Mathematical Aspects of Command Relations

Marcus Kracht
II. Mathematisches Institut
Arnimallee 3
D - 1000 Berlin 33
GERMANY
email: kracht@math.fu-berlin.de

Abstract

In GB, the importance of phrase-structure rules has dwindled in favour of nearness conditions. Today, nearness conditions play a major role in defining the correct linguistic representations. They are expressed in terms of special binary relations on trees called *command relations*. Yet, while the formal theory of phrase-structure grammars is quite advanced, no formal investigation into the properties of command relations has been done. We will try to close this gap. In particular, we will study the intrinsic properties of command relations as relations on trees as well as the possibility to reduce nearness conditions expressed by command relations to phrase-structure rules.

1 Introduction

1.1 Historic Origin

Early transformational grammar consisted of a rather complex generative component and an equally complex and equally imperspicuous transformational component. But since the aim always has been to understand languages rather than describing them, there has been a need for a reduction of these rule systems into preferably few and simple principles. The analysis of transformations as series of movements – an analysis made possible by the introduction of empty categories – was one step. This indeed drastically simplified the transformational component. A second step consisted in simplifying the generative component by reducing the rules in favour of well-formedness conditions, so-called *filters*. While this turned transformational grammar into a real theory now known as GB, the relationship of GB with other syntactic formalisms such as GPSG, LFG, categorial grammar etc. became less and less clear. This in addition to Noam Chomsky's often repeated scepticism with respect to formalizations has led to the common attitude that GB is simply gibberish, unformalizable or hopelessly untractable at best. However, since it is possible to evaluate predictions of theories of GB and have constructive debates over them these theories are if not formal then at least rigorous. Hence, it must be possible to formalize them. Formalizations of GB have been offered, e. g. in [Stabler, 1989] but in a manner that makes GB even less comprehensible. So if formalization means coming to terms with the precise predictions of an otherwise rigorously defined theory in order to understand it better, the project has failed if ever begun. More or less the same criticism applies to [Gazdar *et al.*, 1985]. Even if GPSG is rigorously defined the formalism as laid out in this book leads to no understanding of it's properties. Likewise, categorial grammar seems more to be adequate for a computer than a human. So, in principle GB is right to isolate the basic components of representations rather than jumping into a particular formalization. Yet the need to understand the formal properties of GB and the relationship between all these approaches remains and must be satisfied in order to achieve real progress. The theory of command relations forms part of an investigation that should ultimately lead to such an understanding. The present paper will sketch the theory of command relation and is a distilled version of [Kracht, 1993].

1.2 Relevance of Command Relations

The idea to study the formal properties of command relations is due to [Barker and Pullum, 1990]. There we find a definition of command relations as well as many illustrations of command relations from linguistic theory. In that paper the origins of the notions are also discussed. I guess it is fair to attribute to [Reinhart, 1981] the beginning of the study of domains. Moreover, [Koster, 1986] presents a impressive and thorough study of the role of domains in grammar. Yet all this work is either too specific or too vague to lead to a proper understanding of nearness conditions in grammar. In [Kracht, 1992] I took the case of [Barker and Pullum, 1990] further and proved some more results concerning these relations especially the structure of the heyting algebra of command relations. The latter proved to be of little significance in the light of the questions raised in § 1.1. Instead, it emerged that it is more fruitful to study the properties of command relations under intersection, union and relational composition. They form an algebraic structure called a *distributoid*. The structure of this distributoid can be determined. If the grammar is enriched with enough labels, this distributoid contains enough command relations to express all known nearness conditions. This being so, it becomes an immediate question whether the effect of a nearness condition expressed via command relations can be incorporated into the syntax. This is discussed at length in [Kracht, 1993]. The result is that indeed all such conditions are implementable, but this often requires a lot more basic features. The explosion of the size grammars when translating from GB to GPSG can be explained namely by the necessity to add auxiliary features that secure that the grammar obeys certain nearness restrictions. A typical example is the SLASH-feature which has been invented to guarantee a gap for a displaced filler. Having the main theorems in one's hand means that one is in principle dispensed from writing GPSG-type grammars in order to make available the rich theory of context-free grammars. Now it is possible to transfer this theory to grammars which consist both of a generative context-free component and a set of wellformedness conditions based on command relations. In particular, it is perfectly decidable whether two such grammars generate the same bracketed strings and hence effective comparison between two different theories of natural language – if given in that format – is possible.

2 Grammatical Relations on Trees

2.1 Definitions

A **tree** is an object $\mathbb{T} = \langle T, <, r \rangle$ with r the **root** and $<$ a tree ordering. We write $x \prec y$ if x is immediately dominated by y; in mathematical jargon y is said to **cover** x. A **leaf** is an element which does not cover; x is **interior** if it is neither a leaf nor the root. $int(\mathbb{T})$ is the set of interior nodes of $\mathbb{T}$. We put $\downarrow x = \{y : y \leq x\}$ and $\uparrow x = \{y : y \geq x\}$. $\downarrow x$ is called the **lower** and $\uparrow x$ the **upper cone** of x. If $R \subseteq T^2$ is a binary relation we write $R_x = \{y : xRy\}$ and call R_x the R-**domain** of x. A function $f : T \to T$ is called **monotone** if $x \leq y$ implies $f(x) \leq f(y)$, **increasing** if $x \leq f(x)$ for all x, and **strictly increasing** if $x < f(x)$ for all $x < r$.

Definition 1 *A binary relation $R \subseteq T^2$ is called a* ***command relation*** *(**CR** for short) iff there exists a function $f_R : T \to T$ such that (1), (2) and (3) hold; R is called* ***monotone*** *if in addition it satisfies (4) and* ***tight*** *if it satisfies (5) in addition to (1) - (3). f_R is called the* ***associated function*** *of R.*

(1) $R_x = \downarrow f_R(x)$

(2) $x < f_R(x)$ *for all* $x < r$

(3) $f_R(r) = r$

(4) $x \leq y$ *implies* $f_R(x) \leq f_R(y)$

(5) $x < f_R(y)$ *implies* $f_R(x) \leq f_R(y)$.

(1) expresses that $f_R(x)$ represents R; (2) and (3) express that f_R must be strictly increasing. If (4) holds, f_R is monotone. A tight relation is monotone; for if $x \leq y$ and $y < r$ then $y < f_R(y)$ and so $x < f_R(y)$; whence $f_R(x) \leq f_R(y)$ by (5). For some reason [Barker and Pullum, 1990] do not count monotonicity as a defining property of CRs even though there is no known command relation that fails to be monotone.

Given a set $P \subseteq T$ we can define a function g_P by

$$(\dagger) \quad g_P(x) = min\{y : y \in P, y > x\}$$

We put $min\emptyset = r$; thus $g_P(r) = r$. Let $x\mathsf{P}y$ iff $y \leq g_P(x)$. g_P is the associated function of P, a relation commonly referred to as P-**command**. We call P the **basic set** of g_P as well as P.

Here are some examples. With P the set of branching nodes P is C-command, with $P = T$ we have that P is IDC-command. When we take P to be the set of maximal projections we obtain that P is M-command, and, finally, with P the set of bounding nodes, e. g. $\{\mathsf{NP}, \mathsf{S}\}$, the relation P defined becomes identical to

Lasnik's KOMMAND. Lasnik's KOMMAND is identical to 1-node subjacency under the typical definition of subjacency.

Relations that are of the form P for some P are called **fair**.

Theorem 2 *R is fair iff it is tight. There are $2^{\sharp int(\mathbb{T})}$ distinct tight CRs on $\mathbb{T}$.*

Proof. ($\Rightarrow$) Assume $x < g_P(y) = min\{z \in P : z > y\}$. Then $g_P(x) = min\{z \in P : z > x\} \leq g_P(y)$ since $g_P(y) \in P$. ($\Leftarrow$) Put $P = \{f_R(x) : x \in T\}$. We have to show (†). By (5), however, $f_R(x) = min\{f_R(z) : f_R(z) > x\}$. For the second claim observe first that if P, Q differ only in exterior nodes then $\mathsf{P} = \mathsf{Q}$. If, however, $x \in P - Q$ is interior then $y \prec x$ for some y and $g_P(y) = x$ but $g_Q(y) > x$. ■

Tight relations have an important property; even when the structure of the tree is lost and we know only P we can recover g_P and $<$ to some extent. Notice namely that if $\mathsf{P}_x \neq T$ then $g_P(x)$ is the unique y such that $y \in \mathsf{P}_x$ but the P-domain of y is larger than the P-domain of x. We can then exactly say which elements are dominated by y: exactly the elements of the P-domain of x. By consequence, if we are given T, the root r and we know the IDC-command domains, $<$ can be recovered completely. This is of relevance to syntax because often the tree structures are not given directly but are recovered using domains.

2.2 Lattice Structure

Let f, g be increasing functions; then define

$$\begin{aligned}(f \sqcup g)(x) &= max\{f(x), g(x)\} \\ (f \sqcap g)(x) &= min\{f(x), g(x)\} \\ (f \circ g)(x) &= f(g(x))\end{aligned}$$

Since $f(x), g(x) \geq x$, that is, $f(x), g(x) \in \uparrow x$ and since $\uparrow x$ is linear, the maximum and minimum are always defined. Clearly, with f and g increasing, $f \sqcup g$, $f \sqcap g$ and $f \circ g$ are also increasing. Furthermore, if f and g are strictly increasing, the composite functions are strictly increasing as well.

Lemma 3 $f_{R\cup S} = f_R \sqcup f_S$. $f_{R\cap S} = f_R \sqcap f_S$.

Proof. $z \leq f_{R\cup S}(x)$ iff $x(R \cup S)z$ iff either xRz or xSz iff either $z \leq f_R(x)$ or $z \leq f_S(x)$ iff $z \leq max\{f_R(x), f_S(x)\}$. Analogously for intersection. ■

Theorem 4 *For any given tree $\mathbb{T}$ the command relations over $\mathbb{T}$ form a distributive lattice $\mathfrak{Cr}(\mathbb{T}) = \langle Cr(\mathbb{T}), \cap, \cup\rangle$ which contains the lattice $\mathfrak{Mon}(\mathbb{T})$ of monotone CRs as a sublattice.*

Proof. By the above lemma, the CRs over $\mathbb{T}$ are closed under intersection and union. Distributivity automatically follows since lattices isomorphic to lattices of sets with intersection and union as operations are always distributive. The second claim follows from the fact that if f_R, f_S are both monotone, so is $f_R \sqcup f_S$ and $f_R \sqcap f_S$. We prove one of these claims. Assume $x \leq y$. Then $f_R(x) \leq f_R(y)$ and $f_S(x) \leq f_S(y)$, hence $f_R(x) \leq max\{f_R(y), f_S(y)\}$ as well as $f_S(x) \leq max\{f_R(y), f_S(y)\}$. So $max\{f_R(x), f_S(x)\} \leq max\{f_R(y), f_S(y)\}$ and therefore $f_{R\cup S}(x) \leq f_{R\cup S}(y)$, by definition. ■

Proposition 5 *$g_{P\cup Q} = g_P \sqcap g_Q$. Hence tight relations over a tree are closed under intersection. They are generally not closed under closed union.*

Proof. Let $P, Q \subseteq T$ be two sets upon which the relations P and Q are based. Then the intersection of the relations, $\mathsf{P} \cap \mathsf{Q}$, is derived from the union $P \cup Q$ of the basic sets. Namely, $g_{P\cup Q}(x) = min\{y : y \in P \cup Q, y > x\} = min\{min\{y : y \in P, y > x\}, min\{y : y \in Q, y > x\}\} = min\{g_P(x), g_Q(x)\} = (g_P \sqcap g_Q)(x)$. To see that tight relations are not necessarily closed under union take the union of NP-command and S-command. If it were tight, the nodes of the form $g(x)$ for some x define the set on which this relation must be based. But this set is exactly the set of bounding nodes, which defines Lasnik's kommand. The latter, however, is the intersection, not the union of these relations. ■

The consequences of this theorem are the following. The tight relations form a sub-semilattice of the lattice of command relations; this semi-lattice is isomorphic to $\langle 2^{int(\mathbb{T})}, \cup\rangle$. Although the natural join of tight relations is not necessarily tight, it is possible to define a join in the semi-lattice. This operation is completely determined by the meet-semilattice structure, because this structure determines the partial order of the elements which in turn defines the join. In order to distinguish this join from the ordinary one we write it as $\mathsf{P} \bullet \mathsf{Q}$. The corresponding basic set from which this relation is generated is the set $P \cap Q$; this is the only choice, beacuse the semilattice $\langle 2^{int(\mathbb{T})}, \cup\rangle$ allows only one extension to a lattice, namely $\langle 2^{int(\mathbb{T})}, \cup, \cap\rangle$. The notation for associated functions is the same as for the relations. If g_P and g_Q are associated functions, then $g_P \bullet g_Q = g_{P\cap Q}$ denotes the associated function of the (tight) join.

2.3 Composition

For monotone relations there is more structure. Consider the definition of the relational product

$$R \circ S = \{\langle x, z\rangle : (\exists y)(xRySz)\}$$

Then $f_{R\circ S} = f_S \circ f_R$ (with converse ordering!). For a proof consider the largest z such that $x(R \circ S)z$. Then there exists a y such that $xRySz$. Now let

$\tilde{y}$ be the largest y such that xRy. Then not only $xR\tilde{y}$ but also $\tilde{y}Sz$, since S is monotone. By choice of $\tilde{y}$, $\tilde{y} = f_R(x)$. By choice of z, $z = f_S(\tilde{y})$, since $f_S(\tilde{y}) > z$ would contradict the maximality of z. In total, $z = (f_S \circ f_R)(x)$ and that had to be proved.

From the theory of binary relations it is known that $\circ$ distributes over $\cup$, that is, that we have $R \circ (S \cup T) = (R \circ S) \cup (R \circ T)$ as well as $(S \cup T) \circ R = (S \circ R) \cup (T \circ R)$. But in this special setting $\circ$ also distributes over $\cap$.

Proposition 6 *Let R, S, T be monotone CRs. Then $R \circ (S \cap T) = (R \circ S) \cap (R \circ T), (S \cap T) \circ R = (S \circ R) \cap (T \circ R)$.*

Proof. Let $x(R \circ (S \cap T))z$, that is, $xRy(S \cap T)z$, that is, $xRySz$ and $xRyTz$ for some y. Then, by definition, $x(R \circ S)z$ and $x(R \circ T)z$ and so $x((R \circ S) \cap (R \circ T))z$. Conversely, if the latter is true then $x(R \circ S)z$ and $x(R \circ T)z$ and so there are y_1, y_2 with xRy_1Sz and xRy_2Tz. With $y = max\{y_1, y_2\}$ we have $xRy(S \cap T)z$ since S, T are monotone. Thus $x(R \circ (S \cap T))z$. Now for the second claim. Assume $x((S \cap T) \circ R)z$, that is, $x(S \cap T)yRz$ for some y. Then xSy, xTy and yRz, which means $x(S \circ R)z$ and $x(T \circ R)z$ and so $x((S \circ R) \cap (T \circ R))z$. Conversely, if the latter holds then $x(S \circ R)z$ and $x(T \circ R)z$ and so there exist y_1, y_2 with xSy_1Rz and xTy_2Rz. Put $y = min\{y_1, y_2\}$. Then xSy, xTy, hence $x(S \cap T)y$. Moreover, yRz, from which $x((S \cap T) \circ R)z$. ■

Definition 7 *A* ***distributoid*** *is a structure $\mathfrak{D} = \langle D, \cap, \cup, \circ \rangle$ such that (1) $\langle D, \cap, \cup \rangle$ is a distributive lattice, (2) $\circ$ an associative operation and (3) $\circ$ distributes both over $\cap$ and $\cup$.*

Theorem 8 *The monotone CRs over a given tree form a distributoid denoted by $\mathfrak{Dis}(\mathbb{T})$.* ■

2.4 Normal Forms

The fact that distributoids have so many distributive laws means that for composite CRs there are quite simple normal forms. Namely, if $\mathfrak{R}$ is a CR composed from the CRs $R_1, \ldots, R_n$ by means of $\cap, \cup$ and $\circ$, then we can reproduce $\mathfrak{R}$ in the following simple form. Call $\mathfrak{C}$ a **chain** if it is composed from the R_i using only $\circ$. Then $\mathfrak{R}$ is identical to an intersection of unions of chains, and it is identical to a union of intersections of chains. Namely, by (3), both $\cap$ and $\cup$ can be moved outside the scope of $\circ$. Moreover, $\cap$ can be moved outside the scope of $\cup$ and $\cup$ can be moved outside the scope of $\cap$.

Theorem 9 (Normal Forms) *For every $\mathfrak{R} = \mathfrak{R}(R_1, \ldots, R_n)$ there exist chains $\mathfrak{C}_i^j = \mathfrak{C}_i^j(R_1, \ldots, R_n)$ and $\mathfrak{D}_i^j = \mathfrak{D}_i^j(R_1, \ldots, R_n)$ such that $\mathfrak{R} = \bigcup_i \mathfrak{I}_i$ with $\mathfrak{I}_i = \bigcap_j \mathfrak{C}_i^j$ and $\mathfrak{R} = \bigcap_j \mathfrak{U}_j$ with $\mathfrak{U}_j = \bigcap_i \mathfrak{D}_i^j$.* ■

From the linguistic point of view, tight relations play a key role because they are defined as a kind of topological closure of nodes with respect to the topology induced by the various categories. (However, this analogy is not perfect because the topological closure is an idempotent operation while the domain closure yields larger and larger sets, eventually being the whole tree.) It is therefore reasonable to assume that all kinds of linguistic CRs be defined using tight relations as primitives. Indeed, [Koster, 1986] argues for quite specific choices of fundamental relations, which will be discussed below. It is worthwile to ask how much can be defined from tight relations. This proves to yield quite unexpected answers. Namely, it turns out that union can be eliminated in presence of intersection and composition. We prove this first for the most simple case.

Lemma 10 *Let g_P, g_Q be the associated functions of tight relations. Then*
$g_P \sqcup g_Q = (g_P \circ g_Q) \sqcap (g_Q \circ g_P) \sqcap (g_P \bullet g_Q)$.

Proof. First of all, since $g_P, g_Q \leq g_P \circ g_Q, g_Q \circ g_P, g_P \bullet g_Q$ we have $g_P \sqcup g_Q \leq (g_P \circ g_Q) \sqcap (g_Q \circ g_P) \sqcap (g_P \bullet g_Q)$. The converse inequation needs to be established. There are three cases for a node x. (i) $g_P(x) = g_Q(x)$. Then $(g_P \sqcup g_Q)(x) = g_{P \cap Q}(x) = (g_P \bullet g_Q)(x)$, because the next P-node above x is identical to the next Q-node above x and so is identical to the next $P \cap Q$-node above x. (ii) $g_P(x) < g_Q(x)$. Then with $y = g_P(x)$ we also have $g_Q(y) = g_Q(x)$, by tightness. Hence $(g_P \sqcup g_Q)(x) = (g_Q \circ g_P)(x)$. (iii) $g_P(x) > g_Q(x)$. Then as in (ii) $(g_P \sqcup g_Q)(x) = (g_P \circ g_Q)(x)$. ■

The next case is the union of two chains of tight relations. Let $\mathfrak{g} = g_m \circ g_{m-1} \ldots \circ g_1$ and $\mathfrak{h} = h_n \circ h_{n-1} \ldots \circ h_1$ be two associated functions of such chains. Then define a **splice** of $\mathfrak{g}$ and $\mathfrak{h}$ to be any chain $\mathfrak{k} = k_\ell \circ k_{\ell-1} \ldots \circ k_1$ such that $\ell = m + n$ and $k_i = g_j$ or $k_i = h_j$ for some j and each g_i and h_j occurs exactly once and the order of the g_i as well as the order of the h_i in the splice is as in their original chain. So, the situation is comparable with shuffling two decks of cards into each other. A **weak splice** is obtained from a splice by replacing some number of $g_i \circ h_j$ and $h_j \circ g_i$ by $g_i \bullet h_j$, least tight relation containing both g_i and h_j. In a weak splice, the shuffling is not perfect in the sense that some pairs of cards may be glued to each other. If $\mathfrak{g} = g_2 \circ g_1$ and $\mathfrak{h} = h_2 \circ h_1$ then the following are all splices of $\mathfrak{g}$ and $\mathfrak{h}$: $g_2 \circ g_1 \circ h_2 \circ h_1$, $g_2 \circ h_2 \circ g_1 \circ h_1$, $g_2 \circ h_2 \circ h_1 \circ g_1$. The following are weak splices (in addition to the splices, which are also weak splices): $g_2 \circ g_1 \bullet h_2 \circ h_1$, $g_2 \bullet h_2 \circ g_1 \bullet h_1$. A non-splice is $g_1 \circ h_2 \circ g_2 \circ h_1$, and $g_2 \bullet g_1 \circ h_2 \circ h_1$ is not a weak splice.

Lemma 11 *Let $\mathfrak{g}, \mathfrak{h}$ be two chains of tight relations* *(or their associated functions). Let* $\mathbf{wk}(\mathfrak{g}, \mathfrak{h})$ *be the set of weak splices of* $\mathfrak{g}$ *and* $\mathfrak{h}$. *Then*

$$\mathfrak{g} \sqcup \mathfrak{h} = \bigsqcap \langle \mathfrak{s} : \mathfrak{s} \in \mathbf{wk}(\mathfrak{g}, \mathfrak{h}) \rangle$$

.

Proof. As before, it is not difficult to show that $\mathfrak{g} \sqcup \mathfrak{h} \leq \bigsqcap \langle \mathfrak{s} : \mathfrak{s} \in \mathbf{wk}(\mathfrak{g}, \mathfrak{h}) \rangle$ because $\mathfrak{g}, \mathfrak{h} \leq \mathfrak{s}$ for each weak splice. So it is enough to show that the left hand side is equal to one of the weak splices in any tree for any given node. Consider therefore a tree $\mathbb{T}$ and a node $x \in T$. We define a weak splice $\mathfrak{s}$ such that $\mathfrak{s}(x) = max\{\mathfrak{g}(x), \mathfrak{h}(x)\}$. To this end we define the following nodes. $x_0 = x, y_0 = x$, $x_1 = g_1(x_0), h_1(y_0), \ldots, x_{i+1} = g_{i+1}(x_i), y_{i+1} = h_{i+1}(y_i), \ldots$. The x_i and the y_i each form an increasing sequence. We can also assume that both sequences are strictly increasing because otherwise there would be an i such that $x_i = r$ or $y_i = r$. Then $(\mathfrak{g} \sqcup \mathfrak{h})(x) = r$ and so for any weak splice $\mathfrak{s}(x) = r$ as well. So, all the x_i can be assumed distinct and all the y_i as well. Now we define z_i as follows. $z_0 = x$, $z_1 = min\{x_1, \ldots, x_m, y_1, \ldots, y_n\}, \ldots, z_{i+1} = min(\{x_1, \ldots, x_m, y_1, \ldots, y_n\} - \{z_1, \ldots, z_i\})$. Thus, the sequence of the z_i is obtained by fusing the two sequences along the order given by the upper segment $\uparrow x$. Finally, the weak splice can be defined. We begin with s_1. If $x_1 = y_1$, $s_1 = g_1 \bullet h_1$, if $x_1 < y_1$, $s_1 = g_1$ and if $x_1 > y_1$ then $s_1 = h_1$. Generally, for z_{i+1} there are three cases. First, $z_{i+1} = x_j = y_k$ for some j, k. Then $s_{i+1} = g_j \bullet h_k$. Else $z_{i+1} = x_j$ for some j, but $z_{i+1} \neq y_k$ for all k. Then $s_{i+1} = g_j$. Or else $z_{i+1} = y_k$ for some k but $z_{i+1} \neq x_j$ for all j; then $s_{i+1} = h_k$. It is straightforward to show that $\mathfrak{s}$ as just defined is a weak splice, that $z_{i+1} = s_i(z_i)$ and hence that $\mathfrak{s}(x) = max\{\mathfrak{g}(x), \mathfrak{h}(x)\}$. ■

The tight relations generate a subdistributoid $\mathfrak{Tgr}(\mathbb{T})$ in $\mathfrak{Dis}(\mathbb{T})$ members of which we call **tight generable**.

Theorem 12 *Each tight generable command relation is an intersection of chains of tight relations.* ■

3 Introducing Boolean Labels

3.1 Boolean Grammars

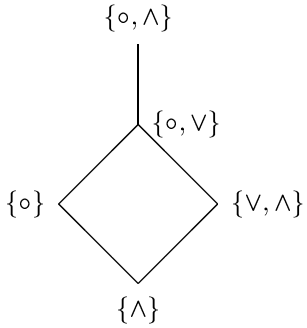
We are now providing means to define CRs uniformly over trees. The trees are assumed to be *labelled.* For mathematical convenience the labels are drawn from a boolean algebra $\mathfrak{L} = \langle L, 0, 1, -, \cap, \cup \rangle$. A **labelling** is a function $\ell : T \to L$. ℓ is called **full** if $\ell(x)$ is an atom of $\mathfrak{L}$ or 0 for every x. If either $\ell(x) = \mathsf{a} = 0$ or $0 < \ell(x) \leq \mathsf{a}$ we say that x is of **category** a. Labelled trees are generated by *boolean grammars.* Since syntax is abstracting away from actual words to word classes named each by its own syntactical label we may forget to discriminate between the terminal labels with impunity. This allows to give all of them the unique value 0, which is now the only terminal, the non-terminals being all elements of $L - \{0\}$. A **boolean grammar** is defined as a triple $\mathbb{G} = \langle \Sigma, \mathfrak{L}, R \rangle$ where R is a finite subset of $(L - \{0\}) \times L^+$ and $\Sigma \in L$. $\mathbb{G}$ **generates** $\mathbb{T} = \langle \mathbb{T}, \ell \rangle$ – in symbols $\mathbb{G} \gg \mathbb{T}$ –, if (r) r is of category Σ, (t) x is of category 0 iff x is a leaf and (nt) if x immediately dominates $y_1, \ldots, y_n$ then with an appropriate order of the indices there is a rule $\mathsf{a} \to \mathsf{b}_1, \ldots, \mathsf{b}_n$ in R such that x is of category a and y_i is of category b_i for all i. Boolean grammars are a mild step away from context free grammars. Namely, if $\mathsf{a} \to \mathsf{b}_1 \ldots \mathsf{b}_n$ is a boolean rule, we may consider it as an abbreviation of the set of rules $\mathsf{a}^* \to \mathsf{b}_1^* \ldots \mathsf{b}_n^*$ where a^* is an atom of $\mathfrak{L}$ below a and b_i^* is an atom of $\mathfrak{L}$ below b_i for each i. Likewise, the start symbol abbreviates a set of start symbols Σ^*, which by familiar tricks can be replaced by a single one denoted by $\aleph$, which is added artificially. In this way we can translate $\mathbb{G}$ into a cfg $\mathbb{G}^*$ over the set of atoms of $\mathfrak{L}$ plus 0 and the new start symbol $\aleph$, which generates the same fully labelled trees – ignoring the deviant start symbol. It is known that there is an effective procedure to eliminate from a cfg labels that never occur in a finite tree generated by the grammar (see e. g. [Harrison, 1978]). This procedure can easily be adapted to boolean grammars. A boolean grammar without such superfluous symbols is called **normal**.

3.2 Domain Specification

Each boolean label a defines the relation of a-**command** on a fully labelled tree via the set of nodes of category a. This is the classical scenario; the label S defines S-command, the label $\mathsf{NP} \cup \mathsf{CP}$ defines Lasnik's Kommand. And so forth. We denote the particular relation induced on $\langle \mathbb{T}, \ell \rangle$ by $\delta_{\mathbb{T}}(\mathsf{a})$. From this basic set of tight CRs we allow to define more complex CRs using the operations. To do this we first define a constructor language that contains a constant a for each $\mathsf{a} \in L$ and the binary symbols $\wedge, \vee$ and $\circ$. (Although we also use $\bullet$, we will treat it as an abbreviation; also, this operation is defined only for tight relations.) Since we assume the equations of distributoids, the symbols a generate a distributoid with $\wedge, \vee, \circ$, namely the so-called **free distributoid**. The map $\delta_{\mathbb{T}}$ can be extended to a homomorphism from this distributoid into $\mathfrak{Dis}(\mathbb{T})$. Simply put

$$\begin{aligned} \delta_{\mathbb{T}}(\mathfrak{d} \wedge \mathfrak{e}) &= \delta_{\mathbb{T}}(\mathfrak{d}) \cap \delta_{\mathbb{T}}(\mathfrak{e}) \\ \delta_{\mathbb{T}}(\mathfrak{d} \vee \mathfrak{e}) &= \delta_{\mathbb{T}}(\mathfrak{d}) \cup \delta_{\mathbb{T}}(\mathfrak{e}) \\ \delta_{\mathbb{T}}(\mathfrak{d} \circ \mathfrak{e}) &= \delta_{\mathbb{T}}(\mathfrak{d}) \circ \delta_{\mathbb{T}}(\mathfrak{e}) \end{aligned}$$

By definition, the image of $\mathfrak{d}$ under $\delta_{\mathbb{T}}$ is tight generable. Hence $\delta_{\mathbb{T}}$ maps all nearness terms into tight generable relations. With $\mathsf{NP}\cup\mathsf{CP}$ being 1-node subjaceny (for English) we find that $(\mathsf{NP}\cup\mathsf{CP})\circ(\mathsf{NP}\cup\mathsf{CP})$ is 2-node subjacency. Using a more complex definition it is possible to define 0- and 1-subjacency in the barriers system on the condition that there are no double segments of a category. If we consider the power of subsystems of this language, e. g. relations definable using only $\wedge$ etc. the following picture emerges.



This follows mainly from Theorem 12 because the map δ is by definition into the distributoid $\mathfrak{Tgr}(\mathbb{T})$ of tight generated CRs. Moreover, $\wedge$ alone does not create new CRs, because of Prop. 5. Each of the inclusions is proper as is not hard to see. So $\vee$ does not add definitional strength in presence of $\circ$ and $\wedge$; although things may be more perspicuously phrased using $\vee$ it is in principle eliminable. By requiring CRs to be intersections of chains we would therefore not express a real restriction at all.

3.3 The Equational Theory

Given a boolean grammar $\mathbb{G}$, a tree $\mathbb{T}$ and two domains $\mathfrak{d}, \mathfrak{e}$ constructed from the labels of $\mathbb{G}$ we write $\mathbb{T} \models \mathfrak{d} = \mathfrak{e}$ if $\delta_{\mathbb{T}}(\mathfrak{d}) = \delta_{\mathbb{T}}(\mathfrak{e})$. The set

$$Eq(\mathbb{G}) = \{\mathfrak{d} = \mathfrak{e} : (\forall \mathbb{T})(\mathbb{G} \gg \mathbb{T} \Rightarrow \mathfrak{T} \models \mathfrak{d} = \mathfrak{e}\}$$

is called the **equational theory** of $\mathbb{G}$. To determine the equational theory of a grammar we proceed through a series of reductions. $\mathbb{G}$ admits the same finite trees as does is normal reduct $\mathbb{G}^n$. So, we might as well assume from start that $\mathbb{G}$ is normal. Second, domains are insensitive to the branching nature of rules. We can replace with impunity any rule $\rho = \mathsf{a} \to \mathsf{b}_1 \ldots \mathsf{b}_n$ by the set of rules $\rho^u = \{\mathsf{a} \to \mathsf{b}_i : i \leq n\}$. We can do this for all rules of the grammar. The grammar $\mathbb{G}^u = \langle \Sigma, \mathfrak{L}, R^u \rangle$ where $R^u = \{\rho^u : \rho \in R\}$ is called the **unary reduct of** $\mathbb{G}$. It has the same equational theory as $\mathbb{G}$ since the trees it generates are exactly the branches of tree generated by $\mathbb{G}$. Next we reduce the unary grammar to an ordinary cfg $\mathbb{G}^{u*}$ in the way described above with an artificially added start symbol $\aleph$. This grammar is completely isomorphic to a transition network alias directed graph with single source $\aleph$ and single sink 0. This network is realized over the set of atoms of $\mathfrak{L}$ plus $\aleph$ and 0. There are only finitely many such networks over given $\mathfrak{L}$ – to be exact, at most $2^{(n+1)^2}$ (!) where n is the number of atoms of $\mathfrak{L}$. Finally, it does not harm if we add some transitions from $\aleph$ and transitions to 0. First, if we do so, the equational theory must be included in the theory of $\mathbb{G}$ since we allow more structures to be generated. But it cannot be really smaller; we are anyway interested in *all* substructures $\uparrow x$ for nodes x, so adding transitions to 0 is of no effect. Moreover, adding transitions from $\aleph$ can only give more equations because the generated trees of this new transition system are branches where some lower and some upper cone is cut off. Thus, rather than taking the grammar $\mathbb{G}^{u*}$ we can take a grammar with some more rules, namely all transitions $\aleph \to \mathsf{A}$, $\mathsf{A} \to 0$ for an atom A plus $\aleph \to 0$. In all, the role of source and sink are completely emptied, and we might as well forget about them. What we keep to distinguish grammars is the directed graph on the atoms of $\mathfrak{L}$ induced by the unary reduct of $\mathbb{G}$. Let us denote this graph by $\mathfrak{Gph}(\mathbb{G})$. We have seen that if two grammars $\mathbb{G}$, $\mathbb{H}$ have the same graph, their equational theory is the same. The converse also holds. To see this, take an atom A and let $\mathsf{A}^{\ominus}_{\mathbb{G}}$ be the disjunction of all atoms B such that $\mathsf{B} \to \mathsf{A}$ is a transition in the graph (or, equivalently, in the unary reduct) of $\mathbb{G}$. Then $\mathsf{A} \circ \mathsf{A}^{\ominus}_{\mathbb{G}} = \mathsf{A} \circ \bot \in Eq(\mathbb{G})$. However, if $\mathsf{C} \neq \mathsf{A}^{\ominus}_{\mathbb{G}}$ then $\mathsf{A} \circ \mathsf{C} = \mathsf{A} \circ \bot \notin Eq(\mathbb{G})$. If $\mathbb{G}$ and $\mathbb{H}$ have different graphs, then there must be an A such that $\mathsf{A}^{\ominus}_{\mathbb{G}} \neq \mathsf{A}^{\ominus}_{\mathbb{H}}$. Consequently, $\mathsf{A} \circ \mathsf{A}^{\ominus}_{\mathbb{G}} = \mathsf{A} \circ \bot \notin Eq(\mathbb{H})$ and $\mathsf{A} \circ \mathsf{A}^{\ominus}_{\mathbb{H}} = \mathsf{A} \circ \bot \notin Eq(\mathbb{G})$.

Theorem 13 $Eq(\mathbb{G}) = Eq(\mathbb{H})$ *iff* $\mathfrak{Gph}(\mathbb{G}) = \mathfrak{Gph}(\mathbb{H})$. *Hence it is decidable for any pair* $\mathbb{G}$, $\mathbb{H}$ *of boolean grammars over the same labels whether or not* $Eq(\mathbb{G}) = Eq(\mathbb{H})$. ∎

The question is now how we can decide whether a given domain equation holds in a grammar. We know by the reductions that we can assume this grammar to be unary. Now take an equation $\mathfrak{d} = \mathfrak{e}$. Suppose this equation is not in the theory and we have a countermodel. This countermodel is a non-branching labelled tree $\mathbb{T}$ a node x such that $\delta_{\mathbb{T}}(\mathfrak{d})_x \neq \delta_{\mathbb{T}}(\mathfrak{e})_x$. Let $Sf(\mathfrak{d})$ denote the set of subformulas of $\mathfrak{d}$ and $Sf(\mathfrak{e})$ the set of subformulas of $\mathfrak{e}$. Put $S = \{f_{\mathfrak{g}}(x) : \mathfrak{g} \in Sf(\mathfrak{d}) \cup Sf(\mathfrak{e})\}$. S is certainly finite and its cardinality is bounded by the sum of the cardinalities of $Sf(\mathfrak{d})$ and $Sf(\mathfrak{e})$. Now let y, z be two points from S such that $y < z$ and for all u such that $y < u < z$ $u \notin S$. Let u_1 and u_2 be two points such that $y < u_1 < u_2 < z$ and such that u_1 and u_2 have the same label. We construct a new labelled tree $\mathbb{U}$ by dropping all nodes from u_1 up until the node immediately below u_2. The following holds of the new model. (i) It is a tree generated by $\mathbb{G}$ and (ii) $\delta_{\mathbb{U}}(\mathfrak{d})_x \neq \delta_{\mathbb{U}}(\mathfrak{e})_x$. Namely, if $w \prec u_1$ then $\ell(u_1) \rightarrow \ell(w)$ is a transition of $\mathbb{G}$, hence $\ell(u_2) \rightarrow \ell(w)$ is a transition of $\mathbb{G}$ as well because $\ell(u_1) = \ell(u_2)$; and so (i) is proved. For (ii) it is enough to prove that for all $\mathfrak{g} \in Sf(\mathfrak{d}) \cup Sf(\mathfrak{e})$ the value $f'_{\mathfrak{g}}(x)$ in the new model is the same as the value $f_{\mathfrak{g}}(x)$ in the old model. (Identification is possible, because these points have not been dropped.) This is done by reduction on the structure of $\mathfrak{g}$. Suppose then that $\mathfrak{g} = \mathfrak{h} \wedge \mathfrak{k}$ and $f'_{\mathfrak{h}}(x) = f_{\mathfrak{h}}(x)$ as well as $f'_{\mathfrak{k}}(x) = f_{\mathfrak{k}}(x)$; then $f'_{\mathfrak{g}}(x) = min\{f'_{\mathfrak{h}}(x), f'_{\mathfrak{k}}(x)\} = min\{f_{\mathfrak{h}}(x), f_{\mathfrak{k}}(x)\} = f_{\mathfrak{g}}(x)$. And similarly for $\mathfrak{g} = \mathfrak{h} \vee \mathfrak{k}$. By the normal form theorem we can assume $\mathfrak{g}$ to be a disjunction of conjunctions of chains, so by the previous reductions it remains to treat the case where $\mathfrak{g}$ is a chain. Hence let $\mathfrak{g} = \mathsf{a} \circ \mathfrak{k}$. We assume $f'_{\mathfrak{k}}(x) = f_{\mathfrak{k}}(x) =: y$. Let $z := f_{\mathfrak{g}}(x)$. Then if $y < r$, $y < z$ and else $y = z$. By construction, z is the first node above y to be of category a and $z \in S$, by which z is not dropped. In the reduced model, z is again the first node of category a above y, and so $f'_{\mathfrak{g}}(x) = f'_{\mathsf{a}}(y) = z$, which had to be shown.

Assume now that we have a tree of minimal size generated by $\mathbb{G}$ in which $\mathfrak{d} = \mathfrak{e}$ does not hold. Then if $y, z \in S$ such that $y < z$ but for no $u \in S$ $y < u < z$, then in between y and z all nodes have different labels. Thus, in between y and z sit no more points than there atoms of $\mathfrak{L}$. Let this number be n; then our model has size $\leq n \cdot S$. Now if we want to decide whether or not $\mathfrak{d} = \mathfrak{e}$ is in $Eq(\mathbb{G})$, all we have to do is to first generate all possible branches of trees of length at most $n \times (\sharp Sf(\mathfrak{d}) + \sharp Sf(\mathfrak{e})) + 2$ and check the equation on them. If it holds everywhere, then indeed $\mathfrak{d} = \mathfrak{e}$ is valid in all trees because otherwise we would have found a countermodel of at most this size.

Theorem 14 *It is decidable whether or not* $\mathfrak{d} = \mathfrak{e} \in Eq(\mathbb{G})$. ■

These theorems tell us that there is nothing dangerous in using domains in grammar as concerns the question whether the predictions made by this theory can effectively be computed; that is, as long as one sticks to the given format of domain constructions, it is decidable whether or not a given grammatical theory makes a certain prediction about domains.

4 Implementations

4.1 Problems of Implementations

The aim set by our theory is to reduce all possible nearness conditions of grammar to some restrictions involving command relations. Thus we treat not only binding theory or case theory but also restrictions on movement. Even though [Barker and Pullum, 1990] did not think of movement and subjacency as providing cases for command relations, the fact that nearness conditions are involved clearly indicates that the theory should have something to say about them. However, there are various obstacles to a direct implementation.

The theory of command relations is not directly compatible with standard nearness relations in GB. A command relation as defined here depends in its size only of the isomorphism type of the linear structure above the node x. So, typical definitions such as those involving the notions of being *governed*, being *bound*, having an *accessible subject* fail to be of the kind proposed here because they involve a node that stands in relation of C-command rather than domination. Nevertheless, if GB would be spelt out fully into a boolean grammar, far more labels have to be used than appear usually on trees displayed in GB books. The reason is that while context-free grammars by definition allow no context to rule the structure of a local tree, in GB the whole tree is implicitly treated as a context. But if it is true that the context for a node reduces to nodes that are C-commanding, it is enough to add for certain primitive labels X another label $\oslash\mathsf{X}$ which translates as *one of my daughters is X*. Here, $\oslash\mathsf{X}$ is not necessarily understood to be a new label but a specific label that guarantees one of the daughters to be of category X. However, 'modals' such as $\oslash$ are somewhat whimsical creatures. Sometimes, $\oslash\mathsf{X}$ is an already existing category, for example $\oslash\mathsf{IP}$ can (with the exception of exceptional case marking constructions) be equated with C'. On other occasions, however, we need to incorporate them into our grammar; prominent modals are $\mathsf{SLASH} : \mathsf{X}$, which has the meaning *somewhere below me is a gap of category X* and $\mathsf{AGR} : \mathsf{X}$ which says *this sentence has a subject of category X*. If a context-free rendering of phrase structure is done properly (as for example in [Gazdar *et al.*, 1985]) a single entry such as V must be split into a vast number of different symbols so we can reasonably assume that our grammar is rich enough to have all the $\oslash\mathsf{X}$ for the X we need; otherwise they must be added artificially. In that case many of the standard nearness relations can be directly encoded using command relations.

A second problem concerns the role of adjunction in the definition of subjacency. If the domain of movement for a node (that is, the domain within which the antecedent has to be found) is tight, then no iteration of movement leads to escaping the original domain. So, the domain for movement must be large. But it cannot be too large either because we loose the necessity of free escape hatches (spec of comp, for example). The typical definitions of subjacency lead to domains that are just about right in size. However, the dilemma must be solved that after moving to spec of comp, an element can move higher than it could from its original position. Different solutions have been offered. The most simple is standard 2-node subjacency which is KOMMAND $\circ$ KOMMAND. This domain indeed allows this type of cyclical movement; cyclic movement from spec of comp to spec of comp is possible – but only to the next spec of comp. However, due to it's shortcomings, this notion has been criticised; moreover, it has been felt that 1-node subjacacency should be superior, largely because of the slogan 'grammar does not count'. Yet, tight domains don't do the jobs and so tricks have been invented. [Chomsky, 1986] formulated rather small domains but included a mechanism to escape them by creating 'grey zones' in which elements are neither properly dominated by a node nor in fact properly non-dominated. This idea has caught on (for example in [Sternefeld, 1991]) but has to be treated cautiously as even the simplest notions such as *category*, *node* etc. receive new interpretations because nodes are not necessarily identical with occurrences of categories as before. A reduction to standard notions should certainly be possible and desired – without necessarily banning adjunction.

4.2 The Koster Matrix

As [Koster, 1986] observed, grammatical relations are typically relations between a dependent element δ and an antecedent α:

$$\cdots \quad \underbrace{\alpha \quad \cdots \quad \delta}_{R} \quad \cdots$$

[Koster, 1986] notes four conditions on such configurations.

a. obligatoriness

b. uniqueness of the antecedent

c. c-command of the antecedent

d. locality

If these conditions are met then this relation has the effect

share property

This has to be understood as follows. (a.) and (b.) express nothing but that δ needs one and only one antecedent. This antecedent, α, must C-command δ. Finally, (d.) states that α must be found in some local domain of δ. Of course, this domain is language specific as well as specific to the syntactic construction, i. e. the category of δ and α. Likewise, the property to be shared depends on the category of α and δ.

The locality restriction expresses that α is found within the R-domain of δ. This relation R is in the unmarked case defined as follows.

Definition 15 *α is **locally accessible**^{1} to δ if $\alpha \leq \beta$, where β is the least maximal projection containing δ and a governor of δ.*

[Koster, 1986] assumes that greater domains are formed by licensed extensions. These extensions are marked constructions; while all languages agree on the local accessibility^1 as the minimal domain within which antecedents must be found, larger domains may also exist but their size is language and construction specific. Nevertheless, the variation is limited. There are only three basic types, namely *locally accessible*^{i} for $i = 1, 2, 3$.

Definition 16 *α is **locally accessible**^{2} to δ if $\alpha \leq \beta$, where β is the least maximal projection containing δ, a governor for δ and some opacity element ω. α is **locally accessible**^{3} to δ if there is a sequence β_i, $1 \leq n$, such that β_1 is locally accessible^2 from δ and β_{i+1} is locally accessible^2 from β_i.*

The opacity elements are drawn from a rather limited list. Such elements are *tense*, *mood* etc. A well-known example are Icelandic reflexives whose domain is the smallest indicative sentence.

4.3 The Command Relations of Koster's Matrix

The local accessibility relations certainly are command relations in our sense. The real problem is whether they are definable using primitive labels of the grammar. In particular the recursiveness of the third accessibility makes it unlikely that we can find a definition in terms of $\wedge, \vee, \circ$. Yet, if it were really an arbitrary iteration of the second accessibility relation it would be completely trivial, because any iteration of a command relation over a tree is the total relation over the tree. Hence, there must be something non-trivial about this domain; indeed, the iteration is stopped if the outer β is ungoverned. This is the key to a non-iterative definition of the third accessibility relation.

Let us assume for simplicity that there is a single type of governors denoted by GOV and that there is a single type of opacity element denoted by OPY. The first hurdle is the clarification of *government*. Normally, government requires a governing element, i. e. an element of category GOV that is close in some sense. How close, is not clarified in [Koster, 1986]. Clearly, by penalty of providing circular definitions, closeness cannot be accessibility^1; really, it must be an even smaller domain. Let us assume for simplicity that it is sisterhood. If then we introduce the modal $\natural\mathsf{X}$ to denote *one of my sisters is of category X*, being governed is equal to being of category $\natural\mathsf{GOV}$. Likewise we will assume that the opacity element must be in C-command relation to δ. We are now ready to define the three accessibility relations, which we denote by LA^1, LA^2 and LA^3.

$$\begin{aligned}
\mathsf{LA}^1 \;=\;& \oslash\mathsf{GOV} \bullet \mathsf{BAR{:}2}\\
&\wedge \oslash\mathsf{GOV} \circ \mathsf{BAR{:}2}\\
\mathsf{LA}^2 \;=\;& \oslash\mathsf{GOV} \bullet \oslash\mathsf{OPY} \bullet \mathsf{BAR{:}2}\\
&\wedge \oslash\mathsf{GOV} \bullet \oslash\mathsf{OPY} \circ \mathsf{BAR{:}2}\\
&\wedge \oslash\mathsf{GOV} \circ \oslash\mathsf{OPY} \bullet \mathsf{BAR{:}2}\\
&\wedge \oslash\mathsf{GOV} \circ \oslash\mathsf{OPY} \circ \mathsf{BAR{:}2}\\
\mathsf{LA}^3 \;=\;& \oslash\mathsf{GOV} \bullet \oslash\mathsf{OPY} \bullet \mathsf{BAR{:}2} \bullet \text{-}\natural\mathsf{GOV}\\
&\wedge \oslash\mathsf{GOV} \bullet \oslash\mathsf{OPY} \circ \mathsf{BAR{:}2} \bullet \text{-}\natural\mathsf{GOV}\\
&\wedge \oslash\mathsf{GOV} \circ \oslash\mathsf{OPY} \bullet \mathsf{BAR{:}2} \bullet \text{-}\natural\mathsf{GOV}\\
&\wedge \oslash\mathsf{GOV} \circ \oslash\mathsf{OPY} \circ \mathsf{BAR{:}2} \bullet \text{-}\natural\mathsf{GOV}
\end{aligned}$$

(Observe that $\bullet$ binds stronger than $\circ$.) For a proof consider a point x of a labelled tree $\mathbb{T}$. Let g denote the smallest node dominating both x and its governor and let m be the smallest maximal projection of g. Then $x < g \leq m$. So two cases arise, namely $g = m$ and $g < m$. In each cases LA^1 picks the right node. Likewise, if o denotes the smallest element containing x and a opacity element that C-commands x, then $x < o$. Three cases are conceivable, $o < g$, $o = g$ and $o > g$. However, if government can take place only under sisterhood, $o < g$ cannot occur. So $x < g \leq o \leq m$. For each of the four cases LA^2 picks the right node. Finally, for LA^3 there is an extra condition on m that it be ungoverned.

Notice that our translation is faithful to Koster's definitions only if the domains defined in [Koster, 1986] are monotone. This is by no means trivial. Namely, it is conceivable that a node has an ungoverned element y locally accessible^2, while the highest locally accessible^2 node, z, is governed. In that case (ignoring the opacity element for a moment) the domain of local accessibility^3 of y is z while the domain of x is strictly larger. We find no answer to this puzzle in the book because the domains are defined only for governed elements. But it seems certain that the monotone definition given here is the intended one.

It should be stressed that GOV and OPY are not specific labels but variables. Their value may change from situation to situation. Consequently, the local accessibility relations are parametrized with respect to the choice of particular governors and particular opacity elements. As an example, recall the Icelandic case again, where certain anaphors whose domain of accessibility^2 (typically the clause) can be extended in case the opacity element is *subjunctive*. Following our reduction, the domain of local accessibility^3 is defined by the first maximal projection that is not subjunctive, hence indicative. We take a primitive label IND to stand for *is indicative*. So, for Icelandic we have the following special domain

$$\begin{aligned}
\mathsf{LA}^3 \;=\;& \oslash\mathsf{GOV} \bullet \oslash\mathsf{IND} \bullet \mathsf{BAR{:}2} \bullet \text{-}\natural\mathsf{GOV}\\
&\wedge \oslash\mathsf{GOV} \bullet \oslash\mathsf{IND} \circ \mathsf{BAR{:}2} \bullet \text{-}\natural\mathsf{GOV}\\
&\wedge \oslash\mathsf{GOV} \circ \oslash\mathsf{IND} \bullet \mathsf{BAR{:}2} \bullet \text{-}\natural\mathsf{GOV}\\
&\wedge \oslash\mathsf{GOV} \circ \oslash\mathsf{IND} \circ \mathsf{BAR{:}2} \bullet \text{-}\natural\mathsf{GOV}
\end{aligned}$$

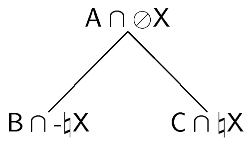
We notice in passing that recent results have put this analysis into doubt (see [Koster and Reuland, 1991]) but this is a problem of Koster's original definitions, not of this translation. What is a problem, however, is the standard opacity factor of an *accessible subject*. While *subject* (or even *SUBJECT*) can be easily handled with a boolean label, the accessibility condition presents real difficulties. First of all it involves indexing and indexes potentially destroy the finiteness of the labelling system; secondly, it is not clear how the accessibility condition (namely, the reqirement that the i/i-Filter is respected after conindexation) can be handled at all in this calculus. This issue is too complex to be tackled here, so we leave it for another occasion.

4.4 Translating Koster's Matrix into Rules

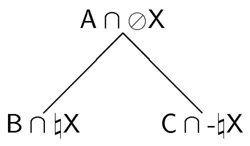
In a final step we show how the nearness conditions of the Koster Matrix can be rewritten into rules of a context-free grammar. To be more precise, we show how they can be implemented into any given boolean cfg. The booleanness, of course, is not essential but is here for convenience. We noticed earlier that the domains in GB really are for the purpose to introduce some limited forms of context-sensitivity. If two nodes relate via some dependency relation R then Koster assumes that a certain property is shared. But context-free grammars do in principle not allow such a sharing except between mother and daughters and between sister nodes. Nevertheless, as we do not require *all* properties to be shared but only *some* it is possible to enrich the grammar in such a way that nodes receive relevant information about parts of the structure that normally cannot be accessed. We will show how.

First, we will assume that *share property* is to be understood as a dependency in the labellings between two elements. We simplify this by assuming that there are special features PRP_i, $i \leq n$, of unspecified nature whose instantiation at the two nodes, δ and α, is somehow correlated. Since the dependent element is structurally lower than the antecedent, and since generation in cfg's is top to bottom, we assume that it is the dependent element that has to set the PRP_i according to the way they are set at the antecedent. We assume here that there is a look-up function f that for every assignment prp of the primitive labels at the antecedents the dependent element must satisfy $f(\mathsf{prp})$. In order to be able to achieve this correlation in a context-free grammar, the dependent element needs to know in which way the atoms PRP_i have been set at α. Thus the problem reduces to a transfer of information from α to δ. If we generate only fully labelled trees the problem is precisely to transfer n bits of information from α to δ. The precise nature of this information is of course irrelevant for the formalization.

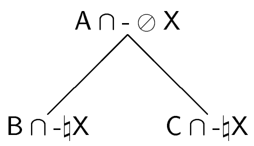
To begin with, we need to be able to recognize antecedent and dependent element by their category. We do this here by assuming two labels ANT and DEP with obvious meaning. Furthermore, our real task is to assure that the labels $\oslash\mathsf{X}$ and $\natural\mathsf{X}$ are correctly distributed. Notice, by the way, that it is only for special choices of X that we need these composite elements, so there is nothing recursive or infinite in this procedure. For the sake of simplicity we assume the grammar to be in Chomsky Normal Form; that is, we only have rules ot type $\mathsf{X} \to \mathsf{YZ}, \mathsf{X} \to \mathsf{Y}, \mathsf{X} \to 0$ for X, Y and Z atoms or $= \aleph$ (see [Harrison, 1978]). For any rule $\rho = \mathsf{A} \to \mathsf{BC}$ and any X we distribute the new labels $\oslash\mathsf{X}$ and $\natural\mathsf{X}$ as follows. If $\mathsf{B} \leq \mathsf{X}$ but $\mathsf{C} \not\leq \mathsf{X}$ then we replace ρ by



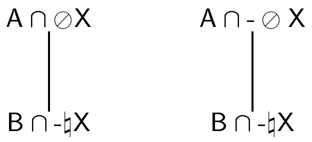
However, if $\mathsf{C} \leq \mathsf{X}$ but $\mathsf{B} \not\leq \mathsf{X}$ then we use this rule



It is clear what we do if both $\mathsf{B}, \mathsf{C} \leq \mathsf{X}$. If neither is the case, however, we have this rule



Likewise the unary rules are expanded. Here, we have either $\mathsf{B} \leq \mathsf{X}$ (left) or $\mathsf{B} \not\leq \mathsf{X}$ (right).



After having inserted enough $\oslash\mathsf{X}$ and $\natural\mathsf{X}$ we can proceed to the domains of accessibility. The general problem is as said above, the transfer of information from α to δ. The problem is attacked by introducing more modal elements. Namely, for certain $\mathfrak{g}$ and certain labels X we introduce the new label $\langle\mathfrak{g}\rangle\mathsf{X}$. Its interpretation is *an element of label X is in my* $\mathfrak{g}$*-domain and neither do I dominate it nor am I dominated by it.* If we succeed in distributing these new labels according to their intended interpretation we can code the Koster Matrix into the grammar. We show its encoding for $\langle\mathsf{F}\rangle\mathsf{Y}$. It is then more or less evident how $\langle\mathfrak{g}\rangle\mathsf{X}$ is encoded for a chain $\mathfrak{g}$ because $\langle\mathfrak{h} \circ \mathsf{F}\rangle\mathsf{X} = \langle\mathfrak{h}\rangle\langle\mathsf{F}\rangle\mathsf{X}$, just as in modal logic. Now for $\langle\mathsf{F}\rangle\mathsf{Y}$ there are two cases. (i) The mother node is of category $\langle\mathsf{F}\rangle\mathsf{Y} \cap \text{-}\mathsf{F}$. Then the information $\langle\mathsf{F}\rangle\mathsf{Y}$ must be passed on to all daughters. (ii) The mother is of category $\text{-}\langle\mathsf{F}\rangle\mathsf{Y} \cup \mathsf{F}$. Then a daughter is $\langle\mathsf{F}\rangle\mathsf{Y}$ if and only if it has a sister of category Y. Thus at all daughters we instantiate $\langle\mathsf{F}\rangle\mathsf{Y} \leftrightarrow \natural\mathsf{Y}$.

It should be quite clear that by a suitable choice of $\langle\mathfrak{g}\rangle\mathsf{X}$ to be added a dependent element δ will have access to the information that it has an antecedent in its domain of local accessibility^i. If it needs to know what category this antecedent has, this information has to be supplied in tandem with the mere property that needs to be shared. One snag remains; namely, it may happen that there are more than one antecedent of required type. In that case we need to manipulate the rules of the grammar as follows. As long as we have an element of category ANT we suppress any other antecedents of category ANT within the same domain. This might be not entirely straightforward, but to keep matters simple here we assume that the grammar takes care of that. We show now how the translation is completed. For accessibility[1] we add the following boolean axiom to the grammar (that is, we 'kill' all rules that do not comply with this axiom):

$$\langle\mathsf{BAR{:}2}\rangle(\mathsf{ANT} \cap \mathsf{prp}) \cap \natural\mathsf{GOV} \cap \mathsf{DEP}. \to .f(\mathsf{prp})$$

For accessibility[2] this axiom is added instead

$$\langle \mathsf{OPY} \circ \mathsf{BAR{:}2} \wedge \mathsf{OPY} \bullet \mathsf{BAR{:}2} \rangle (\mathsf{ANT} \cap \mathsf{prp}) \\ \cap \natural \mathsf{GOV} \cap \mathsf{DEP}. \rightarrow .f(\mathsf{prp})$$

Finally, for accessibility[3], we have to replace $\mathsf{BAR{:}2}$ by $\mathsf{BAR{:}2} \cap -\natural\mathsf{GOV}$.

More details can be found in [Kracht, 1993]. The upshot of this is the following. Suppose that a grammar of some language consists of a basic generative component in form of a cfg $\mathbb{G}$ and a number of Koster Matrices as additional constraints on the structures. If the number of matrices is finite, then finitely many additional labels suffice to create a cfg $\mathbb{G}^+$ from the original grammar that guarantess that it's output trees satisfy the local conditions of $\mathbb{G}$ as well as the nearness conditions imposed by the Koster Matrices. Upper bounds on the number of labels of $\mathbb{G}^+$ (depending both on $\mathbb{G}$ and the additional matrices) can be computed as well.

Acknowledgements

I wish to thank A. and J. for their moral support and F. Wolter for helpful discussions.

References

[Barker and Pullum, 1990] Chris Barker and Geoffrey Pullum. A theory of command relations. *Linguistics and Philosophy*, 13:1–34, 1990.

[Chomsky, 1986] Noam Chomsky. *Barriers.* MIT Press, Cambrigde (Mass.), 1986.

[Gazdar *et al.*, 1985] Gerald Gazdar, Ewan Klein, Geoffrey Pullum, and Ivan Sag. *Generalized Phrase Structure Grammar.* Blackwell, Oxford, 1985.

[Harrison, 1978] Michael A. Harrison. *Introduction to Formal Language Theory.* Addison-Wesley, Reading (Mass.), 1978.

[Koster and Reuland, 1991] Jan Koster and Eric Reuland, editors. *Long-Distance Anaphora.* Cambridge University Press, Cambridge, 1991.

[Koster, 1986] Jan Koster. *Domains and Dynasties: the Radical Autonomy of Syntax.* Foris, Dordrecht, 1986.

[Kracht, 1992] Marcus Kracht. The theory of syntactic domains. Technical report, Dept. of Philosophy, Rijksuniversiteit Utrecht, 1992. Logic Group Preprint Series No. 75.

[Kracht, 1993] Marcus Kracht. Nearness and syntactic influence spheres. Manuscript, 1993.

[Reinhart, 1981] Tanya Reinhart. Definite np-anaphora and c-command domains. *Linguistic Inquiry*, 12:605–635, 1981.

[Stabler, 1989] Edward Jr. Stabler. A logical approach to syntax: Foundation, specification and implementation of theories of government and binding. Manuscript, 1989.

[Sternefeld, 1991] Wolfgang Sternefeld. *Syntaktische Grenzen. Chomsky's Barrierentheorie und ihre Weiterentwicklungen.* Westdeutscher Verlag, Opladen, 1991.