Heads and Phrases. Type Calculus for Dependency and Constituent Structure

Michael Moortgat & Glyn Morrill
Research Institute for Language and Speech (OTS, Utrecht)

Draft Version. September 1991.
Revised version to appear in *Journal of Logic, Language and Information*

Abstract

From a logical perspective, categorial type systems can be situated within a landscape of substructural logics — logics with a structure-sensitive consequence relation. Research on these logics has shown that the inhabitants of the substructural hierarchy can be systematically related by embedding translations on the basis of *structural modalities*. The modal operators offer controlled access to stronger logics from within weaker ones by licensing of structural operations. Linguistic material exhibits structure in dimensions not covered by the standard structural rules. The purpose of this paper is to generalize the modalisation and licensing strategy to two such dimensions: phrasal structure and headedness. Phrasal domain-sensitive type systems capture the notion of *constituent structure*; constituency relaxation can be licensed via an associativity modality. The opposition between heads and non-heads introduces *dependency structure*, an autonomous dimension of linguistic structure which may cross-cut semantic function-argument asymmetry; flexibility with respect to dependency structure can be encoded by a licensing balance modality. The interplay between constituent and dependency sensitivity leads to characterizations of well-formedness for linguistic objects structured as trees or strings, and headed trees or headed strings.

With respect to overall grammatical organization the paper represents a perspective which takes as its basis type systems with the finest structural discrimination that it is required to make, together with the idea that relaxation of structure sensitivity (i.e. increase in structure flexibility), can be reduced to licensing lexical type declarations. We discuss phenomena of non-constituent coordination, prosodic restructuring and metrical organization which support such a view.

1 Substructural logics and structural modalities

This paper is concerned with a family of categorial type systems which, from a logical point of view, can be situated within a landscape of *substructural* logics. Substructural logics are characterized by structure-sensitive consequence relations: whether or not a conclusion A is derivable from assumptions Δ depends not just on the assumptions *per se* but on the way they are configured, i.e. on their organisational structure.

Consider presentation of a consequence relation in terms of Gentzen style sequent calculus. The architecture of sequent calculus consists of three general components:

an identity group, logical rules, and structural rules. The identity group, containing the identity axiom and the Cut rule in (1), capture the reflexivity and (contextualized) transitivity of the derivability relation. The logical rules provide for each of the connectives a rule of *use* and a rule of *proof*. The logical rules are formulated in such a way that Cut is eliminable, in the sense that all theorems are obtainable without use of the Cut rule. From Cut elimination it follows that Cut is a rule *admissible* to the Cut free presentation, i.e. its addition gives rise to no new theorems; but it is not in general *derivable* from Gentzen rules of use and proof. The structural rules tell us what aspects of the representation (usually a list) of assumptions are irrelevant for the characterization of the consequence relation, i.e. which representations signify the same configurations. Different choices of structural rules give rise to different logics even while the logical rules are invariant.[1]

The calculus of syntactic types of Lambek [14] (henceforth **L**) and Intuitionistic Logic (henceforth **IL**) occupy two extremes of a spectrum: the Lambek calculus has an empty set of structural rules, and thus is sensitive to all aspects of the list representation of the database; Intuitionistic Logic admits structural rules of Permutation and Contraction, which remove order-sensitivity and sensitivity for the multiplicity of assumptions, and Weakening, which preserves derivability under growth of the database, i.e. which engenders a *monotonic* consequence relation. Addition of just permutation to **L** gives rise to (intuitionstic) Linear Logic, the implicational fragment of which is also known as Lambek/van Benthem calculus **LP**.

Below we present Gentzen sequent rules for **L** and structural rules the addition of which transforms it into **IL**. Note that addition of permutation already dissolves the distinction in directionality present in the syntactic calculus.

Identity rules

$$[\mathrm{Ax}]\frac{}{A \Rightarrow A} \quad [\mathrm{Cut}]\frac{\mathrm{T} \Rightarrow A \quad \Delta, A, \Gamma \Rightarrow C}{\Delta, \mathrm{T}, \Gamma \Rightarrow C} \tag{1}$$

Logical rules for $/, \bullet, \backslash$

$$[\mathrm{R}/]\frac{\mathrm{T}, B \Rightarrow A}{\mathrm{T} \Rightarrow A/B} \quad \frac{\mathrm{T} \Rightarrow B \quad \Gamma, A, \Lambda \Rightarrow C}{\Gamma, A/B, \mathrm{T}, \Lambda \Rightarrow C}[\mathrm{L}/]$$

$$[\mathrm{L}\bullet]\frac{\Gamma, A, B, \Lambda \Rightarrow C}{\Gamma, A \bullet B, \Lambda \Rightarrow C} \quad \frac{\Delta \Rightarrow A \quad \mathrm{T} \Rightarrow B}{\Delta, \mathrm{T} \Rightarrow A \bullet B}[\mathrm{R}\bullet] \tag{2}$$

$$[\mathrm{R}\backslash]\frac{B, \mathrm{T} \Rightarrow A}{\mathrm{T} \Rightarrow B\backslash A} \quad \frac{\mathrm{T} \Rightarrow B \quad \Gamma, A, \Lambda \Rightarrow C}{\Gamma, \mathrm{T}, B\backslash A, \Lambda \Rightarrow C}[\mathrm{L}\backslash]$$

Structural rules

$$\frac{\Gamma, A, B, \Delta \Rightarrow C}{\Gamma, B, A, \Delta \Rightarrow C}[\mathrm{P}] \tag{3}$$

$$\frac{\Gamma, A, A, \Delta \Rightarrow B}{\Gamma, A, \Delta \Rightarrow B}[\mathrm{C}] \tag{4}$$

$$\frac{\Gamma, \Delta \Rightarrow B}{\Gamma, A, \Delta \Rightarrow B}[\mathrm{W}] \tag{5}$$

[1] The central role of the structural rules is stressed as follows by [12], p. 30:"...contrary to popular belief, these rules are the most important of the whole calculus, for, without having written a single logical symbol, we have practically determined the future behaviour of the logical operations".

The character of the various points in this landscape has already been quite well explored; see for example [3] [5], [6], [22] [23].

The sequent presentation given above shows characterisation of a structure-sensitive consequence relation by taking seriously the representation of a sequent antecedent and being fully explicit about allowable structural manipulation, starting from a database Δ notated by a *list* structure. Alternatively, one can eschew explicit structural rules by reading antecedents as data structures less ornate than their notations. Interpreting Δ in sequents $\Delta \Rightarrow A$ not as a list but as a *multiset* (with the sequent comma as multiset union) builds in the order-insensitivity of the consequence relation — the structural rule of Permutation becomes implicit. In a similar way, one can build in Contraction by shifting to a *set* interpretation of Δ. Finally, irrelevant assumptions are allowed by extending the Axiom sequent $A \Rightarrow A$ to $\Delta, A \Rightarrow A$, under the set interpretation of the database. The structural rule of Weakening thus becomes implicit. See [11] or [21] for some discussion.

As it has been described, the substructural landscape materializes by fixing attention on meta-logical notation. Once the basic point is grasped however, it is apparent that current possibilities are bounded only by the discriminatory power of the representation for antecedents employed so far. We can only make as many structural discriminations as can be represented. But by switching to successively richer notations the ceiling of structure-sensitivity can be raised without limit: the landscape has no end. A general framework for the study of structure-sensitive consequence relations is provided by Gabbay's Labelled Deductive Systems ([9], [10]); see also Dunn's Gaggle theory, [8]).

Linear logic and its relatives have become the focus of intensive study. Two themes of these investigations are especially relevant for the linguistic application of type systems: (i) the decomposition of connectives in the more structure-sensitive logics, and (ii) the introduction of *controlled* access to structural expressivity in the form of structural modalities. We discuss these two themes in the following paragraphs with respect to the rules of Permutation, Contraction and Weakening, and then generalize them beyond these structural options: we introduce more structural discrimination than above, and observe the same development of themes (i) and (ii).

1.1 Decomposition of connectives

In the preceding section we noticed the following general point concerning the traversal of the substructural landscape: structural rules *destroy* the organization of the type assumptions. There is an immediate dependency therefore between the structural component of our sequent calculi and the logical component. Addition of structural rules leads to a decrease in structural discrimination; connectives that can be distinguished in the more structured logics *collapse*; for example we have seen already the collapse under permutation of predictive and retroactive implications. Viewing from the other direction: the withdrawal of structural rules leads to *decomposition* of connectives; connectives with a higher degree of structure-sensitivity become distinguishable. We can illustrate the above with the various metamorphoses of the logical implication connective, which in its linguistic form underlies the functional type-constructors.

In the presence of contraction and weakening, logical implication exhibits a lack of *resource* sensitivity: assumptions can be used in a deduction as often as one wants to use them — or not at all. Withdrawing Contraction/Weakening introduces resource sensitivity, i.e. it marks the move to logics of *occurrences* and deduction on the basis of *multisets* of assumptions. This is the contrast between intuitionistic and linear implication.

Withdrawing Permutation introduces *order* sensitivity in the type inference system. The multiset logic becomes a logic of *sequences* of type occurrences. This

is the minimal structure required for application of the logic to specification of grammatical forms.

In the remainder of the paper, we take up the theme of decomposition of connectives with respect to two dimensions of linguistic structure not covered by the structural representations and rules we have discussed so far.

Firstly, we include a dimension of associativity. The associative type system **L** of [14] is ignorant of any notion of constituency. Withdrawal of the associativity assumption as for the type calculus **NL** of [15] introduces sensitivity to constituent domains. Instead of *strings* the well-formed inhabitants of types become hierarchically structured *bracketed strings*, i.e. trees. The regain of this constituent structure together with modality controlled flexibility enables resolution of puzzling coordination structure and prosodic phrasing phenomena.

Secondly, we note that the standard functional type-constructors for left and right incompleteness derive (by *residuation duality*) from a single concatenation product. Linguistic phenomena of dependency and prominence suggest a cross-cutting of the function-argument asymmetry with an autonomous head-dependent opposition. We obtain this opposition by decomposing a single product into a left-dominant and a right-dominant variant and obtaining residuation duality for both. Each tree then has a distinguished daughter that can be interpreted as the head (versus the dependent). As an example of application, we show how the introduction of headed tree adjunction provides the apparatus for the representation of hierarchical prosodic organization in terms of metrical trees.

1.2 Structural modalities

The important lesson to be learned from Linear Logic is that access to the structural operations can be reintroduced in a *type-controlled* manner in the form of extra connectives, the so-called structural *modalities* or exponentials. The 'of course' modality ('!') in Linear Logic does this for Contraction and Weakening. Kosta Došen has investigated the modalization strategy in its generality, cf. his [7]. The common logical pattern of the modalities is outlined below. As to notation: we use a general modal operator $\Box$, subscripted for specific structural operations, or unsubscripted for rules which apply to the modalities in general.

$$[\Box\mathrm{L}]\frac{\Gamma[A] \Rightarrow B}{\Gamma[\Box A] \Rightarrow B} \qquad \frac{\Box\Gamma \Rightarrow A}{\Box\Gamma \Rightarrow \Box A}[\Box\mathrm{R}] \tag{6}$$

As governers of structural operations from *within* the logic, structural modalities have both structural rules *and* logical rules. In accordance with the sequent architecture, the new modal operators need logical rules of use and proof. The rule of use $[\Box\mathrm{L}]$ tells us that at a certain point in a deduction, a flagged modal assumption $\Box A$ finds itself in a structural configuration where it can be used as an unflagged datum of type A. The rule of proof $[\Box\mathrm{R}]$ allows one to derive a flagged $\Box A$ from a database of which all the components are modalized, provided one can derive an unflagged datum of type A from this fully modalized database. In the logical rules for $\Box$ we recognize the sequent rules of the modal logic S4.

The S4 basis is shared by the distinct modalities. Next to this basis, each modality has an operational rule which performs the manipulation of the its corresponding structural operation. But the structural rules are now subject to the condition that the minimal structural configurations involved contain *licensing* modal assumptions $\Box_i A$.[2]

[2] Notice that the '!' modality (in our notation $\Box_!$) of Linear Logic is in fact a *combination* of Reuse and Waste of resources, which in principle could be considered apart, as is done in ([18], [1]) on linguistic grounds.

- Reusability of resources

$$[\Box_!\mathrm{C}]\frac{\Gamma, \Box_! A, \Box_! A, \Delta \Rightarrow B}{\Gamma, \Box_! A, \Delta \Rightarrow B} \tag{7}$$

- Waste of resources

$$\frac{\Gamma, \Delta \Rightarrow B}{\Gamma, \Box_! A, \Delta \Rightarrow B}[\Box_!\mathrm{W}] \tag{8}$$

- Permutation

$$\frac{\Gamma, A_1, A_2, \Delta \Rightarrow B}{\Gamma, A_2, A_1, \Delta \Rightarrow B}[\Box_p\mathrm{P}] \text{ provided } A_i \text{ is of the form } \Box_p C \tag{9}$$

Once we have defined the family of structural modalities, stronger type logics can be embedded into weaker ones via *embedding translations*. Thus (implicational) Intuitionistic Logic is faithfully embedded in Linear Logic by the following translation, i.e. $\mathbf{IL} \vdash \Gamma \Rightarrow A$ if and only if $\mathbf{ILL} \vdash !|\Gamma| \Rightarrow |A|$.

$$\begin{array}{rcll} |A| & := & A & \text{for } A \text{ atomic} \\ |A \to B| & := & !|A| \multimap |B| & \end{array} \tag{10}$$

Linguistic applications of modalities such as permutors are explored in [18] and [1]. Unidirectional permutors ([18]), one moving left and the other right, could provide potential for treatment of e.g. quantifier floating. And permutors may be used for medial extraction; thus assignment of an object relative pronoun to type $(cn\backslash cn)/(s/np)$ allows only peripheral extraction, but a type $(cn\backslash cn)/(s/\Box_p np)$ would allow also non-peripheral extraction such as *the man that John saw on Monday*. Then parasitic extraction, with multiple gaps: *the paper that John filed without reading*, can be approached by adding also to the gap subtype an orthogonal modality for contraction ([18]), or else having an modality that aggegates licensing of contraction and permutation ([1]).

2 Associativity and constituent domains

2.1 Associative versus non-associative L

The sequent presentation of **L** in the preceding section hides an implicit assumption about the structure-sensitivity of the derivability relation. In order to uncover this implicit assumption, we move from the metalevel presentation (sequent calculus) to a more elementary presentation of the logic of the type constructors, in terms of categorical combinators. Below one finds the categorical presentation of **L** as originally introduced in [14]. We write $f : A \to B$ for a proof that the combinator f maps A into B. It is interesting to study the structure of this combinator presentation. The identity arrow **id** and the composition of arrows are essential components of any well-behaved derivability relation. What we want to focus on is the arrows and rules that capture the properties of the type-constructors with respect to the derivability relation.

$$\mathbf{id} : A \to A \qquad \frac{f : A \to B \quad g : B \to C}{g \circ f : A \to C} \tag{11}$$

$$\alpha : A \bullet (B \bullet C) \leftrightarrow (A \bullet B) \bullet C : \alpha^{-1} \tag{12}$$

$$\frac{f : A \bullet B \to C}{\beta(f) : A \to C/B} \qquad \frac{f : A \bullet B \to C}{\gamma(f) : B \to A\backslash C}$$

$$\frac{g : A \to C/B}{\beta^{-1}(g) : A \bullet B \to C} \qquad \frac{g : B \to A\backslash C}{\gamma^{-1}(g) : A \bullet B \to C} \tag{13}$$

The categorical presentation highlights the central role played by *residuation* in connecting the family of connectives $\{/, \bullet, \backslash\}$. We introduce the concept of residuation in a general setting first, following [8], then present the categorial type constructors in a residuation perspective.

Consider two partially ordered sets $\mathcal{A} = (A, \leq)$ and $\mathcal{B} = (B, \leq')$ with functions $f : A \mapsto B$ and $g : B \mapsto A$. The pair of functions (f, g) is called *residuated* iff

$$fx \leq' y \quad \text{iff} \quad x \leq gy \tag{14}$$

Although in general we want to keep $\mathcal{A}$ and $\mathcal{B}$ distinct, we can for the purposes of this discussion consider them to be equal. An alternative definition of residuation for a pair of functions (f, g) is given by requiring f and g to be monotone (15), and by having the functions satisfy the inequalities of (16).

$$\text{if } x \leq y \text{ then } fx \leq fy \text{ and } gx \leq gy \tag{15}$$

$$fgx \leq x, x \leq gfx \tag{16}$$

How does the concept of residuation apply to the categorial type constructors? To establish the connection, let us interpret the partially ordered set $\mathcal{A}$ $(= \mathcal{B})$ as the set of types, ordered by set-theoretic inclusion. The categorial pairs $(\bullet, /)$ and $(\bullet, \backslash)$ are now easily recognized as binary generalizations of the notion of residuation just defined for unary operations f, g. For the *right residual* pair $(\bullet, /)$ we can read f as $- \bullet B$ and g as $-/B$, i.e. the product and division operations relativized to some fixed parameter type B. The defining equation (14) then becomes

$$A \bullet B \subseteq C \quad \text{iff} \quad A \subseteq C/B \tag{17}$$

Similarly for the *left residual* pair $(\bullet, \backslash)$, where we read f as $A \bullet -$ and g as $A\backslash -$, and obtain

$$A \bullet B \subseteq C \quad \text{iff} \quad B \subseteq A\backslash C \tag{18}$$

When we compare what we have got above in (17) and (18) with the categorical presentation of $\mathbf{L}$, we observe that the *residuation* relation between $\bullet$ and $/$ and $\backslash$, as given in (13), is in fact all that is needed for a *minimal* system of type-forming operators $\{/, \bullet, \backslash\}$. The associativity arrow (12) can be added as an extra structural property of the concatenation operation $\bullet$, giving rise to the standard type calculus of Lambek [14], or it can be withheld, resulting in the non-associative variant of [15].

Consider now the alternative characterization of residuation, in terms of monotonicity (15) and the inequalities of (16). The following derived rules of inference capture the monotonicity properties of the categorial type constructors with respect to the derivability relation ($A \bullet B$ monotone in both the A and B argument, A/B and $B\backslash A$ monotone in A and antitone in B). These derived rules of inference are independent of the associativity assumptions one wants to impose on $\bullet$.
Monotonicity of $\bullet$:

$$\frac{f : A \to B \quad g : C \to D}{f \cdot g : A \bullet C \to B \bullet D} \tag{19}$$

Monotonicity of $/, \backslash$:

$$\frac{f : A \to B \quad g : C \to D}{g\backslash f : D\backslash A \to C\backslash B} \qquad \frac{f : A \to B \quad g : C \to D}{f/g : A/D \to B/C} \tag{20}$$

Reading again, for the right residual pair $(\bullet, /)$ f as $- \bullet B$ and g as $-/B$, and f as $A \bullet -$ and g as $A\backslash -$, for the left residual pair $(\bullet, \backslash)$, the inequalities of (16) turn up as the categorial arrows of (21) below.

$$\mathbf{appr} : A/B \bullet B \to A \qquad \mathbf{appl} : B \bullet B\backslash A \to A \tag{21}$$

$$\mathbf{appr}^{-1} : A \to (A \bullet B)/B \qquad \mathbf{appl}^{-1} : B \to A\backslash(A \bullet B) \tag{22}$$

The reader may recognize in (19), (20) and (21), (22) a slight variant of Zielonka's [24] axiomatization of Lambek's type calculus, which has the application arrows (21) and type *lifting* ((23) instead of (22)) as the characteristic axioms. The associative system **L** is obtained by adding the division arrows (24).

$$\mathbf{liftr} : A \to B/(A\backslash B) \qquad \mathbf{liftl} : A \to (B/A)\backslash B \tag{23}$$

$$\mathbf{divr} : A/B \to (A/C)/(B/C) \qquad \mathbf{divl} : B\backslash A \to (C\backslash B)\backslash(C\backslash A) \tag{24}$$

The intended interpretation of types which is in accordance with the residuation characterization of $/, \bullet, \backslash$ is based on a set U obtained by closing the set of lexical atoms under a concatenation operation '+' which we take as associative for **L** and as non-associative for **NL**. Atomic types pick out arbitrary subsets of U. Complex types get their denotations as follows.

$$\begin{array}{l} D(A \bullet B) = \{\gamma | \exists \alpha \beta [\gamma = \alpha + \beta \ \&\ \alpha \in D(A) \ \&\ \beta \in D(B)]\} \\ D(C/B) = \{\gamma | \forall \alpha \beta [(\alpha = \gamma + \beta \ \&\ \beta \in D(B)) \to \alpha \in D(C)]\} \\ D(A\backslash C) = \{\gamma | \forall \alpha \beta [(\alpha = \beta + \gamma \ \&\ \beta \in D(A)) \to \alpha \in D(C)]\} \end{array} \tag{25}$$

In order to offer the reader some insight into the combinatorics of the systems we are discussing, we show how to derive type lifting from (22) in (26). In the derivation of (27) we show how the division arrows depend on the associativity of $\bullet$. This derivation makes use of the residuation inferences of (13).

$$\frac{\mathbf{appr}^{-1} : A \to (A \bullet A\backslash B)/(A\backslash B) \qquad \dfrac{\mathbf{appl}\text{: } A \bullet A\backslash B \to B \quad \mathbf{id}\text{: } A\backslash B \to A\backslash B}{\mathbf{appl/id}\text{: } (A \bullet A\backslash B)/(A\backslash B) \to B/(A\backslash B)}}{(\mathbf{appl/id}) \circ \mathbf{appr}^{-1} : A \to B/(A\backslash B)} \tag{26}$$

$$\dfrac{\dfrac{\dfrac{((A/B) \bullet (B/C)) \bullet C \to (A/B) \bullet ((B/C) \bullet C) \qquad (A/B) \bullet ((B/C) \bullet C) \to A}{((A/B) \bullet (B/C)) \bullet C) \to A}}{(A/B) \bullet (B/C) \to A/C}}{A/B \to (A/C)/(B/C)} \tag{27}$$

Let us give some linguistic illustrations of the type transitions which we introduced here in an abstract setting. Consider first the distributional differences between ordinary noun phrases, as represented by proper nouns, versus noun phrases with an overt case marking, hence a limited distribution. Proper nouns are assigned type *np*. With this type they will be able to occupy any *np* argument position of functors subcategorizing for that type. In contrast, a personal pronoun like 'I' can be given a higher order lexical type declaration which accounts the distributional restriction to subject argument position. The type declarations below distinguish

between the well-formed *I kissed Mary* and the ill-formed *Mary kissed I*. At the same time the lifting arrow (23) properly relates the inhabitants of type np to those of type $s/(np\backslash s)$.

$$\begin{array}{ll} \text{Mary} & := np \\ \text{I} & := s/(np\backslash s) \\ \text{kissed} & := (np\backslash s)/np \\ \text{needed} & := (np\backslash s)/((s/np)\backslash s) \end{array} \tag{28}$$

For an illustration of type inference which depends on the monotonicity properties of the type constructors, consider transitive verbs of the *need* variety. Although from a purely syntactic point of view one could say that these verbs require a direct object argument, strong arguments have been given that in terms of function-argument structure the direct object has to be of the higher-order generalized quantifier type, say $(s/np)\backslash s$. How then do we derive the sentence *I needed Mary*, where *need* is combined with a simple np type object, rather than a higher order $(s/np)\backslash s$? One can approach the problem in two ways: either by observing that the type transition from np to $(s/np)\backslash s$ is an instance of our lifting schema (23), or by taking the lifting arrow as the premise for a monotonicity inference (cf. 20), which then 'lowers' the direct object argument of the verb to type np.

$$\frac{\mathbf{id} : np\backslash s \to np\backslash s \quad \mathbf{liftl} : np \to (s/np)\backslash s}{\mathbf{id/liftl} : (np\backslash s)/((s/np)\backslash s) \to (np\backslash s)/np} \tag{29}$$

As we remarked at the beginning of this section, the sequent notation of **L** builds in the associativity of the product operator by having an n-place sequent comma which serves as the metasymbol for $\bullet$. In order to obtain a uniform presentation for the more highly structured type logics that will be the focus of our attention below, we recast the sequent formulations in the following structure-sensitive way. A sequent $\Delta \Rightarrow A$ represents a deduction of a type A from a structured configuration of type assumptions Δ represented as bracketed and perhaps labelled. To notate the logic of the type constructors we need a general notion of distinguished occurrence of a subconfiguration Δ' in a structured configuration Δ: we write $\Delta[\Delta']$. Configuations are separated by a sequent comma having a structural interpretation with structural rules to match.

Substitution of structured configurations is represented by means of the distinguished occurrence notation: the Cut rule states that a structure Γ replaces a type A situated in a structure Δ. Below is the full sequent presentation of **NL** where configurations have a *tree* structure, being (unlabelled) bracketed sequences of type assumptions.

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

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

$$[\text{R}\backslash]\frac{(B, \Gamma) \Rightarrow A}{\Gamma \Rightarrow B\backslash A} \quad \frac{\Gamma \Rightarrow B \qquad \Delta[A] \Rightarrow C}{\Delta[(\Gamma, B\backslash A)] \Rightarrow C}[\text{L}\backslash] \tag{30}$$

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

$$[\text{Cut}]\frac{\Gamma \Rightarrow A \quad \Delta[A] \Rightarrow C}{\Delta[\Gamma] \Rightarrow C}$$

To obtain **L** from **NL**, we need an explicit structural rule allowing the rebracketing of assumptions.

$$\frac{\Gamma[(\Delta_1,(\Delta_2,\Delta_3))]\Rightarrow B}{\Gamma[((\Delta_1,\Delta_2),\Delta_3)]\Rightarrow B} \tag{31}$$

For controlled (rather than global) access to rebracketing, we have the modal version of the structural rule, in the form of an Associativity (Restructuring) modality $\Box_a$.[3]

$$\frac{\Gamma[(\Delta_1,(\Delta_2,\Delta_3))]\Rightarrow B}{\Gamma[((\Delta_1,\Delta_2),\Delta_3)]\Rightarrow B}[\Box_a \mathrm{A}] \text{ provided } \Delta_i \text{ is fully modalized} \tag{32}$$

The Restructuring modality operates on an S4 base, which we recapitulate here from (6) in its readjusted format.

$$[\Box_a \mathrm{L}]\frac{\Gamma[A]\Rightarrow B}{\Gamma[\Box_i A]\Rightarrow B} \quad \frac{\Box_i\Gamma\Rightarrow A}{\Box_i\Gamma\Rightarrow \Box_i A}[\Box_a \mathrm{R}] \tag{33}$$

2.2 Constituency: linguistic applications

Let us compare the merits of **L** and **NL**. The associative calculus represents a theory of flexible constituency which offers successful accounts of phenomena such as non-constituent coordination or mismatches between phonological and syntactic structure. On the negative side, this system is insensitive to *domains of locality*, as represented in terms of a rigid constituency concept. The strategy of *licensing* structural modalities allows one to combine the attractive properties of both **L** and **NL**. We start from a non-associative basis, i.e. a domain sensitive system, and explicitly license flexible adaptations of constituency by means of modal type-assignments to the elements that are responsible for the flexibility.

We will illustrate the strategy with two standard examples of structural relaxation: so-called non-constituent coordination phenomena, where the licensing elements are the Boolean particles *and, or*, and prosodic restructuring as licensed by weak function words.

Generalized coordination

Consider first the problem of generalized cordination The type declaration to the Boolean particles has to account in an integrated way for the semantic and syntactic properties of coordination, resolving the tension between semantic generality and syntactic particularities. On the one hand there is generalized type conjoinability. From a basic meaning recipe for coordination in the truth-value domain, coordination of arbitrary conjoinable types is derivable in a systematic fashion. On the other hand however there is (for example, in English) a left/right peripheral incompleteness asymmetry. The semantic characterization of conjoinable types predicts

[3] The general modalization strategy outline above from the sequent perspective can be developed for the combinator presentation as well, as shown in [7]. The characteristic combinators for the structural modalities are given below. For **assl, assr, perm**, we make the proviso that one of the factors A, B, C is modalized, i.e. of the form $\Box D$.

assl : $A \bullet (B \bullet C) \rightarrow (A \bullet B) \bullet C$
assr : $(A \bullet B) \bullet C \rightarrow A \bullet (B \bullet C)$
perm : $A \bullet B \rightarrow B \bullet A$
cont : $\Box A \rightarrow \Box A \bullet \Box A$
exp : $\Box A \bullet \Box A \rightarrow \Box A$
thin : $A \bullet \Box B \rightarrow A$

coordination of non-constituents. However, not all such forms of non-constituent coordination are syntactically well-formed. The observed constraints point to an asymmetry between acceptable incompleteness violating left phrasal edges versus unacceptable violations of right phrasal edges unless accompanied by highly marked intonation. Compare the well-formedness on unmarked intonation of *John saw (Bill on Monday and Mary on Tuesday)* with the symmetrical *(John loves but Peter hates) bagels* which on unmarked intonation would be ill-formed.[4]

It is easy to derive 'non-constituent' Boolean coordination in associative Lambek calculus by assigning coordinating particles a polymorphic type $(A\backslash A)/A$ with A a type with a value s on saturation. (The lexical semantics for such a type-assignment schema induces multiple-bind lambda programs, i.e. controlled (parallel) access to reuse of resources.) The associative character means that *any* contiguous substrings that can form sentences in the same context can be coordinated, leading to such ghastly predictions as *the mother of and Bill thought John arrived.* On the other hand, the non-associative Lambek calculus would allow coordination only of constituents.

The apparently conflicting requirements can be captured in a recursive polymorphic type declaration scheme for the Boolean particles in **NL** with an association modality.

$$\mathbf{and} \in (s\backslash s)/s \quad \rightsquigarrow \quad \lambda x\lambda y.x \wedge y$$

$$\frac{\mathbf{and} \in (A\backslash A)/A}{\mathbf{and} \in ((\Box_a B\backslash A)\backslash(B\backslash A))/(\Box_a B\backslash A)} \quad \rightsquigarrow \quad \frac{\sqcap}{\lambda x\lambda y\lambda z.xz \sqcap yz}$$

$$\frac{\mathbf{and} \in (A\backslash A)/A}{\mathbf{and} \in ((A/B)\backslash(A/B))/(A/B)} \quad \rightsquigarrow \quad \frac{\sqcap}{\lambda x\lambda y\lambda z.xz \sqcap yz}$$

The following derivation illustrates how conjuncts such as *Mary on Tuesday* can be derived as of type $\Box_a(vp/np)\backslash((\Box_a np)\backslash s)$. and can thus be coordinated. By contrast, elements such as *John loves*, while they have type $s/(\Box_a np)$, do not have type s/np, so that the type declaration given for coordinating particles does not recognise them as conjuncts.

(34)
$$\dfrac{\dfrac{np \Rightarrow np \qquad \dfrac{\dfrac{\dfrac{s \Rightarrow s}{[np, vp] \Rightarrow s}\backslash\mathrm{L}}{[\Box_a np, vp] \Rightarrow s}\Box_a\mathrm{L}}{vp \Rightarrow (\Box_a np)\backslash s}\backslash\mathrm{R}}{[vp, vp\backslash vp] \Rightarrow (\Box_a np)\backslash s}\backslash\mathrm{L}}{[[vp/np, np], vp\backslash vp] \Rightarrow (\Box_a np)\backslash s}\backslash\mathrm{E}$$
$$\dfrac{[[\Box_a(vp/np), np], vp\backslash vp] \Rightarrow (\Box_a np)\backslash s}{[\Box_a(vp/np), [np, vp\backslash vp]] \Rightarrow (\Box_a np)\backslash s}\Box_a\mathrm{A}$$
$$\dfrac{[\Box_a(vp/np), [np, vp\backslash vp]] \Rightarrow (\Box_a np)\backslash s}{[np, vp\backslash vp] \Rightarrow \Box_a(vp/np)\backslash((\Box_a np)\backslash s)}\backslash\mathrm{R}$$

Consider the effect of the above type-assignment schema in general: violation of constituent boundaries has to be explicitly *licensed* by means of the Associativity

[4] Historically, Right node raising has received more grammatical accounts than the less marked but more puzzling Left node raising. Whether we wish to model grammaticality with respect to unmarked intonation or grammaticality with respect to a non-specific intonation is a methodological matter; those who insist on the latter should take our example as illustrative only.

modality $\Box_a$. The left-division subtypes carry the modality and so characterize the left edge as the locus for non-constituent phenomena. The right- division subtypes are non-modal — they have to coincide with constituent boundaries.

Prosodic restructuring

As a second application of the licensing modalization strategy, we discuss structural mismatches between prosodic and semantic organization of linguistic material. A characteristic example is presented below.[5]

- Prosodic phrasing:
  (the cat) (that caught the rat) (that stole the cheese)
- Syntactic/semantic organization:
  (the (cat (that caught (the (rat (that stole the cheese))))))

To achieve constituent structure we must work in **NL**. Let us assume the type declarations given below.

$$\begin{array}{ll} \textbf{the} & := np/cn \\ \textbf{cat, rat, cheese} & := cn \\ \textbf{caught, stole} & := (np\backslash s)/np \end{array} \tag{35}$$

Prosodically we would wish to add **that** $\in (np\backslash np)/(np\backslash s)$, but semantically we need $(cn\backslash cn)/(np\backslash s)$. Such conflicts have been influential in the design of grammar architecture, and represent a challenge to simple monostratality, as in categorial grammar. Prosodically governed forms of restructuring such as that above are discussed in [17] from the perspective of the associative type calculus **L**, but full associativity of its nature fails to impose a *preferred* bracketing, being able in principle to resolve any form of mismatch between prosodic and semantic organization of linguistic material.

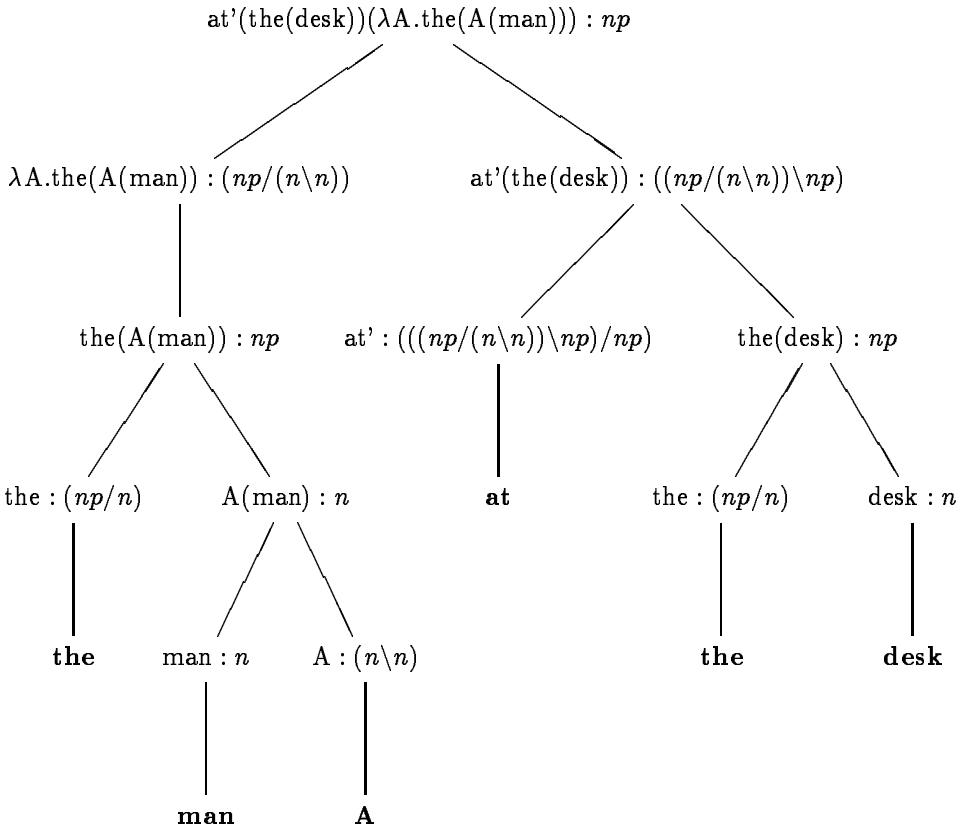
For constituency, we need to take **NL** as a basis, and for the puzzling mismatch we add a pinch of modally controlled associativity. The solution rests on the insight that the type assignment assumed above is not minimal/adequate if it is intended to account for semantic *and* prosodic properties of the expression in a parallel fashion. We present the solution in two steps. First we address the problem of *triggering* restructuring, abstracting from constituent sensitivity, i.e. working from the **L** perspective. Then we present a modalized version of the required type declaration, which pinpoints the exact place where restructuring has to be explicitly licenced within non-associative **NL**.

The prosodically governed restructuring can be *triggered* by a higher-order type assignment to postnominal modifiers. The type assignment $(np/(cn\backslash cn))\backslash np$ (rather than $cn\backslash cn$) forces an attempt to combine determiner and nominal into $np/(cn\backslash cn)$ The argument here is essentially the same as the argument for higher-order type assignment to transitive verbs of the *need* variety. Although in that case a $(np\backslash s)/np$ assignment would be adequate to account for the syntactic behaviour of such an expression, *semantic* considerations require the type to be minimally $(np\backslash s)/((s/np)\backslash s)$. The Natural Deduction derivation below illustrates the syntactic and semantic composition of a noun phrase with postnominal modification.[6] Notice that the lambda

[5] Although in the literature these examples are generally presented as mismatches between *syntactic* and prosodic organization, the relevant non-phonological level involved in fact can be taken as *semantic* function-argument structure, rather than syntax. See [17] for discussion. Crucially, the modifier is within the semantic scope of the determiner: *the cat that caught the rat that stole the cheese sleeps* does not entail a sole cat.

[6] We simplified the example to a postnominal prepositional modifier rather than a relative clause.

recipe for the complete *np* contains the subterm λA.the(A(man)) where the bound variable represents the slot in the scope configuration where the modifier belongs.



The meaning of the higher-order expression *at'* as a matter of fact is not unrelated to the primitive meaning in type *cn\cn*. A higher-order lambda program can be derived from a primitive meaning in *cn\cn* (cf. lifting). Our solution to the restructuring problem then is to lexically type postnominal modifiers as $(np/(cn\backslash cn))\backslash np$ (as their minimal type taking prosodic behaviour into account) and assign them a non-primitive lambda recipe — a meaning recipe expressed in terms of a primitive constant in type *cn\cn*. See the derivation below for the required type transition. The reader will find that substituting the higher order lambda recipe for *at'* in the natural deduction derivation above results in a scope configuration where indeed the modifier is within the scope of the determiner, i.e. where the apparent mismatch between prosodic and semantic organization is resolved.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathrm{A}_{np} \Rightarrow \mathrm{A}_{np} \quad \dfrac{\mathrm{C}_{cn} \Rightarrow \mathrm{C}_{cn} \quad \mathrm{at(A)(C)}_{cn} \Rightarrow \mathrm{at(A)(C)}_{cn}}{\mathrm{C}_{cn}, \mathrm{at(A)}_{(cn\backslash cn)} \Rightarrow \mathrm{at(A)(C)}_{cn}}}{\mathrm{C}_{cn}, \mathrm{at}_{((cn\backslash cn)/np)}, \mathrm{A}_{np} \Rightarrow \mathrm{at(A)(C)}_{cn}}}{\mathrm{at}_{((cn\backslash cn)/np)}, \mathrm{A}_{np} \Rightarrow \lambda\mathrm{C.at(A)(C)}_{(cn\backslash cn)}} \quad \mathrm{B}(\lambda\mathrm{C.at(A)(C)})_{np} \Rightarrow \mathrm{B}(\lambda\mathrm{C.at(A)(C)})_{np}}{\mathrm{B}_{(np/(cn\backslash cn))}, \mathrm{at}_{((cn\backslash cn)/np)}, \mathrm{A}_{np} \Rightarrow \mathrm{B}(\lambda\mathrm{C.at(A)(C)})_{np}}}{\mathrm{at}_{((cn\backslash cn)/np)}, \mathrm{A}_{np} \Rightarrow \lambda\mathrm{B.B}(\lambda\mathrm{C.at(A)(C)})_{((np/(cn\backslash cn))\backslash np)}}}{\mathrm{at}_{((cn\backslash cn)/np)} \Rightarrow \lambda\mathrm{A}\lambda\mathrm{B.B}(\lambda\mathrm{C.at(A)(C)})_{(((np/(cn\backslash cn))\backslash np)/np)}} \tag{36}$$

The argument sofar abstracted from constituent sensitivity: the combination of determiner and noun as $np/(cn\backslash cn)$ requires associativity. Starting from a **NL**

base, restructuring must be licensed (rather than being generally available). Below one finds the modalized version of the restructuring type declaration. Remark that this is not — and of course should not be — a derivable form of modalized lifting from $cn\backslash cn$. But as the structural modalities are semantically neutral, the type deduction above relating lower order primitive *at* and higher order *at'* still goes through.

$$\begin{array}{lll} \text{the} & := np/n & \text{the} \\ \text{man, desk} & := n & \text{man, desk} \\ \text{at} & := ((np/\Box_a(n\backslash n))\backslash np)/np & \lambda A\lambda B.B(\lambda C.at(A)(C)) \end{array} \tag{37}$$

The derivation below makes it clear where the restructuring modality licenses relaxation of constituent structure.

$$\dfrac{\dfrac{n \Rightarrow n \quad np \Rightarrow np}{[(np/n), n] \Rightarrow np} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{n \Rightarrow n \quad np \Rightarrow np}{[(np/n), n] \Rightarrow np}}{[(np/n), [n, (n\backslash n)]] \Rightarrow np}}{[(np/n), [n, \Box_a(n\backslash n)]] \Rightarrow np}}{[[(np/n), n], \Box_a(n\backslash n)] \Rightarrow np}}{[(np/n), n] \Rightarrow (np/\Box_a(n\backslash n))} \qquad np \Rightarrow np}{[[(np/n), n], ((np/\Box_a(n\backslash n))\backslash np)] \Rightarrow np}}{[[(np/n), n], [(((np/\Box_a(n\backslash n))\backslash np)/np), [(np/n), n]]] \Rightarrow np} \tag{38}$$

3 Dependency structure and alternation

3.1 Non-associative type calculus for headed trees

In the preceeding sections we have explored the nature of linear logic and Lambek calculus in characterising inference on resources which are configured as data structures of various kinds: bags, lists, and trees. We introduce now, as an example of a still more highly structured resource domain, type calculus for headed trees, i.e. (binary) trees in which each mother node has a distinguished daughter: either the left or the right subtree it immediately dominates is designated as *head*, and hence the other as *non-head*, or *dependent*.[7] We then consider the interplay between the dimension of constituent structure and structuring in terms of the head/dependent asymmetry. We present associative versions of the head-sensitive type system in which hierarchical organisation is removed; the forms of associativity we consider are such that they preserve certain aspects of the dependency structure while ignoring others. In one particular formulation inference is conditioned on data structured as a list with two distinguished elements, the head and the non-head.

Recall that the non-associative Lambek calculus is defined on the basis of left and right residuation with respect to a non-associative binary operation $+$ which we may take to be adjunction of trees and its setwise generalisation. For the logic of headed trees we begin with two such operations, $+_l$ and $+_r$, being the constructors

[7] It would be possible to formulate a type calculus for n-ary trees, though that is a task beyond the scope of the present article.

of left- and right-headed trees respectively:

$$\begin{array}{l} D(A \blacklozenge B) = \{\gamma | \exists \alpha\beta[\gamma = \alpha +_l \beta \ \&\ \alpha \in D(A) \ \&\ \beta \in D(B)]\} \\ D(A \blacktriangleright B) = \{\gamma | \exists \alpha\beta[\gamma = \alpha +_r \beta \ \&\ \alpha \in D(A) \ \&\ \beta \in D(B)]\} \end{array} \tag{39}$$

$$\begin{array}{l} D(A \bullet\!\!- B) = \{\gamma | \forall \alpha\beta[(\alpha = \gamma +_l \beta \ \&\ \beta \in D(B)) \rightarrow \alpha \in D(A)]\} \\ D(A \circ\!\!- B) = \{\gamma | \forall \alpha\beta[(\alpha = \gamma +_r \beta \ \&\ \beta \in D(B)) \rightarrow \alpha \in D(A)]\} \\ D(B -\!\!\bullet A) = \{\gamma | \forall \alpha\beta[(\alpha = \beta +_r \gamma \ \&\ \beta \in D(B)) \rightarrow \alpha \in D(A)]\} \\ D(B -\!\!\circ A) = \{\gamma | \forall \alpha\beta[(\alpha = \beta +_r \gamma \ \&\ \beta \in D(B)) \rightarrow \alpha \in D(A)]\} \end{array}$$

By way of illustration, if α is of type A, β of type B, and γ of type C, $\alpha +_r (\beta +_l \gamma)$ is of type $A \blacktriangleright (B \blacklozenge C)$; and if δ is of type $(D \circ\!\!- A) \bullet\!\!- B$, $(\delta +_r \beta) +_l \alpha$ is of type D.

The validity of the following categorical presentation of the derivability relation follows as for the Lambek calculus before. Notice that we see here the theme of decomposition of connectives: in the more highly structured resource domain of trees with a designated head daughter the connectives $\blacklozenge$ and $\blacktriangleright$ can be distinguished; in a resource domain where we have no structuring in terms of a head/non-head opposition, these connectives collapse into the $\bullet$ connective.

(40)

$$\mathbf{id} : A \longrightarrow A \qquad \frac{f : A \longrightarrow B \quad g : B \longrightarrow C}{(gf) : A \longrightarrow C}$$

$$\frac{f : A \blacktriangleright B \longrightarrow C}{\beta_r(f) : A \longrightarrow C \circ\!\!- B} \qquad \frac{f : A \blacktriangleright B \longrightarrow C}{\gamma_r(f) : B \longrightarrow A -\!\!\bullet C}$$

$$\frac{g : A \longrightarrow C \circ\!\!- B}{\beta_r^{-1}(g) : A \blacktriangleright B \longrightarrow C} \qquad \frac{g : B \longrightarrow A -\!\!\bullet C}{\gamma_r^{-1}(g) : A \blacktriangleright B \longrightarrow C}$$

$$\frac{f : A \blacklozenge B \longrightarrow C}{\beta_l(f) : A \longrightarrow C \bullet\!\!- B} \qquad \frac{f : A \blacklozenge B \longrightarrow C}{\gamma_l(f) : B \longrightarrow A -\!\!\circ C}$$

$$\frac{g : A \longrightarrow C \bullet\!\!- B}{\beta_l^{-1}(g) : A \blacklozenge B \longrightarrow C} \qquad \frac{g : B \longrightarrow A -\!\!\circ C}{\gamma_l^{-1}(g) : A \blacklozenge B \longrightarrow C}$$

This calculus delivers lifting theorems in the manner illustrated in (41).

$$\dfrac{\dfrac{\mathbf{id} : A -\!\!\circ B \longrightarrow A -\!\!\circ B}{\gamma_l^{-1}(\mathbf{id}) : A \blacklozenge (A -\!\!\circ B) \longrightarrow B}}{\beta_l(\gamma_l^{-1}(\mathbf{id})) : A \longrightarrow B \bullet\!\!- (A -\!\!\circ B)} \tag{41}$$

The other three lifting theorems share the property of complementarity in the 'colouring' of divisors:

$$\begin{array}{l} A \longrightarrow B \circ\!\!- (A -\!\!\bullet B) \\ A \longrightarrow (B \bullet\!\!- A) -\!\!\circ A \\ A \longrightarrow (B \circ\!\!- A) -\!\!\bullet A \end{array} \tag{42}$$

Statements such as $A \longrightarrow B \bullet\!\!- (A -\!\!\bullet B)$ are not valid, and are not theorems. This fact could be verified by means of a decision procedure for the calculus. A sequent presentation is obtained in a manner analogous to that for Lambek calculus. It differs in no respect that could effect Cut-elimination, so that Cut-free proof search

constitutes a decision procedure for the headed type calculus

(43)

$$[\mathrm{Ax}]\frac{}{A \Rightarrow A}$$

$$[\mathrm{R}\bullet\!-]\frac{[_l X, B] \Rightarrow A}{X \Rightarrow A\bullet\!-B} \quad \frac{X \Rightarrow B \qquad Y(A) \Rightarrow C}{Y([_l A\bullet\!-B, X]) \Rightarrow C}[\mathrm{L}\bullet\!-]$$

$$[\mathrm{R}\circ\!-]\frac{[_r X, B] \Rightarrow A}{X \Rightarrow A\circ\!-B} \quad \frac{X \Rightarrow B \qquad Y(A) \Rightarrow C}{Y([_r A\circ\!-B, X]) \Rightarrow C}[\mathrm{L}\circ\!-]$$

$$[\mathrm{R}-\!\bullet]\frac{[_r B, X] \Rightarrow A}{X \Rightarrow B-\!\bullet A} \quad \frac{X \Rightarrow B \qquad Y(A) \Rightarrow C}{Y([_r X, B-\!\bullet A]) \Rightarrow C}[\mathrm{L}-\!\bullet]$$

$$[\mathrm{R}-\!\circ]\frac{[_l B, X] \Rightarrow A}{X \Rightarrow B-\!\circ A} \quad \frac{X \Rightarrow B \qquad Y(A) \Rightarrow C}{Y([_l X, B-\!\circ A]) \Rightarrow C}[\mathrm{L}-\!\circ]$$

$$[\mathrm{L}\blacklozenge]\frac{X([_l A, B]) \Rightarrow C}{X(A \blacklozenge B) \Rightarrow C} \quad \frac{X \Rightarrow A \qquad Y \Rightarrow B}{[_l X, Y] \Rightarrow A \blacklozenge B}[\mathrm{R}\blacklozenge]$$

$$[\mathrm{L}\blacklozenge]\frac{X([_r A, B]) \Rightarrow C}{X(A \blacklozenge B) \Rightarrow C} \quad \frac{X \Rightarrow A \qquad Y \Rightarrow B}{[_r X, Y] \Rightarrow A \blacklozenge B}[\mathrm{R}\blacklozenge]$$

$$[\mathrm{Cut}]\frac{X \Rightarrow A \quad Y(A) \Rightarrow C}{Y(X) \Rightarrow C}$$

3.2 Illustration: dependency structures and metrical trees

In the above section we have introduced the dimension of dependency structure in terms the head/non-head asymmetry in a completely general way, without committing ourselves to any particular domain of application. The important point here is that we consider the dependency asymmetry as an *autonomous* dimension of linguistic organization — a dimension which may cross-cut the distinctions that can be made in terms of the function/argument opposition. Our position then should be contrasted with approaches where headedness (in whatever sense is at issue) is *defined* in terms of function/argument structure, i.e. where it is a derivative concept just employed in elucidation.[8] In taking the head/non-head asymmetry as primitive, our proposal rather squares with recent work of Zwicky's.

To illustrate the calculus of headed tree, and motivate the primitive head concept, we now consider employment of metrical trees in description of the rhythmical properties of speech. A metrical tree is simply a binary tree in which each mother node marks one daughter as strong (s) and the other as weak (w). It is thus a headed tree in our sense.

Metrical trees specify a hierarchical organisation of localised stress assignments interpreted as prosodic prominence contours. The Hoeksema ([13]) interpretation that we shall now assume interprets each path in a metrical tree as a degree of stress subordination equal to the value of the binary numeral obtained by reading w as 1 and s as 0 from leaf to root. The prominence assignments (which are in fact always total orderings) defined by this procedure are illustrated in Figure 1. That

[8] Examples of such a derived notion of head can be found in [2].

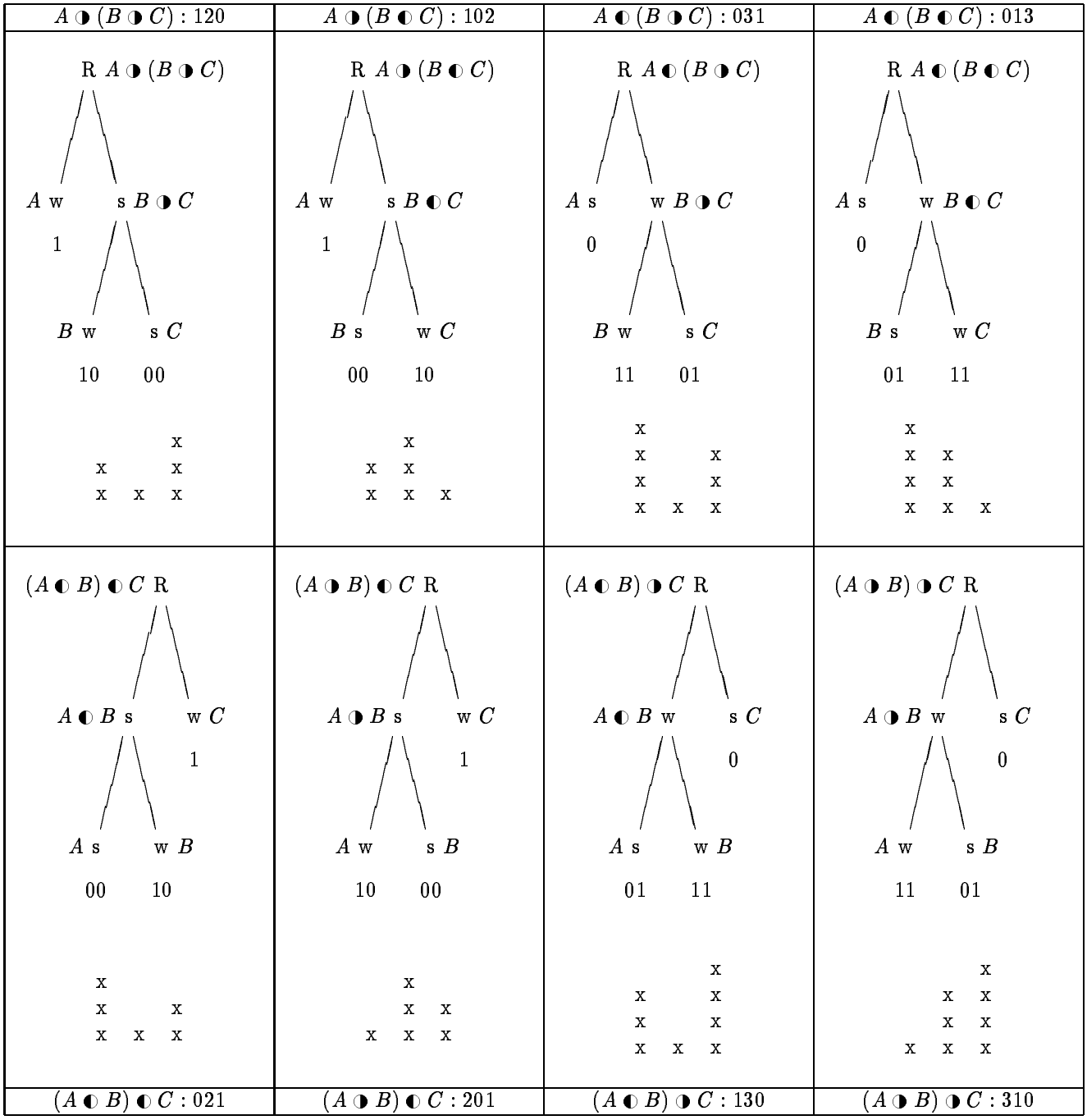


Figure 1: Hoeksema interpretation of metrical trees

this interpretation is one-to-one accords with the invalidity of any restructuring, such as association, in this headed calculus.

As our minimal examples, consider the following, and read them as replies to a question: What happened?

a. John arrived. (44)
b. John left.

Observe that the neutral utterance of (44a) has stress on the subject; stress on the verb phrase puts the verb in focus. Conversely, the neutral utterance of (44b) has stress on the verb phrase; stress on the subject puts the proper name in focus. The metrical configurations for the non-focusing pronunciations are those in (45).

a. $[_l$John arrived] (45)
b. $[_r$John left]

These are given by the lexical asignments in (46). Observe that any attempt to characterize prosodic structuring purely in terms of the function/argument asymmetry would have to treat the two verbs on a par: here then we see an example of the autonomous character of the dependency dimension.

arrived $:= n \multimap s$ (46)
John $:= n$
left $:= n \bullet\!\!- s$

However, a subject pronoun is weak in both cases (on the neutral pronunciation)

a. I arrived. (47)
b. I left.

A characterisation of this (other than the non-solution of conjoining lexical entries) will be given by incorporation of a structural modality on the pattern that has already been seen. Observe that the following structural rule of *alternation* or *balance* collapses the headed calculus into the non-associative calculus **NL**, in the same way that association collapses **NL** into **L**, and contraction and weakening collapse linear logic into intuitionistic logic.

$$\frac{X([_r Y, Z]) \Rightarrow A}{X([_l Y, Z]) \Rightarrow A}\,[\mathrm{B}] \tag{48}$$

We control alternation by means of a structural modality $\Box_b$, requiring that either Y or Z in (48) is $\Box_b$-modal, i.e. that one or other of these subconfigurations contains only $\Box_b$-modal types. As before, $\Box_b$ has the left and right rules of S4 modality and we would expect the non-associative calculus to be faithfully embedded in the headed calculus by the embedding translation (it making no difference which colouring is chosen for implications).

We saw above that subject pronouns get a lifted lexical type declaration capturing their restricted case. What then would be the appropriate modalized type assignment? The stress subordination of subject pronouns with respect to both strong and weak functors is obtained by the following lexical assignment.[9]

he $:= s \circ\!\!- ((\Box_b n) \bullet\!\!- s)$ (49)

In view of the principal connective in its type, it is clear that 'he' can only combine

[9] A type $s \circ\!\!- ((\Box_b n) \multimap s)$ would have done equally well.

as prosodic non-head with a verb phrase. To see that derivation goes through fully, observe the following proof.

(50)

a. He arrived.

b.
$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{s \Rightarrow s}\,\text{id} \quad \dfrac{}{n \Rightarrow n}\,\text{id}}{[_l n, n{-\!\circ} s] \Rightarrow s}\,[{-\!\circ}\text{L}] \quad \dfrac{}{s \Rightarrow s}\,\text{id}}{[_l \Box_b n, n{-\!\circ} s] \Rightarrow s}\,[\Box_b\text{L}]}{[_r \Box_b n, n{-\!\circ} s] \Rightarrow s}\,[\text{B}]}{n{-\!\circ} s \Rightarrow (\Box_b n){-\!\bullet} s}\,[{-\!\bullet}\text{R}]}{[_r s{\circ\!-}((\Box_b n){-\!\bullet} s), n{-\!\circ} s] \Rightarrow s}\,[{\circ\!-}\text{L}]$$

The derivation of 'he left' would be the same, but without the need for an application of alternation.

3.3 Associativity and dependency

It has already been pointed out that the Hoeksema interpretation of metrical trees assigns all distinct trees a distinct prosodic value. In the original interpretation of Liberman and Prince ([16], p259) each leaf is assigned a degree of stress subordination equal to the depth of its lowest dominating w-node.[10] Under such an interpretation there are equations such as the following:

(51)

a. $(a +_l b) +_l c = a +_l (b +_r c)$

b. $a +_r (b +_r c) = (a +_l b) +_r c$

So far as application to prosodics is concerned however, this system appears to be too discriminatory (and the Hoeksema scheme even more so). For instance, a fully right-branching and right-headed tree will assign successive degrees of stress subordination 1, 2, 3, ..., 0 to its left-to-right leaves with as many levels as the yield is long. Accordingly, Liberman and Prince ([16], p. 316) propose a "Relative Prominence Projection Rule" requiring just that the s-head of each s-daughter tree is more highly stressed than the s-head of its w-daughter sister tree.[11] More radically still we may propose just head-preservation, so that any trees are equated that have the same number of leaves, the same one amongst which is the s-head.

For prosodic applications, such s-head preservation may be an appropriate to the description of main stress alone, but a slightly more sensitive tool, and the one with which we shall continue our little illustration, corresponds to an interpretation of metrical trees which distinguishes both the s-head and the w-head as the most and least prominent elements respectively, but makes no other distinctions. This is illustrated in Figure 2.

It is easy to see by induction that every tree has a unique s-head and a distinct unique w-head. Transformations preserving this headedness are valid; these are as follows:

(52)

$$X([_l[_l Y_1, Y_2], Y_3]) \Leftrightarrow X([_l Y_1, [_l Y_2, Y_3]])$$

$$X([_r Y_1, [_r Y_2, Y_3]]) \Leftrightarrow X([_r[_r Y_1, Y_2], Y_3])$$

[10] In the absence of any dominating w-node the stress subordination should be defined to be zero. Liberman and Prince actually increment all degrees by one in order to mimic the stress numbering of cyclic transformational treatment of stress assignment, cf. Chomsky and Halle [4], which counts from one upwards.

[11] In the metrical phonology literature the s-head is usually referred to as the distingished terminal element, DTE.

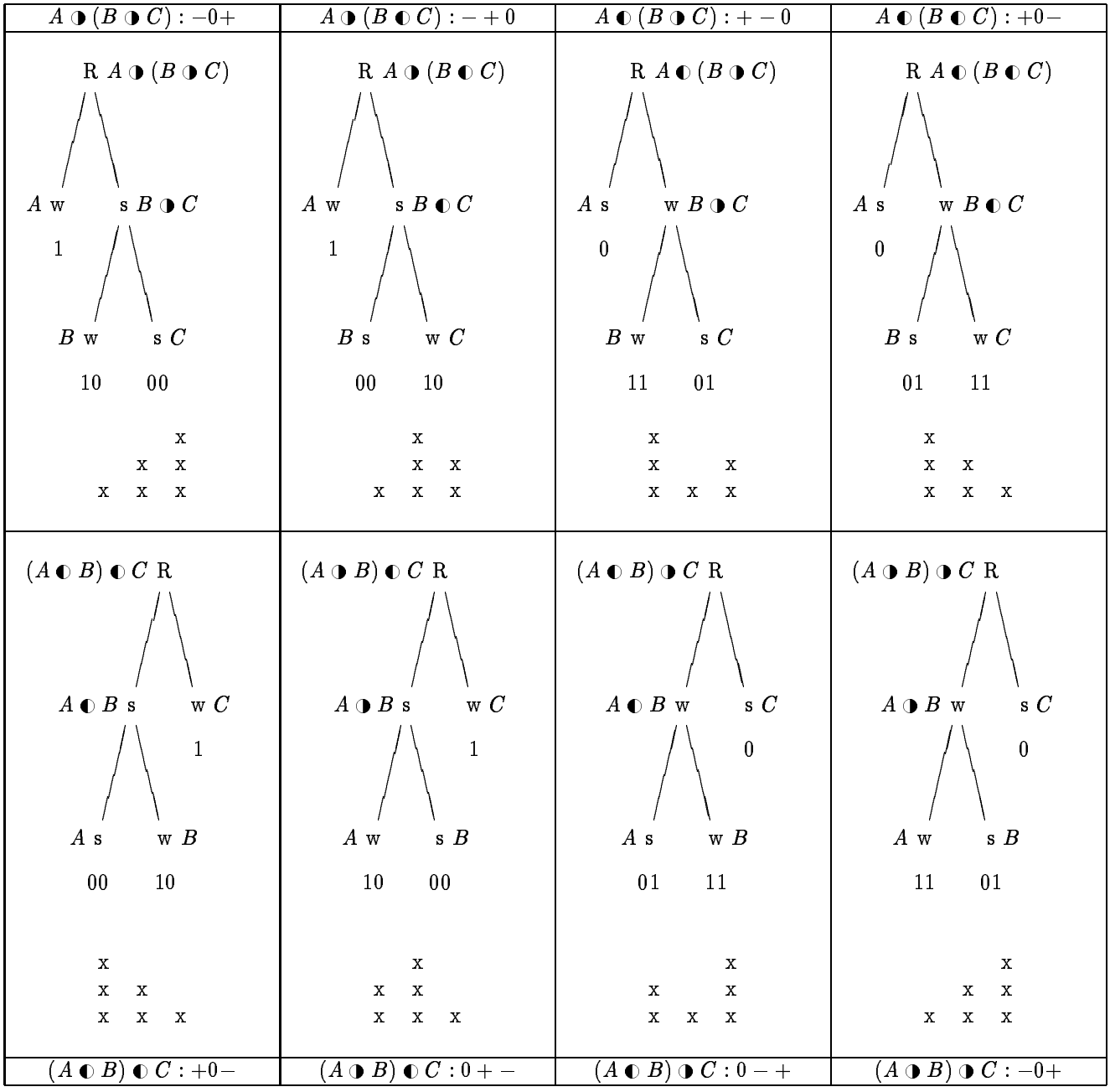


Figure 2: Head-and-foot interpretation of metrical trees

The validity of the top case follows since the paths from the root to leaves in Y_1 change only between –s–s– and –s–, which cannot alter whether or not any path leads to the s-head (clearly none lead to the w-head), and likewise those to leaves in Y_3 change only between –w–w– and –w–, which cannot alter whether or not any path leads to the w-head (clearly none lead to the s-head); and changing the paths into Y_2 between –s–w– and –w–s– makes no difference since neither could lead to either head. The bottom case is symmetric.

This restructuring corresponds to internal associativity for each product:

$$\begin{array}{ll} \text{a.} & (A \blacklozenge B) \blacklozenge C = A \blacklozenge (B \blacklozenge C) \\ \text{b.} & A \blacklozenge (B \blacklozenge C) = (A \blacklozenge B) \blacklozenge C \end{array} \tag{53}$$

We obtain an associative headed calculus by addition of the following axioms to the categorical presentation of the non-headed calculus.

$$\begin{array}{l} \alpha_l : (A \blacklozenge B) \blacklozenge C \longleftrightarrow A \blacklozenge (B \blacklozenge C) : \alpha_l^{-1} \\ \alpha_r : A \blacklozenge (B \blacklozenge C) \longleftrightarrow (A \blacklozenge B) \blacklozenge C : \alpha_r^{-1} \end{array} \tag{54}$$

A sequent presentation is obtained by adding:

$$\begin{array}{l} \dfrac{X([_l[_l Y, Z], W]) \Rightarrow A}{X([_l Y, [_l Z, W]]) \Rightarrow A}[\mathrm{A}_l] \\ \dfrac{X([_r Y, [_r Z, W]]) \Rightarrow A}{X([_r[_r Y, Z], W]) \Rightarrow A}[\mathrm{A}_r] \end{array} \tag{55}$$

Assuming still cut-elimination, a decision procedure is provided by cut-free proof-search on the basis of the sequent presentation with a trivial check on re-cycling amongst the (always finitary) possibilities for re-bracketing. In the appendix a formulation is given which drops unnecessary bracketing and keeps associativity implicit, as in the sequent formulation of associative Lambek calculus.

Consider now (56).

John saw Mary. (56)

The non-focusing prominence contour rises left-to-right. Inspection of figure 2 indicates that type assignments $(n \bullet\!\!- s) \circ\!\!- n$ and $n \bullet\!\!- (s \circ\!\!- n)$ project the appropriate configurations (these two types are mutually derivable). Assume the lexical entry:

$$\text{saw} \quad := (n \bullet\!\!- s) \circ\!\!- n \tag{57}$$

Then derivation for (56) is as follows.

$$\begin{array}{lll} [_r n, [_r (n \bullet\!\!- s) \circ\!\!- n, n]] \Rightarrow s & & [\circ\!\!-\mathrm{L}] \\ \quad n \Rightarrow n & & \mathrm{id} \\ \quad [_r n, n \bullet\!\!- s] \Rightarrow s & & [\bullet\!\!-\mathrm{L}] \\ \qquad n \Rightarrow n & & \\ \qquad s \Rightarrow s & & \end{array} \tag{58}$$

The prominence contour for 'John seeks Mary' is the same. The intensional-object transitive verb 'seek' can be assigned a type:

$$\text{seeks} \quad := (n \bullet\!\!- s) \circ\!\!- ((s \circ\!\!- n) \bullet\!\!- s) \tag{59}$$

Thus:

$$
\begin{array}{ll}
[_r n, [_r (n \bullet\!\!-\!\!\circ s) \circ\!\!- ((s \circ\!\!- n) \bullet\!\!-\!\!\circ s), n]] \Rightarrow s & [\circ\!\!-\mathrm{L}] \\
\quad n \Rightarrow (s \circ\!\!- n) \bullet\!\!-\!\!\circ s & [\bullet\!\!-\!\!\circ \mathrm{R}] \\
\quad\quad [_r s \circ\!\!- n, n] \Rightarrow s & [\circ\!\!-\mathrm{L}] \\
\quad\quad\quad n \Rightarrow n & \mathrm{id} \\
\quad\quad\quad s \Rightarrow s & \mathrm{id} \\
\quad\quad [_r n, n \bullet\!\!-\!\!\circ s] \Rightarrow s & [\bullet\!\!-\!\!\circ \mathrm{L}] \\
\quad\quad\quad n \Rightarrow n & \mathrm{id} \\
\quad\quad\quad s \Rightarrow s & \mathrm{id}
\end{array}
\tag{60}
$$

Not all transitive verbs have the same neutral prosodic context. Oehrle ([19]) notes that (61) again has the object strongest, but the verb and not the subject is weakest.

John passed Mary. (61)

From figure 2, we see that the following lexical assignment is required.

$$\text{passed} \quad := n \circ\!\!-\!\!\bullet (s \circ\!\!- n) \tag{62}$$

When the subject is a pronoun, (63) has the rising contour that would be expected, and which the assignments we have given deliver.

He saw Mary. (63)

Example (64) has the same contour.

He passed Mary. (64)

It is less obvious what results the lexical assignments we have given predict for this case. In fact the appropriate contour is accepted, as the following derivation shows.

$$
\begin{array}{ll}
[_r s \circ\!\!- ((\Box_b n) \bullet\!\!-\!\!\circ s), [_r n \circ\!\!-\!\!\bullet (s \circ\!\!- n), n]] & [\circ\!\!-\mathrm{L}] \\
\quad [_r n \circ\!\!-\!\!\bullet (s \circ\!\!- n), n] \Rightarrow (\Box_b n) \bullet\!\!-\!\!\circ s & [\bullet\!\!-\!\!\circ \mathrm{R}] \\
\quad\quad [_r \Box_b n, [_r n \circ\!\!-\!\!\bullet (s \circ\!\!- n), n]] \Rightarrow s & [\mathrm{A}_{ws}] \\
\quad\quad\quad [_r [_r \Box_b n, n \circ\!\!-\!\!\bullet (s \circ\!\!- n)], n] \Rightarrow s & [\Box_b \mathrm{B}] \\
\quad\quad\quad\quad [_r [_l \Box_b n, n \circ\!\!-\!\!\bullet (s \circ\!\!- n)], n] \Rightarrow s & [\Box_b \mathrm{L}] \\
\quad\quad\quad\quad\quad [_r [_l n, n \circ\!\!-\!\!\bullet (s \circ\!\!- n)], n] \Rightarrow s & [\circ\!\!-\!\!\bullet \mathrm{L}] \\
\quad\quad\quad\quad\quad\quad n \Rightarrow n & \mathrm{id} \\
\quad\quad\quad\quad\quad\quad [_r s \circ\!\!- n, n] \Rightarrow & [\circ\!\!-\mathrm{L}] \\
\quad\quad\quad\quad\quad\quad\quad n \Rightarrow n & \mathrm{id} \\
\quad\quad\quad\quad\quad\quad\quad s \Rightarrow s & \mathrm{id} \\
s \Rightarrow s &
\end{array}
\tag{65}
$$

Some concluding remarks. For a realistic account of phonological structure it would be necessary to represent several levels of phonological *rank* (distinguishing phonological words, phrases, ...). The type-theoretic apparatus could be extended with respect to this dimension of linguistic structure by having several product operators, parametrized for the relevant levels of prosodic organization. The rank domains would then represent the locus for application of the associativity equations one might want to consider. Secondly, as we noted above, the headed type calculus can be used to model different types of dependency structure in natural language (i.e. syntactic, semantic, pragmatic notion of dependency in addition to the application to prosodics discussed here). In a comprehensive grammar, such notions of dependency would interact in a parallel but modular fashion.[12]

[12] A previous version of this paper was presented under the title 'A type calculus for biased trees' at the Utrecht conference on the Syntax/Prosody Interface (OTS, May 1991). The research by the alphabetically second author was partly supported by a grant from the Nederlandse Organisatie voor Wetenschappelijk Onderzoek.

Appendix A. Digital labelling for non-associative dependency calculus

In order to present algorithmic proof theory (theorem proving) for the non-associative dependency calculus, we us the framework of Labelled Deductive Systems (LDS), cf. Gabbay [9],[10]. In a LDS approach the basic declarative unit is not the formula, but the *labelled* formula $t : A$. The traditional consequence relation $A_1, \ldots, A_n \Rightarrow B$ (deduction of B from a database of assumptions $A_1, \ldots, A_n$) is replaced by $t_1 : A_1, \ldots, t_n : A_n \Rightarrow s : B$. Intuitively, the labels represent relevant aspects of the structure of the database of assumptions—structure which the logic is sensitive to. Structured consequence is defined not just in terms of proof rules on the formulas, but in terms of rules that operate on both the formulas and their labels. From a sequent perspective, for every logical rule introducing a connective in the antecedent (database formulae) or the succedent (goal formula), we have to indicate how in the labelling algebra the labels propagate between conclusions and premises.

The syntactic structures in the LDS presentation below are *headed* trees: binary branching structures with a prominence (dependency) ordering among sisters (head versus non-head). In order to associate a sequent deduction with a prominence representation, types are labelled with a binary code, represented as a sequence of digits. (Notation: *digit.tail*). The binary code represents the path from terminals to the root of the headed tree through head ('0') versus non-head ('1') branches. (For simplicity, labelling here is restricted to head-sensitive dominance information. The linear precedence information is represented by the ordering of the database.) In the unfolding of the proofs, the propagation of the dependency labels will in fact compute the structure of the database that has to be assumed to make the derivation go through.

$$[\mathrm{Ax}]\frac{}{\langle A,k\rangle \Rightarrow \langle A,k\rangle}$$

$$[\mathrm{R}\bullet\!\!-]\frac{\mathrm{T}, \langle B,1.k\rangle \Rightarrow \langle A,k\rangle}{\mathrm{T} \Rightarrow \langle A \bullet\!\!- B, 0.k\rangle} \quad \frac{\mathrm{T} \Rightarrow \langle B,1.k\rangle \quad \Gamma, \langle A,k\rangle, \Lambda \Rightarrow \langle C,l\rangle}{\Gamma, \langle A \bullet\!\!- B, 0.k\rangle, \mathrm{T}, \Lambda \Rightarrow \langle C,l\rangle}[\mathrm{L}\bullet\!\!-]$$

$$[\mathrm{R}\circ\!\!-]\frac{\mathrm{T}, \langle B,0.k\rangle \Rightarrow \langle A,k\rangle}{\mathrm{T} \Rightarrow \langle A \circ\!\!- B, 1.k\rangle} \quad \frac{\mathrm{T} \Rightarrow \langle B,0.k\rangle \quad \Gamma, \langle A,k\rangle, \Lambda \Rightarrow \langle C,l\rangle}{\Gamma, \langle A \circ\!\!- B, 1.k\rangle, \mathrm{T}, \Lambda \Rightarrow \langle C,l\rangle}[\mathrm{L}\circ\!\!-]$$

$$[\mathrm{R}-\!\!\circ]\frac{\langle B,0.k\rangle, \mathrm{T} \Rightarrow \langle A,k\rangle}{\mathrm{T} \Rightarrow \langle B -\!\!\circ A, 1.k\rangle} \quad \frac{\mathrm{T} \Rightarrow \langle B,0.k\rangle \quad \Gamma, \langle A,k\rangle, \Lambda \Rightarrow \langle C,l\rangle}{\Gamma, \mathrm{T}, \langle B -\!\!\circ A, 1.k\rangle, \Lambda \Rightarrow \langle C,l\rangle}[\mathrm{L}-\!\!\circ]$$

$$[\mathrm{R}-\!\!\bullet]\frac{\langle B,1.k\rangle, \mathrm{T} \Rightarrow \langle A,k\rangle}{\mathrm{T} \Rightarrow \langle B -\!\!\bullet A, 0.k\rangle} \quad \frac{\mathrm{T} \Rightarrow \langle B,1.k\rangle \quad \Gamma, \langle A,k\rangle, \Lambda \Rightarrow \langle C,l\rangle}{\Gamma, \mathrm{T}, \langle B -\!\!\bullet A, 0.k\rangle, \Lambda \Rightarrow \langle C,l\rangle}[\mathrm{L}-\!\!\bullet]$$

$$[\mathrm{L}\blacklozenge]\frac{\Gamma, \langle A,0.k\rangle, \langle B,1.k\rangle, \Lambda \Rightarrow \langle C,l\rangle}{\Gamma, \langle A \blacklozenge B, k\rangle, \Lambda \Rightarrow \langle C,l\rangle} \quad \frac{\Delta \Rightarrow \langle A,0.k\rangle \quad \mathrm{T} \Rightarrow \langle B,1.k\rangle}{\Delta, \mathrm{T} \Rightarrow \langle A \blacklozenge B, k\rangle}[\mathrm{R}\blacklozenge]$$

$$[\mathrm{L}\blacklozenge]\frac{\Gamma, \langle A,1.k\rangle, \langle B,0.k\rangle, \Lambda \Rightarrow \langle C,l\rangle}{\Gamma, \langle A \blacklozenge B, k\rangle, \Lambda \Rightarrow \langle C,l\rangle} \quad \frac{\Delta \Rightarrow \langle A,1.k\rangle \quad \mathrm{T} \Rightarrow \langle B,0.k\rangle}{\Delta, \mathrm{T} \Rightarrow \langle A \blacklozenge B, k\rangle}[\mathrm{R}\blacklozenge]$$

As an illustration, consider the following lifting theorems. The two directional forms of lifting in **(N)L** each split up in two head-sensitive versions.

$$\frac{a_{[1|A]} \Rightarrow a_{[1|A]} \quad b_A \Rightarrow b_A}{\frac{(b \bullet\!\!- a)_{[0|A]}, a_{[1|A]} \Rightarrow b_A}{a_{[1|A]} \Rightarrow ((b \bullet\!\!- a) \multimap b)_{[1|A]}}}$$

Left, Weak

$$\frac{a_{[0|A]} \Rightarrow a_{[0|A]} \quad b_A \Rightarrow b_A}{\frac{(b \circ\!\!- a)_{[1|A]}, a_{[0|A]} \Rightarrow b_A}{a_{[0|A]} \Rightarrow ((b \circ\!\!- a) \bullet\!\!\multimap b)_{[0|A]}}}$$

Left, Strong

$$\frac{a_{[0|A]} \Rightarrow a_{[0|A]} \quad b_A \Rightarrow b_A}{\frac{a_{[0|A]}, (a \multimap b)_{[1|A]} \Rightarrow b_A}{a_{[0|A]} \Rightarrow (b \bullet\!\!- (a \multimap b))_{[0|A]}}}$$

Right, Strong

$$\frac{a_{[1|A]} \Rightarrow a_{[1|A]} \quad b_A \Rightarrow b_A}{\frac{a_{[1|A]}, (a \bullet\!\!\multimap b)_{[0|A]} \Rightarrow b_A}{a_{[1|A]} \Rightarrow (b \circ\!\!- (a \bullet\!\!\multimap b))_{[1|A]}}}$$

Right, Weak

To remove spurious non-determinism from sequent proof search, one can move to a proof-net presentation as an optimal datastructure for sequent deductions. The relation between sequent calculus and proof nets for occurrence logics is discussed in [20]. We obtain proof nets with digital labelling from the sequent presentation above by adding a *polarity* label which distinguishes antecedent ('**1**') from succedent ('**0**') occurrences of types.

$$[\mathrm{Ax}]\frac{}{\langle A, \mathbf{1}, k\rangle \quad \langle A, \mathbf{0}, k\rangle} \qquad \frac{}{\langle A, \mathbf{0}, k\rangle \quad \langle A, \mathbf{1}, k\rangle}[\mathrm{Ax}]$$

$$[\mathrm{L}\bullet\!\!-]\frac{\langle A, \mathbf{1}, k\rangle \quad \langle B, \mathbf{0}, 1.k\rangle}{\langle A \bullet\!\!- B, \mathbf{1}, 0.k\rangle} \qquad \frac{\langle B, \mathbf{1}, 1.k\rangle \quad \langle A, \mathbf{0}, k\rangle}{\langle A \bullet\!\!- B, \mathbf{0}, 0.k\rangle}[\mathrm{R}\bullet\!\!-]$$

$$[\mathrm{L}\circ\!\!-]\frac{\langle A, \mathbf{1}, k\rangle \quad \langle B, \mathbf{0}, 0.k\rangle}{\langle A \circ\!\!- B, \mathbf{1}, 1.k\rangle} \qquad \frac{\langle B, \mathbf{1}, 0.k\rangle \quad \langle A, \mathbf{0}, k\rangle}{\langle A \circ\!\!- B, \mathbf{0}, 1.k\rangle}[\mathrm{R}\circ\!\!-]$$

$$[\mathrm{L}\CIRCLE]\frac{\langle A, \mathbf{1}, 0.k\rangle \quad \langle B, \mathbf{1}, 1.k\rangle}{\langle A \CIRCLE B, \mathbf{1}, k\rangle} \qquad \frac{\langle B, \mathbf{0}, 1.k\rangle \quad \langle A, \mathbf{0}, 0.k\rangle}{\langle A \CIRCLE B, \mathbf{0}, k\rangle}[\mathrm{R}\CIRCLE]$$

$$[\mathrm{R}\LEFTcircle]\frac{\langle B, \mathbf{0}, 0.k\rangle \quad \langle A, \mathbf{0}, 1.k\rangle}{\langle A \LEFTcircle B, \mathbf{0}, k\rangle} \qquad \frac{\langle A, \mathbf{1}, 1.k\rangle \quad \langle B, \mathbf{1}, 0.k\rangle}{\langle A \LEFTcircle B, \mathbf{1}, k\rangle}[\mathrm{L}\LEFTcircle]$$

$$[\mathrm{R}\circ\!\!\multimap]\frac{\langle A, \mathbf{0}, k\rangle \quad \langle B, \mathbf{1}, 0.k\rangle}{\langle B \circ\!\!\multimap A, \mathbf{0}, 1.k\rangle} \qquad \frac{\langle B, \mathbf{0}, 0.k\rangle \quad \langle A, \mathbf{1}, k\rangle}{\langle B \circ\!\!\multimap A, \mathbf{1}, 1.k\rangle}[\mathrm{L}\circ\!\!\multimap]$$

$$[\mathrm{R}\bullet\!\!\multimap]\frac{\langle A, \mathbf{0}, k\rangle \quad \langle B, \mathbf{1}, 1.k\rangle}{\langle B \bullet\!\!\multimap A, \mathbf{0}, 0.k\rangle} \qquad \frac{\langle B, \mathbf{0}, 1.k\rangle \quad \langle A, \mathbf{1}, k\rangle}{\langle B \bullet\!\!\multimap A, \mathbf{1}, 0.k\rangle}[\mathrm{L}\bullet\!\!\multimap]$$

Appendix B. Partial Bracketing for Associative Headed Type Calculus

The associative headed type calculus deals with inference of headed-and-footed lists. In order to give a common representation to equivalent binary configurations, i.e. those having the same interpretation as a headed-and-footed list, we give a notion of normalisation in an expanded language of headed trees, and then give sequent calculus for normal configurations representing equivalence classes. The expanded language has $n+2$-ary, as opposed to binary, trees.[13] As before however, a mother node assigns either left- or right- headedness to the tree it dominates; any medial nodes are assigned a nil status. And as before the s-head is that (unique) leaf dominated by an uninterrupted chain of s-nodes up to the root, and the w-head that dominated by such a chain of w-nodes. In this conservative extension of binary configurations, the following contractions are valid (i.e., head-and-foot preserving).

$$\begin{array}{ccccc} [_r\Gamma,[_r\Delta]] & \rhd & [_r\Gamma,\Delta] & \lhd & [_r[_r\Gamma],\Delta] \\ [_l[_l\Gamma],\Delta] & \rhd & [_l\Gamma,\Delta] & \lhd & [_l\Gamma,[_l\Delta]] \end{array} \tag{66}$$

Gentzen sequent calculus for the (fully representative) sublanguage of normal forms is given in Figure 3. Note that a little care is needed with the left rules to check against unary bracketing, and with the right rules to preserve normal form. The rules are given in a refinement notation showing how this book-keeping is to be managed in proof search.

[13] Regarding notation, note that we shall sometimes be using binary bracketed antecedents, sometimes unbracketed, and sometimes partially bracketed. We use $A, B, C, \ldots$ to stand for types. *Configurations* are defined thus: a type is a configuration, and a bracketed sequence of types is a configuration, and are represented by X, Y, Z, W. *Antecedents* are defined thus: a configuration is an antecedent, and a sequence of antecedents is an antecedent, and are represented by Γ, Δ. Every type is a configuration, and every configuration is an antecedent: we adopt the convention of using the most specific notation applicable; in particular, for non-associative sequents we write X, Y, Z for configurations as opposed to Γ, Δ for antecedents. (Perhaps this should have come earlier, in the first half of the paper)

$$\begin{array}{l} X([_l\Gamma_1, A\bullet\!\!-B, \Delta, \Gamma_2]) \Rightarrow C \\ \quad l(\Delta) \Rightarrow B \\ \quad X(l(\Gamma_1, A, \Gamma_2)) \Rightarrow C \end{array} \quad [\bullet\!\!-\mathrm{L}] \qquad l(Y_1, \ldots, Y_n) = \begin{cases} Y_1 \text{ if } n = 1 \\ [_l Y_1, \ldots, Y_n] \text{ o.w.} \end{cases}$$

$$\begin{array}{l} X([_r\Gamma_1, A\circ\!\!-B, \Delta, \Gamma_2]) \Rightarrow C \\ \quad r(\Delta) \Rightarrow B \\ \quad X(r(\Gamma_1, A, \Gamma_2)) \Rightarrow C \end{array} \quad [\circ\!\!-\mathrm{L}] \qquad r(Y_1, \ldots, Y_n) = \begin{cases} Y_1 \text{ if } n = 1 \\ [_r Y_1, \ldots, Y_n] \text{ o.w.} \end{cases}$$

$$\begin{array}{l} X([_r\Gamma_1, \Delta, A\!-\!\!\circ B, \Gamma_2]) \Rightarrow C \\ \quad l(\Delta) \Rightarrow B \\ \quad X(l(\Gamma_1, A, \Gamma_2)) \Rightarrow C \end{array} \quad [\bullet\!\!-\mathrm{L}]$$

$$\begin{array}{l} X([_r\Gamma_1, \Delta, A\!-\!\!\bullet\Gamma_2]) \Rightarrow C \\ \quad r(\Delta) \Rightarrow B \\ \quad X(r(\Gamma_1, A, \Gamma_2)) \Rightarrow C \end{array} \quad [\circ\!\!-\mathrm{L}]$$

$$\begin{array}{l} X \Rightarrow A\bullet\!\!-B \\ \quad l'(X, B) \Rightarrow A \end{array} \quad [\bullet\!\!-\mathrm{R}] \qquad l'(X, B) = \begin{cases} [_l Y_1, \ldots, Y_n, B] \text{ if } X = [_l Y_1, \ldots, Y_n] \\ [_l X, B] \text{ o.w.} \end{cases}$$

$$\begin{array}{l} X \Rightarrow A\circ\!\!-B \\ \quad r'(X, B) \Rightarrow A \end{array} \quad [\circ\!\!-\mathrm{R}] \qquad r'(X, B) = \begin{cases} [_r Y_1, \ldots, Y_n, B] \text{ if } X = [_r Y_1, \ldots, Y_n] \\ [_r X, B] \text{ o.w.} \end{cases}$$

$$\begin{array}{l} X \Rightarrow B\!-\!\!\bullet A \\ \quad r''(B, X) \Rightarrow A \end{array} \quad [-\!\!\bullet\mathrm{R}] \qquad r''(B, X) = \begin{cases} [_r B, Y_1, \ldots, Y_n] \text{ if } X = [_r Y_1, \ldots, Y_n] \\ [_r B, X] \text{ o.w.} \end{cases}$$

$$\begin{array}{l} X \Rightarrow B\!-\!\!\circ A \\ \quad l''(B, X) \Rightarrow A \end{array} \quad [-\!\!\circ\mathrm{R}] \qquad l''(B, X) = \begin{cases} [_l B, Y_1, \ldots, Y_n] \text{ if } X = [_l Y_1, \ldots, Y_n] \\ [_l B, X] \text{ o.w.} \end{cases}$$

Figure 3: Gentzen sequent presentation of normal associative headed calculus

References

[1] Barry, G., Hepple, M., Leslie, N. and Morrill, G. (1991) 'Proof figures and structural operators for categorial grammar'. Proceedings COLING91, Berlin.

[2] Barry, G. & M. Pickering (1990) 'Dependency and constituency in Categorial Grammar', in Barry & Morrill (eds.) *Studies in Categorial Grammar*, CCS, Edinburgh.

[3] Benthem, J. van (1991) *Language in Action. Categories, Lambdas, and Dynamic Logic.* Studies in Logic, North-Holland, Amsterdam.

[4] Chomsky, N. and Halle, M (1968) *The Sound Pattern of English*, Harper and Row, New York.

[5] Došen, K. (1988) 'Sequent systems and groupoid models. I', *Studia Logica 47*, 353–385.

[6] Došen, K. (1989) 'Sequent systems and groupoid models. II', *Studia Logica 48*, 41–65.

[7] Došen, K. (1990) 'Modal logic as metalogic'. SNS Report, Tübingen.

[8] Dunn, J.M. (1991) 'Gaggle theory: an abstraction of Galois connections and residuation, with applications to negation, implication, and various logical operators'. In Van Eijck (ed.) *Logics in AI.* JELIA Proceedings. Springer, Berlin.

[9] Gabbay, D. (1991a) *Labelled Deductive Systems.* Draft. Oxford University Press (to appear).

[10] Gabbay, D. (1991b) 'A general theory of structured consequence relations'. Draft. To appear in *Theoretical Foundations for Non-Monotonic Reasoning*–Part 3.

[11] Gallier, J. (1986) *Logic for Computer Science. Foundations of Automatic Theorem Proving.* Harper & Row, New York.

[12] Girard, J.Y., Y. Lafont & P. Taylor (1989) *Proofs and types.* Cambridge Tracts in Theoretical Computer Science 7. Cambridge.

[13] Hoeksema, J. (1985) 'Formal Properties of Stress Representation'. In Hulst and Smith (eds.) *Advances in Nonlinear Phonology.* Foris, Dordrecht.

[14] Lambek, J. (1958) 'The Mathematics of Sentence Structure', *American Mathematical Monthly* 65, pp. 154-170.

[15] Lambek, J. (1961) 'On the Calculus of Syntactic Types', in R. Jakobson (ed.) *Structure of Language and its Mathematical Aspects.* Providence.

[16] Liberman, M. and Prince, A. (1977) 'On Stress and Linguistic Rhythm', *Linguistic Inquiry 8*, **2**, 249–336.

[17] Moortgat, M. (1988) *Categorial Investigations. Logical and Linguistic Aspects of the Lambek Calculus.* Foris, Dordrecht.

[18] Morrill, G., Leslie, N., Hepple, M. and Barry G. (1990) 'Categorial Deductions and Structural Operations'. In Barry & Morrill (eds.) *Studies in Categorial Grammar*, CCS, Edinburgh.

[19] Oehrle, R.T. (1986) 'Sources and Structures of Linguistic Prominence in English'. In *Cognition and Representation*, Westview Press, 209–241.

[20] Roorda, D. (1991) *Resource Logics. Proof-theoretical Investigations.* Ph.D. Dissertation, Amsterdam.

[21] Troelstra, A.S. (1990) *Lectures on Linear Logic.* ITLI Prepublications X–90–15, Amsterdam.

[22] Wansing, H. (1990a) 'Formulas-as-types for a hierarchy of sublogics of intuitionistic propositional logic'. Berichte der Gruppe Logik, Wissenstheorie und Information 9/90. Freie Universität Berlin.

[23] Wansing, H. (1990b) 'Functional completeness for subsystems of intuitionistic propositional logic'. Berichte der Gruppe Logik, Wissenstheorie und Information 10/90. Freie Universität Berlin.

[24] Zielonka, W. (1981) 'Axiomatizability of Ajdukiewicz-Lambek Calculus by means of cancellation Schemes', *Zeitschrift für mathematische Logik und Grundlagen der Mathematik* 27, p. 215-224.