syntactic component, we see that the tree we proposed there can be made precise. We can visualize the combination of the tree and upside down tree given in (70) as below to ease the comparison. Let's take the reading with $OP_1 > OP_2$, the tree looks like the following

(75)



We can now go back to the goal of this paper, viz. use the "parsing as deduction" approach to shed new lights on the question about Reinhart's hypothesis, repeated below in (76). The question we are seeking to answer is: "is this hypothesis correct? And if yes, exactly what kind of syntactic representation and notion of domain bears it out" (Szabolcsi, 2008).

(76) "The scope of a linguistic operator coincides with its domain in some syntactic representation that the operator is part of."

Our answer to this question is that, yes, Reinhart's hypothesis is correct, and that scope operators require to distinguish their syntactic domain from their semantic domain which are governed by dual relations.

Adding the derivational perspective worked out in this paper, we could rephrase (76) in the following way

(77) "The scope of a linguistic operator is *read off* of the (inference) *steps* that put together the expression of which the operator is part of."

5 Related Works and Some Further Intuition

Two points of our work that is most interesting to see in connection with related works are the idea of looking at quantifier as consisting of a syntactic and a semantic component and the interpretation of natural language via Continuation Semantics.

5.1 *QP syntactic category*

The idea of looking at quantifiers as consisting of a syntactic and a scope component is not new. Examples of its use, among others, are the work by Barker and Shan (Barker & Shan, 2008), and the one by Kallmeyer and Joshi (Kallmeyer & Joshi, 2003). In the former the authors give a nice and clear account of donkey anaphora relating it to quantifier binding. The grammar they use, based on continuation semantics, had been previously introduced in (Barker & Shan, 2006), but in (Barker & Shan, 2008) is presented with a simplified "tower" notation that highlights the different role played by the dp-component of the QP and its scope-part. A quantifier receives the category below, that, the author says, "can be read counterclockwise, starting below the horizontal line" –here and in the following brief overview, we indicate the value-s and the argument-s for ease of comparison with our proposal. For a comparison between our and Barker and Shan's proposal see (Bastenhof, 2009a).

$$\frac{s_{val}|s_{arg}}{dp} \quad \text{means} \quad \frac{\ldots\text{to form an S.}|\text{and takes scope at an S}}{\text{The expression functions in local syntax as a DP}}$$

Similarly, in (Kallmeyer & Joshi, 2003), the authors extend Tree Adjoining Grammar with multi-component trees in order to deal with QP scope ambiguity. In this framework, lexical entries receive a tree; a QP receives a tree with two components: "one component corresponds to the contribution of [the quantifier] to the predicate-argument structure [...], and [the other] component contributes to the scope structure."

Finally, Moortgat (Moortgat, 1996) proposes a three-place binding type constructor q which captures the behavior of in-situ binders. The behaviour of this operator was represented by the inference step below. An expression of category $q(A, B, C)$ occupies the position of an expression of category A within a structural context of category B; using the q connective turns the domain of category B into one of type C. In particular, a QP receives the category $q(dp, s_{arg}, s_{val})$.

$$\frac{\Delta[A] \vdash B \quad \Gamma[C] \vdash D}{\Gamma[\Delta[q(A, B, C)]] \vdash D}$$

This inference step properly characterizes natural language scope operators. However, this operator is not part of the logical language, and it was not known which are the logical operators that capture its behaviour. Our work is an answer to this question. Notice, the limitations encountered by the q-operators discussed in (Carpenter, 1998), e.g. its relation with coordination, are overcome by the proposal presented in this paper.

The idea of looking at QPs has an effect of splitting a given expressions into parts is also behind the work in (Morrill, Fadda, & Valentin, 2007; Morrill & Fadda, 2008). In these papers the starting point is the Associative Lambek calculus (Lambek, 1961) which is extended with discontinuous product and their residuals, infixation ($\downarrow$) and extraction ($\uparrow$), and separators to mark the holes generated by the extraction and where other expression can be substitute in. In this framework, called Discontinuous Lambek Calculus, a QP receives the category $(s_{arg} \uparrow dp) \downarrow s_{val}$: the infixation substitutes the QP in the hole generated by the extraction of a dp from the predicate-argument structure; the hole is marked by a separator. It is interesting to notice that this extension has been shown to overcome the limitations mentioned in Section 1, being able to account both for quantifier scope construals and non-peripheral extractions and cross-serial dependencies. See (Bastenhof, 2009b) and (Moot, 2007) for a discussion of these phenomena in LG .

We would like to draw attention on an important difference between the solutions mentioned so far. In Morrill and Fadda and Barker and Shan's categories the dp and s_{arg} form a sub-formula together in the QP category, whereas both in Kallmeyer and Joshi and in our's category the dp category is taken apart. We believe that is a point of interest that deserves further in depth analysis.

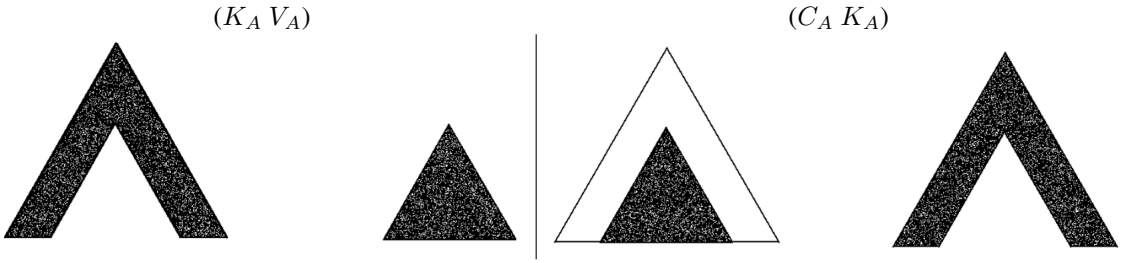
5.2 *Continuation Semantics*

Barker and Shan, and Philippe de Groote have shown in several papers (see for instance (Barker, 2002; Shan, 2005; Barker & Shan, 2006, 2008; de Groote, 2001, 2006)) that continuation semantics is what is needed to interpret quantifier phrases as well as other related phenomena, like donkey anaphora, wh-extraction etc. Important keywords of this semantics are "values", "continuation" and "computations". Below we try to give an intuitive idea of their difference and we briefly discuss how continuation semantics properly interprets the LG derivations discussed so far.

Values, Continuation and Computation Work done within Curry Howard Correspondence to Classical Logic has brought to the attention the duality distinction holding between terms and contexts. We believe that this "duality" view can help further shaping the syntax-semantics interface and the connection between natural and formal languages. This duality is clearly explained in (Sacerdoti Coen, 2006):

- A context (aka, *continuation*) is an expression with one placeholder for a "missing" term (aka, *value*). The placeholder can be filled with a term (a value) to obtain a placeholder-free expression. ($K_A\ V_A = R$)
- Dually, a term (aka, *computation*) can be seen as an expression with exactly one placeholder for a "missing context" that is "all around" the term. The placeholder can be filled with a context to obtain a placeholder-free expression. ($C_A\ K_A = R$)

We will abbreviate *values*, *continuation* and *computation* of type A as V_A, K_A and C_A, respectively. We can think of "a placeholder-free expression" as an "observable object" and assign to it the type R (the category of an unfocused entailment when the communication between the two merge-relations is established, ie. the conclusion of a Cut-rule (See example in (49)). Both the application of a continuation to a value ($K_A\ V_A$) and of a computation to a continuation ($C_A\ K_A$) result into a "placeholder-free expression" (R). Since we are interested in composing linguistic structures (tree-structures), the above applications can be visualized as below. A value is a tree, a continuation and a computation are a tree with a hole on the bottom and on the top part, respectively.



The picture highlights the new view on compositionality, and its connection with the two merging relations discussed so far might be clear at this point: the $\otimes$ composes values whereas the $\oplus$ composes continuations. The $\bar{\lambda}\mu\tilde{\mu}$-calculus in Curry Howard Correspondence with LG has the μ and $\tilde{\mu}$ operator to bring the focus on elements of the $\oplus$ and $\otimes$ structures, respectively.

6 Conclusion

Our goal was to define the mathematical principles of natural language structures. We have claimed that the fundamental principles needed are the (pure) residuation principle, its dual and the distributivity principles which establish the interaction among them. We have shown that, proof-theoretically, this means having structures both on the antecedent and the succedent of the entailment relation of a sequent. We have also briefly discussed the connection between this extended calculus and Continuation Semantics, that, in several papers by different authors, has been shown to be a proper framework for the interpretation of natural language structures containing inverse scope operators and the alike.

Acknowledgment I would like to express my gratitude to Chris Barker, Marco Baroni, Anna Szabolcsi and Roberto Zamparelli for their valuable feedback on earlier versions of the present paper. My thanks go to Chris Barker, Philippe de Groote, Glyn Morrill, Richard Moot, Ian Pratt, Ken Shan, and Mark Steedman for stimulating discussions on the QP puzzle, and on the approach to natural language analysis undertaken in this paper. Finally, the work reported in this paper is part of the adventure started with Rajeev Goré and Michael Moortgat back at ESSLLI 2004 and brought ahead with Michael Moortgat through all these years. The work reported has been carried out jointly with Michael Moortgat.

Notes

[1]Following the same procedure, *every* is assigned the type $(e \to t) \to ((e \to t) \to t)$ and the term $\lambda Z.\lambda Y.\forall\ \lambda x.((Z\ x) \Rightarrow (Y\ x))$. An alternative notation is $\lambda Z.\lambda Y.\forall x.((Z\ x) \Rightarrow (Y\ x))$.

[2]The mapping from syntactic categories to semantic types, $\texttt{type} : \texttt{CAT} \to \texttt{TYPE}$, is given below.

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

[3]In Natural Deduction, the sequent rule introducing an operator on the right corresponds to the Introduction rule, whereas the introduction of the operator on the left of the $\vdash$ corresponds to the Elimination rule.

[4]The cut rule involves an unknown formula that disappear from the conclusion (given the problem to solve, $\Gamma[\Delta] \vdash C$, it is not known what is the cut-formula, B.) Hence, for decidable proof search, people check whether in a given system the cut rule is admissible, viz. all the theorems that can be proved with the cut rule, can be proved also without it.

[5]This failure is due to the monotonicity property of the implication, that is downward monotonic in the argument position and upward monotonic in the value position, viz. $-\backslash+$ and $+/-$.

[6]In (Bernardi & Szabolcsi, 2008) this observation is exploited to account for the different scope behaviours of quantifiers. The argument-s determines what the operator can immediately scope over. The value-s determines what the operator can be in the immediate scope of.

[7]Note that the distributivity we are used to in mathematics, is the one of $\times$ over the $+$, $(A + B) \times C = (A \times C) + (B \times C)$. It involves resource duplication (see the number of occurrence of C differs in the two side of the equation), whereas we are working in a resource sensitive system hence no formula is duplicated.

[8]Below we unfold the steps behind the compiled logical rules. To help understanding the steps, we take Δ to consists of only two $\oplus$ in the $(/R)$ case, and Γ of only two $\otimes$ in the $(\oslash)$ case. But the two rules apply for arbitrarily big structures.

$$\dfrac{\dfrac{\dfrac{\Gamma \otimes B \vdash (D \oplus C) \oplus A}{\Gamma \vdash (D \oplus C) \oplus A)/B}\ (RES)}{\Gamma \vdash ((D \oplus C)/B) \oplus A}\ (\oplus MC)}{\Gamma \vdash (D \oplus (C/B)) \oplus A}\ (\oplus MA) \qquad \dfrac{\Gamma \otimes B \vdash \Delta[C]}{\Gamma \vdash \Delta[C/B]}\ (/R)$$

$$\dfrac{\dfrac{\dfrac{A \otimes (C \otimes D) \vdash B \oplus \Delta}{B \oslash (A \otimes (C \otimes D)) \vdash \Delta}\ (DRES)}{A \otimes (B \oslash (C \otimes D)) \vdash \Delta}\ (\otimes MC)}{A \otimes ((B \oslash C) \otimes D) \vdash \Delta}\ (\otimes MA) \qquad \dfrac{\Gamma[C] \vdash B \oplus \Delta}{\Gamma[B \oslash C] \vdash \Delta}\ (\oslash L)$$

[9]The s_{val} and s_{arg} of s_{val}/s_{arg} are in a positive and negative polarity position, respectively $(+/-$, $-\backslash+$ and similarly for their dual, $-\oslash+$, and $+\oslash-)$. Compare $s_1/(dp\backslash s_2)$ with $((s_2 \oslash s_1) \oslash dp)$, the polarity of the atomic formulas is the same, and s_1 is the s_{val} and s_2 is the s_{arg}. Hence, $(s_{arg} \oslash s_{val})$.

[10]Notice that in the dual calculus too, we could follow either a top-down or a bottom-up approach to derive the sequents. For instance, in (68) we could have applied first the internal function to match the s_6 formula first.

References

Areces, C., & Bernardi, R. (2004). Analyzing the Core of Categorial Grammar. *Journal of Logic, Language and Information (JoLLI)*, *13*(2), 121–137.

Barker, C. (2002). Continuations and the nature of quantification. *Natural language semantics*, *10*, 211–242.

Barker, C., & Shan, C. (2006). Explaining crossover and superiority as left-to-right evaluation. *Linguistics and Philosophy*, *29*(1), 91-134.

Barker, C., & Shan, C. (2008). Donkey anaphora is in-scope binding. *Semantics and Pragmatics*, *1*, 1–46.

Bastenhof, A. (2007). *Quantifier scope and coordination in flexible and continuation-based approaches to natural language semantics*. BPhil thesis, Utrecht University.

Bastenhof, A. (2009a). *Continuation in natural language syntax and semantics*. MPhil thesis, Utrecht University.

Bastenhof, A. (2009b). Extraction in the Lambek-Grishin calculus. In *ESSLLI 09 student session.* Bordeaux.

Bernardi, R. (2002). *Reasoning with Polarity in Categorial Type Logic*. Utrecht: Utrecht Institute of Linguistics OTS.

Bernardi, R., & Moortgat, M. (2007). Continuation semantics for Symmetric Categorial Grammar. In D. Leivant & R. de Queiroz (Eds.), *Proceedings of the 14th Workshop on Logic, Language, Information and Computation (WOLLIC'07)* (pp. 53–71). Berlin/Heidelberg: Springer.

Bernardi, R., & Moortgat, M. (2009). Continuation semantics for the Lambek-Grishin calculus. *Information and Computation*. (To appear)

Bernardi, R., & Szabolcsi, A. (2008). Optionality, scope, and licensing: An application of partially ordered categories. *JoLLI*, *17*(3).

Carpenter, B. (1998). *Type-logical semantics*. Cambridge MA: MIT Press.

Chomsky, N. (1976). Conditions on Rules of Grammar. *Linguistic Analysis*, 303-351.

de Groote, P. (2001). Type raising, continuations, and classical logic. In R. van Rooy & M. Stokhof (Eds.), *Proceedings of the Thirteenth Amsterdam Colloquium* (pp. 97–101). Amsterdam: University of Amsterdam.

de Groote, P. (2006). Towards a Montagovian account of dynamics. In *Proceedings of Semantics and Linguistic Theory XVI.*

Goré, R. (1997). Substructural Logics on Display. *Logic Journal of IGPL*, *6*(3), 451–504.

Grishin, V. (1983). Studies in Nonclassical Logics and Formal Systems. In (pp. 315–334). Moscow: Nauka. ([English translation in Abrusci and Casadio (eds.) Proceedings 5th Roma Workshop, Bulzoni Editore, Roma, 2002])

Hendriks, H. (1993). *Studied flexibility. categories and types in syntax and semantics.* Amsterdam: ILLC, Amsterdam University.

Howard, W. A. (1980). The formulae-as-types notion of construction. In *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism* (pp. 480–490). London: Academic Press.

Kallmeyer, L., & Joshi, A. (2003). Factoring predicate argument and scope semantics: Underspecified semantics with LTAG. *Research on Language and Computation*, *1*(1-2), 3–58.

Kurtonina, N., & Moortgat, M. (2007). Relational semantics for the Lambek-Grishin calculus. In M. Kracht, G. Penn, & E. Stabler (Eds.), *10th Conference on Mathematics of Language.* Los Angeles: UCLA Working Papers in Linguistics.

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

Lambek, J. (1961). On the calculus of syntactic types. In R. Jakobson (Ed.), *Structure of Languages and its Mathematical Aspects* (pp. 166–178). American Mathematical Society.

Lambek, J. (1993). From categorial to bilinear logic. In K. D. P. Schröder-Heister (Ed.), *Substructural Logics* (p. 207-237). UK: Oxford University Press.

May, R. (1977). *The Grammar of Quantification*. Unpublished doctoral dissertation, MIT.

Montague, R. (1974). *Formal Philosophy: Selected papers of Richard Montague* (R. Thomason, Ed.). New Haven: Yale University Press.

Moortgat, M. (1988). *Categorial Investigations. Logical and Linguistic Aspects of the Lambek calculus*. Dordrecht: Foris.

Moortgat, M. (1996). Generalized quantifiers and discontinuous type constructors. In H. Bunt & A. van Horck (Eds.), *Discontinuous Constituency* (pp. 181–207). De Gruyter.

Moortgat, M. (2007). Symmetries in natural language syntax and semantics: the Lambek-Grishin calculus. In D. Leivant & R. de Queiros (Eds.), *Proceedings 14th Workshop on logic, Language, Information and Computation (WoLLIC'07)* (p. 264-284). Berlin/Heidelberg: Springer.

Moortgat, M. (2009). Symmetric Categorial Grammar. *Journal of Philosophical Logic*. (To appear)

Moot, R. (2007). *Proof nets for display logic.* (CoRR abs/0711.2444)

Morrill, G., & Fadda, M. (2008). Proof nets for basic discontinuous Lambek calculus. *Journal of Logic and Computation*.

Morrill, G., Fadda, M., & Valentin. (2007). Nondeterministic Discontinuous Lambek Calculus. In *Proceedings of the Seventh International Workshop on Computational Semantics (IWCS7).* Tilburg.

Reinhart, T. (1979). Syntactic domains for semantic rules. *Formal Semantics and Pragmatics for Natural Languages*, 107-130.

Reinhart, T. (1997). Quantifier Scope: How Labor is Divided Between QR and Choice Functions. *Linguistic and Philosophy*, *20*(4), 335-397.

Ruys, E., & Winter, Y. (2009). Handbook of Philosophical Logic. In D. Gabbay (Ed.), (chap. Quantifier Scope in Formal Linguistics). (To appear)

Sacerdoti Coen, C. (2006). Explanetion in Natural Language of $\bar{\lambda}\mu\widetilde{\mu}$ terms. In *Fourth International Conference on Mathematical Knowledge Management* (Vol. 3863). Berlin/Heidelberg: Springer.

Schröder. (1980). *Verlesungen über die Algebra der Logik (Teachings on the Algebra of Logic)* (Vol. 1). Leibzig: Teubner Press.

Shan, C. (2005). *Linguistic side effects*. Harvard University.

Steedman, M. (2000). *The Syntactic Process*. Cambridge, MA: MIT Press.

Szabolcsi, A. (Ed.). (1997). *Ways of scope taking*. Berlin/Heidelberg: Springer.

Szabolcsi, A. (2008). Scope and Binding. In C. Maienborn, K. von Heusinger, & P. Portner (Eds.), *Semantics: An International Handbook of Natural Language Meaning.* Mouton de Gruyter.

van Benthem, J. (1987). Categorial Grammar and Lambda Calculus. In D. Skordev (Ed.), *Mathematical Logic and its Applications* (p. 39-60). Plenum, New York.

van Benthem, J. (1988). The Lambek Calculus. In R. Oehrle, E. Bach, & D. Wheeler (Eds.), *Categorial Grammars and Natural Language Structures* (p. 35-68). Dordrecht: Reidel Publishing Company.

Vermaat, W. (2005). *The logic of variation. a cross-linguistic account of wh-question formation*. Utrecht: UiL OTS, Utrecht.