

Structural equations in language learning

Michael Moortgat

Utrecht Institute of Linguistics — OTS
Trans 10, 3512 JK Utrecht, The Netherlands
Michael.Moortgat@let.uu.nl

Abstract. In categorial systems with a fixed structural component, the learning problem comes down to finding the solution for a set of type-assignment equations. A hard-wired structural component is problematic if one wants to address issues of structural variation. Our starting point is a type-logical architecture with separate modules for the logical and the structural components of the computational system. The logical component expresses *invariants* of grammatical composition; the structural component captures variation in the realization of the correspondence between form and meaning. Learning in this setting involves finding the solution to both the type-assignment equations and the structural equations of the language at hand. We develop a view on these two subtasks which pictures learning as a process moving through a two-stage cycle. In the first phase of the cycle, type assignments are computed statically from structures. In the second phase, the lexicon is enhanced with facilities for structural reasoning. These make it possible to dynamically relate structures during on-line computation, or to establish off-line lexical generalizations. We report on the initial experiments in [15] to apply this method in the context of the Spoken Dutch Corpus.

For the general type-logical background, we refer to [12]; §1 has a brief recap of some key features.

1 Constants and variation

One can think of type-logical grammar as a functional programming language with some special purpose features to customize it for natural language processing tasks. Basic constructs are demonstrations of the form $\Gamma \vdash A$, stating that a structure Γ is a well-formed expression of type A . These statements are the outcome of a process of computation. Our programming language has a built-in vocabulary of logical constants to construct the type formulas over some set of atomic formulas in terms of the indexed unary and binary operations of (1a). Parallel to the formula language, we have the structure-building operations of (1b) with $(\cdot \circ_i \cdot)$ and $\langle \cdot \rangle^j$ as counterparts of \bullet_i and \diamond_j respectively. The indices i and j are taken from given, finite sets I, J which we refer to as composition *modes*.

- (1) a. $\text{Typ} ::= \text{Atom} \mid \diamond_j \text{Typ} \mid \square_j \text{Typ} \mid \text{Typ} \bullet_i \text{Typ} \mid \text{Typ} /_i \text{Typ} \mid \text{Typ} \setminus_i \text{Typ}$
b. $\text{Struc} ::= \text{Typ} \mid \langle \text{Struc} \rangle^j \mid \text{Struc} \circ_i \text{Struc}$

In presenting the rules of computation characterizing the derivability relation, we keep logical and structural aspects apart. The different composition modes all have the same *logical rules*. But we can key access to different *structural rules* by means of the mode distinctions.

Let us consider the logical component first. For each type-forming operation in (1a) there is a *constructor* rule (rule of use, assembly) and a *destructor* rule (rule of proof, disassembly). These rules can be presented in a number of equivalent ways: algebraically, in Gentzen or natural deduction format, or in a proof net presentation. The assembly/disassembly duality comes out particularly clearly in the algebraic presentation, where we have the *residuation laws* of (2). In the natural deduction format, these laws will turn up as Introduction/Elimination rules; in the Gentzen format as Left/Right introduction rules.

$$(2) \quad \begin{aligned} \diamond_j A \vdash B &\text{ iff } A \vdash \square_j B \\ A \vdash C /_i B &\text{ iff } A \bullet_i B \vdash C \quad \text{iff } B \vdash A \setminus_i C \end{aligned}$$

The composition of natural language *meaning* proceeds along the lines of the Curry-Howard interpretation of derivations, which reads off the meaning assembly from the logical inference steps that make up a computation. In this sense, the composition of meaning is *invariant* across languages — it is fully determined by the elimination/introduction rules for the grammatical constants. Languages show variation in the structural realization of the correspondence between form and meaning. Such variation is captured by the structural component of the computational system. Structural rules have the status of *non-logical axioms* (or postulates). The structural rules we consider in this paper are *linear transformations*:¹ they reassemble grammatical material, but they cannot duplicate or waste it. The familiar rules of Associativity and Commutativity in (3) can serve as illustrations. In a global form, these rules destroy essential grammatical information. But we will see in §2.2 how they can be tamed.

$$(3) \quad \begin{aligned} A \bullet B \vdash B \bullet A \\ (A \bullet B) \bullet C \dashv\vdash A \bullet (B \bullet C) \end{aligned}$$

To obtain a type-logical grammar over a terminal vocabulary Σ , we have to specify a lexicon $\text{Lex} \subseteq \Sigma \times \text{Type}$, assigning each vocabulary item a finite number of types. A grammar, then, is a structure $G = (\text{Lex}, \text{Op})$, where Op is the union of the logical and the structural rules. Let $L(G, B)$ be the set of strings of type B generated by G , and let $\text{Struc}(A_1, \dots, A_n)$ be the set of structure trees with yield A_1, \dots, A_n . For a string $\sigma = w_1 \dots w_n \in \Sigma^+$, we say that $\sigma \in L(G, B)$ iff there are A_1, \dots, A_n and $T \in \text{Struc}(A_1, \dots, A_n)$ such that for $1 \leq i \leq n$, $\langle w_i, A_i \rangle \in \text{Lex}$, and $T \vdash B$. To obtain $L(G)$, the language generated by the type-logical grammar G , we compute $L(G, B)$ for some fixed (finite set of) goal type(s)/start symbol(s) B .

¹ That is, we do not address multiple-use issues like parasitic gaps here, which might require (a controlled form of) Contraction.

2 Structural reasoning in learning

The modular treatment of logical and structural reasoning naturally suggests that we break up the learning problem in two subtasks. One task consists in finding appropriate categorization for the words in Σ as the elementary building blocks for meaning composition. This is essentially the problem of computing type-assignments as we know it from classical categorial learning theory. The second subtask addresses the question: What is the dynamic potential of words in syntax? Answering this question amounts to solving structural equations within a space set by universal grammar.

In tackling the second subtask, we will rely heavily on the unary constant \diamond and its residual \square . As we have seen in §1, the binary product \bullet captures the composition of grammatical parts, while the residual implications $/$ and \backslash express incompleteness with respect to the composition relation. Extending the vocabulary with the unary constants \diamond, \square substantially increases the analytical power of the categorial type language. This can be seen already in *the base logic* (i.e. the pure residuation logic, with empty structural module): the unary operators make it possible to refine type-assignments that would be overgenerating without the unary decoration. Moreover, in systems with a non-empty structural component, the unary operators can provide lexically anchored control over structural reasoning. We discuss these two aspects in turn.

In the base logic, the fundamental derivability pattern created by the unary operators is $\diamond\square A \vdash A \vdash \square\diamond A$. One can exploit this pattern to obtain the agreement configurations of (4).

$$(4) \quad \diamond\square A \bullet \begin{cases} \diamond\square A \backslash B \vdash B \\ A \backslash B \vdash B \\ \square\diamond A \backslash B \vdash B \end{cases} \quad A \bullet \begin{cases} \diamond\square A \backslash B \not\vdash B \\ A \backslash B \vdash B \\ \square\diamond A \backslash B \vdash B \end{cases} \quad \square\diamond A \bullet \begin{cases} \diamond\square A \backslash B \not\vdash B \\ A \backslash B \not\vdash B \\ \square\diamond A \backslash B \vdash B \end{cases}$$

The treatment of polarity sensitive items in [2] illustrates this use of modal decoration. Consider the contrast between ‘Nobody left yet’ with the negative polarity item ‘yet’ and ‘*Somebody left yet’. The negative polarity trigger ‘nobody’ is assigned the type $s/(np \backslash \square\diamond s)$, whereas ‘somebody’ has the undecorated type $s/(np \backslash s)$. The negative polarity item ‘yet’ is typed as $\square\diamond s \backslash \square\diamond s$ — it requires a trigger such as ‘nobody’ to check the $\square\diamond$ decoration in its result subtype. In the base logic, we have $s/(np \backslash \square\diamond s) \vdash s/(np \backslash s)$, i.e. the $\square\diamond$ decoration on argument subtypes can be simplified away, allowing a derivation of e.g. ‘Nobody left’ where there is no polarity item to be checked. This strategy of unary decoration is extended in [3] to lexically enforce constraints on the scopal possibilities of generalized quantifier expressions such as discussed in [1].

For the use of unary type decoration to provide controlled access to structural reasoning, we can rely on the results of [11]. In that paper, we present embedding translations \cdot^{\natural} from source logics \mathcal{L} to target logics $\mathcal{L}'\diamond$, in the sense that $A \vdash B$ is derivable in \mathcal{L} iff $A^{\natural} \vdash B^{\natural}$ is derivable in $\mathcal{L}'\diamond$. The translations \cdot^{\natural} decorate the type assignments of the source logic with the unary operators \diamond, \square in such a way that they licence access to restricted versions of the structural rules.

In the following sections, we use this modalization strategy to develop our two-stage view on the learning process. The first stage consists of learning from structures in the base logic. Because the base logic has no facilities for structural reasoning, the lexical ambiguity load in this phase soon becomes prohibitive. In the second stage, the lexicon is optimized by shifting to modalized (\diamond, \square) type assignments. The modal decoration is designed in such a way that lexical ambiguity is reduced to derivational polymorphism. We will assume that the learning process cycles through these two stages. To carry out this program, a number of questions have to be answered:

- What kind of MODAL DECORATION do we envisage?
- What is the STRUCTURAL PACKAGE which delimits the space for variation?

We discuss these questions in §2.2. First, we address the problem of learning from structures in the base logic.

2.1 Solving type equations by hypothetical reasoning

The unification perspective on learning type assignments from structures is well understood — we refer the reader to the seminal work of [5], and to [9]. Here we present the problem of solving type assignment equations from a Logic Programming perspective in order to highlight the role of hypothetical reasoning in the process.

Consider the standard abstract interpreter for logic programs (see for example [16]). The resolution algorithm takes as input a program P and a goal G and initializes the resolvent to be the input goal G . While the resolvent is non-empty, the algorithm chooses a goal A from the resolvent and a matching program clause $A' \leftarrow B_1, \dots, B_n$ ($n \geq 0$) from P such that A and A' unify with mgu θ . A is removed from the resolvent and B_1, \dots, B_n added instead, with θ applied to the resolvent and to G . As output, the algorithm produces $G\theta$, if the resolvent is empty, or failure, if the empty clause cannot be derived.

[6] presents a variant on this refutation algorithm which does not return failure for an incomplete derivation, but instead extracts information from the non-empty resolvent which provides the *conditional answer* that would make the goal G derivable. In [6] the conditional answer approach is illustrated with the polymorphic type inference problem from lambda calculus. This illustration can be straightforwardly adapted to our categorial type inference problem. Writing $\Gamma \triangleleft \Delta$ for functor-argument structures with the functor as the left component and $\Gamma \triangleright \Delta$ for such structures with the functor as the right component, the ‘program clauses’ for categorial type assignment appear as (5).

$$(5) \Gamma \triangleleft \Delta \vdash A \text{ if } \Gamma \vdash A/B \text{ and } \Delta \vdash B \quad \Gamma \triangleright \Delta \vdash A \text{ if } \Gamma \vdash B \text{ and } \Delta \vdash B \setminus A$$

In order to derive the empty clause, the program would need a lexicon of type assignment facts. In the absence of such a lexicon (or in the case of an incomplete lexicon), the conditional answer derivation returns the resolvent with the type

assignment goals that would be needed for a successful refutation. The conditional answer is optimized by *factoring*, i.e. by the contraction of unifiable type assignment goals. A sample run of the algorithm is given below.

input Alice ▷ dreams, Lewis ▷ dreams, (the ◁ girl) ▷ dreams, Alice ▷ (knows ◁ Lewis), Lewis ▷ (knows ◁ Alice), (the ◁ mathematician) ▷ (knows ◁ Alice), Alice ▷ (knows ◁ (the ◁ mathematician)), Alice ▷ (irritates ◁ (the ◁ mathematician)), (the ◁ mathematician) ▷ (irritates ◁ Alice), Alice ▷ (dreams ▷ (about ◁ Lewis)), Lewis ▷ (wrote ◁ (the ◁ book)), Lewis ▷ (knows ◁ (the ◁ girl)), Lewis ▷ (wrote ◁ (the ◁ (nice ◁ book))), (the ◁ girl) ▷ (knows ◁ (the ◁ book)), (the ◁ girl) ▷ (knows ◁ (the ◁ (book ▷ (which ◁ (irritates ◁ (the ◁ mathematician)))))), (the ◁ girl) ▷ (knows ◁ (the ◁ (book ▷ ((which ◁ (the ◁ mathematician)) ◁ wrote))))), ...

output The term assignments of (6). With the gloss $A = n$, $B = np$, $C = s$ for the type variables, these will look familiar enough.

$$(6) \quad \begin{array}{ll} \text{Alice} \vdash B & \text{Lewis} \vdash B \\ \text{dreams} \vdash B \setminus C & \text{the} \vdash B/A \\ \text{girl} \vdash A & \text{mathematician} \vdash A \\ \text{book} \vdash A & \text{nice} \vdash A/A \\ \text{about} \vdash ((B \setminus C) \setminus (B \setminus C))/B & \text{irritates} \vdash (B \setminus C)/B \\ \text{knows} \vdash (B \setminus C)/B & \text{wrote} \vdash (B \setminus C)/B \\ \text{which} \vdash (A \setminus A)/(B \setminus C) & \text{which} \vdash ((A \setminus A)/((B \setminus C)/B))/B \end{array}$$

The interesting point of this run is the two type assignments for ‘which’: one for subject relativization (obtained from ‘... which irritates Alice’), the other for object relativization (from ‘... which Lewis wrote’) — and, of course, many others for different structural contexts. Factoring (unification) cannot relate these assignments: this would require structural reasoning. To see that the learning algorithm is missing a generalization here, consider the meaning assembly for these two type assignments to the relative pronoun. The lambda program of (7a), expressing property intersection, would be adequate for the subject relativization type. To obtain appropriate meaning assembly for the object relativization assignment, one would need the lambda program of (7b).

$$(7) \quad \begin{array}{ll} a. \text{ which} \vdash (n \setminus n)/(np \setminus s) & \lambda x \lambda y \lambda z. (x \ z) \wedge (y \ z) \quad (= \mathbf{wh}) \\ b. \text{ which} \vdash ((n \setminus n)/((np \setminus s)/np))/np & \lambda x' \lambda y' \lambda y \lambda z. ((y' \ z) \ x') \wedge (y \ z) \end{array}$$

The point is that these two meaning programs are not unrelated. We can see this by analysing them from the perspective of **LP** (or Multiplicative Linear Logic) — a system which removes all structural obstacles to meaning composition in the sense that free restructuring and reordering under Associativity and Commutativity are available. In **LP**, the different type assignments are simply structural realizations of one and the same meaning. See the derivation of Figure 1, which produces the proof term of (8). Unfortunately, **LP** is of little use as a framework for natural language analysis: apart from the required meaning assembly of (8), there is a second derivation for the type transition of Figure 1

producing the proof term $\lambda y_1. \lambda x_2. (\mathbf{wh} (x_2 y_1))$, which gets the thematic roles of subject and object wrong. Can we find a way of controlling structural reasoning, so that we can do with a single type assignment to the relative pronoun, while keeping the thematic structure intact?

$$\begin{array}{c}
\frac{\frac{[p_1 \vdash np]^2 \quad \frac{[r_1 \vdash (np \setminus s)/np]^3 \quad [r_0 \vdash np]^4}{r_1 \circ r_0 \vdash np \setminus s} [/E]}{p_1 \circ (r_1 \circ r_0) \vdash s} [Ass]}{\frac{(p_1 \circ r_1) \circ r_0 \vdash s}{r_0 \circ (p_1 \circ r_1) \vdash s} [Comm]}{p_1 \circ r_1 \vdash np \setminus s} [Ass]} \quad \frac{\frac{\text{which}}{(n \setminus n)/(np \setminus s)}}{[I]^1} \\
\frac{[p_2 \vdash n]^4 \quad \frac{\text{which} \circ (p_1 \circ r_1) \vdash n \setminus n}{[E]}}{p_2 \circ (\text{which} \circ (p_1 \circ r_1)) \vdash n} [Ass]}{\frac{p_2 \circ ((\text{which} \circ p_1) \circ r_1) \vdash n}{(\text{which} \circ p_1) \circ r_1 \vdash n \setminus n} [I]^4}[\text{which} \circ p_1 \vdash (n \setminus n)/(np \setminus s)/np] [I]^3}[\text{which} \vdash ((n \setminus n)/((np \setminus s)/np))/np] [I]^2}
\end{array}$$

Fig. 1. Relating subject and object relativization in **LP**.

$$(8) \quad \mathbf{wh} \vdash_{\mathbf{LP}} \lambda y_1. \lambda x_2. (\mathbf{wh} \lambda z_0. ((x_2 z_0) y_1)) \quad =_{\beta} \\
\lambda y_1. \lambda x_2. \lambda z_3. \lambda x_4. (((x_2 x_4) y_1) \wedge (z_3 x_4))$$

2.2 Modal decorations for solving structural equations

To answer this question, we turn to the second phase of the learning process. Let $\mathbf{type}(w)$ be the set of types which the algorithm of §2.1 associates with a word w in the base logic lexicon: $\mathbf{type}(w) = \{A \mid \langle w, A \rangle \in \mathbf{Lex}\}$. The type assignments found in §2.1 are built up in terms of the binary connectives $/$ and \setminus : they do not exploit the full type language, and they do not appeal to structural reasoning. In the second phase of the learning cycle, these limitations are lifted. We translate the question at the end of the previous section as follows: Can we find a modal decoration \cdot^{\sharp} and an associated structural package \mathcal{R} which would allow the learner to identify a $B \in \mathbf{type}(w)$ such that $B^{\sharp} \vdash A$ for all $A \in \mathbf{type}(w)$? Or, if we opt for a weaker package \mathcal{R} that makes unique type-assignment unattainable, can we at least reduce the cardinality of $\mathbf{type}(w)$ by removing some derivable type assignments from the type-set? In what follows, we consider various options for \cdot^{\sharp} and for \mathcal{R} .

Consider first the decorations $[\cdot], [\cdot] : \mathbf{B} \mapsto \mathbf{B}\diamond$ of (9) for *input* and *output* polarities respectively.²

$$(9) \quad \begin{array}{l} [p] = p \qquad [p] = p \\ [A \bullet B] = \diamond\Box[A] \bullet \diamond\Box[B] \quad [A \bullet B] = [B] \bullet [A] \\ [A/B] = \diamond\Box[A]/[B] \quad [A/B] = [A]/\diamond\Box[B] \\ [B\backslash A] = [B]\backslash\diamond\Box[A] \quad [B\backslash A] = \diamond\Box[B]\backslash[A] \end{array}$$

The effect of $[\cdot], [\cdot]$ is to prefix all input subformulas with $\diamond\Box$. In the absence of structural rules, this modal marking would indeed be a pure embellishment, in the sense that $\mathbf{B} \vdash \bullet\Gamma \Rightarrow A$ iff $\mathbf{B}\diamond \vdash [\bullet\Gamma] \Rightarrow [A]$. But we are interested in the situation where the $\diamond\Box$ decoration gives access to structural reasoning. As a crude first attempt, consider the postulate package (10) which would be the modal analogue of (3), i.e. it allows full restructuring and reordering under \diamond control. In the Associativity postulates, one of the factors A_i ($1 \leq i \leq 3$) is of the form $\diamond A'$, with the rule label indexed accordingly.

$$(10) \quad \begin{array}{l} A_1 \bullet (A_2 \bullet A_3) \vdash (A_1 \bullet A_2) \bullet A_3 \quad [A_i] \\ (A_1 \bullet A_2) \bullet A_3 \vdash A_1 \bullet (A_2 \bullet A_3) \quad [A_i^{-1}] \\ \\ B \bullet \diamond A \vdash \diamond A \bullet B \quad [C] \\ \diamond B \bullet A \vdash A \bullet \diamond B \quad [C^{-1}] \end{array}$$

With the package (10), we again have the embedding

$$\mathbf{LP} \vdash \bullet\Gamma \Rightarrow A \quad \text{iff} \quad \mathbf{B}\diamond + (10) \vdash [\bullet\Gamma] \Rightarrow [A]$$

Consider again the type assignments we computed in (6) for the relative pronoun:

$$\text{type}(\text{which}) = \{(n\backslash n)/(np\backslash s), \\ ((n\backslash n)/((np\backslash s)/np))/np, \\ \dots\}$$

Calculating $[\cdot]$ for the first of these, we obtain (11)

$$(11) \quad \begin{aligned} [(n\backslash n)/(np\backslash s)] &= \diamond\Box[n\backslash n]/[np\backslash s] \\ &= \diamond\Box([n]\backslash\diamond\Box[n])/(\diamond\Box[np]\backslash[s]) \\ &= \diamond\Box(n\backslash\diamond\Box n)/(\diamond\Box np\backslash s) \end{aligned}$$

which indeed gives the type transformation

$$\diamond\Box(n\backslash\diamond\Box n)/(\diamond\Box np\backslash s) \vdash ((n\backslash n)/((np\backslash s)/np))/np$$

In Figure 2, we give an example derived from the modalized type assignment to the relative pronoun. We concentrate on the subderivation that realizes non-peripheral extraction via \diamond controlled structural reasoning. The modal decoration implements the ‘key and lock’ strategy of [14]. For a constituent of type

² \mathbf{B} is the base logic for the binary connectives: the pure residuation logic for $/, \bullet, \backslash$, with no structural postulates at all. $\mathbf{B}\diamond$ is the extended system with the unary connectives \diamond, \Box .

$\diamond \Box A$, the \diamond component provides access to the structural postulates in (10). At the point where such a marked constituent has found the structural position where it can be used by the logical rules, the \diamond key unlocks the \Box lock: the control feature is cancelled through the basic law $\diamond \Box A \vdash A$.

$$\begin{array}{c}
\frac{\frac{\text{dedicated}}{((np \setminus s)/pp)/np} \quad \frac{[p_2 \vdash \Box np]^5}{\langle p_2 \rangle \vdash np} \quad [\Box E] \quad \frac{\text{to} \quad \text{Alice}}{pp/np \quad np}}{\text{dedicated} \circ \langle p_2 \rangle \vdash (np \setminus s)/pp} \quad [/\!E] \quad \frac{\text{to} \circ \text{Alice} \vdash pp}{pp/np \quad np} \quad [/\!E]}{\frac{\text{Lewis}}{np} \quad \frac{\text{dedicated} \circ \langle p_2 \rangle \vdash (np \setminus s)/pp}{(dedicated \circ \langle p_2 \rangle) \circ (to \circ Alice) \vdash np \setminus s} \quad [/\!E]}{Lewis \circ ((dedicated \circ \langle p_2 \rangle) \circ (to \circ Alice)) \vdash s} \quad [/\!E]} \\
\frac{\text{Lewis} \circ ((dedicated \circ \langle p_2 \rangle) \circ (to \circ Alice)) \vdash s}{\text{Lewis} \circ (dedicated \circ (\langle p_2 \rangle \circ (to \circ Alice))) \vdash s} \quad [A_2] \\
\frac{\text{Lewis} \circ (dedicated \circ (\langle p_2 \rangle \circ (to \circ Alice))) \vdash s}{\text{Lewis} \circ (dedicated \circ ((to \circ Alice) \circ \langle p_2 \rangle)) \vdash s} \quad [C] \\
\frac{\text{Lewis} \circ (dedicated \circ ((to \circ Alice) \circ \langle p_2 \rangle)) \vdash s}{\text{Lewis} \circ ((dedicated \circ (to \circ Alice)) \circ \langle p_2 \rangle) \vdash s} \quad [A_3^{-1}] \\
\frac{\text{Lewis} \circ ((dedicated \circ (to \circ Alice)) \circ \langle p_2 \rangle) \vdash s}{(\text{Lewis} \circ (dedicated \circ (to \circ Alice))) \circ \langle p_2 \rangle \vdash s} \quad [A_3^{-1}] \quad \dagger \\
\frac{[r_1 \vdash \diamond \Box np]^4 \quad \frac{\langle p_2 \rangle \circ (\text{Lewis} \circ (dedicated \circ (to \circ Alice))) \vdash s}{\langle p_2 \rangle \circ (\text{Lewis} \circ (dedicated \circ (to \circ Alice))) \vdash s} \quad [C^{-1}]}{r_1 \circ (\text{Lewis} \circ (dedicated \circ (to \circ Alice))) \vdash s} \quad [/\!E]^5}{\text{Lewis} \circ (dedicated \circ (to \circ Alice)) \vdash \diamond \Box np \setminus s} \quad [/\!I]^4
\end{array}$$

Fig. 2. Non-peripheral extraction under \diamond control: ‘(the book which) Lewis dedicated to Alice’. The (\dagger) sign marks the entry point for an alternative derivation, driven from an assignment $(n \setminus n)/(s/\diamond \Box np)$ for the relative pronoun. See the discussion in §2.3.

2.3 Calibration

The situation we have obtained is a crude first approximation in two respects. First, the *modal decoration* is overly rich in the sense that every input subformula is given a chance to engage in structural reasoning. Second, the structural package (10) is not much better than the global structural reasoning of §1 in that it allows full reordering and restructuring, this time under \diamond control. The task here is to find the proper trade-off between the degree of lexical ambiguity one is prepared to tolerate, and the expressivity of the structural package. We discuss these two considerations in turn.

STRUCTURAL REASONING Consider first the structural component. The package in (12) seems to have a pleasant balance between expressivity and structural constraint. We refer the reader to the discussion of extraction asymmetries between head-initial and head-final languages in [14], Dutch verb-raising in [13], and the analysis of French cliticization in [10], all of which are based essentially on the structural features of (12). In this section, we discuss the postulates in their schematic form — further fine-tuning in terms of mode distinctions for the \bullet and \diamond operations is straightforward and will be taken into consideration in §3.

$$\begin{aligned}
(12) \quad & \diamond A \bullet (B \bullet C) \dashv\vdash (\diamond A \bullet B) \bullet C & (Pl1) \\
& \diamond A \bullet (B \bullet C) \dashv\vdash B \bullet (\diamond A \bullet C) & (Pl2) \\
& (A \bullet B) \bullet \diamond C \dashv\vdash (A \bullet \diamond C) \bullet B & (Pr2) \\
& (A \bullet B) \bullet \diamond C \dashv\vdash A \bullet (B \bullet \diamond C) & (Pr1)
\end{aligned}$$

The postulates can be read in two directions. In the \vdash direction, they have the effect of *revealing* a \diamond marked constituent, by promoting it from an embedded position to a position where it is visible for the logical rules: the immediate left or right daughter of the structural root node.³ In the $\dashv\vdash$ direction, they *hide* a marked constituent, pushing it from a visible position to an embedded position. Apart from the $\dashv\vdash$ asymmetry, the postulates preserve the left-right asymmetry of the primitive operations $/$ and \backslash : the *Pl* postulates have a bias for left branches; for the *Pr* postulates only right branches are accessible.

We highlight some properties of this package.

Linearity The postulates rearrange a structural configuration; they cannot duplicate or waste grammatical material.

Control The postulates operate under \diamond control. Because the logic doesn't allow the control features to enter a derivation out of the blue, this means they have to be lexically anchored.

Locality The window for structural reasoning is strictly local: postulates can only see two products in construction with each other (with one of the factors bearing the licensing \diamond feature).

Recursion Non-local effects of structural reasoning arise through recursion.

In comparison with universal package (10), the postulates of (12) represent a move towards *more specific* forms of structural reasoning. One can see this in the deconstruction of *Pr2* (similarly, *Pl2*) as the compilation of a sequence of structural inferences in (10). The postulate *C* of (10) is removed from (12) as an independent structural inference; instead, a restricted use of it is *encapsulated* in *Pr2* (or *Pl2*).

$$(A \bullet B) \bullet \diamond C \vdash (A \bullet \diamond C) \bullet B \quad (Pr2)$$

$$\text{Combinator: } Pr2 = \gamma^{-1}(\gamma(A_2) \circ C) \circ A_3^{-1}$$

$$(A \bullet B) \bullet \diamond C \stackrel{A_3^{-1}}{\vdash} A \bullet (B \bullet \diamond C) \stackrel{C}{\vdash} A \bullet (\diamond C \bullet B) \stackrel{A_2}{\vdash} (A \bullet \diamond C) \bullet B$$

The careful reader may have noticed that the package (12) is too weak to allow the derivation of the extraction example in Figure 2. The modalized type assignment for the relative pronoun in (11) has $(\diamond \square np \backslash s)$ as the subtype for the relative clause body: the $\diamond \square np$ hypothesis is withdrawn to the left. But

³ The reader should keep in mind that, as a result of the cut rule, it is the pattern to the *left* of \vdash that shows up in the conclusion of the natural deduction inferences we have given.

this means that complement positions on right branches are inaccessible for the $\diamond\Box np$ gap hypothesis, if we want to stay within the limits of (12). Accessing a right branch position from the launching point of $\diamond\Box np$ would require the extra postulate *Pl3* (and by symmetry *Pr3*), establishing communication between the left- and right-biased options. Again, these are forms of structural reasoning encapsulating a controlled amount of *C*.

$$(13) \quad \begin{array}{l} \diamond A \bullet (B \bullet C) \dashv\vdash B \bullet (C \bullet \diamond A) \quad (Pl3)? \\ (A \bullet B) \bullet \diamond C \dashv\vdash (\diamond C \bullet A) \bullet B \quad (Pr3)? \end{array}$$

$$\frac{\frac{\frac{\text{dedicated}}{((np \setminus s)/pp)/np} \quad \frac{[s_1 \vdash \Box np]^2}{\langle s_1 \rangle \vdash np} \quad [\Box E] \quad \frac{\text{to} \quad \text{Alice}}{pp/np \quad np} \quad [/\!E]}{\text{dedicated} \circ \langle s_1 \rangle \vdash (np \setminus s)/pp} \quad [/\!E] \quad \frac{\text{to} \circ \text{Alice} \vdash pp}{np} \quad [/\!E]}{\frac{\text{Lewis}}{np} \quad \frac{(\text{dedicated} \circ \langle s_1 \rangle) \circ (\text{to} \circ \text{Alice}) \vdash np \setminus s}{\text{Lewis} \circ ((\text{dedicated} \circ \langle s_1 \rangle) \circ (\text{to} \circ \text{Alice})) \vdash s} \quad [/\!E]}}{\frac{\text{Lewis} \circ ((\text{dedicated} \circ (\text{to} \circ \text{Alice})) \circ \langle s_1 \rangle) \vdash s}{\text{Lewis} \circ (\text{dedicated} \circ (\text{to} \circ \text{Alice})) \circ \langle s_1 \rangle \vdash s} \quad [Pr2]}{\frac{[q_1 \vdash \diamond np]^1 \quad (\text{Lewis} \circ (\text{dedicated} \circ (\text{to} \circ \text{Alice}))) \circ \langle s_1 \rangle \vdash s \quad [Pr1]}{\text{Lewis} \circ (\text{dedicated} \circ (\text{to} \circ \text{Alice})) \circ q_1 \vdash s} \quad [/\!E]^2}}{\frac{\text{Lewis} \circ (\text{dedicated} \circ (\text{to} \circ \text{Alice})) \circ q_1 \vdash s}{\text{Lewis} \circ (\text{dedicated} \circ (\text{to} \circ \text{Alice})) \vdash s / \diamond \Box np} \quad [/\!I]^1}$$

Fig. 3. Non-peripheral extraction in terms of the package (12). Compare with the derivation in Figure 2.

There is a lexical alternative to strengthening the postulate package which is obtained from a directional variant of the type assignment to the relative pronoun, with $(s/\diamond\Box np)$ as the subtype for the relative clause body. Under this alternative, the lexicon assigns two types to ‘which’: one for subject relativization, one for non-subject cases. See the derivation in Figure 3. Notice the trade-off here between increasing the size of the lexicon (storage) versus simplification of the on-line computation (the structural package). Different learners could make different choices with respect to this trade-off: we do not want to assume that the solution for the lexicon and the structural module has to be unique. Individual solutions can count as equally adequate as long as they associate the same forms with the same meaning. As we have noticed in §1, meaning composition is fully determined by the logical introduction/elimination rules for the type-logical constants modulo directionality (i.e. / and \ are indented).

The two alternatives above make different choices with respect to the distribution of grammatical complexity over the lexicon and the structural module. For an example of alternative solutions that are essentially of the same complexity, we refer to the analysis of Dutch verb raising (VR) in [13], where a leftwing and a rightwing solution to the verb raising puzzle are presented. A schematic

comparison is given in Figure 4. We use mode 0 for the composition of the lexical cluster, and mode 1 for phrasal composition. The leftwing approach treats VR-triggers (modals/auxiliaries) as $\square_0(vp/{}_0inf)$, and relates the surface order to the configuration required for meaning assembly in terms of $Pl2$. The rightwing solution assigns VR-triggers a type $\square_0(inf\wedge{}_0vp)$, and obtains the phrasal reconfiguration in terms of $Pr1$. The first step, in the two derivations, is a feature percolation postulate checking lexicality of the verb cluster — we do not go into this aspect of the analysis here.

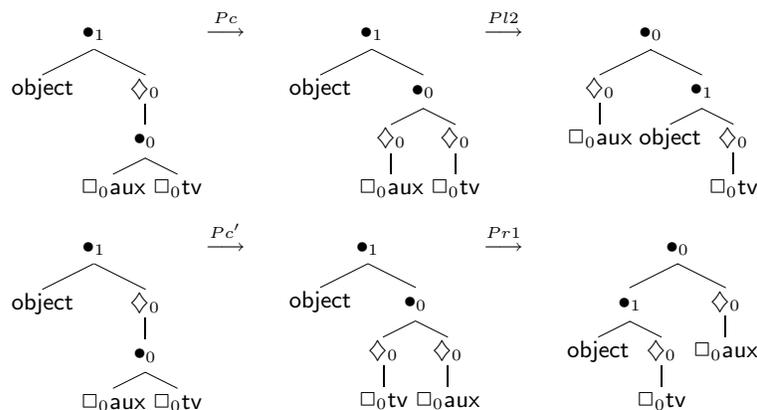


Fig. 4. Two views on Dutch VR. Surface order on the left. Meaning composition on the right. Phrasal composition: \bullet_1 , lexical cluster formation: \bullet_0 .

MODAL DECORATION The decoration of (9) marks every input subtype of a type formula. These subtypes will become active, in some stage of the derivation, in the structural part (antecedent) of a sequent, where they have the potential to trigger structural reasoning. Various options for a sparser style of modal decoration present themselves. One could choose to mark only *terminal* input formulae, i.e. lexical and hypothetical assumptions. Compare the full decoration for the relative pronoun type in (11) $\diamond\square(\diamond\square(n\wedge\square n)/(\diamond\square np\wedge s))$ with the terminal-inputs-only marking $\diamond\square((n\wedge n)/(\diamond\square np\wedge s))$. Or one could mark subtypes that one wants to consider as ‘major phrases’ — atomic types, and maybe some others, such as the relative clause type $n\wedge n$ in our example which, with a modal prefix, could be extraposed via the Pr family. In the next section, we compare some of these options in a concrete application.

A second aspect of the modal decoration strategy has to do with the dynamics of the learning process, as it goes through its two-phase cycle. In fine-tuning the lexicon, a reasonable strategy would be to go for the *most general* type compatible with the data. Modal decoration that turns out to be non-functional, in this sense, ‘dies off’ in the course of the learning process. In our relative pronoun

example, if all $\diamond\Box$ marks remain inert, except for the $\diamond\Box np$ gap hypothesis, the learner at a certain point is justified in applying the following pruning type transformation.

$$\diamond\Box(\diamond\Box(n\backslash\diamond\Box n)/(\diamond\Box np\backslash s)) \vdash (n\backslash n)/(\diamond\Box np\backslash s)$$

3 Testcase: the Spoken Dutch Corpus

In the previous section, we have sketched some options for the structural package and for the modal decoration that gives access to this package. We are currently exploring these options in the context of the Spoken Dutch Corpus project (CGN) — our initial experiments are reported in [15], on which this section is based.

The CGN project is a Dutch-Flemish collaborative effort to put together an annotated Dutch speech corpus of 10 million words — some 1000 hours of audio. Upon its completion in 2003 the CGN corpus will be a major resource for R&D in language and speech technology. A rich part-of-speech annotation is provided for the complete corpus. In addition, a core corpus of one million words is annotated syntactically (cf. [8]). The annotation is designed in such a way that it can be easily translated into the analysis formats of the various theoretical frameworks that want to use the CGN treebank to train and test computational grammars.

The CGN annotation provides information on two levels: syntactic constituent structure and the semantic dependencies between them. Because these two dimensions often do not run in parallel, the annotation format has to be rich enough to naturally represent dependency relations also where they are at odds with surface constituency. The DAG (directed acyclic graphs) data structure has the required expressivity.

Figure 5 is an example of a CGN annotation graph for the sentence ‘Wat gaan we doen het komend uur?’ (‘What shall we do the next hour?’). The nodes of the graph are labeled with syntactic category information (circled in Figure 5) — part-of-speech labels for the leaves, phrasal category labels for the internal nodes. The edges carry dependency labels (boxed in the picture), indicating the grammatical function of the immediate subconstituents of a phrase. In the dependency dimension, the basic distinctions are between the head of a phrase, its complements, and its modifiers.

The annotation graph of Figure 5 illustrates how the specific features of DAG’s (as compared to trees) are exploited. In the example, we want to express the fact that the interrogative pronoun ‘wat’ (‘what’) serves as direct object of the transitive infinitive ‘doen’ (‘do’). This is a discontinuous dependency, which leads to *crossing branches* in the annotation graph. At the same time, we want to indicate that the question word is responsible for projecting the top node WHQ (constituent question) and in this sense is acting as the head of the WHQ phrase. This means that a constituent can carry *multiple dependency roles*. Finally, an annotation DAG can consist of disconnected parts. This makes it possible to

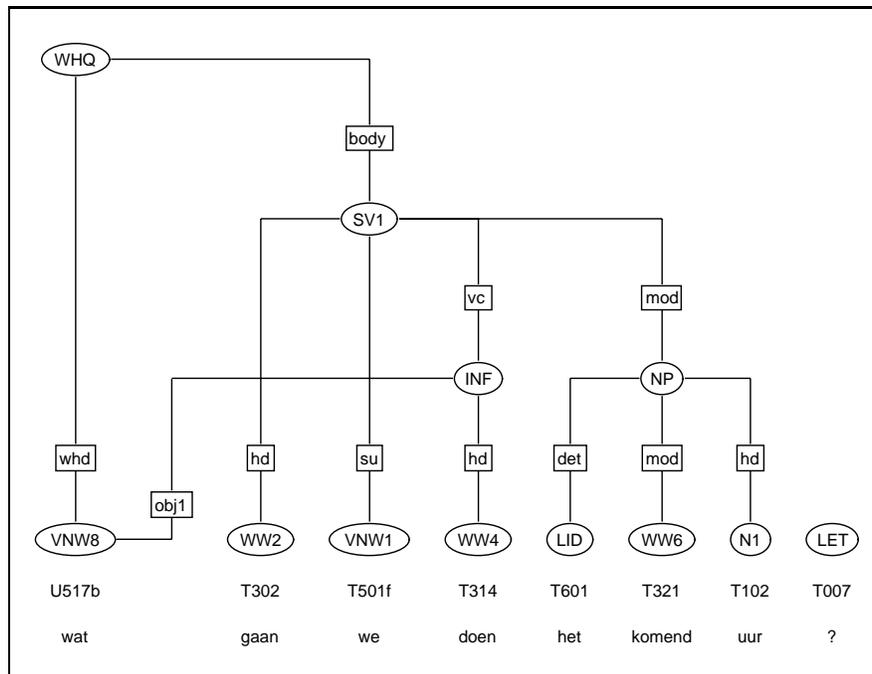


Fig. 5. A CGN annotation graph.

accommodate a number of phenomena that are very frequent in spontaneous speech: discourse fragments, interjections, etc.

The algorithm for the extraction of a type-logical lexicon out of the CGN annotation graphs is set up in such a way that one can easily experiment with the options we have discussed in the previous sections. The following parameters can be manipulated.

- Node labels. The choice here is which of the category labels are to be maintained as atomic types in the categorial lexicon.
- Edge labels. The dependency labels provide a rich source of information for mode distinctions. A ‘light’ translation implements the dependency labels as mode indices on the binary composition operation \bullet . One can furthermore keep the head component implicit by starting from a basic distinction between leftheaded and rightheaded products. An intransitive main clause verb, for example, is typed $np \setminus_{r(su)} s$: it creates a rightheaded configuration, where the np complement bears the subject role with respect to the head.
- Thematic hierarchy. One can fix the canonical order of complements within a dependency domain in terms of the degree of coherence with the head.
- Head position. For the various clausal types, one can determine the directional orientation of the head with respect to its complements.
- Licensing structural reasoning. Targets for $\diamond \square$ decoration.

In (14) the reader sees the effect of some of these settings on the lexicon that is extracted out of the annotation graph of Figure 5.

$$\begin{aligned}
(14) \quad & \text{doen} : \diamond_{hd} \square_{hd} (np \backslash_{r(obj1)} inf) \\
& \text{gaan} : \diamond_{hd} \square_{hd} ((s1 /_{l(vc)} inf) /_{l(su)} np) \\
& \text{het} : \diamond_{det} \square_{det} (\diamond_{mod} \square_{mod} (s1 \backslash s1) /_{l(hd)} np) \\
& \text{komend} : \diamond_{mod} \square_{mod} (\diamond_{mod} \square_{mod} (s1 \backslash s1) / \diamond_{mod} \square_{mod} (s1 \backslash s1)) \\
& \text{uur} : np \\
& \text{wat} : \diamond_{whd} \square_{whd} (whq /_{l(body)} (\diamond_{se} \square_{se} np \backslash_{r(obj1)} s1)) \\
& \text{we} : np
\end{aligned}$$

The *np* hypothesis in the type assignment to the question word ‘wat’ (the gap hypothesis) gains access to structural reasoning by means of its *se* (for secondary edge) decoration. Relating the hypothesis to the direct object complement for the transitive infinitive ‘doen’ (‘do’) requires the mode-instantiated form of *Pl1* in (15). We present a derivation in Figure 6. Note the structural move in step 20, which establishes the required configuration for the logical introduction/elimination steps.

$$(15) \quad \diamond_{se} A \bullet_{r(obj1)} (B \bullet_{l(vc)} C) \vdash B \bullet_{l(vc)} (\diamond_{se} A \bullet_{r(obj1)} C) \text{ (Pl1)}$$

At the point of writing, the first set of syntactically annotated CGN data has been released, for some 50,000 words. We hope to report on the effect of different choices for the structural module and the parameters for the lexicon extraction algorithm in future work.

4 Conclusion

This paper represents an attempt to decompose the learning problem into a two-phase cycle, in line with the architecture of multimodal type-logical grammar. The first phase computes type-assignments from structured input, unifying type-assignment solutions in structurally similar contexts. The second phase enhances the lexicon with control features licencing structural reasoning. Lexical ambiguity is reduced by dynamically relating structural environments.

Needless to say, the ideas in this paper are in a stage of initial exploration. The only work we are aware of which also attributes a role to the unary connectives in learning is [7]. This author gives lambda terms as input to the learning algorithm, as a form of ‘semantic bootstrapping’. Although we agree that semantics cannot be ignored in learning, we think the lambda term input is too rich — it gives away too much of the learning puzzle. We mention some areas for future investigation that naturally suggest themselves. As we remarked in the Introduction, classical Gold-style categorical learning theory does not address issues of structural variation. Could one recast the two-stage learning cycle of this paper in terms of the identification-in-the-limit paradigm? A second area worth exploring is the relation between the deductive view on learning in this paper and results in the field of (human) language acquisition. The test case in

| | |
|---|--------------------------|
| 1. $\text{wat} : \diamond_{whd} \square_{whd} (whq /_{l(body)} (\diamond_{se} \square_{se} np \backslash_{r(obj1)} s1))$ | <i>Lex</i> |
| 2. $r_0 : \square_{whd} (whq /_{l(body)} (\diamond_{se} \square_{se} np \backslash_{r(obj1)} s1))$ | <i>Hyp</i> |
| 3. $\langle r_0 \rangle^{whd} : whq /_{l(body)} (\diamond_{se} \square_{se} np \backslash_{r(obj1)} s1)$ | $\square E$ (2) |
| 4. $\text{wat} : whq /_{l(body)} (\diamond_{se} \square_{se} np \backslash_{r(obj1)} s1)$ | $\diamond E$ (1, 2, 3) |
| 5. $s_0 : \diamond_{se} \square_{se} np$ | <i>Hyp</i> |
| 6. $\text{doen} : \diamond_{hd} \square_{hd} (np \backslash_{r(obj1)} inf)$ | <i>Lex</i> |
| 7. $\text{gaan} : \diamond_{hd} \square_{hd} (s1 /_{l(vc)} inf) /_{l(su)} np$ | <i>Lex</i> |
| 8. $s_1 : \square_{hd} (s1 /_{l(vc)} inf) /_{l(su)} np$ | <i>Hyp</i> |
| 9. $\langle s_1 \rangle^{hd} : (s1 /_{l(vc)} inf) /_{l(su)} np$ | $\square E$ (8) |
| 10. $\text{we} : np$ | <i>Lex</i> |
| 11. $\langle s_1 \rangle^{hd} \circ_{l(su)} \text{we} : s1 /_{l(vc)} inf$ | $/E$ (9, 10) |
| 12. $q_1 : \square_{se} np$ | <i>Hyp</i> |
| 13. $\langle q_1 \rangle^{se} : np$ | $\square E$ (12) |
| 14. $r_2 : \square_{hd} (np \backslash_{r(obj1)} inf)$ | <i>Hyp</i> |
| 15. $\langle r_2 \rangle^{hd} : np \backslash_{r(obj1)} inf$ | $\square E$ (14) |
| 16. $\langle q_1 \rangle^{se} \circ_{r(obj1)} \langle r_2 \rangle^{hd} : inf$ | $\backslash E$ (13, 15) |
| 17. $\langle s_1 \rangle^{hd} \circ_{l(su)} \text{we} \circ_{l(vc)} (\langle q_1 \rangle^{se} \circ_{r(obj1)} \langle r_2 \rangle^{hd}) : s1$ | $/E$ (11, 16) |
| 18. $(\text{gaan} \circ_{l(su)} \text{we}) \circ_{l(vc)} (\langle q_1 \rangle^{se} \circ_{r(obj1)} \langle r_2 \rangle^{hd}) : s1$ | $\diamond E$ (7, 8, 17) |
| 19. $(\text{gaan} \circ_{l(su)} \text{we}) \circ_{l(vc)} (\langle q_1 \rangle^{se} \circ_{r(obj1)} \text{doen}) : s1$ | $\diamond E$ (6, 14, 18) |
| 20. $\langle q_1 \rangle^{se} \circ_{r(obj1)} ((\text{gaan} \circ_{l(su)} \text{we}) \circ_{l(vc)} \text{doen}) : s1$ | <i>PI</i> (19) |
| 21. $s_0 \circ_{r(obj1)} ((\text{gaan} \circ_{l(su)} \text{we}) \circ_{l(vc)} \text{doen}) : s1$ | $\diamond E$ (5, 12, 20) |
| 22. $(\text{gaan} \circ_{l(su)} \text{we}) \circ_{l(vc)} \text{doen} : \diamond_{se} \square_{se} np \backslash_{r(obj1)} s1$ | $\backslash I$ (5, 21) |
| 23. $\text{wat} \circ_{l(body)} ((\text{gaan} \circ_{l(su)} \text{we}) \circ_{l(vc)} \text{doen}) : whq$ | $/E$ (4, 22) |

Proof term: $(\text{wat } \lambda x_1. ((\text{gaan we}) (\text{doen } x_1)))$

Fig. 6. Extraction in terms of (15).

§3 takes the machine learning perspective of grammar induction from an annotated corpus. But of course, we are interested in this test case because it opens a window on the effect of parameters that find their motivation in the cognitive setting of language acquisition. There is an affinity here between the proposals in this paper and the work on language acquisition and co-evolution in [4]. Briscoe’s approach is formulated in terms of a rule-based categorial framework. The connection with our logic-based approach needs further investigation.

References

1. Beghelli F. and T. Stowell, ‘Distributivity and Negation: The syntax of each and every’. In Szabolcsi (ed.) *Ways of Scope Taking*, Kluwer, 1997, pp. 72–107.
2. Bernardi, R., ‘Polarity items in resource logics. A comparison’. Proceedings Student Session, ESSLLI2000, Birmingham.
3. Bernardi, R. and R. Moot, ‘Generalized quantifiers in declarative and interrogative sentences’. Proceedings ICoS2.
4. Briscoe, E.J., ‘Grammatical Acquisition: Inductive Bias and Coevolution of Language and the Language Acquisition Device’, *Language* 76.2, 2000.

5. Buszkowski, W. & G. Penn, 'Categorial grammars determined from linguistic data by unification'. *Studia Logica* **49**, 431–454.
6. Emden, M.H. van, 'Conditional answers for polymorphic type inference'. In Kowalski and Bowen (eds.) *Proceedings 5th International Conference on Logic Programming*, 1988.
7. Fulop, S., *On the Logic and Learning of Language*. PhD Thesis, UCLA.
8. Heleen Hoekstra, Michael Moortgat, Ineke Schuurman, Ton van der Wouden, Syntactic Annotation for the Spoken Dutch Corpus Project (CGN). Proceedings CLIN2000.
9. Kanazawa, M., *Learnable classes of categorial grammars*. PhD Dissertation, Stanford, 1994.
10. Kraak, E., 'A deductive account of French object clitics'. In Hinrichs, Kathol & Nakazawa (eds.) *Complex Predicates in Nonderivational Syntax*. Syntax and Semantics, Vol 30. Academic Press.
11. Kurtonina, N. and M. Moortgat, 'Structural control'. In Blackburn and de Rijke (eds.) *Specifying Syntactic Structures*. CSLI Publications, 1997.
12. Moortgat, M., 'Categorial type logics'. Chapter 2 in Van Benthem and ter Meulen (eds.) *Handbook of Logic and Language*. Elsevier, 1997, pp. 93–177.
13. Moortgat, M., 'Meaningful patterns'. In Gerbrandy, Marx, de Rijke and Venema (eds.) *JFAK. Essays dedicated to Johan van Benthem on the occasion of his 50th birthday*, UAP, Amsterdam.
14. Moortgat, M., 'Constants of grammatical reasoning'. In Bouma, Hinrichs, Kruijff and Oehle (eds.) *Constraints and Resources in Natural Language Syntax and Semantics*, CSLI, Stanford, 1999.
15. Moortgat, M. and R. Moot, CGN to Grail. Extracting a type-logical lexicon from the CGN Annotation. Proceedings CLIN2000.
16. Sterling, L. and E. Shapiro, *The Art of Prolog*. MIT Press, Cambridge, MA.