2 Hybrid Type-Logical Categorial Grammar

2.1 Introduction

In this chapter, we introduce *Hybrid Type-Logical Categorial Grammar* (Hybrid TLCG; Kubota 2010, 2014, 2015; Kubota and Levine 2012, 2013a,b, 2014a, 2016a). Hybrid TLCG is a variant of *categorial grammar* (CG; Ajdukiewicz 1935; Bar-Hillel 1953; Lambek 1958) that belongs to the tradition of Type-Logical Categorial Grammar (Morrill 1994; Moortgat 1997). We provide a gentle introduction of our framework for syntacticians and semanticists without prior familiarity with CG. Although familiarity with notions in (discrete) mathematics, logic, and computer science would be helpful, we hope to have made the ensuing exposition accessible to anybody with background in graduate introductory-level syntax and (formal) semantics.

As discussed in the previous chapter, the notion of phrase structural constituency is central to both transformational and nontransformational variants of generative grammar. For this reason, to make the ensuing exposition maximally transparent to our readers, we present our fragment in a step-by-step manner, starting with an equivalent of simple context-free grammar (known as the *AB Grammar* [Ajdukiewicz 1935, Bar-Hillel 1953]) and then working it up to a fragment that can (at an abstract level) be thought of as a formalization of the familiar movement-based model of syntax (in various incarnations of transformational grammar in the mainstream syntax literature). Up to this point, there is nothing that is fundamentally different from the mainstream approach as far as the basic architecture of the grammar is concerned, where the only major differences if any are notational and methodological (e.g., the explicit encoding of valence information in syntactic categories and an insistence on providing explicit compositional semantics that goes with the syntactic component). We illustrate the workings of this extended system with the basic analyses of quantification (for "covert movement") and topicalization (for "overt movement"). The analyses crucially exploit the notion of *hypothetical reasoning* that is characteristic of the *type-logical* variants of CG (as opposed to "rule-based" systems such as Combinatory Categorial Grammar [CCG; Steedman 1996, 2000]; see chapter 12 for a comparison with CCG).

The final part of the framework exposition embodies a radical departure from the familiar assumptions in standard approaches in generative grammar. Here, we show that a conceptually natural extension of the "AB Grammar + movement" fragment developed up to that point effectively amounts to abandoning the notion of phrase structural constituency as a primitive notion in the combinatoric system for natural language syntax. This is admittedly a radical move (though corresponding to the more dominant view within the CG literature), but we show that a number of important and interesting consequences directly follow from making this move. The present chapter illustrates some key cases involving nonconstituent coordination, and the ensuing chapters demonstrate the empirical payoff of this system with case studies with greater complexity.

The central property of Hybrid TLCG is that it employs the notion of hypothetical reasoning (a fundamental tool of logic-based systems of syntax) not just for the treatment of movement-like phenomena but for capturing the regularities in the local combinatorics too. This results in a "hybrid" implication system in which grammatical composition via local and nonlocal combinatorics are both governed by logical principles and in which the two types of grammatical composition smoothly interact with one another. The present chapter illustrates how this architecture offers a straightforward analysis of some of the basic phenomena we deal with more extensively in later chapters, namely, the patterns of nonconstituent coordination and the syntax-semantics interface of quantifier scope. Later chapters exploit this property of Hybrid TLCG more extensively, in the analyses of more complex linguistic phenomena, especially Gapping, the interaction of coordination and scope, and the so-called respective readings and related phenomena (including the semantics of *same* and *different*). Later chapters will also illustrate how the flexible notion of constituency is useful not only in the analyses of coordination (as is already well-known in the CG literature) but also in the characterization of elliptical linguistic expressions in Gapping, pseudogapping, and Stripping (the analysis of Stripping in Hybrid TLCG is due to Puthawala [2018]). The empirical phenomena we treat in this book have all posed recalcitrant problems for both transformational and nontransformational approaches in the previous syntactic literature. We show how the flexible and systematic syntax-semantics interface of Hybrid TLCG enables simple analyses of these phenomena that overcome the problems of previous proposals while at the same time integrating the key analytic insights from these previous proposals.

2.2 The AB Grammar

We start with a simple fragment of CG called the *AB Grammar*, and introduce some key concepts and notations along the way that are shared by many variants of CG. Following the standard practice in CG, linguistic expressions are written as triples of prosodic representation (i.e., "PF"—we gloss over fine details of this component, so

this is just a string of words for all practical purposes except in chapter 11, in which we introduce a more sophisticated model of the prosodic component), semantic interpretation, and syntactic category, notated $\langle \pi; \sigma; \gamma \rangle$ (angle brackets will be omitted below). We assume that the set of syntactic categories is infinite and is recursively defined. This assumption reflects the view that natural language syntax is a kind of logic, a perspective that is particularly salient in the "type-logical" variants of CG (which Hybrid TLCG is an instance of). The underlying idea is that syntactic categories are like formulas in propositional logic. Just as the set of well-formed formulas in propositional logic is infinite in order to deal with unboundedly complex logical inferences, the set of syntactic categories is infinite in order to deal with (potentially) unboundedly complex combinatory properties of linguistic expressions.

We start with the following definition of *syntactic categories* (or *syntactic types*; following the convention in TLCG, we use the terms *syntactic type* and *syntactic category* interchangeably):[1]

(8) a. N, NP, and S are categories.
b. If A and B are categories, then so are A/B and $B\backslash A$.
c. Nothing else is a category.

The categories in (8a) are called *atomic categories* The ones involving slashes built from atomic categories via clause (8b) are called *complex categories*.

Note that this definition allows for an infinite set of syntactic categories. For example, by (8b) with $A = B = \mathrm{NP}$, we have NP/NP as a category. Applying (8b) again, this time with $A = B = \mathrm{NP}/\mathrm{NP}$, yields $(\mathrm{NP}/\mathrm{NP})/(\mathrm{NP}/\mathrm{NP})$. Doing the same thing once again yields $((\mathrm{NP}/\mathrm{NP})/(\mathrm{NP}/\mathrm{NP}))/((\mathrm{NP}/\mathrm{NP})/(\mathrm{NP}/\mathrm{NP}))$. And so on. The definition of syntactic categories becomes slightly more complex once we introduce the vertical slash ($|$) below (see appendix A.1 for a complete definition), but for now, the simple definition in (8) (standard in the CG literature) suffices.

One important feature of CG is that it lexicalizes the valence (or subcategorization) properties of linguistic expressions via the use of complex syntactic categories. For example, lexical entries for intransitive and transitive verbs in English will look like the following (semantics is omitted here but will be supplied later):

(9) a. ran; $\mathrm{NP}\backslash\mathrm{S}$
b. read; $(\mathrm{NP}\backslash\mathrm{S})/\mathrm{NP}$

1. We assume a small number of syntactic features for atomic categories, notated as subscripts as in NP_{nom} (for nominative NP) and NP_{acc} (for accusative NP). One way to formalize the notion of syntactic features in CG is by means of dependent types (Martin-Löf 1984; Ranta 1994), along the lines discussed in Morrill (1994) and Pogodalla and Pompigne (2012).

(9a) says that the verb *ran* combines with its argument NP *to its left* to become an S. Likewise, (9b) says that *read* first combines with an NP *to its right* and then another NP to its left to become an S. Thus, the slashes encode both the linear order in which the verb looks for its arguments and the number (and type) of arguments that it subcategorizes for. It is useful to introduce some terminology here: we say that complex categories (like the ones in (9)) designate *functors* that combine with *arguments* to return *results*.

As already noted, the distinction between the *forward slash* ($/$) and the *backward slash* ($\backslash$) corresponds to the distinction in the directions (in the surface word order) in which functors look for arguments. That is, A/B is a functor looking for a B *to its right* (to become an A), whereas $B\backslash A$ is a functor looking for a B *to its left*. We adopt the so-called Lambek-style notation for syntactic categories, in which arguments are always written "under the slash." In the alternative notation adopted in CCG, the result category is always written on the left, with the consequence that the backward slash is written in the opposite way from the Lambek-style notation; that is, our $B\backslash A$ will be written $A\backslash B$ in the CCG notation. We omit outermost parentheses and parentheses for a sequence of the same type of slash, assuming that / and $\uparrow$ (introduced below) are left associative and $\backslash$ is right associative. Thus, $\mathrm{S}/\mathrm{NP}/\mathrm{NP}$, $\mathrm{NP}\backslash\mathrm{NP}\backslash\mathrm{S}$, and $\mathrm{S}\uparrow\mathrm{NP}\uparrow\mathrm{NP}$ are abbreviations of $((\mathrm{S}/\mathrm{NP})/\mathrm{NP})$, $(\mathrm{NP}\backslash(\mathrm{NP}\backslash\mathrm{S}))$, and $((\mathrm{S}\uparrow\mathrm{NP})\uparrow\mathrm{NP})$, respectively.

We first introduce the two basic rules in the grammar, namely, *Slash Elimination* rules for forward and backward slashes. In the so-called labeled deduction format of natural deduction (cf. Oehrle 1994; Morrill 1994), these rules are formulated as in (10).

(10) a. Forward Slash Elimination

$$\frac{a;\, A/B \qquad b;\, B}{a \circ b;\, A}\,/\mathrm{E}$$

b. Backward Slash Elimination

$$\frac{b;\, B \qquad a;\, B\backslash A}{b \circ a;\, A}\,\backslash\mathrm{E}$$

The inputs to the rule are written above the horizontal line and the output is written below the line. In line with the analogy between language and logic underlying TLCG, we call the inputs of these rules *premises* and the outputs *conclusions*. The linear order between the two premises above the line has only mnemonic significance; as will become clear below, what *is* significant to word order is instead the order of a and b in the prosody of the expression obtained as the conclusion. The labeled deduction presentation is so called since, in addition to the syntactic categories of the premises and conclusions, the rules are also annotated (or labeled) with how the semantics and prosody of the conclusion are computed given the semantics and prosody of the premises.

The *proof* (or *derivation*; following the CG tradition, we use these two terms interchangeably, but it should be kept in mind that the notion of derivation in CG is quite different from that in standard derivational approaches) in (11) illustrates how larger linguistic expressions are built from smaller ones using the rules just introduced. Here,

a transitive verb, of category $(\mathrm{NP}\backslash\mathrm{S})/\mathrm{NP}$, is combined with its two arguments on the right (object) and left (subject).

(11)
$$\dfrac{\mathsf{john};\ \mathrm{NP} \qquad \dfrac{\mathsf{mary};\ \mathrm{NP} \quad \mathsf{loves};\ (\mathrm{NP}\backslash\mathrm{S})/\mathrm{NP}}{\mathsf{loves}\circ\mathsf{mary};\ \mathrm{NP}\backslash\mathrm{S}}\,/\mathrm{E}}{\mathsf{john}\circ\mathsf{loves}\circ\mathsf{mary};\ \mathrm{S}}\,\backslash\mathrm{E}$$

Note that the object NP *Mary* is placed to the left of the verb in the proof tree, but this does not have any significance in the linear order of the string derived. Thus, unlike linguistic trees in ordinary syntactic theories (or in CCG), the left-to-right yield of the proof trees does not correspond to word order.

From a "logical" point of view, the two slashes should be thought of as directional variants of implication (that is, both A/B and $B\backslash A$ essentially mean '*if* there is a B, *then* there is an A'), and the two rules of Slash Elimination should be thought of as directional variants of *modus ponens* ($B \to A, B \vdash A$). Thus, (11) is literally a proof of the fact that a complete sentence exists if there is one NP to the right and one NP to the left of the verb *loves*.

We now turn to the syntax-semantics mapping. For expository ease, we assume a standard Montagovian model-theoretic semantics.[2] As is standard in CG, we assume a homomorphic mapping from syntactic categories to semantic types.[3] This means that, for each linguistic expression, given its syntactic category, one can determine uniquely and unambiguously its semantic type (e.g., individuals, sets of individuals, sets of sets of individuals, etc.). To work this out, we first assume that semantic types are recursively built from the basic types e (individuals) and t (truth values) in the standard fashion:[4]

(12) a. e and t are semantic types.
b. If α and β are semantic types, then so is $\alpha \to \beta$.

2. Since we do not deal with phenomena that crucially involve intensionality, we assume an extensionalized fragment throughout. Also, our approach is in principle compatible with more sophisticated semantic theories such as dynamic semantics. See, for example, Martin (2013) and Martin and Pollard (2014) for a proposal for incorporating a compositional dynamic semantics to Linear Categorial Grammar (LCG), a version of CG closely related to Hybrid TLCG. See also Kubota et al. (2019) for a proposal for coupling Hybrid TLCG with Dependent Type Semantics (Bekki 2014; Bekki and Mineshima 2017), a recent proof-theoretic approach to compositional dynamic semantics.

3. Technically, this is ensured in TLCG by the homomorphism from the syntactic type logic to the semantic type logic (the latter of which is often implicit) and the so-called Curry-Howard correspondence (Howard 1969) between proofs and terms (van Benthem 1988b).

Linear Categorial Grammar (Mihalicek and Pollard 2012; Worth 2016) is different from other variants of CG in that it does not assume a functional mapping from syntactic types to semantic types. Instead, the mapping is relational (see in particular Worth [2014, section 1.1] for an explicit statement of this point).

4. We replace Montague's (1973) notation $\langle\beta,\alpha\rangle$ with $\beta \to \alpha$, so that the notation more transparently reflects the fact that the complex type is a functional type.

c. Nothing else is a semantic type.

Then, we can define the function Sem that returns, for each syntactic category given as input, its semantic type:

(13) (Base Case)
a. $\mathsf{Sem}(\mathrm{NP}) = e$
b. $\mathsf{Sem}(\mathrm{N}) = e \to t$
c. $\mathsf{Sem}(\mathrm{S}) = t$

(14) (Recursive Clause)
For any complex syntactic category of the form A/B (or $B\backslash A$),
$\mathsf{Sem}(A/B)$ (= $\mathsf{Sem}(B\backslash A)$) = $\mathsf{Sem}(B) \to \mathsf{Sem}(A)$

(14) says that the semantic type of a complex syntactic category A/B (or $B\backslash A$) is that of a function that takes as its argument an expression that has the semantic type of its syntactic argument (i.e., $\mathsf{Sem}(B)$) and returns an expression that has the semantic type of its result syntactic category (i.e., $\mathsf{Sem}(A)$).

We can now write full lexical entries that specify the semantics as well:

(15) a. ran; **ran**; NP\S
b. saw; **saw**; (NP\S)/NP

Given the functional mapping from syntactic categories to semantic types in (14), we see that the intransitive verb *ran* is semantically of type $e \to t$ (set of individuals) and that the transitive verb *saw* is of type $e \to (e \to t)$ (a function from individuals to sets of individuals, i.e., a curried two-place relation).

Syntactic rules with semantics can then be written as in (16) and a sample derivation with semantic annotation is given in (17).

(16) a. Forward Slash Elimination

$$\frac{a;\ \mathcal{F};\ A/B \qquad b;\ \mathcal{G};\ B}{a \circ b;\ \mathcal{F}(\mathcal{G});\ A}\ /\mathrm{E}$$

b. Backward Slash Elimination

$$\frac{b;\ \mathcal{G};\ B \qquad a;\ \mathcal{F};\ B\backslash A}{b \circ a;\ \mathcal{F}(\mathcal{G});\ A}\ \backslash\mathrm{E}$$

(17)

$$\frac{\mathsf{john};\ \mathbf{j};\ \mathrm{NP} \qquad \dfrac{\mathsf{loves};\ \mathbf{love};\ (\mathrm{NP}\backslash\mathrm{S})/\mathrm{NP} \qquad \mathsf{mary};\ \mathbf{m};\ \mathrm{NP}}{\mathsf{loves} \circ \mathsf{mary};\ \mathbf{love}(\mathbf{m});\ \mathrm{NP}\backslash\mathrm{S}}\ /\mathrm{E}}{\mathsf{john} \circ \mathsf{loves} \circ \mathsf{mary};\ \mathbf{love}(\mathbf{m})(\mathbf{j});\ \mathrm{S}}\ \backslash\mathrm{E}$$

Note that the semantic effect of Slash Elimination is *function application* in both of the two rules in (16).

A system of CG with only the Slash Elimination rules such as the fragment above is called the *AB Grammar*, because it corresponds to the earliest form of CG formulated by Ajdukiewicz (1935) and Bar-Hillel (1953).

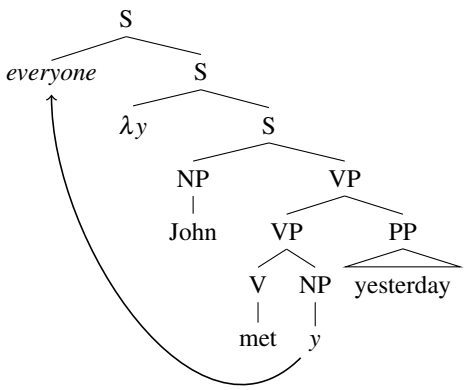
2.3 Adding the Vertical Slash to the AB Grammar

The AB Grammar introduced above is basically equivalent to phrase structure grammar (PSG) without elaborate additional mechanisms such as (overt or covert) movement, the SLASH feature inheritance mechanism in Generalized/Head-Driven Phrase Structure Grammar (G/HPSG), or quantifier storage due to Cooper (1983). In fact, the formal equivalence between the AB Grammar and context-free grammar (i.e., simple PSG) was proved as early as Bar-Hillel et al. (1960). This means that the AB Grammar is too impoverished for modeling complex empirical phenomena in natural language that have motivated the notion of "movement" in transformational grammar and various alternative mechanisms (like the SLASH feature and Cooper storage) in other frameworks.

In this section, we extend the AB Grammar by adding just one new mechanism—specifically, a new type of slash called the *vertical slash* ($|$), which, unlike the forward and backward slashes, does not encode linear order in syntactic categories (see Moortgat [1990] for a related earlier proposal involving the "q constructor" $\Uparrow$). This extended fragment can naturally model the notion of movement in derivational approaches. The mechanism of the vertical slash in Hybrid TLCG is moreover more general than that of movement in such approaches, and this difference becomes crucial in the analysis of Gapping we present in chapter 3. We will come back to this point in that chapter.

The problem with the AB Grammar can be illustrated by the failed derivation for the sentence *John met everyone yesterday* in (19), which attempts to mimic the Quantifier Raising (QR)-based analysis in a derivational framework in (18).

(18)



(19)

$$\dfrac{\text{everyone; NP} \qquad \dfrac{\text{john; NP} \qquad \dfrac{\dfrac{\text{met; } (\text{NP}\backslash\text{S})/\text{NP} \quad \varphi\text{; NP}}{\text{met} \circ \varphi\text{; NP}\backslash\text{S}}/\text{E} \qquad \dfrac{\text{yesterday;}}{(\text{NP}\backslash\text{S})\backslash(\text{NP}\backslash\text{S})}}{\text{met} \circ \varphi \circ \text{yesterday; NP}\backslash\text{S}}\backslash\text{E}}{\text{john} \circ \text{met} \circ \varphi \circ \text{yesterday; S}}\backslash\text{E}}{\text{john} \circ \text{met} \circ \text{everyone} \circ \text{yesterday; S}}????$$

In (19), to model the covert movement of the quantifier, we start by positing a dummy expression, whose prosody is represented by the symbol φ. The derivation proceeds up to the point where a complete sentence is formed containing this "trace."
What we then need to do is combine the quantifier with this sentence so that it semantically scopes over the whole sentence but appears in the "trace" position in the prosodic form (since this is an instance of "covert movement").

However, things go wrong in all of the three components of the tripartite linguistic sign. First, syntactically, with the two Slash Elimination rules we have introduced above, we can only combine functors (i.e., expressions of the form A/B or $B\backslash A$, involving either the forward or the backward slash) with arguments, but neither S nor NP is a functor that takes the other as its argument. Prosodically, all we can do with the two rules we have introduced above is concatenate the two strings. But here, what we need to do is a more elaborate type of string manipulation whereby we replace the dummy string φ with the string of the quantifier *everyone* (i.e., an operation corresponding to Montague's (1973) syncategorematic rule of "quantifying-in"). Finally, semantically, the variable corresponding to the "trace" needs to be bound by the λ-operator and the property thus obtained given as an argument to the quantifier, but we currently do not have any rule in the grammar that allows for such a complex meaning assembly.

Are we then stuck? In a sense, yes, since the above illustration clearly shows that there is no natural way of modeling the notion of covert movement in the AB Grammar. Essentially the same point can be made with overt movement. But in fact, our attempt with the failed derivation in (19) was *almost* on the right track. Specifically, all that is missing is a mechanism that enables the last step to go through. And it turns out that such a mechanism is already available in CG, albeit outside of the AB Grammar and in a relatively recent line of development which has not yet gained due recognition in the linguistic literature.

Although approaches that distinguish word order by the forward and backward slashes (like the AB Grammar fragment formulated above) have been the mainstream in CG research, there is a relatively recent strand of research that relocates the word order information from the syntactic types to the prosodic component (Oehrle 1994; de Groote 2001; Muskens 2003; Mihalicek and Pollard 2012). Following Pollard (2013), we call this family of approaches *Linear Categorial Grammar* (LCG). LCG is characteristic for its use of the λ-calculus for modeling the prosody of linguistic expressions, and this plays a key role in encoding word order information in the prosodic representation directly.

Part of the motivation for LCG comes precisely from the recognition that variants of CG based on the distinction between forward and backward slashes are suboptimal for handling phenomena that are essentially insensitive to word order, such as quantification and extraction (see Muskens [2003] for a particularly lucid discussion on this

point). We incorporate the key insight of LCG into our AB fragment and show that the extended fragment indeed gives us exactly the right tool to "fill in the gap" in the derivation in (19). The reason we make this move (of combining AB with LCG), rather than simply switching to LCG, will become clear later—it mainly has to do with the analysis of coordination. In a purely "nondirectional" calculus of LCG, the analysis of coordination quickly becomes extremely unwieldy, and, although there are some attempts at addressing this issue (Worth 2016; Kanazawa 2015; Pollard and Worth 2015), at this point it is still considerably unclear whether a completely general solution for this problem can be obtained in LCG. See chapter 12, section 12.3 for a more detailed discussion on this point.

The new mechanism we incorporate from LCG is an order-insensitive mode of implication $\uparrow$, called *vertical slash*. We introduce the *Introduction* and *Elimination* rules for this slash, which are formulated as follows (as with /, we write the argument to the right for $\uparrow$; the harpoon is there as a visual aide indicating that the right category is the argument):

(20) a. Vertical Slash Introduction

$$\frac{\begin{array}{c}\vdots\ [\varphi; x; A]^n\ \vdots \\ \vdots\qquad\vdots\qquad\vdots \\ b; \mathcal{F}; B\end{array}}{\lambda\varphi.b; \lambda x.\mathcal{F}; B{\uparrow}A}\uparrow\mathrm{I}^n$$

b. Vertical Slash Elimination

$$\frac{a; \mathcal{F}; A{\uparrow}B \qquad b; \mathcal{G}; B}{a(b); \mathcal{F}(\mathcal{G}); A}\uparrow\mathrm{E}$$

The workings of these rules can be best illustrated with examples. We show in (21) the derivation for the sentence *John saw everyone yesterday*.

(21)

$$\textcircled{3}\rightarrow\frac{\lambda\sigma.\sigma(\mathsf{everyone});\ \mathbf{V}_{\mathbf{person}};\ \mathrm{S}{\uparrow}(\mathrm{S}{\uparrow}\mathrm{NP}) \qquad \textcircled{2}\rightarrow\dfrac{\mathsf{john};\ \mathbf{j};\ \mathrm{NP} \qquad \textcircled{1}\rightarrow\dfrac{\dfrac{\mathsf{saw};\ \mathbf{saw};\ (\mathrm{NP}\backslash\mathrm{S})/\mathrm{NP} \qquad \begin{bmatrix}\varphi;\\ x;\\ \mathrm{NP}\end{bmatrix}^1}{\mathsf{saw}\circ\varphi;\ \mathbf{saw}(x);\ \mathrm{NP}\backslash\mathrm{S}}/\mathrm{E} \qquad \dfrac{\mathsf{yesterday};\ \mathbf{yest};\ (\mathrm{NP}\backslash\mathrm{S})\backslash(\mathrm{NP}\backslash\mathrm{S})}{}}{\mathsf{saw}\circ\varphi\circ\mathsf{yesterday};\ \mathbf{yest}(\mathbf{saw}(x));\ \mathrm{NP}\backslash\mathrm{S}}\backslash\mathrm{E}}{\mathsf{john}\circ\mathsf{saw}\circ\varphi\circ\mathsf{yesterday};\ \mathbf{yest}(\mathbf{saw}(x))(\mathbf{j});\ \mathrm{S}}\backslash\mathrm{E}}{\lambda\varphi.\mathsf{john}\circ\mathsf{saw}\circ\varphi\circ\mathsf{yesterday};\ \lambda x.\mathbf{yest}(\mathbf{saw}(x))(\mathbf{j});\ \mathrm{S}{\uparrow}\mathrm{NP}}\uparrow\mathrm{I}^1}{\lambda\sigma.[\sigma(\mathsf{everyone})](\lambda\varphi.\mathsf{john}\circ\mathsf{saw}\circ\varphi\circ\mathsf{yesterday});\ \mathbf{V}_{\mathbf{person}}(\lambda x.\mathbf{yest}(\mathbf{saw}(x))(\mathbf{j}));\ \mathrm{S}}\uparrow\mathrm{E}$$

$$\lambda\varphi.[\mathsf{john}\circ\mathsf{saw}\circ\varphi\circ\mathsf{yesterday}](\mathsf{everyone});\ \mathbf{V}_{\mathbf{person}}(\lambda x.\mathbf{yest}(\mathbf{saw}(x))(\mathbf{j}));\ \mathrm{S}$$

$$\mathsf{john}\circ\mathsf{saw}\circ\mathsf{everyone}\circ\mathsf{yesterday};\ \mathbf{V}_{\mathbf{person}}(\lambda x.\mathbf{yest}(\mathbf{saw}(x))(\mathbf{j}));\ \mathrm{S}$$

The main new ingredient here is a type of inference called *hypothetical reasoning*. In ordinary kinds of logic, hypothetical reasoning is a type of proof in which one draws the conclusion $A \rightarrow B$ on the basis of a proof of B by first hypothetically assuming A.

As will become clear below (especially when we introduce hypothetical reasoning for the directional slashes in section 2.4), hypothetical reasoning is a very powerful (yet systematic) tool that is deeply rooted in the CG conception of natural language syntax as a kind of logic and which distinguishes CG from phrase structure–based theories of syntax (including both derivational and non-derivational approaches).

In (21), we first hypothesize an object NP with prosody φ and semantics x. In derivations, hypotheses are indicated by square brackets and are indexed. The indices are for keeping track of where the hypothesis is withdrawn in the whole proof—see below for more on this convention. The derivation proceeds in the same way as in (19) up to the point where the whole sentence is formed (①). At this point, we know that the string containing the prosodic variable φ is a full-fledged sentence. We then apply the *Vertical Slash Introduction* rule (20a) to withdraw this hypothesis (②). In this step, we are essentially concluding that the string *John saw __ yesterday* would be a well-formed sentence of English *if* there were an NP in the gap position notated by __. Note that this conclusion itself does not depend on the assumption that there is in fact an NP in the object position. It is in this sense that Vertical Slash Introduction *withdraws* the hypothesis posited at an earlier step in the proof. The hypothesis is there only for the sake of drawing this conclusion, and it doesn't play any other role in the grammar (and in this sense, it has a very different status from traces in derivational approaches, which are representational objects); rather, the hypothesis and its withdrawal are the type-logical analogue of the Implication Elimination rule in natural deduction formulations of intuitionistic propositional logic. The ↑I step is coindexed with the hypothesis that is withdrawn so as to keep track of which hypothesis is withdrawn at which step in the proof. The vertical dots around the hypothesis in the rule in (20a) abbreviate an arbitrarily complex proof structure. Thus, (20a) simply says that a hypothesis posited at some previous step can be withdrawn by ↑I at any step in the proof.

Let us now examine the effect of the Vertical Slash Introduction rule more closely. In the prosody of the derived expression, the gap position is explicitly represented by the prosodic variable φ bound by the λ-operator, resulting in a function from string to string (of type **st→st**, with **st** the type of strings). Correspondingly, the semantic action of ↑I is also variable binding—the variable x posited as the semantic placeholder for the "trace" is bound. Note that this creates the right property to be given as an argument to the quantifier. Finally, the syntactic category S↑NP, with the vertical slash ↑, indicates that the whole expression is a sentence missing an NP ('if there is an NP, then there is an S'), just like S/NP and NP\S. The difference from the directional slashes is that the vertical slash does not indicate where within the whole string the NP is missing, since that information is represented in the prosodic term itself via the use of λ-binding.

The quantifier then takes this **st→st** function as an argument and embeds its string component in the gap position (③). This step is licensed by the *Vertical Slash Elim-*

ination rule (20b). The dotted lines show reduction steps for the prosodic term (we often omit these in the derivations below) and should not be confused with the application of logical rules (of Slash Elimination and Introduction) designated by solid lines; unlike the latter, purely from a formal perspective, these reduction steps are redundant. Since the quantifier takes an **st→st** function as its argument, it has a higher-order prosody (of type **(st→st)→st**) itself. Semantically, the quantifier denotes a standard generalized quantifier (GQ) meaning of type $(e \to t) \to t$. $\mathbf{\forall_{person}}$ abbreviates the term $\lambda P.\forall x[\mathbf{person}(x) \to P(x)]$ (similarly for the existential quantifier $\mathbf{\exists_{person}}$). Since the $\uparrow$E rule does function application both in the semantic and prosodic components, the right string/meaning pair is assigned to the whole sentence by this derivation.

The analysis of scope ambiguity is then straightforward and is parallel to quantifying-in and QR. (22) shows the derivation for the inverse scope ($\forall > \exists$) reading of *Someone talked to everyone yesterday*:

(22)

$$\dfrac{\lambda\sigma.\sigma(\text{someone});\ \mathbf{\exists_{person}};\ \text{S}\uparrow(\text{S}\uparrow\text{NP}) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\begin{bmatrix}\varphi_2;\\ x_2;\\ \text{NP}\end{bmatrix}^2 \qquad \dfrac{\text{talked}\circ\text{to};\ \mathbf{talked\text{-}to};\ (\text{NP}\backslash\text{S})/\text{NP} \qquad \begin{bmatrix}\varphi_1;\\ x_1;\\ \text{NP}\end{bmatrix}^1}{\text{talked}\circ\text{to}\circ\varphi_1;\ \mathbf{talked\text{-}to}(x_1);\ \text{NP}\backslash\text{S}}/\text{E}}{\varphi_2\circ\text{talked}\circ\text{to}\circ\varphi_1;\ \mathbf{talked\text{-}to}(x_1)(x_2);\ \text{S}}\backslash\text{E} \qquad \text{yesterday};\ \mathbf{yest};\ \text{S}\backslash\text{S}}{\varphi_2\circ\text{talked}\circ\text{to}\circ\varphi_1\circ\text{yesterday};\ \mathbf{yest}(\mathbf{talked\text{-}to}(x_1)(x_2));\ \text{S}}\backslash\text{E}}{\lambda\varphi_2.\varphi_2\circ\text{talked}\circ\text{to}\circ\varphi_1\circ\text{yesterday};\ \lambda x_2.\mathbf{yest}(\mathbf{talked\text{-}to}(x_1)(x_2));\ \text{S}\uparrow\text{NP}}\uparrow\text{I}^2}{\text{someone}\circ\text{talked}\circ\text{to}\circ\varphi_1\circ\text{yesterday};\ \mathbf{\exists_{person}}(\lambda x_2.\mathbf{yest}(\mathbf{talked\text{-}to}(x_1)(x_2)));\ \text{S}}\uparrow\text{E}}{\lambda\varphi_1.\text{someone}\circ\text{talked}\circ\text{to}\circ\varphi_1\circ\text{yesterday};\ \lambda x_1.\mathbf{\exists_{person}}(\lambda x_2.\mathbf{yest}(\mathbf{talked\text{-}to}(x_1)(x_2)));\ \text{S}\uparrow\text{NP}}\uparrow\text{I}^1$$

$$\lambda\sigma.\sigma(\text{everyone});\ \mathbf{\forall_{person}};\ \text{S}\uparrow(\text{S}\uparrow\text{NP})$$

$$\dfrac{\text{someone}\circ\text{talked}\circ\text{to}\circ\text{everyone}\circ\text{yesterday};\ \mathbf{\forall_{person}}(\lambda x_1.\mathbf{\exists_{person}}(\lambda x_2.\mathbf{yest}(\mathbf{talked\text{-}to}(x_1)(x_2))));\ \text{S}}{\uparrow\text{E}}$$

The point here is that the scopal relation between multiple quantifiers depends on the order of application of the hypothetical reasoning steps involving $\uparrow$ to introduce quantifiers. We get the inverse scope reading in this derivation since the subject quantifier is introduced in the derivation first.

On the present approach, the difference between overt movement and covert movement comes down to a lexical difference in the "prosodic action" of the operator that triggers the "movement" operation in question. As shown above, covert movement is modeled by an operator which embeds some (non-empty) string in the gap position,

whereas overt movement is modeled by an operator which embeds the empty string in the gap position. Thus, as shown by Muskens (2003), extraction can be analyzed quite elegantly in this type of CG with hypothetical reasoning for $\uparrow$, as in the derivation in (24) for the topicalization example (23).

(23) Bagels$_i$, Kim gave t_i to Chris.

(24)

$$\dfrac{\text{bagels};\ \mathbf{b};\ \text{NP} \qquad \dfrac{\lambda\sigma\lambda\varphi.\varphi\circ\sigma(\boldsymbol{\varepsilon});\ \lambda\mathcal{F}.\mathcal{F};\ (\text{S}\uparrow X)\uparrow(\text{S}\uparrow X) \qquad \textcircled{1}\rightarrow \dfrac{\dfrac{\text{kim};\ \mathbf{k};\ \text{NP} \qquad \dfrac{\dfrac{\text{gave};\ \mathbf{gave};\ (\text{NP}\backslash\text{S})/\text{PP}/\text{NP} \qquad \begin{bmatrix}\varphi;\\ x;\\ \text{NP}\end{bmatrix}^1}{\text{gave}\circ\varphi;\ \mathbf{gave}(x);\ (\text{NP}\backslash\text{S})/\text{PP}}/\text{E} \qquad \text{to}\circ\text{chris};\ \mathbf{c};\ \text{PP}}{\text{gave}\circ\varphi\circ\text{to}\circ\text{chris};\ \mathbf{gave}(x)(\mathbf{c});\ \text{NP}\backslash\text{S}}/\text{E}}{\text{kim}\circ\text{gave}\circ\varphi\circ\text{to}\circ\text{chris};\ \mathbf{gave}(x)(\mathbf{c})(\mathbf{k});\ \text{S}}\backslash\text{E}}{\lambda\varphi.\text{kim}\circ\text{gave}\circ\varphi\circ\text{to}\circ\text{chris};\ \lambda x.\mathbf{gave}(x)(\mathbf{c})(\mathbf{k});\ \text{S}\uparrow\text{NP}}\uparrow\text{I}^1}{\lambda\varphi.\varphi\circ\text{kim}\circ\text{gave}\circ\text{to}\circ\text{chris};\ \lambda x.\mathbf{gave}(x)(\mathbf{c})(\mathbf{k});\ \text{S}\uparrow\text{NP}}\uparrow\text{E}}{\text{bagels}\circ\text{kim}\circ\text{gave}\circ\text{to}\circ\text{chris};\ \mathbf{gave}(\mathbf{b})(\mathbf{c})(\mathbf{k});\ \text{S}}\uparrow\text{E}$$

In (24), a gapped sentence is derived just in the same way as in the quantifier example above via hypothetical reasoning for $\uparrow$ (①). The difference from the quantifier example is that the topicalization operator, grayed in (24), embeds an empty string $\boldsymbol{\varepsilon}$ in the gap position, thereby closing off the gap, and then concatenates the string of the topicalized NP to the left of that string.

In short, what we have here can be thought of as a logical reconceptualization of transformational grammar; what we used to call "trace" in derivational approaches is just a hypothesis in a proof, and "movement" is just a metaphorical name for a particular type of hypothetical reasoning.[5] This key insight, which seems to stem from the early 1990s when the natural deduction formulations of various extensions of the Lambek calculus were introduced (see, e.g., Hepple 1990; Morrill 1994) and which is expressed particularly clearly in Oehrle (1994), not only is theoretically illuminating, capturing the tight correlation between the semantic and prosodic effects of quantification and extraction transparently, but also has a number of empirical advantages. Specifically, this approach enables an explicit and precise characterization of more complex types of scope-taking such as *parasitic scope* of symmetrical predicates (Barker 2007; Pollard and Smith 2012) and *split scope* of negative quantifiers (Kubota and Levine 2016a) and

5. But caution is in order here; this is just a rough and crude analogy. In fact, one important property of Slash Introduction rules (including the Vertical Slash Introduction rule above) is that, as inference rules in the logical deductive system, they *define* the properties of the slashes together with the Slash Elimination rules. For this reason, the grammar rules in (TL)CG have very different statuses conceptually from "corresponding" rules in other theories.

number determiners (Pollard 2014). Gapping is especially interesting in this connection in that it exhibits the properties of both "overt" and "covert" movement simultaneously, thus constituting a case that goes beyond the analytic possibilities available in the standard derivational architecture. We illustrate these points in later chapters.

2.4 Hypothetical Reasoning for All Slashes: Hybrid Type-Logical Categorial Grammar

At this point, we extend our fragment once more, this time by adding the Introduction rules for the forward and backward slashes. This gives us the full Hybrid TLCG, complete with both the Introduction and Elimination rules for all three slashes $/$, $\backslash$, and $\uparrow$. The main motivation for extending the system with the Introduction rules for the directional slashes comes from the analysis of coordination—including cases of nonconstituent coordination (Right-Node Raising [RNR] and Dependent Cluster Coordination [DCC]), as we illustrate below.

The Slash Introduction rules for $/$ and $\backslash$ are formulated as follows:

(25) a. Forward Slash Introduction

$$\dfrac{\begin{array}{c}\vdots\ [\varphi;\, x;\, A]^n\ \vdots \\ \vdots\qquad\vdots\qquad\vdots \\ b\circ\varphi;\, \mathcal{F};\, B\end{array}}{b;\, \lambda x.\mathcal{F};\, B/A}\,/\mathrm{I}^n$$

b. Backward Slash Introduction

$$\dfrac{\begin{array}{c}\vdots\ [\varphi;\, x;\, A]^n\ \vdots \\ \vdots\qquad\vdots\qquad\vdots \\ \varphi\circ b;\, \mathcal{F};\, B\end{array}}{b;\, \lambda x.\mathcal{F};\, A\backslash B}\,\backslash\mathrm{I}^n$$

The difference between the Introduction rule for the vertical slash and the Introduction rules for the directional slashes is that, unlike in $\uparrow$I, in $/$I and $\backslash$I, the prosodic variable φ for the hypothesis is simply thrown away in the conclusion on the condition of its presence at the (right or left) periphery of the prosody of the premise, instead of explicitly being bound by the λ-operator. The position of the missing expression is instead recorded in the forward versus backward slash distinction in the syntactic category.

As will become clear in a moment, with the Introduction rules for $/$ and $\backslash$, it becomes possible to reanalyze any substring of a sentence as a (derived) constituent. We first illustrate the workings of these rules with analyses of RNR and DCC and then come back to the relevant formal details. (27) shows how the string *John loves* in the RNR example in (26) is assigned the syntactic category S/NP via hypothetical reasoning involving $/$.

(26) John loves, and Bill hates, Mary.

(27)

$$\textcircled{1}\rightarrow\dfrac{\textcircled{2}\rightarrow\dfrac{\mathsf{john};\, \mathbf{j};\, \mathrm{NP}\qquad\dfrac{[\varphi;x;\mathrm{NP}]^1\quad \mathsf{loves};\, \mathbf{love};\, (\mathrm{NP}\backslash\mathrm{S})/\mathrm{NP}}{\mathsf{loves}\circ\varphi;\, \mathbf{love}(x);\, \mathrm{NP}\backslash\mathrm{S}}\backslash\mathrm{E}}{\mathsf{john}\circ\mathsf{loves}\circ\varphi;\, \mathbf{love}(x)(\mathbf{j});\, \mathrm{S}}/\mathrm{E}}{\mathsf{john}\circ\mathsf{loves};\, \lambda x.\mathbf{love}(x)(\mathbf{j});\, \mathrm{S}/\mathrm{NP}}/\mathrm{I}^1$$

By hypothesizing a direct object NP, we first prove an S (①). At this point, since the prosody of the hypothesized NP φ appears on the right periphery, we can apply /I to withdraw the hypothesis (②). Intuitively, what is going on here can be paraphrased as follows: since we've proven that there is a complete S by assuming that there is an NP on the right periphery (①), we know that, without this hypothetical NP, what we have is something that becomes an S *if* there is an NP to its right (②). Thus, this is another instance of hypothetical reasoning. The difference from the case for ↑ is that, instead of explicitly keeping track of the position of the missing expression via λ-binding in the prosodic representation, the prosodic variable φ is simply thrown away, and the slash in the syntactic category records the fact that the NP is missing on the right periphery. Note also that the lambda abstraction on the corresponding variable in semantics assigns the right meaning (of type $e \to t$) to the derived S/NP.

In the CG analysis of RNR (see, e.g., Morrill 1994; the original analytic insight goes back to Steedman [1985]) such nonconstituents are directly coordinated as constituents and then combined with the RNR'ed expression as in (28):

(28)

$$\dfrac{\dfrac{\dfrac{\vdots}{\text{john} \circ \text{loves};\ \lambda x.\mathbf{love}(x)(\mathbf{j}); \text{S/NP}} \quad \dfrac{\text{and};\ \lambda\mathcal{W}\lambda\mathcal{V}.\mathcal{V} \sqcap \mathcal{W};\ (X\backslash X)/X \qquad \dfrac{\vdots}{\text{bill} \circ \text{hates};\ \lambda x.\mathbf{hate}(x)(\mathbf{b}); \text{S/NP}}\ /\text{E}}{\text{and} \circ \text{bill} \circ \text{hates};\ \lambda\mathcal{V}.\mathcal{V} \sqcap \lambda x.\mathbf{hate}(x)(\mathbf{b}); (\text{S/NP})\backslash(\text{S/NP})}\ \backslash\text{E}}{\text{john} \circ \text{loves} \circ \text{and} \circ \text{bill} \circ \text{hates};\ \lambda x.\mathbf{love}(x)(\mathbf{j}) \sqcap \lambda x.\mathbf{hate}(x)(\mathbf{b});\ \text{S/NP}} \quad \begin{array}{l}\text{mary};\\ \mathbf{m}; \text{NP}\end{array}}{\text{john} \circ \text{loves} \circ \text{and} \circ \text{bill} \circ \text{hates} \circ \text{mary};\ \mathbf{love}(\mathbf{m})(\mathbf{j}) \wedge \mathbf{hate}(\mathbf{m})(\mathbf{b});\ \text{S}}\ /\text{E}$$

Note in particular that this analysis assigns the right meaning to the whole sentence compositionally. ⊓ designates *generalized conjunction* (Partee and Rooth 1983), defined as follows:[6]

(29) a. For p and q of type t, $p \sqcap q =_{def} p \wedge q$
b. For $\mathcal{P}$ and $\mathcal{Q}$ of any conjoinable type other than t (where $\mathcal{P}$ and $\mathcal{Q}$ are of the same type), $\mathcal{P} \sqcap \mathcal{Q} =_{def} \lambda\mathcal{R}.\mathcal{P}(\mathcal{R}) \sqcap \mathcal{Q}(\mathcal{R})$

6. A conjoinable type is t and any functional type that "bottoms out" at t, such as $e \to t$, $t \to t$, $et \to t$, $et \to et$, etc. We illustrate generalized conjunction with some more examples in (i).

(i) a. $\mathbf{bought}_{e\to et} \sqcap \mathbf{ate}_{e\to et}$
$= \lambda x.\mathbf{bought}(x)_{et} \sqcap \mathbf{ate}(x)_{et}$ (by (29b))
$= \lambda x\lambda y.\mathbf{bought}(x)(y)_t \sqcap \mathbf{ate}(x)(y)_t$ (by (29b))
$= \lambda x\lambda y.\mathbf{bought}(x)(y) \wedge \mathbf{ate}(x)(y)$ (by (29a))

b. $\mathbf{yesterday}_{et\to et} \sqcap \mathbf{today}_{et\to et}$
$= \lambda P.\mathbf{yesterday}(P)_{et} \sqcap \mathbf{today}(P)_{et}$ (by (29b))
$= \lambda P\lambda x.\mathbf{yesterday}(P)(x)_t \sqcap \mathbf{today}(P)(x)_t$ (by (29b))
$= \lambda P\lambda x.\mathbf{yesterday}(P)(x) \wedge \mathbf{today}(P)(x)$ (by (29a))

Thus, $\lambda u.\mathbf{love}(u)(\mathbf{j}) \sqcap \lambda x.\mathbf{hate}(x)(\mathbf{b}) = \lambda w.\mathbf{love}(w)(\mathbf{j}) \wedge \mathbf{hate}(w)(\mathbf{b})$.[7]

This analysis of RNR immediately extends to DCC. (30) shows the "reanalysis" of the string *Bill the book* as a derived constituent. This involves first hypothesizing a ditransitive verb and withdrawing that hypothesis after a whole verb phrase is formed (here, VP abbreviates NP\S):

(30)
$$\dfrac{\dfrac{\dfrac{[\varphi; f; \mathrm{VP/NP/NP}]^1 \quad \mathsf{bill};\ \mathbf{b};\ \mathrm{NP}}{\varphi \circ \mathsf{bill};\ f(\mathbf{b});\ \mathrm{VP/NP}}/\mathrm{E} \quad \mathsf{the} \circ \mathsf{book};\ \iota(\mathbf{bk});\ \mathrm{NP}}{\varphi \circ \mathsf{bill} \circ \mathsf{the} \circ \mathsf{book};\ f(\mathbf{b})(\iota(\mathbf{bk}));\ \mathrm{VP}}/\mathrm{E}}{\mathsf{bill} \circ \mathsf{the} \circ \mathsf{book};\ \lambda f.f(\mathbf{b})(\iota(\mathbf{bk}));\ (\mathrm{VP/NP/NP})\backslash\mathrm{VP}}\backslash\mathrm{I}^1$$

Then, after like-category coordination, the missing verb and the subject NP are supplied to yield a complete sentence (the last step is omitted):

(31)
$$\dfrac{\mathsf{gave};\ \mathbf{gave};\ \mathrm{VP/NP/NP} \qquad \dfrac{\vdots \qquad \dfrac{\mathsf{and};\ \lambda\mathcal{W}\lambda\mathcal{V}.\mathcal{V} \sqcap \mathcal{W};\ (X\backslash X)/X \qquad \dfrac{\vdots}{\mathsf{john} \circ \mathsf{the} \circ \mathsf{record};\ \lambda f.f(\mathbf{j})(\iota(\mathbf{rc}));\ (\mathrm{VP/NP/NP})\backslash\mathrm{VP}}}{\mathsf{and} \circ \mathsf{john} \circ \mathsf{the} \circ \mathsf{record};\ \lambda\mathcal{V}.\mathcal{V} \sqcap \lambda f.f(\mathbf{j})(\iota(\mathbf{rc}));\ ((\mathrm{VP/NP/NP})\backslash\mathrm{VP})\backslash((\mathrm{VP/NP/NP})\backslash\mathrm{VP})}/\mathrm{E}}{\mathsf{bill} \circ \mathsf{the} \circ \mathsf{book} \circ \mathsf{and} \circ \mathsf{john} \circ \mathsf{the} \circ \mathsf{record};\ \lambda f.f(\mathbf{b})(\iota(\mathbf{bk})) \sqcap \lambda f.f(\mathbf{j})(\iota(\mathbf{rc}));\ (\mathrm{VP/NP/NP})\backslash\mathrm{VP}}\backslash\mathrm{E}}{\mathsf{gave} \circ \mathsf{bill} \circ \mathsf{the} \circ \mathsf{book} \circ \mathsf{and} \circ \mathsf{john} \circ \mathsf{the} \circ \mathsf{record};\ \mathbf{gave}(\mathbf{b})(\iota(\mathbf{bk})) \sqcap \mathbf{gave}(\mathbf{j})(\iota(\mathbf{rc}));\ \mathrm{VP}}\backslash\mathrm{E}$$

We now turn to some formal details about the Introduction rules in (25). First, since $\circ$ is string concatenation, if we remove the rules for the vertical slash (i.e., $\uparrow$I and $\uparrow$E) from the present fragment, the system is identical to the product-free (associative) Lambek calculus **L** (Lambek 1958). This means that, with the /I and \I rules, a hypothesis can be withdrawn as long as its prosody appears on either the left or the right periphery of the prosody of the premise.[8] Note also that in this formulation, the prosodic term

7. A note is in order regarding the meaning of conjunction. Here, we have assumed that the word *and* denotes (generalized) boolean conjunction. In chapter 5, we examine various interactions between coordination and expressions denoting "structured" objects such as "respective" readings of plurals and symmetrical and summative predicates. We revise the meaning of *and* to a tuple-forming operator there ((181) in section 5.3.1), but keep the simpler generalized conjunction outside of that chapter since our discussion in the rest of the book does not hinge directly on the analysis of plurals and related phenomena in chapter 5.

8. One might worry about overgeneration in this regard. For example, note that the infamous Dekker's puzzle arises in our setup, just as in the Lambek calculus, overgenerating the following string as a sentence:

(i) *[Bill thinks]$_{\mathrm{S/VP/NP}}$, and [the brother of]$_{\mathrm{S/VP/NP}}$, John walks.

The standard response to this issue in contemporary CG is to control the availability of associativity in the prosodic calculus by the notion of *multi-modality* (Moortgat and Oehrle 1994; Dowty 1996b; Baldridge

labeling, rather than the left-to-right order of the premises in the proof tree, is relevant for the applicability conditions of the /I and \I rules (the latter presentation is more common in the literature of mathematical linguistics; so far as we are aware, Morrill (1994) was the first to recast the Lambek calculus in the former format). This point should be clear from the proof in (27), where we have deliberately placed the hypothetical object NP to the *left* of the verb in the proof tree. This also means that the order of the two premises in the Elimination rules does not play any role, as we have already noted above. In practice, we often write premises in an order reflecting the actual word order, but it should be kept in mind that this is only for the sake of maintaining the readability of derivations.

2.5 A Note on the Linearity of the Calculus

Since the treatment of quantification via prosodic λ-binding in the present framework is very powerful and flexible, one might worry about potential overgeneration of the following kind. Suppose we hypothesize the same variable in the two conjuncts of a conjoined sentence and bind them at once after the conjoined sentence is formed:

(32) $\lambda\varphi.\varphi \circ \mathsf{is} \circ \mathsf{male} \circ \mathsf{or} \circ \varphi \circ \mathsf{is} \circ \mathsf{female};\ \lambda x.\mathbf{male}(x) \vee \mathbf{female}(x);\ \mathrm{S}\uparrow\mathrm{NP}$

Then, by lowering the quantifier *everyone*, we seem to obtain the following:

(33) $\mathsf{everyone} \circ \mathsf{is} \circ \mathsf{male} \circ \mathsf{or} \circ \mathsf{everyone} \circ \mathsf{is} \circ \mathsf{female};$
$\mathbf{V}_{\mathbf{person}}(\lambda x.\mathbf{male}(x) \vee \mathbf{female}(x));\ \mathrm{S}$

In other words, we (apparently) incorrectly predict that *Everyone is male or everyone is female* has the reading 'everyone is either male or female.'

A sign such as (32) is not derivable in Hybrid TLCG. This follows from the fact that the vertical slash (as well as the forward and backward slashes) is a variant of *linear*

2002; Muskens 2007; Kubota 2010). This will, for example, prevent the inference $\mathrm{S}/\mathrm{S} \vdash \mathrm{S}/\mathrm{VP}/\mathrm{NP}$ (valid in **L**) in the enriched calculus. Hybrid TLCG can be extended along these lines, as we discuss in chapter 11 (see also Kubota [2010, 2014] for a more detailed discussion of this general architecture of grammar enriching the morpho-phonological component).

Another possibility is to seek explanation of the ill-formedness of (i) in processing terms. That is, in order to be able to interpret this sentence, speakers are required to process *John* simultanously as the subject and *part* of the subject of a shared VP. This cognitive task is arguably something implausible for speakers to manage in the limited real-time window made available in working memory: the speaker, having encountered *Bill thinks, and*, is waiting for another expression of the same type S/S and a shared remnant, but what s/he finds instead is NP/NP followed by a complete S, very plausibly triggering backtracking/reparsing difficulties sufficient to make such examples unprocessable.

This is an important issue, but a detailed investigation is beyond the scope of the present work. We leave it for future work to determine exactly how much overgeneration to rule out in the purely combinatoric component of grammar.

implication.[9] To put it differently, the three implication connectives $/$, $\backslash$, and $\uparrow$ can bind only one occurrence of a hypothesis at a time. The basic underlying linguistic intuition is that one token of a linguistic expression can fill in only one argument position of a subcategorizing predicate. The derivation above, in which one token of the quantifier *everyone* fills in the subject argument position of two distinct verbs simultaneously, is a textbook example of a violation of resource sensitivity and hence is ruled out in the present framework (as in any other variant of CG).[10]

This of course raises the question of how to treat ATB extraction and parasitic gaps:

(34) a. John met a man who Mary likes __ but Sue hates __.
b. Which paper did John file __ without reading __?

This is indeed a nontrivial problem. The analysis of *wh* extraction by Muskens (2003) via prosodic λ-binding, while capturing elegantly the basic patterns of overt movement, does not extend to multiple-gap phenomena straightforwardly. We will address the issue of multiple gaps in chapter 7.

2.6 Computational Issues

Although Hybrid TLCG is primarily meant to be a theory of competence grammar, and, moreover, we do not take practical applicability in parsing to be one of our primary goals (which sets it apart, we believe, from the research program of CCG), considerations of computational properties are an important issue for formally explicit frameworks of grammar. Moreover, the existence of an actually implemented parser—even one that does not compete realistically with efficient parsers designed to be applicable to practical purposes—would be useful for the purpose of grammar checking and can potentially provide a quite valuable tool for working linguists. In this section, we address these computational/implementation-related issues briefly. For an accessible

9. Note that, unlike (32), the following is a well-formed sign derivable in Hybrid TLCG:

(i) $\lambda\varphi_1\lambda\varphi_2.\varphi_1 \circ \mathsf{is} \circ \mathsf{male} \circ \mathsf{or} \circ \varphi_2 \circ \mathsf{is} \circ \mathsf{female}$; $\lambda x\lambda y.\mathbf{male}(x) \vee \mathbf{female}(y)$; $\mathrm{S}\uparrow\mathrm{NP}\uparrow\mathrm{NP}$

Deriving (32) from (i) would be like deriving $A \to B$ from $A \to A \to B$ in classical propositional logic (where it is a theorem). In classical propositional logic, this is possible since using the same hypothesis A twice to cancel the two occurrences of A in the premise $A \to A \to B$ is allowed. The resource sensitivity of linear logic prohibits exactly this type of reuse of material.

10. This does not exclude a possibility in which a nonlinear semantic term is assigned as the translation for some linguistic expression (e.g., $\mathbf{V}_{\mathbf{man}}(\lambda x.\mathbf{love}(x)(x))$ for *Every man loves himself*). This is possible in TLCG since nonlinear terms can be introduced as the translations of specific lexical items (such as reflexives). By the same token, nothing blocks the nonlinear term $\mathbf{V}_{\mathbf{person}}(\lambda x.\mathbf{male}(x) \vee \mathbf{female}(x))$ from being assigned as the translation for *Everyone is either male or female*. Here, the generalized conjunction meaning for *and* is the source of nonlinearity.

introduction to the logical and computational aspects of TLCG, see Moot and Retoré (2012).

It is known that a certain restricted form of Hybrid TLCG—specifically, one which does not contain empty operators or polymorphic specifications of lexical entries—is decidable and NP complete (Moot and Stevens-Guille 2019), just like related approaches in TLCG such as Displacement Calculus (Morrill et al. 2011) and NL_λ (Barker and Shan 2015). Decidability is an important property when considering the formal computational properties of a grammatical framework, since it ensures that the search space for the parser is finite in size for any given string.

As will become clear in the following chapters, we formulate analyses of linguistic phenomena making somewhat extensive use of empty operators and polymorphic specifications. Whether these empty operators and polymorphic specifications of lexical entries can be eliminated without sacrificing too much the generality of the linguistic analyses proposed below is a somewhat delicate question. There are a couple of points worth noting in connection to this issue. In certain cases (such as in the analysis of VP ellipsis and pseudogapping in chapter 6), the empty operator we introduce can be lexicalized, at the trivial cost of increasing lexical redundancy (but this is a bad consequence only on the linguistically unreasonable assumption that the lexicon is an unstructured list of words). But there are some instances of empty operators which do not seem to lend themselves to lexicalization as easily. An example of this is the type of empty operators that we extensively rely on in our analysis of "respective" readings and related phenomena in chapter 5. In general, there does seem to be a trade-off between computational concerns and elegance of linguistic analysis in the use of empty operators. For example, in the literature of plurality in theoretical linguistics in which computational issues are not the foremost concerns, empty operators are extensively employed for issues closely related to those we address in chapter 5. While we acknowledge this to be an important issue, addressing it properly is beyond the scope of the present work. We just note here that addressing this issue properly requires first and foremost making the assumptions about the relationship between the competence grammar and a theory of linguistic performance sufficiently explicit.

Hybrid TLCG has a parser, developed by Richard Moot as a component of the LinearOne system, which is a theorem prover for "first-order" linear logic. In particular, the module (contained in the LinearOne package, which is available at https://github.com/RichardMoot/LinearOne) that translates Hybrid TLCG into first-order linear logic is specific to Hybrid TLCG, and the rest of the parsing is done as proof search in first-order linear logic. Theoretical underpinnings of the translation from Hybrid TLCG to first-order linear logic is described in detail in Moot (2014), and the repository for the parser currently contains a toy grammar that can parse some sample sentences

(mostly from the two articles on Gapping—Kubota and Levine [2012] and Kubota and Levine [2013b], which are earlier versions of chapter 3) with a brief documentation.

It is perhaps worth emphasizing here that the availability of a working parser is a significant advantage for theory development, since it gives us an indispensable tool for checking the consistency of handwritten theoretical analyses. In this respect, the similarity between Hybrid TLCG and the mainstream derivational architecture of grammar is especially noteworthy. Given the transparent correspondence between the two (in most cases), Hybrid TLCG + the LinearOne parser can, in addition to other purposes to which it can be put, be used as a practical tool for grammar checking for (the greater part of) mainstream syntax-semantics work.

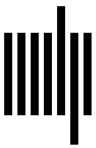
Type-Logical Syntax

By: Yusuke Kubota, Robert D. Levine

Citation:

Type-Logical Syntax
By: Yusuke Kubota, Robert D. Levine
DOI: 10.7551/mitpress/11866.001.0001
ISBN (electronic): 9780262360807
Publisher: The MIT Press
Published: 2020

The open access edition of this book was made possible by generous funding and support from Arcadia – a charitable fund of Lisbet Rausing and Peter Baldwin



The MIT Press

The open access edition of this book was made possible by generous funding from Arcadia—a charitable fund of Lisbet Rausing and Peter Baldwin.

This book was set in Syntax and Times Roman by the authors.

Library of Congress Cataloging-in-Publication Data

Names: Kubota, Yusuke, author. | Levine, Robert, 1947- author.
Title: Type-logical syntax / Yusuke Kubota, Robert D. Levine.
Description: Cambridge, Massachusetts : The MIT Press, [2016] | Includes bibliographical references and index.
Identifiers: LCCN 2020000483 | ISBN 9780262539746 (paperback)
Subjects: LCSH: Categorial grammar. | Grammar, Comparative and general–Syntax.
Classification: LCC P161 .K83 2016 | DDC 415–dc23
LC record available at https://lccn.loc.gov/2020000483

ISBN: 978-0-262-53974-6