TOWARDS A MORE LEXICAL AND FUNCTIONAL TYPE-LOGICAL THEORY OF GRAMMAR

Miltiadis Kokkonidis
University of Oxford

Proceedings of the LFG07 Conference

Miriam Butt and Tracy Holloway King (Editors)

2007

CSLI Publications

http://csli-publications.stanford.edu/

Abstract

Type-Logical Lexical Functional Grammar is a new, radically lexicalist, and formally parsimonious theory, in essence a re-incarnation of Lexical Functional Grammar (Kaplan and Bresnan, 1982) in a type-logical formal framework very similar in formal nature to that of Type-Logical Categorial Grammar (Morrill, 1994; Moortgat, 1997). It puts emphasis on having a simple logical foundation as its formal basis and no empirically unmotivated primitives, representations, and mappings between them. It differs from TLCG in basing syntactic analyses on functional rather than constituent structure, to both LFG and TLCG in that it rejects syntactic categories as primitives, and to LFG in that it rejects c-structure as a linguistically significant representation and in being radically lexicalist. The present paper presents TL-LFG, the sequence of developments that lead to it, and its key differences from LFG.

1 Introduction

Type-Logical Lexical Functional Grammar is a new radically lexicalist and formally parsimonious theory of grammar, deeply influenced by Lexical Functional Grammar (Kaplan and Bresnan, 1982), but similar in formal nature to Type-Logical Categorial Grammar (Morrill, 1994; Moortgat, 1997). Its very existence serves as a reminder that certain associations between theories and formal settings are not a necessary consequence of their respective nature. LFG is model-theoretical, but TL-LFG is not. Type-Logical Categorial Grammar, unfortunately, often goes by the name Type-Logical Grammar, but TL-LFG is a type-theoretical theory of grammar that does not have syntactic categories but grammatical functions as primitives.

TL-LFG is the outcome of a series of developments related mostly to LFG's Glue syntax-semantics interface theory (Dalrymple et al., 1993; Dalrymple, 1999, 2001). Developments in Glue have emphasised formal elegance and simplicity, and this line of development carries over to TL-LFG. TL-LFG, being in essence LFG encoded in Glue, inherits the formal simplicity and elegance of Glue. Contrasting its design with the design of LFG highlights the various redundancies and unnecessary layers in the latter.

The design of TL-LFG pushes forward the idea that a theory should only have primitives that are empirically motivated. What is immediately observable is the written or pronounced word sequence and the meaning it has. What lies in between

[†]I am thankful to Avery Andrews, Anders Soegaard, Ash Asudeh, Doug Arnold, and Ron Kaplan for commenting on ideas seen in this paper at LFG 2007 or on other occasions. Special thanks go to Mary Dalrymple, for her always encouraging and insightful comments and excellent advise. Special thanks also to the two editors, Miriam Butt and Tracy Holloway King, whose professionalism, efficiency, and kindness never seizes to amaze me. This work has been supported by AHRC grant 2005/118667.

is theory-internal and must be justified. TL-LFG, as presented here,[1] assumes the principles of the Montagovian programme for natural language semantics and the LFG functional structure primitives. These are its foundational stones and what anyone accepting the theory would have to also consider a solid basis for any further development. Given those two elements, syntactic trees and 'semantic' projections as separate levels of representation are considered redundant in TL-LFG.

LFG claims that it is a functional theory based on the fact that it has f-structure in addition to the tree structures shared by many other theories (c-structure); TL-LFG claims that it is more functional than TL-LFG because it only relies on f-structure representations. LFG claims that it is a lexical theory because it deals with certain phenomena in the lexicon rather than in terms of transformations; TL-LFG makes the claim that it is more lexical as it deals with the entire syntax in the lexicon, having no syntactic rules as a formal object as such. These claims do represent a real difference between the formalisms on some level, but at the same time the design of TL-LFG offers an opportunity for the consequences of these differences to be examined in a new light.

The beauty, from a formal perspective, of TL-LFG is that it is based on a simple logical formal framework. Given that Glue is but a small piece in the jigsaw puzzle that is the LFG formalism, it is interesting to see that its simplest version to date (Kokkonidis, 2006), appropriately used, can replace much of the formal machinery of LFG.

There are two ways in which this paper discusses how one arrives at TL-LFG. Section 2 explains how the recent developments on the Glue type-system lead to TL-LFG. Section 3 discusses the differences between TL-LFG and LFG, and how one can peel off layers of the LFG architecture to get to the core of the theory. Conclusions are drawn in Section 4.

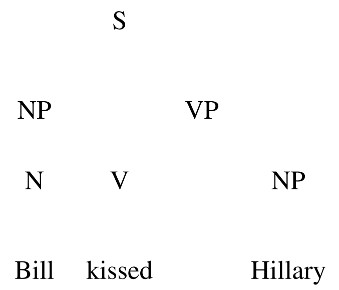
2 From Glue to TL-LFG

Lexical Functional Grammar was developed in the '70s by Ron Kaplan and Joan Bresnan (Dalrymple et al., 1995a). A remarkable fact about LFG is that in the three decades of use and development of the theory, its formal foundation has remained remarkably close to what Kaplan and Bresnan (1982) had proposed. While (sometimes significant) extensions and modifications to the theory have been proposed, the original architectural conception has by and large withstood the test of time. Moreover, the theory has been used by a diverse group of researchers that have found it particularly appealing for their line of work.

[1]In this paper the emphasis is on getting a basic LFG architecture in a type-logical setting. Argument structure, information structure, phonological structure, and morphological structure are not discussed, not because they are peripheral, nor because they are problematic, but because widening the scope of the discussion would not benefit making the basic points the paper intends to make. These are best made when a simpler LFG (closer to the original c-structure + f-structure proposal) is considered.

(1) Bill kissed Hillary.

(2)



$$f:\left[\begin{array}{ll} \textsc{pred} & p\text{: `kiss}\langle\textsc{subj}, \textsc{obj}\rangle\text{'} \\ \textsc{subj} & s\text{: }\left[\begin{array}{ll}\textsc{pred} & \text{`bill'}\end{array}\right] \\ \textsc{obj} & o\text{: }\left[\begin{array}{ll}\textsc{pred} & \text{`hillary'}\end{array}\right] \end{array}\right]$$

One area that had been problematic for some time for LFG was its syntax-semantics interface. A first difficulty lay in the fact that f-structure consists of unordered attribute-value pairs, whereas traditional compositional semantics was carried out on the nodes of trees with ordered children nodes. Although this issue had been addressed, one way or the other, early approaches did not deal in detail with the issues raised by interactions between scope and bound anaphora, or non-clausal quantification scopes arising from complex NPs and from intensional verbs with NP complements (Dalrymple, 1999). Glue was a particularly elegant proposal for the syntax-semantics interface in LFG. The developments outlined here brought enough encoding power to Glue to enable it to encode the information necessary for its purposes directly; TL-LFG is essentially Glue encoding f-structure information.

2.1 Early Glue (Dalrymple et al., 1993, 1995b)

Glue (Dalrymple, 1999, 2001) is a theory of the syntax-semantics interface based on linear logic (Girard, 1987). It was originally designed to solve the problem of f-structure-based compositional semantics for LFG. Developments in the following years were in various directions. One was the expansion of the fragment which the Glue syntax-semantics interface theory covers. Another was the expansion of the range of theories of grammar Glue was proposed as the syntax-semantics interface for: LTAG (Frank and van Genabith, 2001), HPSG (Asudeh and Crouch, 2002), CG (Asudeh and Crouch, 2001), and CFG (Asudeh and Crouch, 2001). A third direction was formal simplification. This was quite remarkable in light of the above two developments that one would assume would bring in additional requirements which in turn would necessitate enrichment after enrichment of the formalism with whatever added complexity such developments would come with.

As was the case with LFG, the foundational intuitions behind Glue have changed very little since its first appearance. In the case of LFG, changes and additions to the framework have not, overall, resulted in a simpler formal framework. Changes and additions in LFG, such as functional uncertainty, were motivated by the need to provide a means to enable the theory to deal with phenomena in a better way, so this statement is not meant as a criticism. But it is interesting to note that while

Glue analyses broadening its empirical coverage have been constantly appearing over the years, Glue followed a path constantly heading towards simplification.

The original Glue system (G0) of Dalrymple et al. (1993) was quickly superseded by the simpler system later introduced by Dalrymple et al. (1995b). The system presented there (G1) did away with G0-style rules and the use of the linear logic '!' ("of course!") modality.

The '!' modality was used in the G0 Glue system for specifying argument mapping principles such as the following:

(3) $$\begin{array}{l} !(\forall f.\, \forall X.\, \forall Y. \\ \quad (((f\ \textsc{subj})_\sigma = X) \otimes ((f\ \textsc{obj})_\sigma = Y)) \multimap \\ \quad (agent((f\ \textsc{pred})_\sigma, X) \otimes theme((f\ \textsc{pred})_\sigma, Y))) \end{array}$$

This would be used together with the semantic contributions from the three words in (1) to give its meaning.

(4)
['Bill'] :
$s_\sigma = bill,$

['kissed'] :
$\forall X.\, \forall Y.\, (agent(p_\sigma, X) \otimes theme(p_\sigma, Y) \multimap (f_\sigma = kiss(X, Y)),$

['Hillary'] :
$o_\sigma = hillary.$

While Dalrymple et al. (1993) use the '!' modality for their argument mapping principles, they also show, in a footnote, how this usage can be avoided. According to that analysis, adopted subsequently in G1, the lexical entries for the three words in (1) would simply make the following three semantic contributions in G0 with no need for argument mapping rules:

(5)
['Bill'] :
$s_\sigma = bill,$

['kissed'] :
$\forall X.\, \forall Y.\, (s_\sigma = X) \otimes (o_\sigma = Y) \multimap (f_\sigma = kiss(X, Y)),$

['Hillary'] :
$o_\sigma = hillary.$

Notice that given some f-structure f, in G0, its semantic projection f_σ is its meaning. This changes in G1. Compare (5) with (6). In G1, meanings are not assigned to semantic projections but associated with them through the '$\rightsquigarrow$' relation. Another difference was that as the mapping rules of Dalrymple et al. (1993) were abandoned by the time G1 was proposed, the '!' modality was not made part of the G1 logic. This simplified the Glue formalism considerably.

(6)

[‘Bill’] :
$s_\sigma \rightsquigarrow bill,$

[‘kissed’] :
$\forall X. \forall Y. (s_\sigma \rightsquigarrow X) \otimes (o_\sigma \rightsquigarrow Y) \multimap (f_\sigma \rightsquigarrow kiss(X, Y)),$

[‘Hillary’] :
$o_\sigma = hillary.$

In G0, semantic projections were meanings. In G1, ‘semantic projections’ were (not particularly interesting) feature structures. While in most cases, they have no internal structure and are simply empty, the ‘semantic projection’ of a noun phrases with a generalised quantifier would have a VAR and a RESTR attribute both having a feature structure that happens to be empty as their value. Given that two empty feature structures (functions from attributes to values) are always equal and two feature structures with only two empty feature structure valued features VAR and RESTR are also equal, there appears to have been a slight formal oversight in the move from G0 to G1 in this regard (Mary Dalrymple 2005, personal communication). One way to solve this problem would be to assume that for each f its semantic projection f_σ has an implicit copy of its PRED feature and that if f_σ has VAR and RESTR attributes their values are f-structures that also contain a copy of the PRED feature but also a feature VAR_OR_RESTR having the value ‘var’ and ‘rest’ respectively.[2]

The reason why G1 ‘semantic projections’ are feature structures is that Dalrymple et al. (1995b) wanted to be able to talk about a variable (entity) and a restrictor (truth value) associated with an f-structure, but did not want to introduce VAR and RESTR features in f-structure as they are semantic in nature, whereas f-structure is a syntactic structure.[3] In G0, there never was such a thing as a VAR or RESTR attribute, but there was no analysis for noun phrases containing determiners and common nouns either.[4]

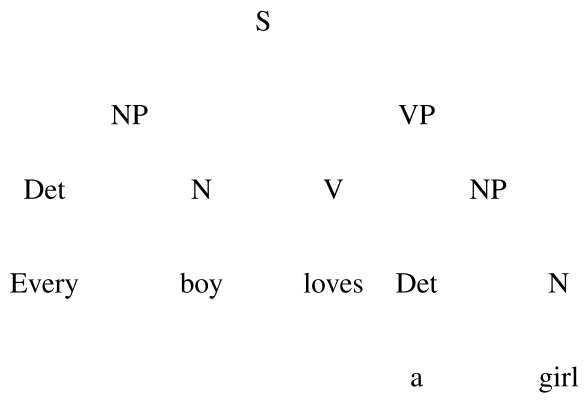
(7) Every boy loves a girl.

[2]A much simpler way of using these attributes but not G1/G2-style ‘semantic projections’ would be to have f-structures with a single VAR or RESTR feature outside their corresponding f-structure f with it as their value, instead of having VAR and RESTR as features of f. This solution was inspired by a combination of work on TL-LFG and one of the different solutions Kokkonidis (2007b) discusses for eliminating semantic projections.

[3]One could argue that the PRED features carrying an f-structure’s ‘semantic form’ also have a semantic flavour to them. Then the reason for not having VAR and RESTR attributes in the f-structure is that they are only needed by Glue; having ‘semantic projections’ as separate structures means that they only appear in Glue analyses and can be ignored by those working with other parts of LFG.

[4]Kokkonidis (2005, 2007b) demonstrates how such an analysis can be obtained without using those attributes; this analysis would have been expressible in G0 too, which would mean that the change from G0-style semantic projections to G1-style ‘semantic’ projections would not have been necessary.

(8)



$$
\text{f:}\left[\begin{array}{ll}
\text{PRED} & \text{`love}\langle\text{SUBJ, OBJ}\rangle\text{'} \\
\text{SUBJ} & \text{s:}\left[\begin{array}{ll}\text{SPEC} & \text{`every'} \\ \text{PRED} & \text{`boy'}\end{array}\right] \\
\text{OBJ} & \text{o:}\left[\begin{array}{ll}\text{SPEC} & \text{`a'} \\ \text{PRED} & \text{`girl'}\end{array}\right]
\end{array}\right]
$$

$$
f_\sigma : [\ \ ] \qquad s_\sigma : \left[\begin{array}{ll}\text{VAR} & \left[\ \ \right] \\ \text{RESTR} & \left[\ \ \right]\end{array}\right] \qquad o_\sigma : \left[\begin{array}{ll}\text{VAR} & \left[\ \ \right] \\ \text{RESTR} & \left[\ \ \right]\end{array}\right]
$$

(9) [‘every’] :
$\forall H.\,\forall R.\,\forall S.$
$(\forall X.\,((s_\sigma \text{VAR}) \rightsquigarrow_e X) \multimap ((s_\sigma \text{RESTR}) \rightsquigarrow_t R(X)))$
$\otimes$
$(\forall Y.\,((s_\sigma \rightsquigarrow_e Y)) \ \multimap (H \rightsquigarrow_t S(Y)))$
$\multimap$
$(H \rightsquigarrow_t \forall x.\, R(x) \rightarrow S(x)),$

[‘boy’] :
$\forall X.\,((s_\sigma \text{VAR}) \rightsquigarrow_e X) \multimap ((s_\sigma \text{RESTR}) \rightsquigarrow_t \mathit{boy}(X))$

[‘loves’] :
$\forall X.\,\forall Y.\,(s_\sigma \rightsquigarrow_e X) \otimes (o_\sigma \rightsquigarrow_e Y) \multimap (f_\sigma \rightsquigarrow_t \mathit{love}(X, Y)$

[‘a’] : $\forall H.\,\forall R.\,\forall S.$
$(\forall X.\,((o_\sigma \text{VAR}) \rightsquigarrow_e X) \multimap ((o_\sigma \text{RESTR}) \rightsquigarrow_t R(X)))$
$\otimes$
$(\forall Y.\,((o_\sigma \rightsquigarrow_e Y)) \ \multimap (H \rightsquigarrow_t S(Y)))$
$\multimap$
$(H \rightsquigarrow_t \exists y.\, R(y) \wedge S(y)),$

[‘girl’] :
$\forall Y.\,((o_\sigma \text{VAR}) \rightsquigarrow_e Y) \multimap ((o_\sigma \text{RESTR}) \rightsquigarrow_t \mathit{girl}(Y))$

2.2 G2: The First Type-logical Glue System (Dalrymple et al., 1997)

A further development was placing Glue on a type-logical setting with System F (Girard, 1989) as its basis (Dalrymple et al., 1997). The type-theoretic notation of this new system, G2, was neater, more concise and more readable than the notation of it predecessor, G1. Although, originally introduced in an effort to relate Glue to Categorial Grammar approaches, its popularity grew quickly to the point of replacing G1.

(10)

[‘every’] :
“$\lambda R.\, \lambda S.\, \forall x.\, S(x) \rightarrow R(x)$” : $\forall H.\, ((s_\sigma \text{VAR})_e \multimap (s_\sigma \text{RESTR})_t) \otimes (s_{\sigma e} \multimap H) \multimap H$

[‘boy’] :
“$\lambda x.\, boy(x)$” : $(s_\sigma \text{VAR})_e \multimap (s_\sigma \text{RESTR})_t$

[‘loves’] :
“$\lambda(x, y).\, loves(x, y)$” : $s_{\sigma e} \otimes o_{\sigma e} \multimap f_{\sigma t}$

[‘a’] :
“$\lambda R.\, \lambda S.\, \exists y.\, S(y) \wedge R(y)$” : $\forall H.\, ((o_\sigma \text{VAR})_e \multimap (o_\sigma \text{RESTR})_t) \otimes (o_{\sigma e} \multimap H) \multimap H$

[‘girl’] :
“$\lambda y.\, girl(y)$” : $(o_\sigma \text{VAR})_e \multimap (o_\sigma \text{RESTR})_t$

At the core of the type-logical approach to the syntax-semantics interface is the Curry-Hoard isomorphism (Howard, 1980) linking logics to the λ-calculus and type systems. The original Curry-Howard isomorphism was between proofs in intuitionistic logic and (well-typed) λ-terms of the simply-typed λ-calculus. The simply-typed λ-calculus has a type system that mirrors propositional intuitionistic logic. The G2 type system mirrors a higher-order logic with two sorts (e and t) but with various restrictions on quantification.

Using the terminology of the type-logical setting, the core idea behind the Glue theory of the syntax-semantics interface is that each atomic semantic contribution is assigned an appropriate syntax-semantics interface type. Given a word sequence, each typed atomic semantic contribution it makes is picked up and placed into Γ, the Glue typing context for that word sequence. A Glue implementation, in turn, finds all distinct (up to α-equivalence) normal-form terms M that have the target syntax-semantics type T for the word-sequence:

$$\Gamma \vdash M : T$$

(where Γ and T are given, and M is one of a number of possible compositions of type T of the atomic meanings in Γ).

2.3 First-Order Glue (Kokkonidis, 2007b)

Kokkonidis (2007b) proposed a first-order system (G3). The design of G3 does not rely on the ad-hoc restrictions and extensions Dalrymple et al. (1997) placed on System F to obtain G2; it is exactly what it appears to be: a first-order linear type system. While being formally simpler, G3 is also significantly more powerful than its predecessor due to its ability to encode arbitrarily complex hierarchical structures (using functions in the syntax for individuals).

The first step towards TL-LFG, however, comes from the alternative analyses of common nouns Kokkonidis (2007b) proposed that did not use VAR and RESTR attributes. These attributes were the most prominent example of some degree of structure in the so-called "semantic projections" that came with G1 and G2. These attributes could have been included in the f-structure but they were considered semantic in nature, therefore foreign to f-structure. Without internal structure, G1/G2-style "semantic projections" had no reason for existence. If f-structures were used directly, not only would the formalism be conceptually simpler, but the formal problem of non-uniqueness of 'semantic structures' mentioned earlier would have also been avoided. There would still be a formal complication as f-structures are complex formal objects that are not related directly to what the syntax of first-order logic individuals describes. This is why Kokkonidis (2007b) proposed a mapping from f-structures to simple atomic labels. These labels are the constants that can appear in expressions that can be arguments to base types in First-Order Glue.

However, this is not the only way things can be done. TL-LFG is based on the type system of First-Order Glue and encodes f-structures in it (Kokkonidis, 2007a). The basic idea is this: there is a finite number of attributes such as SUBJ, OBJ etc. In LFG, an f-structure is a (potentially partial) function from attributes to values. For every partial function f_p there is a corresponding total function f_t such that $f_t(x) = f_p(x)$ if f_p is defined for x and $f_t(x) = \perp$ otherwise, where $\perp$ is a special element of the range of f_t not in the range of f_p. This total function can be represented as a tuple whereby each position corresponds to a particular attribute and its value is the value of the attribute. In terms of first-order logic syntax for individuals, it can be represented as an N-ary function applied to its N-arguments. For an example, the f-structure value of a CASE feature for a noun that has either accusative or dative case in a language with cases NOM, ACC, GEN, DAT would look something like this:[5]

$$\mathit{fstr}(\perp, \perp, \ldots, \overbrace{-, \beta, -, \delta}^{casemarking}, \ldots, \perp, \perp).$$

Given the commutativity of linear logic (order-insensitivity with regards to the premises), and the importance of word order in natural languages, the question of how word-order constraints are captured arises. Inspiration for an answer can readily come either from the Prolog implementation of Definite Clause Grammars

[5]The analysis of Dalrymple et al. (2006) is used here.

(difference lists) or (the option taken here) from the basic setup of chart parsing (spans) (Kokkonidis, 2007a).

2.4 Instant Glue (Kokkonidis, 2006)

An ability to express word-order constraints in terms of simple features and an ability to encode arbitrarily nested feature structures brought First-Order Glue particularly close to being able to function as the basis for a grammar formalism, rather than as just the syntax-semantics interface. However, something was missing still: unification-based underspecification.

The Instant Glue implementation of Glue was based on a simple type system ($G3_i$) that only inhabited types with normal-form λ-calculus terms (Kokkonidis, 2006). That type system was chosen as the formal foundation for TL-LFG, both because of its normal-form property, but also because it is based on unification rather than quantification.

What this made possible is worth noting. Just like its predecessors,[6] the original First-Order Glue system, G3, as defined by Kokkonidis (2007b) only has universal quantification. But even if it did include existential quantification it would not be quite what one would want as the formal foundation for a type-logical LFG.

Let us first see what cannot be expressed without existential quantification.

(11) $_0$ Every $_1$ boy $_2$ loves $_3$ a $_4$ girl $_5$

The typing context for the above example would look similar to what one gets in First-Order Glue, except that instead of s, o, f, etc.[7] being labels for f-structures they would be the actual f-structures encoded in First-Order Glue. The question is then what is the target type. In Glue, it is t_f where f is the label of a pre-built f-structure. In TL-LFG, f is not pre-constructed; it is meant to be built up as part of the concurrent syntactic analysis / semantic composition process. So there are no concrete values (except for the span and even for that in an incremental processing scenario the end point would be unknown). The natural solution would be to have existentially quantified variables as values for every attribute with an unknown value. But then, the actual value used would be subject to existential abstraction and therefore unavailable at the end of the derivation. So the entire functional syntactic analysis would just go to waste.

Unification provides a simple and elegant solution. In $G3_i$, all variables are free and equated with values, including other variables, on demand, using an assignment function that is updated throughout the course of the derivation. While the premises can be straightforwardly interpreted as having all the variables in their

[6]Existential quantification was considered as an option in the early days of Glue, and was even used in an analysis, but a dispreferred one.

[7]There would actually also be a number of intermediate structures, but that is a detail with respect to the present discussion.

$$(\multimap Intro.)$$

$$\frac{\Gamma, X : T \vdash E : T'}{\Gamma \vdash \lambda X.E : (T \multimap T')}$$

$$(\multimap Elim.)$$

$$\frac{\Gamma_1 \vdash A_1 : T'_1 \quad \ldots \quad \Gamma_N \vdash A_N : T'_N}{F : T_1 \multimap \ldots \multimap T_{N+1}, \Gamma_1, \ldots, \Gamma_N \vdash F\ A_1\ \ldots\ A_N : T_{N+1[\sigma]}}$$

$[T_{1[\sigma]}{=}T'_{1[\sigma]}, \ldots, T_{N[\sigma]}{=}T'_{N[\sigma]}$, and T_{N+1} is a base type.]

where σ is some total function
from variables to individual denoting expressions
such that for any variable V, $\sigma(V) \neq V$.

Figure 1: TL-LFG (G3$_i$) Type-Inference Rules

types implicitly universally quantified, the interpretation of the variables in the target type is a bit more open ended. Both an interpretation assuming implicit universal quantification and another one assuming implicit existential quantification are possible, and both are useful. All uninstantiated[8] variables of the target type as originally specified can be thought of as universally quantified and all instantiated ones as existentially quantified.

(12)

[‘every’] :
“$\lambda R.\, \lambda S.\, \forall x.\, S(x) \rightarrow R(x)$” : $(e_s \multimap t_s) \multimap (e_s \multimap t_\alpha) \multimap t_\alpha$

[‘boy’] :
“$\lambda x.\, boy(x)$” : $e_s \multimap t_s$

[‘loves’] :
“$\lambda(x, y).\, loves(x, y)$” : $e_s \multimap e_o \multimap t_f$

[‘a’] :
“$\lambda R.\, \lambda S.\, \exists y.\, S(y) \wedge R(y)$” : $(e_o \multimap t_o) \multimap (e_o \multimap t_\beta) \multimap t_\beta$

[‘girl’] :
“$\lambda y.\, girl(y)$” : $e_o \multimap t_o$

Kokkonidis (2007b) investigated the two opposing trends with regards to having the ‘$\otimes$’ connective (tensor) in Glue, explained to what extend Glue analyses can avoid using it, but, targeting the second-order aspect of G2, chose to take a neutral stand with regards to whether the tensor should be included or excluded. Based on that discussion, I will assume the tensor to not be necessary for the purposes of either Glue or TL-LFG. This assumption leads to a simpler system. While a version of Instant Glue that includes the tensor exists, the version without it (Figure 1) is as simple as a first-order type system for Glue gets.

[8]A variable V is instantiated iff there is an X such that $(V, X) \in \sigma^*$ and X is a non-variable.

3 Differences with LFG

TL-LFG aims to be a simpler theory than LFG. That is a rather ambiguous statement. Formal simplicity does not necessarily come with ease of expressing linguistic facts and generalisations. It has been a priority for the LFG community to have intuitive representations and ways of expressing constraints. TL-LFG tries to build on this tradition, pushing even further both formal simplicity and ease of use.

3.1 From words to meanings in TL-LFG and LFG: An architectural comparison

LFG comes with a modular architectural design, based on separate representations (projections), linked through correspondence functions. While Figure 1 does not mention all the various different projections that have been assumed in the literature, it already gives a picture of the architectural complexity of LFG as described by Dalrymple (2001) (where f-structure was the only input to semantics in the analyses presented [9] as intended originally by Kaplan and Bresnan (1982)).

TL-LFG's architecture is a much more light-weight theory. In TL-LFG there is only one intermediate layer between a sequence of words and their meanings: atomic meanings with their syntax-semantics interface types.

LFG+Glue	TL-LFG
• input: word sequence • π: a mapping from strings to c-structure. • c-structure • ϕ: a mapping from c-structure to f-structure. • f-structure • σ: a mapping from f-structure to 'semantic' structures • meanings and Glue types • output: meanings	• input: word sequence • meanings and TL-LFG types • output: meanings

Table 1: Layers in LFG+Glue and TL-LFG

[9] Of course, Dalrymple (2001) does not fail to mention the understanding that other projections could be contributing to the semantic composition process. But the simplified picture the concrete examples of LFG syntax-semantics analyses in her book present is in line with the level of detail for the comparison between TL-LFG and LFG in the present paper.

3.2 No c-structure

Whether TL-LFG has phrase-structure rules and/or Immediate Dominance / Linear Precedence rules is an interesting question. The easy answer is to say that it does not; the unificational first-order Glue type-system that is its formal basis does not include such ID/LP rules. But this does not mean TL-LFG has no way of expressing the constraints such rules are used to express.

LFG as originally presented by Kaplan and Bresnan (1982) came with the following phrase structure rules for English:

(13)

$$\mathrm{S} \rightarrow \underset{(\uparrow \textsc{subj}) = \downarrow}{\mathrm{NP}} \quad \underset{\uparrow = \downarrow}{\mathrm{VP}}$$

$$\mathrm{VP} \rightarrow \mathrm{V} \left(\underset{(\uparrow \textsc{obj}) = \downarrow}{\mathrm{NP}}\right) \left(\underset{(\uparrow \textsc{obj2}) = \downarrow}{\mathrm{NP}}\right) \left(\underset{(\uparrow (\downarrow \textsc{pcase})) = \downarrow}{\mathrm{PP}}\right) \left(\underset{(\uparrow \textsc{vcomp}) = \downarrow}{\mathrm{VP'}}\right)$$

$$\mathrm{NP} \rightarrow \underset{\uparrow = \downarrow}{\mathrm{Det}} \quad \underset{\uparrow = \downarrow}{\mathrm{N}}$$

(14) John snores.

The relevant lexicon entries for (14) assuming this old c-structure analysis together with a standard (G2) Glue analysis (with first-order logic as the semantic representation language) are:[10]

'John'	NP	$(\uparrow \textsc{pred}) = \text{`}\textsc{john}\text{'}$ $[\underline{john}] : e_{\uparrow_\sigma}$
'snores'	V	$(\uparrow \textsc{pred}) = \text{`}\textsc{snore}\ (\textsc{subj})\text{'}$ $[\lambda x.\, \underline{snore}(x)] : e_{(\uparrow \textsc{subj})_\sigma} \multimap t_{\uparrow_\sigma}$

The TL-LFG grammar that expresses this analysis consists of two lexical entries but no separate syntactic rules: all grammatical knowledge resides in the lexicon. The emphasis is on having few but effective primitives. Grammatical functions such as SUBJect and OBJect are primitives in TL-LFG and so are the semantic concepts of entity and truth value. 'John' makes a semantic contribution corresponding to a particular entity, $john$, and 'snore' one corresponding to a function taking an entity x and returning a truth value (true or false, depending on whether x is snoring or not).

[10]For the purpose of illustrating differences of the frameworks in practice, a simplistic view of syntax and semantics will be sufficient; any additional level of detail would complicate analyses at least equally for the two frameworks and, I claim, not more for TL-LFG than for LFG.

(15) $\underbrace{\text{'John'}}_{j}$ $\quad$ '$john'$' : e_j

(16) $\underbrace{s\ \text{'snores'}}_{f}$ $\quad$ '$\lambda x.\, snore(x)$' : $e_s \multimap t_f$ $\quad$ where $f = \left[\text{SUBJ} \quad s \right]$

On the left-hand side one finds a schematic representation for the ORTHOgraphy and SPAN attributes. The one for 'snores' states that its SUBJect s is expected to precede it. Note that if we take the orthography, the SVO constraint, and the meaning with its type (function from entities to truth values) as observable facts, the only appearance of a theory-specific primitive is the SUBJ feature. This schematic representation for spans possibly augmented with explicit linear-precedence constraints corresponds to LFG's linear precedence constraints.

The other point to be made here is that the specification of word-order constraints used bears some resemblance to LFG's phrase structure rules. The word-order constraint for 'snores' is closely associated with the 'S $\rightarrow$ NP VP' rule of (13) as found in early LFG work (Kaplan and Bresnan, 1982). However, it lacks any mention of syntactic categories, only being concerned with the essential facts of word-order: the subject must precede the verb 'snore'. The stipulation that the subject is an NP in LFG is redundant given the f-structure and semantic type information available, i.e. that the semantic type of the subject is e, and also its f-structure has $\perp$ as the value of its FORM feature meaning that it is not a prepositional phrase. In TL-LFG the concept of a noun phrase, just like that of a noun, is a concept definable in terms of its semantic and functional primitives.

One advantage of the TL-LFG approach is that it relieves the grammar writer from the burden of an additional layer of specification. It also provides a more abstract view of constituent structure that represents exactly what is necessary for determining the semantics. The LFG examples in this paper have been using a rather dated theory of c-structure. In more recent work Inflectional Phrases would be making their appearance, and in the works of some authors Determiner Phrases. The point is that if updating the theory of c-structure does not affect f-structure or semantic composition, c-structure is a redundant intermediate step from the word sequence input to semantics and vice versa. There are cases where updating the theory of c-structure will affect the syntax-semantics interface, namely when the grouping of words changes as this will normally mean that the semantics has to change, and it is exactly this fact that TL-LFG captures.

If the details of c-structure are not important for the syntax-semantics interface, they have no place in TL-LFG, which aims to be a minimalist theory of grammar. It has been one of the key ideas of LFG that a functional structure representation (a feature structure providing information about grammatical functions such as SUBJ and OBJ) is to be maintained in addition to a constituent structure one (a tree representing the phrase structure of the input string). It is easy to claim that TL-LFG is a more 'functional' theory than LFG because it only has f-structure as its syntactic representation.

There is substance to the claim. It was the original intent of Kaplan and Bresnan (1982) that f-structure be the sole input to semantics. This is true for TL-LFG, but not necessarily for LFG. If it were then LFG f-structure would encode all important syntactic relations Glue needs to have available. That is the case with relations such as SUBJ, OBJ, etc. but not necessarily, for instance, with modification relations. The LFG approach, of dumping adjuncts in a set feature helps keep the f-structure for a modified phrase very similar to that of the same phrase with the modification removed. The LFG approach has a positive impact with regard to complexity of grammar writing as the description of functional constraints that are not influenced by the presence of modifiers does not need to make special provisions in order to work when modifiers happen to be present. However, in LFG, convenience comes at a high price: f-structure does not encode syntactic relations relating to modification.[11] So, it encodes some grammatical relations but not all and can not be the sole input to semantics. It is able to capture the difference between 'John likes Mary' and 'Mary likes John', but not the difference between 'a fake golden gun' and 'a golden fake gun'. The current LFG view that this is acceptable is questionable, especially for a theory that claims to put emphasis on functional structure. TL-LFG is more 'functional' because its f-structure captures such syntactic relations.

Moreover, the TL-LFG analysis of scoping modification (Kokkonidis, 2007c) achieves having a sufficiently detailed f-structure representation without loosing the elegance and simplicity of LFG's f-structures. Trivial as it may seem, the key is using a more basic data-structure, lists, that unlike LFG sets, do not disregard the order in which modifiers are encountered in the input.

TL-LFG also comes with the claim that it is more 'lexical' than LFG because it does not have at its formal foundation phrase-structure rules or ID/LP rules. This is indeed true as one can see in (18), the lexical specification for 'snores' in raw TL-LFG lacking the syntactic sugar of the appealing presentation used in (16) and elsewhere in this paper. Indeed, that specification brings to mind theories such as Type-Logical Categorial Grammar where radical lexicalism reigns supreme and no phrase-structure rules as such exist. Yet can this also be said about the syntactically sugared version of TL-LFG used in (16)? Arguably, there is no separate syntactic rule as such. What is expressed on the left-hand side of the lexical entry is simply a constraint that applies to that particular lexical entry. Also the syntactic sugar for span specifications is only a way of expressing certain constraints in a more intuitive way; syntactically sugared TL-LFG is the same theory as TL-LFG,

[11]This is the case in prominent places of the LFG literature inviting criticism and solutions (Andrews and Manning; Andrews, 1993; 2004), but not a weakness of LFG as such. My impression is that when it comes to theory, there are strong voices supporting a simplified version of f-structure and more use of the inverse ϕ mapping, and when it comes to grammar engineering, LFG f-structure is much more detailed and autonomous. My criticism is directed towards LFG with insufficiently detailed f-structures. LFG-based grammar engineering (at least amongst the members of the ParGram community) tends to put the same kind of emphasis on f-structure that TL-LFG does and if one takes that version of LFG as the standard one then much of this criticism is inapplicable as such and should rather be seen as support for that approach to the role of f-structures in LFG.

much the same way as choosing not to display spans in a grammar when using a chart parser or difference lists in Prolog DCGs is a matter of presentation and convenience rather than of essence.

(17) $_0$ John $_1$ snores$_2$

(18) 'snores' $\quad$ '$\lambda x.\, snore(x)$' : $e_s \multimap t_f$

$$\text{where } f = \begin{bmatrix} \text{SPAN} & \begin{bmatrix} \text{START} & start \\ \text{END} & \nabla + 1 \end{bmatrix} \\ \text{ORTH} & \text{`snores'} \end{bmatrix},$$

$$s = \begin{bmatrix} \text{SPAN} & \begin{bmatrix} \text{START} & start \\ \text{END} & \nabla \end{bmatrix} \\ \text{ORTH} & \\ \ldots & \end{bmatrix}, \text{ and}$$

where ∇ is the current position in the word sequence.

The line of division between radical lexicalism and having phrase structure rules (or an equivalent) is pretty thin in TL-LFG. While TL-LFG follows, at the formal foundation level, the paradigm of radical lexicalism as found in, say, Type-Logical Categorial Grammar, the pretty straightforward (and familiar from chart parsing and/or Prolog DCGs) syntactic sugar that hides the underlying representation for the word sequence and positions within it allows for syntactic constraints to be expressed in a way that combines the best aspects of both CG and LFG approaches.

3.3 No 'semantic' forms and no 'semantic' projections

Developments within Glue have lead to the term 'semantic projection' being used (in G1 and G2) for rather uninteresting feature structures. Their intended use in G2 was just that they would distinguish between $e/1$ and $t/1$ base types of different f-structures. There is nothing semantic about them – different random numbers would do. Moreover, they fail to be unique as explained earlier. What so-called semantic projections were meant to do is import syntactic information of a very abstract nature (relations between distinct parts of an f-structure) into the Glue type system. In TL-LFG, there are no intermediaries; f-structures themselves are arguments to the syntax-semantics interface base-type constructors.

The incorporation of Glue into LFG meant also that the role of LFG's 'semantic forms' changed. In early LFG, 'semantic forms' had a clear syntax-semantics interface role. In current LFG with Glue, semantic constructors (the elements of the typing context with their corresponding meaning) have taken up the most essential roles of 'semantic constructions', not leaving much semantic substance to 'semantic forms'.

Investigating the role of these no-longer semantic 'semantic forms' reveals three facts: (i) they are used in relation to *syntactic* completeness and coherence;

(ii) they are used to make the f-structure containing them unique; (iii) they are used for presentation reasons. None of the above three roles has anything to do with semantics. In TL-LFG, it is clear where the semantics is specified; it is not in the functional *syntactic* structures but on the meaning side of semantic contributions (the left hand-side of the colon, the right-hand side being the syntax-semantics interface type).

Therefore it is not surprising that 'semantic forms', one of the most important concepts of LFG, is not part of TL-LFG. For presentation reasons having a feature such as ORTH seems more appropriate. The following section discusses completeness and coherence in TL-LFG and LFG; in TL-LFG the resource sensitivity of the formalism guarantees those principles without stipulation and the syntax-semantics interface types are instrumental in that. That leaves 'semantic forms' a single role in LFG, important for LFG to work, but not related to semantics: distinguishing between different f-structures. Again this is something different arbitrary numbers would achieve equally well.

TL-LFG was inspired by the elegance of the Glue syntax-semantics interface. Two pieces of formal machinery of LFG called 'semantic' ('semantic' projections and 'semantic' forms) are reducible to distinct but otherwise arbitrary numbers. The need for such formal hacks stems from the fact that f-structures and semantic projections have no direct connection to the word sequence they correspond to. In TL-LFG, f-structures have this direct connection in the form of the span feature (or an equivalent in terms of difference lists).

3.4 Completeness and Coherence

Completeness and Coherence are two very fundamental and important principles in LFG. However, these principles are not intrinsic to the formal framework. Nothing in the formal setup stops the syntactic rules of (13) from forming f-structures examples (19)–(21). There needs to be a piece of stipulation, the Completeness Principle, in order to mark example (20) as ungrammatical. There needs to be another piece of stipulation, the Coherence Principle, in order to mark example (21) as ungrammatical.

(19) John likes Mary.

(20) * John likes.

(21) John snores Mary.

While there is nothing objectionable about linguistic principles, the nature of Completeness and Coherence as additional pieces of stipulation shows that something was missing from the formal framework proper. In TL-LFG, (syntactic and semantic) Completeness and Coherence are automatically enforced due to the resource sensitivity of the Glue type system (Dalrymple et al., 1993). They are a consequence of the overall setup and type-logical formal foundation of the theory, rather than something that had to be added to it.

4 Conclusions

TL-LFG rejects the bulky formal (and theoretical) machinery LFG comes with, but not the importance LFG attaches to functional structure and functional constraints. Indeed it attaches more importance to f-structure than LFG and claims to be a more 'functional' theory as a result.

A very obvious argument in support of this claim is on the basis of TL-LFG having no c-structure representation. But if that is the case, the sceptic may wonder whether this is so simply because f-structure was turned into a kind of c-structure with features as is the case for HPSG.

If one concentrates on the structural organisation of a TL-LFG f-structure, it becomes obvious that this is not the case. TL-LFG f-structures have, in general, the same structural organisation as their corresponding LFG f-structures which in turn is quite different from that of their corresponding c-structure trees (in general f-structures are more flat).

As for what information goes into f-structures, the crucial addition to f-structure is the span attribute. Responding to a possible criticism that span information in the f-structure is a way of importing c-structure information into f-structure reveals interesting facts about both TL-LFG and LFG.

Spans relate to the word-sequence, not to any tree-structured analysis of it. They help relate the f-structure to the word-sequence in a simple and intuitive way. It is the span information that helps distinguish between the f-structures of the two occurrences of 'the' in a sentence like 'The coach praised the players'. If instead of encoding this relation in the f-structure itself, an LFG-style correspondence function was used for that purpose, there would need to be some other way of distinguishing between them. Indeed this could come from the semantics (uniqueness of 'semantic forms' for example), but it is not at all clear why this would be a better approach.

Moreover, the f-structure would have to contain this information. A correspondence function from f-structures to semantic structures would not help. This is why 'semantic forms' guaranteed to be non-equal even when their semantic content is the same are a part of f-structures in LFG. This discussion relates to why G1/G2 'semantic projections' fail to serve their purpose and why even if the obvious step of moving the 'semantic form' into the 'semantic projection' would not be a good idea.

TL-LFG, not only does not need to import c-structure information into f-structure in the guise of the SPAN feature,[12] it also keeps semantic information out of the f-structure. LFG distinguishes itself on the basis of using separate representations for linguistically different kinds of information, yet it had semantic information inside a syntactic structure. Moreover, had this not been the case, i.e. had semantic forms

[12]The main reason TL-LFG does not need to add c-structure information into its f-structures is that LFG f-structures tend to already contain enough information to distinguish between f-structures corresponding to different word sequences. This is due to the fact that LFG performs a number of checks at the level of f-structure, just like TL-LFG.

been placed inside semantic projections, in the most straightforward manner possible, the whole system would collapse because it would be unable to distinguish between f-structures that were meant to be different but in the absence of PRED features would be equal. This would be a direct consequence of doing the right thing, with respect to the projection architecture, and relying on a correspondence function rather than embedding the semantic information inside the f-structure using a feature (as is now the case).

While it was never the intent of this paper to challenge the projection architecture of LFG as such, it seems that suspiciously much depends on the PRED attributes and their 'semantic form' features inside f-structure to keep the LFG system together. One important role they play is in setting the subcategorisation requirements for Completeness and Coherence. In many ways, the f-structure PRED features have a syntax-semantics interface role. However, unlike the case with TL-LFG, in LFG there is nothing in the formal foundation of the theory that guarantees the Completeness and Coherence principles. In TL-LFG the resource-sensitive type-logical formal foundation of the theory does exactly that without further stipulation.

Returning to the claim that TL-LFG is more 'functional', the argument that this is so because c-structure disappears has a certain immediate appeal, but the essence is in examining the role and function of f-structure in the two theories. In TL-LFG it is there to capture any and all grammatical relations that would be important for the semantics; in LFG it captures some but not all of them.

Traditional phrase structure rules and the immediate dominance part of immediate dominance / linear precedence rule system are not a part of LFG and neither is the syntactic category system of LFG. Linear precedence rules are. In TL-LFG, all grammatical knowledge resides in the lexicon which makes it more 'lexical' than LFG. However, a bit of TL-LFG syntactic sugar hides low-level details of spans and gives a way of specifying spans and linear precedence constraints in an intuitive manner. To the extent that such constraints can be factored out of the lexicon TL-LFG could be seen as having rules and even constructional meaning. The point is that this is more a matter of presentation and convenience than theoretical essence.

Neither being more 'functional' nor being more 'lexical' mean much in themselves. It is TL-LFG's formal simplicity and parsimony combined with some of the best aspects of LFG that give these comparisons substance. Starting with the syntax-semantics interface and then building the details of the syntax based on a very successful theory lead to a re-incarnation of that theory in a different formal setting which was but a small fragment of the original theory's formal arsenal. Not only is the formal framework now simpler, but so is the conceptual framework: accounting for the facts involves fewer theory-internal concepts and representations, something achieved without complicating the part of the original theory preserved in the new theory. Finally, the new type-logical formal framework captures linguistic intuitions that the original framework left to stipulation.

References

Andrews, A. D. and Manning, C. D. 1993. Information-spreading and levels of representation in LFG. Technical Report CSLI-93-176, CSLI, Stanford University.

Andrews, Avery D. 2004. Glue Logic vs. Spreading Architecture in LFG. In Christo Mostovsky (ed.), *Proceedings of the 2003 Conference of the Australian Linguistics Society*.

Asudeh, Ash and Crouch, Richard. 2001. Glue semantics: A general theory of meaning composition. Talk given at Stanford Semantics Fest 2, March 16, 2001.

Asudeh, Ash and Crouch, Richard. 2002. Glue semantics for HPSG. In Frank van Eynde, Lars Hellan and Dorothee Beermann (eds.), *Proceedings of the 8th International HPSG Conference*, Stanford, CA., CSLI Publications.

Dalrymple, Mary (ed.). 1999. *Semantics and Syntax in Lexical Functional Grammar: The Resource Logic Approach*. MIT Press.

Dalrymple, Mary. 2001. *Lexical Functional Grammar*. Syntax and Semantics Series, No. 42, Academic Press.

Dalrymple, Mary, Gupta, Vineet, Pereira, Fernando C.N. and Saraswat, Vijay. 1997. Relating Resource-based Semantics to Categorial Semantics. In *Proceedings of the Fifth Meeting on Mathematics of Language (MOL5)*, Schloss Dagstuhl, Saarbrücken, Germany, an updated version was printed in (Dalrymple, 1999).

Dalrymple, Mary, Kaplan, Ronald M., Maxwell, III, John T. and Zaenen, Annie (eds.). 1995a. *Formal Issues in Lexical-Functional Grammar*. Stanford, CA: CSLI Publications.

Dalrymple, Mary, King, Tracy Holloway and Sadler, Louisa. 2006. Indeterminacy by underspecification. Poster presented at the LFG06 Conference.

Dalrymple, Mary, Lamping, John, Pereira, Fernado C.N. and Saraswat, Vijay. 1995b. A deductive account of quantification in LFG. In Kanazawa Makoto, Christopher J. Pinón and Henriette de Swart (eds.), *Quantifiers, Deduction and Context*, Center for the Study of Language and Information, Stanford, California.

Dalrymple, Mary, Lamping, John and Saraswat, Vijay. 1993. LFG semantics via constraints. In *Proceedings of the Sixth Meeting of the European ACL*, pages 97–105, European Chapter of the Association for Computational Linguistics, University of Utrecht.

Frank, Anette and van Genabith, Josef. 2001. GlueTag: Linear Logic based semantics construction for LTAG — and what it teaches us about the relation between

LFG and LTAG —. In *Proceedings of the LFG01 Conference*, CSLI Publications.

Girard, Jean-Yves. 1987. Linear logic. *Theoretical Computer Science* 50, 1–102.

Girard, Jean-Yves. 1989. *Proofs and Types*. Cambridge University Press.

Howard, William A. 1980. The formulae-as-types notion of construction. In J.R. Hindley and J.P. Selden (eds.), *To H.B. Curry: Essays on combinatory logic, lambda calculus and formalism*, Academic Press, conceived in 1969. Sometimes cited as Howard (1969).

Kaplan, Ronald M. and Bresnan, Joan. 1982. Lexical Functional Grammar: A formal system for grammatical representation. In Joan Bresnan (ed.), *The Mental Representation of Grammar Relations*, pages 173–281, MIT Press.

Kokkonidis, Miltiadis. 2005. Why glue your donkey to an f-structure when you can constrain and bind it instead? In Miriam Butt and Tracy Holloway King (eds.), *Proceedings of the LFG05 Conference*, CSLI Publications.

Kokkonidis, Miltiadis. 2006. A Simple Linear First-Order System for Meaning Assembly. In *Proceedings of the Second International Congress on Tools for Teaching Logic*, Salamanca, Spain.

Kokkonidis, Miltiadis. 2007a. Encoding LFG f-structures in the TL-LFG type system. In *Proceedings of the Second International Workshop on Typed Feature Structure Grammars*, Tartu, Estonia.

Kokkonidis, Miltiadis. 2007b. First-Order Glue. *Journal of Logic, Language and Information* To appear in print. DOI: 10.1007/s10849-006-9031-0.

Kokkonidis, Miltiadis. 2007c. Scoping and Recursive Modification in Type-Logical Lexical Functional Grammar. In *Proceedings of the 12th Conference on Formal Grammar*, to appear.

Moortgat, Michael. 1997. Categorial Type Logics. In Johan van Benthem and Alice ter Meulen (eds.), *Handbook of Logic and Language*, Elsevier.

Morrill, Glyn. V. 1994. *Type Logical Grammar: Categorial Logic of Signs*. Dordrecht: Kluwer.