Constraining Scope Ambiguity in LFG+Glue

Matthew Gotham
University of Oxford

Proceedings of the LFG'19 Conference

Australian National University

Miriam Butt, Tracy Holloway King, Ida Toivonen (Editors)

2019

CSLI Publications

pages 111–129

http://csli-publications.stanford.edu/LFG/2019

Keywords: scope, quantification, Glue semantics

Gotham, Matthew. 2019. Constraining Scope Ambiguity in LFG+Glue. In Butt, Miriam, King, Tracy Holloway, & Toivonen, Ida (Eds.), *Proceedings of the LFG'19 Conference, Australian National University*, 111–129. Stanford, CA: CSLI Publications.

Abstract

A major strength of the Glue approach to semantic composition for LFG is that it accounts for quantifier scope ambiguity without the need for additional assumptions. However, quantifier scope is more rigid in some languages and constructions than Glue would lead us to expect. I propose a mechanism for constraining scope ambiguity in LFG+Glue, based on ideas taken from Abstract Categorial Grammar. Unlike existing proposals, this account does not depend on representational constraints on linear logic derivations or meaning representations.

1 Introduction

1.1 Scope ambiguity

Famously, sentences like (1) are ambiguous in English between a 'surface scope' and an 'inverse scope' interpretation.

(1) A police officer guards every exit.

The surface scope interpretation can be paraphrased as 'there is a police officer who guards every exit', and is represented logically in (2-a). The inverse scope interpretation can be paraphrased as 'every exit is guarded by a police officer', and is represented logically in (2-b).

(2) a. $\exists x.\text{officer}'x \wedge \forall y.\text{exit}'y \rightarrow \text{guard}'xy$
b. $\forall y.\text{exit}'y \rightarrow \exists x.\text{officer}'x \wedge \text{guard}'xy$

Pre-theoretically, we can refer to the ambiguity of (1) under consideration as 'scope ambiguity'. The surface scope interpretation is so called because the order of quantificational expressions on the surface matches the order of quantifers in the interpretation, and *mutatis mutandis* for the inverse scope interpretation. A question that immediately arises is whether or not this ambiguity corresponds to any distinction in possible syntactic representations or derivations of (1), or, more generally and succinctly:

Q Is quantifier scope ambiguity syntactic ambiguity?

Montague (1973) famously answered 'yes' to **Q**, as indeed the categorial grammar perspective forces one to. Cooper (1983) answered 'no', and explicitly criticized Montague on those grounds. Since May (1985), mainstream Chomskyan syntacticians and semanticists have tended to side with Montague (on this issue!), viewing (1) as having two different representations at a level of syntax known as

[†]Thanks to Mary Dalrymple, Jamie Findlay, John Payne, Adam Przepiórkowski and the audiences at LFG'19 and SE-LFG28 for helpful comments and discussion. This research is supported by an Early Career Fellowship from the Leverhulme Trust.

Logical Form (LF), each corresponding to one of the interpretations in (2). But note that the question of the existence of LF, and the connected debate over 'direct compositionality' (Jacobson, 2002; Barker & Jacobson, 2007), is not the same as **Q**. The systems for handling quantifier scope ambiguity outlined by Barker & Shan (2014) and Jacobson (2014) are both directly compositional, and yet they postulate a difference in syntactic derivation to account for the two different interpretations of (1).

Whenever **Q** has been asked in an LFG context, it has been answered in the negative. For example, the difference between interpretations (2-a) and (2-b) is not taken to correspond to any difference in the structure of (1) at any syntactic level of description. Instead, in the framework of Glue semantics, lexical and syntactic information from a sentence contribute a collection of linear logic premises, and every interpretation the sentence has corresponds to a different proof constructible from those premises.

In the case of (1), the premises contributed are shown in (3), both before and after instantiation based on its f-structure as shown in Figure 1.[1,2]

$$f:\begin{bmatrix}\textsc{pred} & \text{'guard'} \\ \textsc{subj} & g:\begin{bmatrix}\textsc{pred} & \text{'police officer'} \\ \textsc{spec} & i:\begin{bmatrix}\textsc{pred} & \text{'a'}\end{bmatrix}\end{bmatrix} \\ \textsc{obj} & h:\begin{bmatrix}\textsc{pred} & \text{'exit'} \\ \textsc{spec} & j:\begin{bmatrix}\textsc{pred} & \text{'every'}\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

Figure 1: F-structure of (1)

(3) $a \rightsquigarrow \lambda P.\lambda Q.\exists x.Px \wedge Qx : ((\textsc{spec}\uparrow) \multimap \uparrow) \multimap (((\textsc{spec}\uparrow) \multimap \%A) \multimap \%A)$

$\%A = ((\textsc{path}\uparrow)\,\textsc{spec}^*)$

$\Rightarrow \lambda P.\lambda Q.\exists x.Px \wedge Qx : (g \multimap i) \multimap ((g \multimap \%A) \multimap \%A)$

police officer $\rightsquigarrow$ officer$' : \uparrow \multimap (\uparrow \textsc{spec})$

$\Rightarrow$ officer$' : g \multimap i$

guards $\rightsquigarrow$ guard$' : (\uparrow \textsc{subj}) \multimap ((\uparrow \textsc{obj}) \multimap \uparrow)$

$\Rightarrow$ guard$' : g \multimap (h \multimap f)$

[1] I have chosen to define meaning constructors based on f-structure instead of s-structure, which is more common in the Glue literature. This choice has been made for simplicity of exposition and does not affect the argument at any point.

[2] Throughout this paper, expressions of the lambda calculus are given in $\beta\eta$-normal form. So for example, officer$' \equiv_\eta \lambda x.$officer$'x$ and guard$' \equiv_\eta \lambda x.\lambda y.$guard$'xy$.

$$\textit{every} \rightsquigarrow \lambda P.\lambda Q.\forall y.Py \rightarrow Qy : ((\text{SPEC} \uparrow) \multimap \uparrow) \multimap (((\text{SPEC} \uparrow) \multimap \%B) \multimap \%B)$$
$$\%B = ((\text{PATH} \uparrow)\ \text{SPEC}^*)$$
$$\Rightarrow \lambda P.\lambda Q.\forall y.Py \rightarrow Qy : (h \multimap j) \multimap ((h \multimap \%B) \multimap \%B)$$
$$\textit{exit} \rightsquigarrow \text{exit}' : \uparrow \multimap (\uparrow \text{SPEC})$$
$$\Rightarrow \text{exit}' : h \multimap j$$

The proofs corresponding to surface scope and inverse scope interpretations are then shown in Figures 2 and 3 respectively.[3] Note that the lexical entries for the two determiners include local names ($\%A$ and $\%B$ respectively) specified by an inside-out functional uncertainty (IOFU) PATH.[4] For the time being we can take PATH to be maximally unconstrained, i.e. GF*. In any case, the only sensible value for either $\%A$ or $\%B$ in Figure 1 is f, and this is reflected in the two proofs.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[x : g]^1 \quad \text{guard}' : g \multimap (h \multimap f)}{\text{guard}'x : h \multimap f} \quad \dfrac{\text{exit}' : h \multimap j \quad \text{every}' : (h \multimap j) \multimap ((h \multimap f) \multimap f)}{\text{every}'\text{exit}' : (h \multimap f) \multimap f}}{\text{every}'\text{exit}'(\text{guard}'x) : f}}{\lambda x.\text{every}'\text{exit}'(\text{guard}'x) : g \multimap f}\ 1 \quad \dfrac{\text{officer}' : g \multimap i \quad \text{a}' : (g \multimap i) \multimap ((g \multimap f) \multimap f)}{\text{a}'\text{officer}' : (g \multimap f) \multimap f}}{\text{a}'\text{officer}'(\lambda x.\text{every}'\text{exit}'(\text{guard}'x)) : f}}{\equiv \exists x.\text{officer}'x \wedge \forall y.\text{exit}'y \rightarrow \text{guard}'xy : f}$$

Figure 2: Proof for the surface scope interpretation of (1)

It is a major attraction of the Glue approach to semantic composition that it accounts for scope ambiguity like this, without the need for any additional assumptions such as are required by all the other theories already mentioned. However, that advantage can be seen as a disadvantage in cases where we do not see the same scope ambiguity as in (1).

[3] These proofs have been pared down in two ways to save space:

- The meaning representations of the two quantifiers have been written as a′ and every′, and only expanded in the final step.
- The inferential steps are not fully annotated. No annotation at all means that $\multimap$ elimination has been applied, and the number n means that $\multimap$ introduction has been applied, discharging the hypothesis numbered n.

[4] I follow the suggestion of Andrews (2010) and use IOFU, rather than quantification in the linear logic fragment, to specify scope level. The reason is mainly for cleanness of presentation, given that quantification will be introduced for other reasons later, although a potential advantage of this approach emerges in Section 1.3.

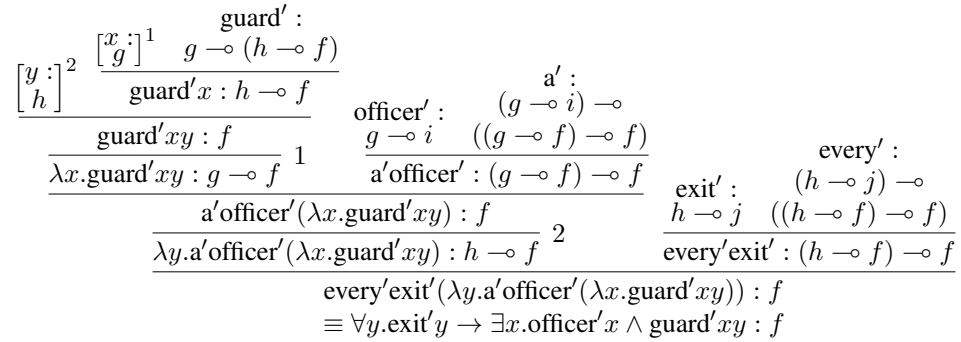

Figure 3: Proof for the inverse scope interpretation of (1)

1.2 Scope rigidity

To take just one example, the German translation of (1) given in (4) does not show the same ambiguity as (1). Specifically, it **only** has the surface scope interpretation (2-a).

(4) Ein Polizist bewacht jeden Ausgang.
a.NOM police officer guards every.ACC exit

This is a general property of German transitive clauses where the subject precedes the object, although not of clauses where the object scrambles over the subject. For example, (5) constrasts with (4) in not being scope-rigid. The argument structure is the same (in particular, exits are still being guarded) but both the surface scope (2-a) and the inverse scope (2-b) interpretation are possible again.

(5) Jeden Ausgang bewacht ein Polizist.
every.ACC exit guards a.NOM police officer

For reference, partial c-to-f-structure mappings for (4) and (5) are shown in Figures 4 and 5 respectively.

In fact, we do not have to go outside English to find cases where quantifier scope is more rigid than Glue would lead us to expect. For example, quantifier scope is 'frozen' in double object constructions: (6) **only** has the interpretation represented in (6-a), not that represented in (6-b).

(6) Hilary gave a student every grade.
a. $\Rightarrow \exists y$.student′$y \wedge \forall x$.grade′$x \rightarrow$ give′hilary′xy
b. $\not\Rightarrow \forall x$.grade′$x \rightarrow \exists y$.student′$y \wedge$ give′hilary′xy

The purpose of this paper is to describe an extension to Glue semantics that will enable us to account both for the ambiguity of sentences like (1) and (5), and for the non-ambiguity of sentences like (4) and (6). But first, I need to set aside a potential red herring.

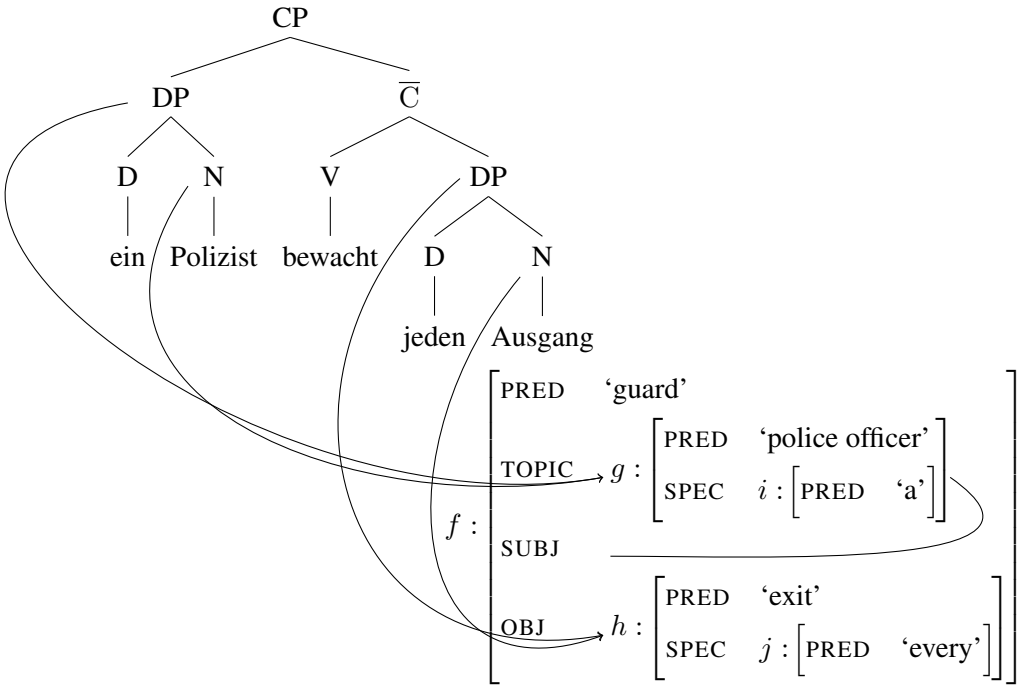

Figure 4: C- to f-structure of (4)

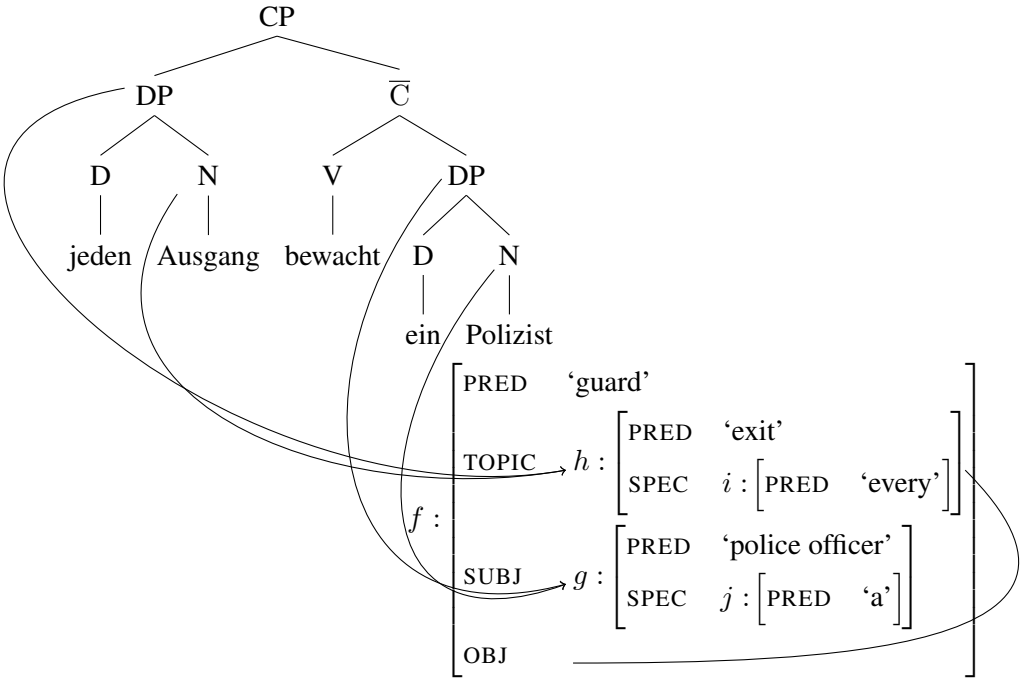

Figure 5: C- to f-structure of (5)

1.3 Not scope 'islands'

There are other examples of scope non-ambiguity that are not so hard to account for in Glue. For example, (7) only has the interpretation shown in (7-a), not that given in (7-b).

(7) If every student passes, the lecturer will be happy.
   a. $\Rightarrow (\forall y.\text{student}'y \rightarrow \text{pass}'y) \rightarrow \text{happy}'(\imath x.\text{lecturer}'x)$
   b. $\not\Rightarrow \forall y.\text{student}'y \rightarrow (\text{pass}'y \rightarrow \text{happy}'(\imath x.\text{lecturer}'x))$

That is to say, in the interpretation of (7), 'every' cannot take wider scope than 'if'.

$$f:\begin{bmatrix}\text{PRED} & \text{'happy'}\\ \text{SUBJ} & [\text{"the lecturer"}]\\ \text{ADJ} & \left\{ g:\begin{bmatrix}\text{PRED} & \text{'pass'}\\ \text{COMPFORM} & \text{if}\\ \text{SUBJ} & h:\begin{bmatrix}\text{PRED} & \text{'student'}\\ \text{SPEC} & i:\begin{bmatrix}\text{PRED} & \text{'every'}\end{bmatrix}\end{bmatrix}\end{bmatrix}\right\}\end{bmatrix}$$

Figure 6: F-structure of (7)

An explanation of this datum becomes obvious once we inspect the f-structure of (7) given in Figure 6 in combination with the lexical semantics of *every* as assumed in (3). To get the unavailable interpretation (7-b), the local name $\%B$ would have to resolve to f. This can be ruled out by making PATH suitably constrained—by not allowing it to pass through ADJ, for example.[5] $\%B$ would then have to resolve to g, which would allow for the derivation of (7-a) but not (7-b), as desired.

However, this kind of explanation is not available in the cases we're interested in. Take (4), the f-structure of which is shown in Figure 4, and assume that the meaning constructor for *jeden* is the same as for *every*. In Figure 4, as in Figure 1, the only way to get an interpretation of **either** the surface **or** (empirically unavailable) inverse scope interpretation is for $\%B$ to resolve to f. (4) and (6) are both examples of intra-clausal scope rigidity, and so in these cases the unwanted readings cannot be ruled out by constraining IOFU paths.

2 Previous work

I am aware of two existing proposals for constraining scope ambiguity in LFG+Glue: Crouch & van Genabith (1999) and Cook & Payne (2006).

[5] As hinted in Footnote 4, this explanation depends on the use of IOFU, rather than linear logic quantification, to fix scope level. If quantification is preferred for this purpose, an alternative explanation of scope islands could be to use extra modalities in the linear logic fragment, as in Gotham (2017).

2.1 Node orderings

According to Crouch & van Genabith (1999), in addition to meaning constructors, the linguistic system may also contribute certain admissibility conditions on linear logic proofs. Proofs failing to meet those conditions would then in some sense be filtered out. For example, the unavailable inverse scope interpretation of (4) could be ruled out by giving the main verb the (partial) lexical entry shown in (8).[6]

(8) *bewacht* V

$(\uparrow \text{PRED}) = \text{'guard'}$
$\text{guard}' : (\uparrow \text{SUBJ}) \multimap ((\uparrow \text{OBJ}) \multimap \uparrow)$
$\boxed{(\uparrow \text{SUBJ}) <_f (\uparrow \text{OBJ}) \Rightarrow (\uparrow \text{SUBJ}) \succ (\uparrow \text{OBJ})}$

The consequent of the boxed constraint in (8) is a **node ordering**. The intended interpretation of it is that in every licit linear logic derivation, the node labelled $(\uparrow \text{SUBJ})$ must be ordered higher than that labelled $(\uparrow \text{OBJ})$. Simplifying somewhat, for a linear logic formula x to be ordered higher than another y in a derivation d means that no instance of y occurs strictly lower down in d than every instance of x (Crouch & van Genabith, 1999, 132). So for example, if we apply (8) to Figure 4 or 5, we end up with

$$g <_f h \Rightarrow g \succ h$$

Since g f-precedes h in Figure 4, we then look at the candidate derivations in Figures 2 and 3.[7] We then see that the derivation in Figure 3 violates the constraint: there is an instance of h occurring strictly lower down than every instance of g. Therefore, this derivation is ruled out by the node ordering, and scope rigidity in enforced.

2.2 Objections to node orderings

2.2.1 Reifying representations?

As mentioned above, node orderings are constraints on derivations. But what, exactly, is a derivation? To inspect (8), in which the node ordering appears in a constraining equation, it seems that derivations have to be treated as levels in the projection architecture like c- or f-structure and that, just as with c- or f-structure, we can impose properly linguistic constraints on that structure:

> A derivation is a tree-like structure of sequents [...] Represent derivations $\mathcal{D}$ as triples $\langle S, >_S, \$ \rangle$ where S is the set of points in the tree,

[6] I have had to slightly reconstruct the proposal of Crouch & van Genabith (1999) around other assumptions made in this paper. I do believe that this is a fair representation of their proposal.

[7] These are candidate derivations for (4) just as much as for (1) because the f-structure of (4) (Figure 4) does not differ from that of (1) (Figure 1) in any way that affects the contribution of meaning constructors.

> $>_S$ is a transitive , asymmetric ordering over them, and \$ is a function mapping the points onto their corresponding sequents. (Crouch & van Genabith, 1999, 131)

But from a logical and conceptual perspective this is deeply unsatisfactory, since it ties our use of the logic to a particular format for writing proofs out: namely, one with this kind of 'tree-like structure'. But, as Corbalán & Morrill (2016, fn. 4, emphasis mine) put it,

> Gentzen calculus, labelled and unlabelled natural deductions, proof nets, categorical calculus, etc. are all of repute, all have their respective advantages and disadvantages, and are **all notations for the same theory**.

Put differently, natural deduction derivations are representations of proofs, not the proofs themselves. The definition that Crouch & van Genabith (1999) provide requires us to write out proofs in a particular natural deduction format and not, for example, in sequent calculus or with proof nets.

Now, it should be pointed out that in all likelihood an equivalent notion of node ordering could be defined for all these other proof formats.[8] The real issue, though, is that if we have properly linguistic constraints on the form of derivations, we are not really working with proofs and therefore not doing logic any more. This clashes with an appealing (to me at least) picture according to which linear logic inference takes linguistic input and then operates entirely on its own terms, without the linguistic system needing to see the internal structure of any inferences.

2.2.2 Generate and filter?

Even if one sees no problem with treating derivations as linguistic objects in this way, one might well view node orderings with with some suspicion. The boxed clause in (8), for example, does not function like a normal constraining equation. There is no minimal linear logic derivation against which it has to be checked, on pain of failure—if there were, there would be no scope ambiguity! Instead, as mentioned above, the picture is that all logically-acceptable derivations are produced, but then (potentially) some are discarded.

What we need for a Glue-based theory of scope rigidity is a method for assigning linear logic formulae to lexical items such that all and only the desired interpretations of a sentence have a corresponding proof, rather that filtering out proofs by non-logical means based on derivations. Section 3 presents such a method.

[8] In fact, I have done just this in that the proof format used in this paper is not exactly that used by Crouch & van Genabith (1999).

2.3 Partition of meaning representations

In the theory of Cook & Payne (2006), topicality is not a grammatical function but rather a binary-valued f-structure attribute $\pm$T. Along with $\pm$N(ew) and $\pm$C(ontrastive), it constitutes the presence of information structure in f-structure. Word order in German is determined by the interaction of Optimality-Theoretic constraints referencing this information structure, as well as the GF hierarchy. The upshot of those constraints for a transitive clause is that, if the subject is $+$T and the object is $-$T then either of (4) or (5) is a possible word order, depending on other factors. But if the object is $+$T and the subject is $-$T, then only the scrambled order in (5) is possible. On the side of interpretation, there is a constraint to the effect that $+$T constituents must take scope over $-$T constituents; together, these constraints predict that (5) is scopally ambiguous but (4) is scope-rigid.

When we look at the implementation of that constraint on interpretation, we see that it is similar in spirit to Crouch & van Genabith's (1999) proposal. Instead of a constraint on the form of derivations, it is a constraint on the form of meaning representations. The constraint, called a partition, is shown in (9).

(9) $+\text{T}(-\text{T})$

This states that, in the final meaning representation of the clause, the meaning of any $-$T constituent must be contained in something that the meaning of a $+$T constituent is applied to. Given that the word order produced in (4) requires that a′officer′ be associated with $+$T and every′exit′ with $-$T, this predicts that (10-a) ($\equiv$ (2-a)) is a permitted meaning representation, but (10-b) ($\equiv$ (2-b)) is not.[9]

(10) a. a′officer′$(\lambda x.$every′exit′$(\text{guard}'x))$
b. every′exit′$(\lambda y.$a′officer′$(\lambda x.\text{guard}'xy))$

2.4 Objections

While Crouch & van Genabith's (1999) proposal reifies representations of proofs, Cook & Payne's (2006) proposal reifies representations of meanings. As Montague (1973) stressed, the step of translating natural language into a logical language like the lambda calculus, which in turn is interpreted in a model, should in principle be dispensable. The model-theoretic interpretations of (10-a) and (10-b) do not have any internal structure that could be used to distinguish them in the way envisaged in (9).

An alternative interpretation of Cook & Payne's (2006) proposal is as a constraint not on meaning representations but on the final step of how they are put together, given that application corresponds to $\multimap$ elimination by the Curry-Howard correspondence. Interpreted like that, it would amount to essentially the same thing as Crouch & van Genabith's (1999) proposal, as it would tie us to a proof format

[9] Cook & Payne (2006) in fact use quite different meaning representations, but that does not affect the substance of the argument.

that uses introduction and elimination rules (like natural deduction) rather than, say, left and right rules like sequent calculus.

3 A theory of scope rigidity

In general terms, the proposed account of scope rigidity proceeds as follow. First, we expand the fragment of linear logic used such that f-structure nodes are linear logic predicates (not formulae). We then use the arguments to those predicates to keep track of the order of application of quantifiers. Finally, in the lexicon, we set things up so that only by applying quantifiers in the desired order can a valid proof be constructed. This approach is inspired by work in Abstract Categorial Grammar (Pogodalla & Pompigne, 2012; Kanazawa, 2015).

3.1 Linear logic fragment

Given a set **P** of predicates (f-structure nodes) and a set **V** of variables (for which I wil use Greek letters), the fragment of linear logic used is as defined in (11).

(11)
$$\begin{array}{rcll} n & ::= & \mathbf{V} \mid 0 \mid \mathsf{s}n & \text{(terms)} \\ \phi, \psi & ::= & \mathbf{P}n \mid \phi \multimap \psi \mid \forall \mathbf{V}.\phi & \text{(formulae)} \end{array}$$

Formally, for much of what we want this fragment to do s can be an unanalyzed function symbol. However, for readability, in what follows we will treat it as the successor function and represent $\mathsf{s}0$ as 1, $\mathsf{s}(\mathsf{s}0)$ as 2 etc.

3.2 Lexicon

We can now enforce scope rigidity for (4) by means of the mini German lexicon shown in (12), assuming the f-structure in Figure 4. In order to save space and improve readability, I have written arguments to linear logic predicates and functions in subscript, e.g. instead of writing $g0$ I write g_0.

(12)
$$\begin{array}{l} \textit{ein} \rightsquigarrow \lambda P.\lambda Q.\exists x.Px \wedge Qx : \forall \iota.((\textsc{spec} \uparrow)_0 \multimap \uparrow_0) \multimap \\ \qquad\qquad (((\textsc{spec} \uparrow)_{\mathsf{S}\iota} \multimap \%A_{\mathsf{S}\iota}) \multimap \%A_\iota) \\ \qquad \%A = ((\textsc{path} \uparrow)\ \textsc{spec}^*) \\ \quad \Rightarrow \lambda P.\lambda Q.\exists x.Px \wedge Qx : \forall \iota.(g_0 \multimap i_0) \multimap \\ \qquad\qquad ((g_{\mathsf{S}\iota} \multimap \%A_{\mathsf{S}\iota}) \multimap \%A_\iota) \\ \textit{Polizist} \rightsquigarrow \text{officer}' : \uparrow_0 \multimap (\uparrow \textsc{spec})_0 \\ \quad \Rightarrow \text{officer}' : g_0 \multimap i_0 \\ \textit{bewacht} \rightsquigarrow \text{guard}' : \forall \iota.\forall \eta.(\uparrow \textsc{subj})_\iota \multimap ((\uparrow \textsc{obj})_\eta \multimap \uparrow_\eta) \\ \quad \Rightarrow \text{guard}' : \forall \iota.\forall \eta.g_\iota \multimap (h_\eta \multimap f_\eta) \end{array}$$

$$\textit{jeden} \rightsquigarrow \lambda P.\lambda Q.\exists x.Px \wedge Qx : \forall \iota.((\text{SPEC} \uparrow)_0 \multimap \uparrow_0) \multimap (((\text{SPEC} \uparrow)_{\mathsf{S}\iota} \multimap \%B_{\mathsf{S}\iota}) \multimap \%B_\iota)$$
$$\%B = ((\text{PATH} \uparrow)\ \text{SPEC}^*)$$
$$\Rightarrow \lambda P.\lambda Q.\forall y.Py \to Qy : \forall \iota.(h_0 \multimap j_0) \multimap ((h_{\mathsf{S}\iota} \multimap \%B_{\mathsf{S}\iota}) \multimap \%B_\iota)$$
$$\textit{Ausgang} \rightsquigarrow \text{exit}' : \uparrow_0 \multimap (\uparrow \text{SPEC})_0$$
$$\Rightarrow \text{exit}' : h_0 \multimap j_0$$

3.3 Derivations

With the lexicon shown in (12), the surface scope interpretation can be derived, as shown in Figure 7, but the inverse scope interpretation cannot. Figure 8 shows one failed attempt to do so. Intuitively, the effect of the lexicon is to introduce a 'counter' as the argument to linear logic predicates, which forces quantifiers to apply in a particular order.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\text{guard}' : \forall \iota.\forall \eta.g_\iota \multimap (h_\eta \multimap f_\eta)}{\text{guard}' : g_1 \multimap (h_2 \multimap f_2)}\ \forall_E \quad \textit{jeden Ausgang} : (h_2 \multimap f_2) \multimap f_1}{\text{guard}'x : h_2 \multimap f_2} \quad \textit{Ein Polizist} : (g_1 \multimap f_1) \multimap f_0}{\text{every}'\text{exit}'(\text{guard}'x) : f_1}}{\lambda x.(\text{every}'\text{exit}'(\text{guard}'x)) : g_1 \multimap f_1}\ 1}{\text{a}'\text{officer}'(\lambda x.(\text{every}'\text{exit}'(\text{guard}'x))) : f_0}$$

Figure 7: Proof for the (surface scope) interpretation of (4)

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\text{guard}' : \forall \iota.\forall \eta.g_\iota \multimap (h_\eta \multimap f_\eta)}{\text{guard}' : g_2 \multimap (h_1 \multimap f_1)}\ \forall_E}{\text{guard}'x : h_1 \multimap f_1}}{\text{guard}'xy : f_1}}{\lambda x.\text{guard}'xy : g_2 \multimap f_1}\ 1 \quad \textit{ein Polizist} : (g_2 \multimap f_2) \multimap f_1}{*}$$

Figure 8: Failed attempt to derive an inverse scope interpretation for (4)

The lexical entries for the determiners mean that applying a quantifier reduces the counter by one (from $\mathsf{S}\iota$ to ι). It follows that for quantifier Q_1 to immediately outscope quantifier Q_2 (apply immediately after it), you have to set the counter for Q_1 to one lower than for Q_2. Therefore, to get the inverse scope reading of (4), you

would have to set the counter for the subject position one higher than for the object position; this is how the failed attempt in Figure 8 starts. However, the lexical entry for the verb guarantees that if you do that, no proof can be constructed: it sets the counter for the clause to the same as for the object position, which makes it impossible to apply the subject quantifier first.

4 Loosening the restrictions

In the above example, scope rigidity is enforced by a combination of the lexical entries for the determiners and the verb. The restriction could, therefore, be relaxed by tweaking either lexical entry. For instance, assuming that English uses the same fragment of linear logic for its syntax-semantics interface as German (which seems reasonable), the English data can be accounted for by assigning to *guards* the meaning constructor shown in (13), and otherwise importing the translation of the German lexicon.

(13) $\text{guard}' : \forall\iota.\forall\eta.\forall\kappa.(\uparrow \text{SUBJ})_\iota \multimap ((\uparrow \text{OBJ})_\eta \multimap \uparrow_\kappa)$

The meaning constructor assigned to *guards* in (13) differs from that assigned to *bewacht* in (12) in that in (13) the counter for the clause is not tied to the same value as either argument position. It could be instantiated with the same value as the object, enabling a surface scope interpretation, or to the subject, enabling an inverse scope interpretation.

However, in many instances we want the flexibility we allow for to be more fine-grained than this.

4.1 German scrambling

So far, the theory in Section 3 does not distinguish between (4) and (5); they are predicted to both only have the interpretation (2-a). What it seems that we would like to have is conditional introduction of meaning constructors: some statement to the effect that *bewacht* introduces the meaning constructor given in (12) if its subject precedes its object, and the English-style meaning constructor (13) otherwise. I do not see a way to do that directly in the formal architecture of LFG+Glue, but there is a workaround: give a German transitive verb the lexical entry shown in (14),

(14) *bewacht* V

$(\uparrow \text{PRED}) = \text{‘guard’}$
$\text{guard}' : \forall\iota.\forall\eta.(\uparrow \text{SUBJ})_\iota \multimap ((\uparrow \text{OBJ})_\eta \multimap \uparrow_\eta)$
$(@\,\text{RESET})$

where RESET is the template defined in (15).

(15) RESET $:= (\uparrow \text{OBJ}) <_f (\uparrow \text{SUBJ})$

$$\lambda p.p : \forall \iota . \forall \eta . \uparrow_\iota \multimap \uparrow_\eta$$

The RESET template is so called because it can be used to reset the counter. As specified in (14), the introduction of this template is optional. Clearly, if the subject f-precedes the object, as in Figure 4, then calling RESET would cause failure. So in the case of (4) RESET cannot be called, and scope rigidity is enforced as described in Section 3.

However, if the object f-precedes the subject, as in Figure (5), then RESET may or may not be called. If it is called, then both scope orderings are possible since the counter can be changed. Figure 9 shows a derivation of the (2-b) interpretation of (5) based on Figure 5, picking up from a decision point in the failed attempt for (4) shown in Figure 8.

(see figure 8)

$\begin{bmatrix} y : \\ h_1 \end{bmatrix}^2, \begin{bmatrix} x : \\ g_2 \end{bmatrix}^1$, *bewacht* ⋮ $\text{guard}'xy : f_1$

(RESET) $\dfrac{\lambda p.p : \forall \iota . \forall \eta . f_\iota \multimap f_\eta}{\lambda p.p : f_1 \multimap f_2}\ \forall_E$

$$\dfrac{\text{guard}'xy : f_1 \qquad \lambda p.p : f_1 \multimap f_2}{\text{guard}'xy : f_2}$$

$$\dfrac{\text{guard}'xy : f_2}{\lambda x.\text{guard}'xy : g_2 \multimap f_2}\ 1$$

ein Polizist ⋮ $\text{a}'\text{officer}' : (g_2 \multimap f_2) \multimap f_1$

$$\dfrac{\lambda x.\text{guard}'xy : g_2 \multimap f_2 \qquad \text{a}'\text{officer}' : (g_2 \multimap f_2) \multimap f_1}{\text{a}'\text{officer}'(\lambda x.\text{guard}'xy) : f_1}$$

$$\dfrac{\text{a}'\text{officer}'(\lambda x.\text{guard}'xy) : f_1}{\lambda y.\text{a}'\text{officer}'(\lambda x.\text{guard}'xy) : h_1 \multimap f_1}\ 2$$

jeden Ausgang ⋮ $\text{every}'\text{exit}' : (h_1 \multimap f_1) \multimap f_0$

$$\dfrac{\lambda y.\text{a}'\text{officer}'(\lambda x.\text{guard}'xy) : h_1 \multimap f_1 \qquad \text{every}'\text{exit}' : (h_1 \multimap f_1) \multimap f_0}{\begin{array}{c}\text{every}'\text{exit}'(\lambda y.\text{a}'\text{officer}'(\lambda x.\text{guard}'xy)) : f_0 \\ \equiv \forall y.\text{exit}'y \rightarrow \exists x.\text{officer}'x \wedge \text{guard}'xy : f_0\end{array}}$$

Figure 9: An interpretation of (5) that is not available for (4)

4.2 English double object constructions

In order to analyze scope freezing in English double object constructions, we have to generalize from the example in (6) by seeing what interpretations are available when there is a quantifier in subject position too, as in (16).

(16) A teacher gave most students every grade.

Judgements get a little unclear for examples like this, but Bruening (2001) reports the results shown in (17), and I will proceed on that basis. In summary, the secondary object may not take wider scope than the primary object, but otherwise the relative scope of the three quantifiers is free.[10]

(17) a. (16) $\Rightarrow \exists x.\text{teacher}'x \wedge \text{most}'\text{student}'(\lambda z.\forall y.\text{grade}'y \rightarrow \text{give}'xyz)$

[10]Just to be clear: give$'xyz$ should be interpreted as saying that x gives y to z (or equivalently, the x gives z y).

b. $(16) \Rightarrow \text{most}'\text{student}'(\lambda z.\exists x.\text{teacher}'x \wedge \forall y.\text{grade}'y \rightarrow \text{give}'xyz)$
c. $(16) \Rightarrow \text{most}'\text{student}'(\lambda z.\forall y.\text{grade}'y \rightarrow \exists x.\text{teacher}'x \wedge \text{give}'xyz)$
d. $(16) \not\Rightarrow \exists x.\text{teacher}'x \wedge \forall y.\text{grade}'y \rightarrow \text{most}'\text{student}'(\lambda z.\text{give}'xyz)$
e. $(16) \not\Rightarrow \forall y.\text{grade}'y \rightarrow \exists x.\text{teacher}'x \wedge \text{most}'\text{student}'(\lambda z.\text{give}'xyz)$
f. $(16) \not\Rightarrow \forall y.\text{grade}'y \rightarrow \text{most}'\text{student}'(\lambda z.\exists x.\text{teacher}'x \wedge \text{give}'xyz)$

An attempt to account for the data in (17), which undergenerates slightly, is to assign to *gave* the meaning constructor shown in (18).

(18) $\text{give}' : \forall\iota.\forall\eta.\forall\kappa.(\uparrow \text{SUBJ})_\iota \multimap ((\uparrow \text{OBJ}_\theta)_\eta \multimap ((\uparrow \text{OBJ})_\kappa \multimap \uparrow_\eta))$

Using (18) forces the secondary object to take narrowest scope, and otherwise leaves the relative scope of the quantifiers free. That correctly rules out the unavailable readings of (16), and predicts two of the three available ones, but incorrectly rules out the interpretation shown in (17-c).

There are, of course, various ways of tackling this shortcoming.[11] The one I will present here involves once more expanding the fragment of linear logic used, by adding to the language defined in Section 3.1 a ternary function symbol C, to be interpreted as shown in (19).[12]

(19) $$\mathsf{C}mno = \begin{cases} m \text{ if } m > n > o \\ n \text{ otherwise} \end{cases}$$

We can now capture the data in (17) by assigning *gave* the meaning constructor shown in (20).

(20) $\text{give}' : \forall\iota.\forall\eta.\forall\kappa.(\uparrow \text{SUBJ})_\iota \multimap ((\uparrow \text{OBJ}_\theta)_\eta \multimap ((\uparrow \text{OBJ})_\kappa \multimap \uparrow_{\mathsf{C}\iota\eta\kappa}))$

For illustration, derivations of the interpretations (17-b) and (17-c) are shown in Figures 11 and 12, respectively, based on the f-structure in Figure 10. By contrast, a failed attempt to derive (17-d) is shown in Figure 13.

$$f : \begin{bmatrix} \text{PRED} & \text{‘give’} \\ \text{SUBJ} & g : \left[\text{“a teacher”}\right] \\ \text{OBJ} & h : \left[\text{“most students”}\right] \\ \text{OBJ}_\theta & i : \left[\text{“every grade”}\right] \end{bmatrix}$$

Figure 10: F-structure of (16)

[11] Some will be hinted at in Section 5.

[12] We now have to take seriously that S is the successor function, since we have $>$ at the meta-level.

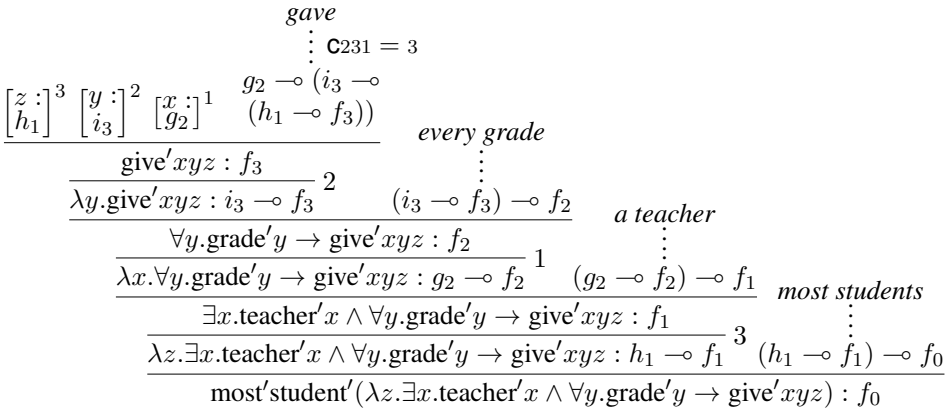

Figure 11: Proof of (17-b)

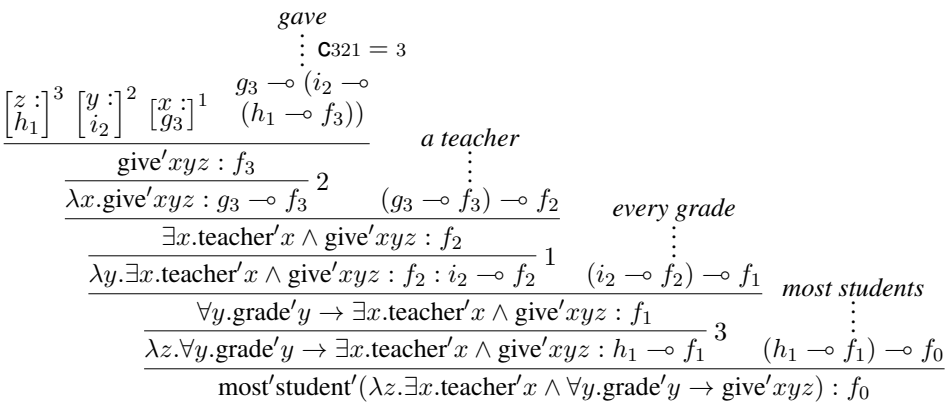

Figure 12: Proof of (17-c)

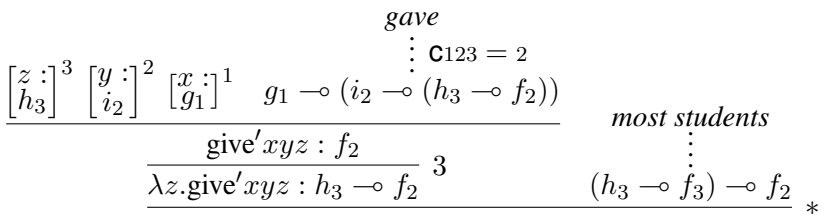

Figure 13: A failed attempt to derive (17-d)

5 Discussion

This has primarily been a theoretical paper, describing and applying certain formal tools for constraining scope ambiguity in LFG+Glue. I have argued that those tools are conceptually preferable to the ones previously put forward in the literature. In a nutshell, the account of scope rigidity presented in this paper is that a verb may

specify, in its lexical entry, which of its arguments must or may take narrowest scope. Ideally, at least for the cases considered in this paper, we would like the specifications to be more general than that, such that they would apply to every transitive (for German) or ditransitive (for English) verb in the language. Presumably the desired effect can be achieved, at least at the descriptive level, with the use of templates (Dalrymple et al., 2004).

Of course, this paper has not even scratched the surface of the empirical data to be accounted for in a theory of scope rigidity. One major limitation is that we have only considered cases where the constraints on scope ordering can be described purely in terms of grammatical functions. But there are cases where the possibility of scope ambiguity seems to depend on precisely which quantifiers occupy which of a verb's argument positions. For example, while English generally allows inverse scope in simple transitive sentences, it is not clear that this is always possible when there is a monotone-decreasing quantifier in object position, as in (21).

(21) Most students completed fewer than three assignments.

The right approach to this datum (if it is one) may be to treat English more like German, only with a less constrained RESET-like template for loosening restrictions. Or alternatively, it may be more profitable to treat the quantifiers themselves as contributing the necessary restrictions, as in Fry's (1999) account of NPI licensing. Many details would need to be worked out, though: as Fry (1999) acknowledges, his theory has the power to force certain items (NPIs) to take scope under other items (licensors), but no power to require licensors to be more prominent than licensees at any level of grammatical description.

There are other possible alternatives that could be explored. For example, an account of the kind given for scope islands in Section 1.3 *could* be extended to scope rigidity, but it would require some quite drastic foundational changes to the LFG+Glue architecture: either by providing f-/s-structure with more internal structure (cf. Andrews (2018) on the relative scope of adjectives), or by having linear logic formulae read off c-structure instead. I cannot seen either of these options being popular.

References

Andrews, Avery D. 2010. Propositional glue and the projection architecture of LFG. *Linguistics and Philosophy* 33. 141–170. doi:10.1007/s10988-010-9079-9.

Andrews, Avery D. 2018. Sets, heads, and spreading in LFG. *Journal of Language Modelling* 6(1). 131–174. doi:10.15398/jlm.v6i1.175.

Barker, Chris & Pauline Jacobson (eds.). 2007. *Direct compositionality* (Oxford Studies in Theoretical Linguistics 14). Oxford: Oxford University Press.

Barker, Chris & Chung-chieh Shan. 2014. *Continuations and natural language* (Oxford Studies in Theoretical Linguistics 53). Oxford: Oxford University Press.

Bruening, Benjamin. 2001. QR obeys superiority: Frozen scope and ACD. *Linguistic Inquiry* 32(2). 233–273. doi:10.1162/00243890152001762.

Cook, Philippa & John Payne. 2006. Information structure and scope in German. In Miriam Butt & Tracy Holloway King (eds.), *Proceedings of the LFG'06 conference, University of Konstanz*, Stanford, CA: CSLI Publications.

Cooper, Robin. 1983. *Quantification and syntactic theory* (Studies in Linguistics and Philosophy 21). Dordrecht: D. Reidel.

Corbalán, María Inés & Glyn Morrill. 2016. Overtly anaphoric control in type logical grammar. In Annie Foret, Glyn Morrill, Reinhard Muskens, Rainer Osswald & Sylvain Pogodalla (eds.), *Formal grammar* (Lecture Notes in Computer Science 9804), 183–199. Berlin, Heidelberg: Springer. doi:10.1007/978-3-662-53042-9_11.

Crouch, Richard & Josef van Genabith. 1999. Context change, underspecification and the structure of Glue language derivations. In Mary Dalrymple (ed.), *Semantics and syntax in Lexical Functional Grammar*, 117–189. Cambridge, MA: MIT Press.

Dalrymple, Mary, Ronald M. Kaplan & Tracy Holloway King. 2004. Linguistic generalizations over descriptions. In Miriam Butt & Tracy Holloway King (eds.), *Proceedings of the LFG'04 conference, University of Canterbury*, 199–208. Stanford, CA: CSLI Publications.

Fry, John. 1999. Proof nets and negative polarity licensing. In Mary Dalrymple (ed.), *Semantics and syntax in Lexical Functional Grammar*, 91–116. Cambridge, MA: MIT Press.

Gotham, Matthew. 2017. Glue semantics and locality. In Miriam Butt & Tracy Holloway King (eds.), *Proceedings of the LFG'17 conference, University of Konstanz*, 230–242. Stanford, CA: CSLI Publications.

Jacobson, Pauline. 2002. The (dis)organization of the grammar. *Linguistics and Philosophy* 25. 601–626. doi:10.1023/A:1020851413268.

Jacobson, Pauline. 2014. *Compositional semantics*. Oxford: Oxford University Press.

Kanazawa, Makoto. 2015. Syntactic features for regular constraints and an approximation of directional slashes in abstract categorial grammars. In Yusuke Kubota & Robert Levine (eds.), *Empirical advances in categorial grammar*, 34–70. https://web.archive.org/web/20170705131723/https://www.u.tsukuba.ac.jp/~kubota.yusuke.fn/cg2015-proceedings.pdf. Last accessed 2019-10-07.

May, Robert. 1985. *Logical form* (Linguistic Inquiry Monographs 12). Cambridge, MA: MIT Press.

Montague, Richard. 1973. The proper treatment of quantification in ordinary English. In Patrick Suppes, Julius Moravcsik & Jaakko Hintikka (eds.), *Approaches to natural language*, 221–242. Dordrecht: D. Reidel.

Pogodalla, Sylvain & Florent Pompigne. 2012. Controlling extraction in abstract categorial grammars. In Philippe de Groote & Mark-Jan Nederhof (eds.), *Formal grammar* (Lecture Notes in Computer Science 7395), 162–177. Berlin, Heidelberg: Springer. doi:10.1007/978-3-642-32024-8_11.