Chapter 8

The Lambek Calculus

8.1 Introduction

This chapter describes the logical approach in categorial grammar, a tradition that originated in Ajdukiewicz (1935) and was further developed in Bar-Hillel (1953) and Lambek (1958). This chapter will only give a brief introduction, Casadio (1988) gives an account of the historical roots of this field, while Moortgat (1997) and Buszkowski (1997) offer a more comprehensive overview.

General combinatory grammar is essentially a rule based approach to linguistic analysis, where a finite collection of unary type transitions and binary type combinators are postulated as *primitive* rule schemata, some examples of these rules were given in Chapter 6.

In contrast to GCG, Lambek systems do not need explicit rules defining grammatical composition. Instead, they rely on a fixed 'logical' component and a variable 'structural' component. The pure logic of residuation **NL** ('non-associative Lambek') captures the fixed logical component:

Definition 8.1 *The pure logic of residuation* **NL** *(Lambek (1961)).*

$$\begin{array}{rl} \text{(REFL)} & A \to A, \\ \text{(TRANS)} & \textit{if } A \to B \textit{ and } B \to C\textit{, then } A \to C, \\ \text{(RES)} & A \to C/B \textit{ if and only if } A \bullet B \to C \textit{ if and only if } B \to A\backslash C. \end{array}$$

This version of categorial grammar is known as the *deductive* approach. Among the reasons in favor of this approach is systematicity of the relation between syntax and semantics.[1] Semantics in Montague's tradition associates each syntactic rule with a semantic one; in the deductive approach this correspondence is more strict than the usual notion of compositionality (see Janssen

[1] Note that Lambek calculi have two kinds of semantics; the 'meaning' kind of semantics discussed here, and a *relational* semantics that yields soundness and completeness of the calculi with respect to Kripke-style relational models, see e.g. Došen (1992).

(1997)), it makes use of the Curry-Howard correspondence between proofs and types.[2] Also note that Tiede (1998) argues for the necessity of introduction rules for any compositional treatment of natural language along these lines.

By relaxing sensitivity of **NL** in a number of (linguistically relevant) dimensions one can obtain other categorial type logics. By adding to the pure logic of residuation a postulate licensing commutative resource management, one obtains freedom in the dimension of linear precedence. The resulting logic is called **NLP**. Adding a postulate licensing associative resource management yields the logic **L**, which provides freedom in the dimension of immediate dominance. Combining these postulates yields the logic **LP**, which obviously licenses both commutative and associative resource management.

Definition 8.2 ***Associativity and Commutativity Postulates***

$$\begin{array}{rl} \text{(ASS)} & (A \bullet B) \bullet C \leftrightarrow A \bullet (B \bullet C), \\ \text{(COMM)} & A \bullet B \rightarrow B \bullet A \end{array}$$

By necessity, GCG systems are only approximations of logics such as **L** and **LP**. These logics have been shown not to be *finitely axiomatizable* (Zielonka (1989, 1981)), this means no finite number of combinators together with Modus Ponens can equal their deductive strength.

As noted in Lambek (1988a), Lifting is a closure operation as it enjoys the following properties (we write A^B for either $B/(A\backslash B)$ or $(B/A)\backslash B$):

$$\begin{gathered} A \rightarrow A^B, \\ (A^B)^B \rightarrow A^B, \\ A \rightarrow C, \text{ implies } A^B \rightarrow C^B. \end{gathered}$$

Note that in general $A^B \not\rightarrow A$, which implies that, during a derivation, once a primitive type is lifted it cannot be lowered anymore. By convention, the typeraising of A to $X/(A\backslash X)$ may be written as $A^{l,X}$.

The type-raising laws can be used to lift the proper noun category (n) to the nominal phrase category ($s/(n\backslash s)$), as in Montague (1974) and van Benthem (1983, 1984), also see Dowty (1988) for other examples in linguistic analysis.

The calculus **LP** was introduced in van Benthem (1986a) because of its natural relation with a fragment of the lambda calculus, but there is also linguistic motivation for introducing commutativity. Also see van Benthem (1987).

All permutation closures of context-free languages are recognizable in **LP** (van Benthem (1991)). Also note that the languages expressible in **NL** are precisely the context-free languages (Buszkowski (1986), also see Kandulski (1988)), the same holds for **L** (Pentus (1993b)). These formalisms do not have the necessary expressive power to capture natural languages (which require at

[2]The Curry-Howard correspondence originated in the influential Howard (1980) and Curry and Feys (1958). The former, and a section from the latter, have been reprinted in the collection de Groote and Lamarche (1995), which is highly recommended to anyone interested in this topic. Also see Sørensen and Urzyczyn (1998).

1	Application:	$A/B \bullet B \to A,\ B \bullet B\backslash A \to A$
2	Co-application:	$A \to (A \bullet B)/B,\ A \to B\backslash(B \bullet A)$
3	Monotonicity of $\bullet$:	if $A \to B$ and $C \to D$, then $A \bullet C \to B \bullet D$
4	Isotonicity of $\cdot/C$, $C\backslash\cdot$:	if $A \to B$, then $A/C \to B/C$
		if $A \to B$, then $C\backslash A \to C\backslash B$
5	Antitonicity of $C/\cdot$, $\cdot\backslash C$:	if $A \to B$, then $C/B \to C/A$
		if $A \to B$, then $B\backslash C \to A\backslash C$
6	Lifting:	$A \to B/(A\backslash B),\ A \to (B/A)\backslash B$
7	Geach (main functor):	$A/B \to (A/C)/(B/C)$,
		$B\backslash A \to (C\backslash B)\backslash(C\backslash A)$
8	Geach (secondary functor):	$B/C \to (A/B)\backslash(A/C)$,
		$C\backslash B \to (C\backslash A)/(B\backslash A)$
9	Composition:	$A/B \bullet B/C \to A/C,\ C\backslash B \bullet B\backslash A \to C\backslash A$
10	Restucturing:	$(A\backslash B)/C \leftrightarrow A\backslash(B/C)$
11	(De)Currying:	$A/(B \bullet C) \leftrightarrow (A/C)/B$,
		$(A \bullet B)\backslash C \leftrightarrow B\backslash(A\backslash C)$
12	Permutation:	if $A \to B\backslash C$ then $B \to A\backslash C$
13	Exchange:	$A/B \leftrightarrow B\backslash A$
14	Preposing/Postposing:	$A \to B/(B/A),\ A \to (A\backslash B)\backslash B$
15	Mixed Composition:	$A/B \bullet C\backslash B \to C\backslash A,\ B/C \bullet B\backslash A \to A/C$

Figure 8.1: Characteristic theorems and derived inference rules for **NL** (1-6); **L** (1-11); **NLP** (1-6, 12-14); **LP** (1-15).

least mild context-sensitivity). Therefore more expressive variants have been proposed, for example the multi-modal variant (MMCG) where applicability of postulates is controlled through the use of modal operators in the lexicon. This variant, without restrictions on postulates, is a Turing-complete system (Carpenter (1999)).[3] Recently some restrictions on postulates have been proposed that restrict expressive power to (mild) context-sensitivity, see Moot (2002), we will discuss these in Subsection 8.5.

The presentation of **LP** used here is due to Kurtonina and Moortgat (1997), it takes **NL**$\Diamond$, which is the system **NL** (Figure 8.2) extended with unary connectives (Figure 8.3) and modalities for the binary connectives, as the 'base logic'[4] and adds associativity and commutativity postulates (Figure 8.4).

[3]The proof of this fact is based on an embedding of the full system of multiplicative linear logic in MMCG, which necessitates the introduction of copying and deletion postulates. Such postulates have little linguistic use (see also the discussion in Section 9.6) and do not fit into the program of exploring the *substructural* landscape between the base logic **NL** and **LP**. A remark pertaining to this issue can be found in Lambek (1993).

[4]Note that, unless otherwise stated, the empty sequent is not allowed, i.e. $\vdash A$ may not occur in any derivation. Lambek variants which allow the empty sequent have $\emptyset$ added as subscript, for example **NL** with empty sequent is written as $\mathbf{NL}_\emptyset$. Allowing the empty sequent permits clearly undesirable derivations such as

A	very	book	
$np/n,$	$(n/n)/(n/n),$	n	$\vdash_{\mathbf{L}_\emptyset} np$

however, these calculi are sometimes used in proofs for technical reasons.

$$\frac{}{A \vdash A} \quad \frac{\Delta \vdash A \qquad \Gamma[A] \vdash C}{\Gamma[\Delta] \vdash C}\,[Cut]$$

$$[/I]\,\frac{(\Gamma, B) \vdash A}{\Gamma \vdash A/B} \quad \frac{\Gamma \vdash A/B \qquad \Delta \vdash B}{(\Gamma, \Delta) \vdash A}\,[/E]$$

$$[\backslash I]\,\frac{(B, \Gamma) \vdash A}{\Gamma \vdash B\backslash A} \quad \frac{\Gamma \vdash B \qquad \Delta \vdash B\backslash A}{(\Gamma, \Delta) \vdash A}\,[\backslash E]$$

$$[\bullet I]\,\frac{\Gamma \vdash A \qquad \Delta \vdash B}{(\Gamma, \Delta) \vdash A \bullet B} \quad \frac{\Delta \vdash A \bullet B \qquad \Gamma[(A, B)] \vdash C}{\Gamma[\Delta] \vdash C}\,[\bullet E]$$

Figure 8.2: Sequent-style presentation of the natural deduction rules for **NL** (with product).

$$[L\Diamond_i]\,\frac{\Gamma[\langle A\rangle^i] \vdash C}{\Gamma[\Diamond_i A] \vdash C} \quad \frac{\Gamma \vdash C}{\langle\Gamma\rangle^i \vdash \Diamond_i C}\,[R\Diamond_i]$$

$$[L\Box_i^{\downarrow}]\,\frac{\Gamma[A] \vdash C}{\Gamma[\langle\Box_i^{\downarrow} A\rangle^i] \vdash C} \quad \frac{\langle\Gamma\rangle^i \vdash C}{\Gamma \vdash \Box_i^{\downarrow} C}\,[R\Box_i^{\downarrow}]$$

Figure 8.3: Unary connectives.

This notation facilitates some of the steps in our (syntactic) proofs of non-learnability in Chapter 9, and makes the derivations more explicit.

Each of the systems **NL**, **L**, **LP** and **NLP** have their virtues in linguistic analysis, but in isolation none of them provides a basis for a plausible theory of grammar. By combining properties of these systems in a controlled way, mixed, *multimodal* systems are obtained that overcome the limitations of these simple systems. This multimodal style of reasoning was developed in Moortgat and Morrill (1991), Moortgat and Oehrle (1993), Moortgat and Oehrle (1994) and Hepple (1994), among others.

While the logical approach to categorial grammar is perhaps not really part of the linguistic mainstream, it has however influenced the development of other grammar formalisms: Johnson (1999) gave a reinterpretation of Lexical

$$[comm]\,\frac{(\Gamma, \Delta) \vdash A}{(\Delta, \Gamma) \vdash A} \quad \frac{((\Gamma, \Delta), \Theta) \vdash A}{(\Gamma, (\Delta, \Theta)) \vdash A}\,[ass]$$

Figure 8.4: Postulates for **LP**.

Functional Grammar as a type-logic, Partial Proof Tree Grammars (Joshi and Kulick (1997)), and a multi-modal interpretation of Derivational Minimalism (Stabler (1997)) in Lecomte (2001). There are also formalisms strongly related to Lambek systems, like pregroup grammars (Lambek (1999); Casadio and Lambek (2002)), pomset-logic (Lecomte and Retoré (1995); Schena (1997)), linear logic and proof nets, these are all outside the scope of this thesis and ware only mentioned for completeness.[5]

8.2 Models for the Lambek Calculus

Algebraic interpretations of the Lambek calculi will be used in Chapter 9, in this section the definitions we are interested in[6] are given. Two kinds of models are defined: free groups and powerset residuated groupoids (or semi-groups), a special case of residuated groupoids (see Buszkowski (1997) for details).

Free Group Interpretation. Let FG denote the free group with generators Pr, operation . and neutral element I. We associate with each formula C an element in FG written $[C]$ as follows: $[p] = p$ for p atomic, $[C_1 \backslash C_2] = [C_1]^{-1}.[C_2]$, $[C_1/C_2] = [C_1].[C_2]^{-1}$, $[C_1 \bullet C_2] = [C_1].[C_2]$. We extend the notation to sequents by $[C_1, C_2, \ldots, C_n] = [C_1].[C_2]. \ldots .[C_n]$. The free group FG provides models for L: if $\Gamma \vdash_L C$ then $[\Gamma] =_{FG} [C]$.

Powerset Residuated Groupoids and Semigroups. Let $(M, .)$ be a groupoid, and let $\mathcal{P}(M)$ denote the powerset of M. A *powerset residuated groupoid* over $(M, .)$ is the structure $(\mathcal{P}(M), \circ, \Rightarrow, \Leftarrow, \subseteq)$ such that for $X, Y \subseteq M$:

$$\begin{array}{lcl} X \circ Y & = & \{x.y \mid x \in X, y \in Y\} \\ X \Rightarrow Y & = & \{y \in M \mid (\forall x \in X) x.y \in Y\} \\ Y \Leftarrow X & = & \{y \in M \mid (\forall x \in X) y.x \in Y\} \end{array}$$

If $(M, .)$ is a semi-group (. is associative), then this structure is a *powerset residuated semi-group*. If $(M, .)$ has a unit I (i.e. $\forall x \in M, I.x = x.I = x$) this structure is a *powerset residuated groupoid with unit* (with $\{I\}$ as unit).

Interpretation. Given a powerset residuated groupoid $(\mathcal{P}(M), \circ, \Rightarrow, \Leftarrow, \subseteq)$, an *interpretation* is a map from primitive types p to elements $[\![p]\!]$ in $\mathcal{P}(M)$ that is extended to types and sequences in a natural way:

$$\begin{array}{rcl} [\![C_1 \backslash C_2]\!] & = & [\![C_1]\!] \Rightarrow [\![C_2]\!] \\ [\![C_1/C_2]\!] & = & [\![C_1]\!] \Leftarrow [\![C_2]\!] \\ [\![C_1 \bullet C_2]\!] & = & [\![C_1]\!] \circ [\![C_2]\!] \\ [\![C_1, C_2, \ldots, C_n]\!] & = & [\![C_1]\!] \circ [\![C_2]\!] \circ \ldots \circ [\![C_n]\!] \end{array}$$

[5] Negative learnability results for classes of pregroup grammars can be found in Bechet and Foret (submitted).

[6] There are other models available, most notably the relational model based on Kripke frames, see Kurtonina (1995). These are outside the scope of this thesis and will not be discussed in any detail.

If $(M, .)$ is a groupoid with an identity I, we add $[\![\Lambda]\!] = \{I\}$ for the empty sequence Λ and get a model property for $\mathbf{NL}_\emptyset$: if $\Gamma \vdash_{\mathbf{NL}_\emptyset} C$ then $[\![\Gamma]\!] \subseteq [\![C]\!]$. If $(M, .)$ is a semi-group, we have a similar model property for $\mathbf{L}$: if $\Gamma \vdash_{\mathbf{L}} C$ then $[\![\Gamma]\!] \subseteq [\![C]\!]$.

8.3 Substitution in the Lambek Calculus

It would be natural to use Kanazawa's approach to learnability questions for CCG grammars to Lambek grammars, since these frameworks seem to have so much in common. In order to do this, precise notions of substitution (unification) as well as a notion of structure language need to be defined. The latter will be dealt with in the next section. This section addresses the former, by discussing work from Foret (2001a,b).

Definition 8.3 *The* join-equivalence*, written $\sim$, is defined as*

$$t \sim t' \text{ if and only if } \exists t_1, \ldots, t_n \text{ such that } t = t_1, t' = t_n, (t_i \vdash t_{i+1} \vee t_{i+1} \vdash t_i),$$

for $i < n$. Types $t_1, \ldots, t_n$ are said to be conjoinable *whenever there is a type t such that $t_i \vdash t$, for each $i \leq n$. In this case t is said to be a* join *for $t_1, \ldots, t_n$.*

The following proposition, defining what is known as the *Diamond property*, is from Lambek (1958):[7].

Proposition 8.4 *Let t_1 and t_2 be two categorial types. The following statements are equivalent:*

1. *t_1 and t_2 are conjoinable.*
2. *there exists a type t' such that $t' \vdash t_1$ and $t' \vdash t_2$.*

Proposition 8.5 *Let t_1 and t_2 be two categorial types. The statement $t_1 \sim t_2$ is equivalent to 1 and 2 of Proposition 8.4.*

We will use this proposition combined with the following completeness result by Pentus which characterizes $\sim$ by groups:

Theorem 8.6 *For any two types t, t', $t \sim [\![t']\!] \leftrightarrow =_{FG} [\![t']\!]$.*

Definition 8.7 *The relation $\|=$ on types t_1, t_2 is defined by:*

$$t_1 \|= t_2 \text{ if and only if there is a substitution } \sigma \text{ such that } t_1 \vdash_{\mathbf{L}} \sigma[t_2].$$

Note that $\|=$ is reflexive and transitive.

[7] An alternative proof of this proposition can be found in Pentus (1993a).

8.3.1 $\|=$-Unifiability

Definition 8.8 *Two types t_1, t_2 are said to be* $\|=$-unifiable *whenever there exists a type t and substitution σ such that $t \vdash \sigma[t_1]$ and $t \vdash \sigma[t_2]$.*

The substitution σ is said to be a $\|=$-unifier *of t_1 and t_2 and t is a* $\|=$-unificand *of t_1 and t_2.*

There are strong links between $\|=$-unification and E-unification, i.e. unification under an equational theory.[8] Depending on the nature of the equational theory, E-unification of two terms can yield an infinite number of mgu's (for example under associativity), and even when there are only finitely many mgu's, finding them may be a computationally difficult task.

Therefore, a naive application of Kanazawa's approach to classes of Lambek grammars yields a host of problems. In fact, as we shall see in Chapter 9, even the rigid subclasses of **L**, **NL** and **LP** are not learnable (from strings).[9]

8.4 The Structure Languages of Non-Associative Lambek Grammars

In Tiede (1999a, 1998) a notion of structure language (i.e., strong generative capacity or SGP) for both **L** and **NL** was proposed. This is a highly non-trivial matter, since in the context of a logical grammar formalism derivations are *proofs*, and proofs are generally (spuriously) nondeterministic.[10]

His approach differs from the one found in Buszkowski (1997) which defines functor-argument structures (so-called f-structures) and phrase structures (p-structures). In the case of Lambek grammars this definition has undesirable consequences, most notably the structural completeness theorem: any (associative) Lambek grammar that generates string s can assign any binary tree that has $|s|$ (non-ε) leaves as structural description for s.[11] This fact has caused some researchers to turn their attention to **NL**-grammars, since these

[8] E-unification plays an important role in automated theorem provers with 'built-in' theories (see e.g., Plotkin (1972)), Stickel (1985)) and in logic programming with equality (see e.g., Jaffar et al. (1984)). See Baader and Siekmann (1993) for a comprehensive overview.

[9] In this context we'd also like to mention the exploratory work in Moortgat (2001), where an algorithm is proposed that takes the unification approach 'as far as possible' and, in the case that no rigid grammar can be obtained this way, invents postulates that equate non-unifiable types assigned to the same symbol. There are as of yet no learnability results or independent characterization of learnable classes in relation tho this approach.

[10] One way of dealing with this spurious non-determinism is the use of *proofnets*, see eg. Moot (2002). However, for the present purposes proofnets are not very useful: depending on the calculus one proofnet may be applicable to several proofs that are non-equivalent, for example when product is used.

They also do not offer a (tree) automata-theoretic perspective, since they produce graphs, and, to the best of the author's knowledge, there exists little work on learning graph languages.

[11] This is easy to see; lifting allows reversal of the direction of application, and associativity allows any rearrangement of the derivation tree.

are not structurally complete. However, Buszkowski (1997) also shows that **NL**-grammars cannot extend the strong generative capacity of context-free grammars, which is unfortunate given the current trend in computational linguistics to 'squeeze more strong power out of a formal system without increasing its weak generative power'.

In Tiede (1999a, 1998) it was shown that considering (the normal forms of) *natural deduction proof trees* as the structures assigned by Lambek grammars yields a notion of SGP that goes beyond that of context-free grammars[12] (and may allow characterization of *crossing dependencies*), but does not suffer from a collapse into structural completeness.[13] Another pleasant property is that this notion of SGP distinguishes between proofs if and only if the semantic term they produce differs (cf Hendriks (1993)), just like proofnets.

A direct definition of natural deduction proof trees for **L** would involve either defining proof trees to contain parenthesised structures or giving a complicated list of side conditions for the introduction rules. The tree format will therefore be defined *indirectly* by defining a translation from SND to to natural deduction trees. Well-formed trees for **NL** will be defined as a subset of well-formed trees for **L**.

Definition 8.9 *Let t be a proof tree of the associative Lambek calculus. We define its translation into a proof in SND, t^*, as follows:*

1. *if $t = A$, i.e. t is an instance of $[ID]$, then $t^* = A \vdash A$,*

2. *if* $t = \dfrac{t_1 \quad t_2}{A}\ [/E]$ *where* $t_1 = \dfrac{\vdots}{A/B}$ *and* $t_2 = \dfrac{\vdots}{B}$.

   We have $t_1^* = \dfrac{\vdots}{\Gamma \vdash A/B}$ *and* $t_2^* = \dfrac{\vdots}{\Delta \vdash B}$, *where* Γ *and* Δ *are the uncancelled assumptions of* t_1 *and* t_2, *respectively, in the order in which they occur. We define* $t^* = \dfrac{t_1^* \quad t_2^*}{(\Gamma, \Delta) \vdash A}\ [/E]$.

3. *If* $t = \dfrac{t_1}{A/B}\ [/I]$, *where* $t_1 = \dfrac{\begin{array}{c} B \\ \vdots \\ A \end{array}}{}$, *then consider* $t_1^* = \dfrac{\vdots}{(\Gamma, B) \vdash A}$. *We define* $t^* = \dfrac{t_1^*}{\Gamma \vdash A/B}\ [/I]$.

4. $[\backslash E]$ *and* $[\backslash I]$ *are defined analogously.*

[12] In van Benthem (1991) the possibility of extended SGP was mentioned as a motivation for studying Lambek grammars; the *weak* generative capacity of CCG, **L** and **NL** is exactly that of context-free grammars.

[13] But see Joshi (2002) for a critique; the crossing dependencies allowed by this definition of SGP are 'very degenerate', i.e. they can only connect a lexical item with a lexically empty element.

Definition 8.10 *A natural deduction proof tree t is a well-formed proof of* **NL** *if its translation t^* (according to Definition 8.9) is a well-formed proof according to Figure 8.2.*

Since there are strong non-learnability results for **L**, we will only be concerned with **NL** in Chapter 9.

Definition 8.11 *A natural deduction proof tree is* in (β-η-)normal form *(or* (β-η-)normal*) if none of its subtrees are of the form of any of the following trees:*

$$\dfrac{\dfrac{\begin{array}{c}[B]\\ \vdots\\ A\end{array}}{A/B}\,[/I] \quad B}{A}\,[/E] \qquad \dfrac{B \quad \dfrac{\begin{array}{c}[B]\\ \vdots\\ A\end{array}}{B\backslash A}\,[\backslash I]}{A}\,[\backslash E] \qquad \dfrac{\dfrac{A/B \quad [B]}{A}\,[/E]}{A/B}\,[/I] \qquad \dfrac{[B] \quad \dfrac{B\backslash A \quad [B]}{A}\,[\backslash E]}{B\backslash A}\,[\backslash I]$$

Note that every proof tree can be converted into a unique normal form proof tree. The following proposition concerns the very important *subformula property*:

Proposition 8.12 ***Subformula property*** *Every formula that occurs in a normal form natural deduction proof or cut-free sequent calculus proof is either a subformula of the uncanceled assumptions or of the conclusion.*

Definition 8.13 *A* track *of a proof tree T is a sequence of formulae $A_0, \ldots, A_n$ such that*

1. *A_0 is a leaf of T,*
2. *for $0 \leq i < n$,*
   (a) *A_{i+1} is immediately below A_i,*
   (b) *A_i is not the minor premise of an $[\backslash E]$ or $[/E]$ application,*
3. *T is maximal, i.e. not a proper part of another track.*

Note that in a normal proof every formula occurence belongs to a track.

The following result has a well-known analogy in the field of linear logic:

Proposition 8.14 *(Also see Prop 2.28 in Tiede (1999a)) Every track that is part of an* **L** *or* **NL** *normal form proof tree begins*[14] *with an E-part, i.e. $A_0, \ldots, A_{i-1}$, has a minimal formula A_i after the E-part and may have an I-part after that, $A_{i+1}, \ldots, A_n$.*

[14] When read top-down.

Proposition 8.15 *The number of different tracks occuring in normal form proofs for a given* **L** *or* **NL** *grammar is finite.*

Definition 8.16 *An* incomplete proof tree *is either*

1. *a track, or*
2. *an incomplete proof tree with a track t inserted such that t starts with formula A and the insertion point is the minor premise of an application of $[/E]$ or $[\backslash E]$, where the major premise is B/A or $A\backslash B$, respectively.*

Definition 8.17 *An incomplete proof tree is* saturated *just if for every occurence of $[/I]$ or $[\backslash I]$ in the tree has a corresponding leaf in the tree that it cancels.*

Definition 8.18 *A saturated incomplete proof tree (or* **NL***-track) is* minimal *just if*

1. *it is a track that contains no introductions, or*
2. *a saturated incomplete proof tree T that contains introduction rules, such that removing any subtree that is an incomplete proof tree would cause T to not be saturated.*

Proposition 8.19 *The number of minimal saturated incomplete proof trees to which a track in a normal form proof tree for an* **NL** *grammar can be extended is finite.*

Since **NL**-tracks behave just like tracks in classical categorial grammar, Kanazawa's approach can be applied in a straightforward way, as we shall see in Chapter 9.

8.5 Multimodal Systems

As was noted in the beginning of this chapter, it is possible to control structural relaxation with the use of modal operators. However, the resulting calculi can be overly expressive,[15] to the point of undecidability. In Moot (2002) a natural restriction on structural rules is proposed that turns out to restrict expressive power to more linguistically plausible levels.

The restriction requires that the left-hand side of a structural conversion contains at least as many unary connectives as the right-hand side, the resulting postulates are called *non-expanding*. First 'length' is defined in terms of unary connectives:

[15] See Kurtonina and Moortgat (1997) for the relations between the different systems obtained by the use of such operators.

Aso see Jäger (1998) for results on the *strong* generative capacity of multimodal systems.

Definition 8.20 *Given an antecedent Ξ, its length is defined as*

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

This definition can be generalized over n-ary configurations: $length(*(\Delta_1, \ldots, \Delta_n)) = length(\Delta_1) + \ldots + length(\Delta_n) + n$.

Definition 8.21 *the logic $\mathbf{NL}\Diamond_{\mathcal{R}-}$ is the logic $\mathbf{NL}\Diamond_{\mathcal{R}}$ where for every structural rule $R \in \mathcal{R}$*

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

the following holds:

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

Structural rules with this property are called non-expanding.

Note that in the case of binary modalities this restricts postulates to be *linear*, and that a structural rule for -for example- strong distributivity is non-expanding according to this definition.

An embedding for lexicalized context-sensitive grammars is offered in Moot (2002):

Definition 8.22 *From a lexicalized context-sensitive grammar G we generate the corresponding multimodal Lambek calculus $\mathcal{M}(G)$ as follows: $\mathcal{M}(G)$ has one unary mode for every nonterminal of G and a single binary mode.*

Every lexicalization rule $A \mapsto \beta$ correspond to a lexical entry of the form

$$lex(\beta) = a \backslash \Box^{\downarrow}_{A} a$$

The goal formula of $\mathcal{M}(G)$ is $a\backslash\Box^{\downarrow}_{S}a$ where S is the mode corresponding to the start symbol of G. The calculus $\mathcal{M}(G)$ has a structural rule for every grammar rule $A_1 \ldots A_n \mapsto_{R1} B_1 \ldots B_m$ of G.

$$\frac{\Gamma[\langle \ldots \langle\Delta\rangle^{A_n} \ldots\rangle^{A_1}] \vdash C}{\Gamma[\langle \ldots \langle\Delta\rangle^{B_m} \ldots\rangle^{B_1}] \vdash C}\ [R1]$$

Because $m > 0$ and $n \leq m$, ths is a valid $\mathbf{NL}\Diamond_{\mathcal{R}-}$ rule.

Furthermore, the structural rule component of $\mathcal{M}(G)$ contains one of the structural rules for associativity for the single binary mode:

$$\frac{\Gamma[\Delta_1 \circ (\Delta_2 \circ \Delta_3)] \vdash C}{\Gamma[(\Delta_1 \circ \Delta_2) \circ \Delta_3] \vdash C}\ [Ass2]$$

and the structural rule of $[K1]$ for every mode $A \in N$:

$$\frac{\Gamma[\langle\Delta_1\rangle^A \circ \Delta_2] \vdash C}{\Gamma[\langle\Delta_1 \circ \Delta_2\rangle^A] \vdash C}\ [K1]$$

Lemma 8.23 *Given a lexicalized context-sensitive grammar G, the corresponding (by Definition 8.22) multimodal Lambek calculus $\mathcal{M}(G)$ generates the same language as G.*

Theorem 8.24 *The parsing problem for $\mathbf{NL}\Diamond_{\mathcal{R}-}$ is quivalent to the parsing problem for context-sensitive grammars.*

Note that, although these are obviously interesting results, the grammars obtained through this embedding are in some sense unnatural. They basically abuse postulates by treating them as rules, resulting in a rather circuitous way of using a rule-based system. We will use this embedding in the next chapter to define a learnable class of multimodal CG, but this should be regarded as a quite trivial result.

Identity

$$\frac{}{A \vdash A}\ [Ax] \qquad \frac{\Gamma[B] \vdash C \quad \Delta \vdash B}{\Gamma[\Delta] \vdash C}\ [Cut]$$

Unary Connectives

$$\frac{\Gamma[\langle A \rangle^i] \vdash C}{\Gamma[\Diamond_i A] \vdash \Diamond_i C}\ [L\Diamond_i] \qquad \frac{\Gamma \vdash C}{\langle \Gamma \rangle^i \vdash C}\ [R\Diamond_i]$$

$$\frac{\Gamma[A] \vdash C}{\Gamma[\langle \Box_i A \rangle^i] \vdash C}\ [L\Box_i^{\downarrow}] \qquad \frac{\langle \Gamma \rangle^i \vdash C}{\Gamma \vdash \Box_i C}\ [R\Box_i^{\downarrow}]$$

Binary Connectives

$$\frac{\Gamma[A \circ_i B] \vdash C}{\Gamma[A \bullet_i B] \vdash C}\ [L\bullet_i] \qquad \frac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma \circ_i \Delta \vdash A \bullet_i B}\ [R\bullet_i]$$

$$\frac{\Delta \vdash B \quad \Gamma[A] \vdash C}{\Gamma[A/_i B \circ_i \Delta] \vdash C}\ [L/_i] \qquad \frac{\Gamma \circ_i B \vdash A}{\Gamma \vdash A/_i B}\ [R/_i]$$

$$\frac{\Delta \vdash B \quad \Gamma[A] \vdash C}{\Gamma[\Delta \circ_i B \backslash_i A] \vdash C}\ [L\backslash_i] \qquad \frac{B \circ_i \Gamma \vdash A}{\Gamma \vdash B \backslash_i A}\ [R\backslash_i]$$

Structural Rules

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

Figure 8.5: The sequent calculus $\mathbf{NL}\Diamond_{\mathcal{R}}$.