Scope alternation and Type Logical Grammar

QNP scope and its implications for deductive grammar

Thesis submitted to the University of London in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Hiroyuki Uchida
Department of Phonetics and Linguistics
University College London

February, 2008

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

Declaration

I, Hiroyuki Uchida, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Abstract

This thesis provides an analysis of QNP scope which naturally explains the characteristic tensed-clause locality constraint.

Linguistically, I argue that QNP scope is not explained by A-bar movement or A-movement. A-bar movement is different from QNP scope in terms of the locality constraints and A-movement is not flexible enough to explain scope ambiguity.

Instead of reducing QNP scope to A-bar or A-movement phenomena, I argue that scope ambiguity is a result of an independent operation that allows us to merge QNPs as syntactic arguments of the local functors even though they are semantically operators. To instantiate this idea, I reformulate Hendriks' argument raising as a special rule (called Argument Slot Raising or ASR) in a non-associative grammar NL. ASR only affects the local functor of the QNP in question. Thus, the QNP's scope is predicted to stay within the final output of this functor. For control/auxiliary/raising constructions, I extend NL by introducing an association rule that is restricted by a pair of merge modes in Moortgat's (1997) Multi-Modal grammar. This structural rule may postpone the saturation of NP argument slots, allowing a complex predicate to be formed. Application of ASR to such complex predicates may switch QNP scope within each finite TP.

Structural rules for A-bar movement are introduced in terms of introduction and discharge of modally marked hypothetical categories, which explains the longer distance nature of A-bar extraction in comparison to QNP scope.

Finally, I explain the apparent 'exceptional scope' of indefinites in terms of their domain restriction and anaphoric dependency on other operators. This maintains the uniform locality of QNP scope.

The underlying claim is that the QNP scope switch mechanism itself does not involve structural ambiguity. The thesis considers the implications for natural language grammar of implementing this idea.

Preface

As I have stated in the abstract, the linguistic goal of the thesis is to provide an analysis of QNP scope which naturally explains why QNP scope is generally blocked by a tensed clause. A theoretical goal is to set up a Type Logical Grammar system so that the three linguistically distinct phenomena, that is, QNP scope, A-movement and A-bar movement phenomena, are taken care of by different kinds of theoretical tools. Because of the grand goal, readers with different interests might concentrate on different parts of the thesis.

Generally, readers who are familiar with Type Logical Grammar as in Moortgat (1997) can either skip or skim through the introductions of TLG, which appear in section 1.5 of chapter 1 and sections 3.1-3.3 of chapter 3. The thesis does not propose any innovation in the logical grammar system itself. However, because I have interpreted Type Logical Grammar from a viewpoint of Universal Grammar, logicians might still want to have a brief look at those sections to grasp a different conception of logical grammar.

For linguists, the readers who are interested in my analysis of QNP scope can get a general idea by reading chapter 1 and chapter 2 only. Chapter 8 and Chapter 9 discuss the issues which I argue are distinct from QNP scope, and thus such readers might skip these chapters.

For those who are interested in how I have distinguished the above mentioned three kinds of linguistic phenomena in formalism, the essential chapters are 2, 7 and 8.

For those who are interested in how I deal with 'exceptional scope' of indefinites, chapter 9 is reasonably self-contained, though those who are not familiar with Gentzen Sequent presentation might need to read chapter 3 to understand the TLG proofs/derivations. The essential structural rules are repeated in chapter 9.

A different version of chapter 4 appears as Uchida (2005a). Uchida (2006) is an informal version of chapter 5, and Uchida (2005b) is chapter 9, though it was written in a Combinatory Categorial Grammar framework in the proceedings.

I have presented a preliminary version of chapter 9 at the Strategies of Quantification conference organized by University of York in July, 2004, also at ESSLLI'04 in Nancy, France, at LAGB'04 at Roehampton University and at ConSOLE XIII at Tromsø University in different versions. I have presented an earlier version of chapter 4 at ESSLLI'05 in Edinburgh, and have given a talk about initial versions of chapter 5 at CamLING'06 at Cambridge University and at LAGB'06 at Newcastle University.

This provides a seamless transition to acknowledgments. After finding out that some might actually read acknowledgments, Rob Truswell understandably got worried about what to put. Being careless and forgetful, I am more worried about misspelling the names of those who I do include in mine. Predicting such mistakes, I apologize to anyone I've inadvertently left out and to anyone whose name I have misspelt.

Though she might have regretted her decision by now, Annabel Cormack kindly agreed to supervise my PhD research. Without her help, this thesis would have become much less sophisticated, especially on the linguistic side. Having extensive knowledge about Montague Grammar, Combinatory Categorial Grammar and natural language syntax, she has been an ideal supervisor for me. Her sheer enthusiasm towards Chomskian linguistics always impressed me.

Without Robyn Carston, my first supervisor, I would not have been able to start and continue my research at UCL in the first place. I would not have been able to attend various conferences which have led to this thesis without her help, either. I am grateful to Robyn for her constant support during my years at UCL and for her prompt and efficient dealing with the many administrative aspects of my studies and my stay in the UK. I also appreciate her great patience with my continuous inefficiency. As well as her administrative and psychological support, her comments on aspects of my thesis have been also invaluable. General presentations of the crucial ideas have been significantly improved because of her help.

Neil Smith and Michael Brody were the reason why I decided to study linguistics at UCL. I especially owe a lot to Neil, another supervisor of mine. Having discussion with him, whether about linguistics, logic, math or life, has been simply the best pleasure I have experienced in London. I am hoping that I have learned something from his extremely careful and step-wise development of theories.

I also thank Yushiro Inouchi, who supervised my undergraduate thesis and encouraged me to pursue linguistic research in the UK.

Attending Ruth Kempson and Wilfried Meyer-Viol's lectures/reading groups and having discussions with them at King's College London have had major impact

in my use of logic and formal grammar in theoretical linguistics. Ruth's analysis of VP ellipsis in terms of copied procedures and Wilfried's treatment of Dutch cross-serial constructions in Dependency Grammar are two of my favorite analyses.

I feel much gratitude to Michael Moortgat for providing a brief supervision about the basic idea of my thesis at ESSLLI'06, and Glyn Morrill for having brief but insightful discussions with me about Type Logical Grammar and proof nets at conference sites. Michael's symmetrical Type Logical Grammar and Glyn's generalized wrapping have shown me representative models of deductive grammar systems.

Turning to fellow students in the present and in the past, Nicholas Allott, Anna Pollard and Rob Truswell have read the whole or part of the thesis and given me many valuable comments. Nick and Rob were generous enough to discuss proofs and algebraic semantics with me which helped clear my thoughts. Anna's patient revision advice which covered the whole thesis has helped me express the thoughts I really wanted to express in English.

Comparing Dirk Bury's Constituent Structure Sets with relational structures, and giving a related talk with him at tenth Mathematics of Language conference at UCLA have had an indirect but important effect on this thesis. For that conference, I also thank Marcus Kracht for his encouragements and comments, and Natasha Kurtonina for having discussion with me about the development of Type Logical Grammar as symmetrical grammar.

I am grateful to Dorota Glowacka for showing how to recognize A/B-accents. It helped me abstract away from the effect of intonation contours when I considered quantifier scope. Discussing CFG, MCFG and computational/algebraic complexity with her during my evaluation of TLG has also been helpful in formulating my grammar system.

To make up for my naivety as an empirical linguist, Ad Neeleman and Reiko Vermeulen have shown me how to analyze language data in a theory neutral manner. Marc Richards has helped me understand the spirit of Chomskian syntax. Marc's views about phase edges, superiority of Wh-movement and VP structures across VO/OV languages have influenced my structural rule introductions. I also thank Alison Hall for answering questions related to the thesis in her capacity as a philosopher of language. Her views about pragmatic free enrichment have been informative as well. Kriszta Szendroi's comments about Hungarian data have been helpful beyond the Hungarian data included.

I thank all those who attended my presentations related to this thesis. Comments and questions from Klaus Abels, Michael Brody, Ronnie Cann, Paul Egré,

Judit Gervain, Janneke Huitink, Alex Lascarides, Angelika Kratzer, Edward Keenan, Marc Richards, Mark Steedman, Øystein Nilsen, Rob Truswell, George Tsoulas and Hans van de Koot were especially memorable.

Among various teaching experiences, a non-credit formal semantics course I taught with Nicholas Allott to PhD students at UCL has directly influenced the presentation of Type Logical Grammar in chapter 1 and 3 of this thesis. I thank Rob Truswell and Reiko Vermeulen for making important contributions to the lectures as students. For my visiting lectureship at University of Westminster, I thank Andrew Caink and Anand Syeaa, who have always been supportive and accessible for help. Working with them has taught me a lot, not only in teaching, but also in academic research as well. It has been great fun as well.

I owe a lot to those who generously shared with me the invaluable knowledge that they have acquired in their own fields. Some of the knowledge that I have acquired in this way has not been explicitly used in my thesis, but I can nonetheless sense their influence in some way or other. Other than the staff/students that I have already mentioned, I appreciate the help of Richard Breheny and Nathan Klinedinst in semantics, Deirdre Wilson and Ingrid Falkum in pragmatics, Andrew Gargett, Eleni Gregoromichelaki and Udo Klein in Dynamic Syntax, Emily Robin and Kayoko Yanagisawa in phonology/phonetics, Eric Carlson in computer science, Hui Cao, Victoria Janke, Jieun Joe-Kiaer, Ivona Kučerová, Ann Law, Marika Lekakou, Matthew Reeve, Kate Scott and Hitoshi Shiraki in natural language syntax, Thiago Galery, George Powell and Eva Pils in philosophy of language, Judy Humphrey in physics, and Amanda D'Souza and Karen Pamela James in neuroscience and speech therapies.

Many thanks to Stefanie Anyadi and Molly Bennett for their efficient administrative support. I have also hugely appreciated Stephen Nevard's technical support.

I thank all the other staff/students that I have interacted with either academically or socially during my research, as well as the PhD room in GS21, which has contributed to the improved research/social experience among research students.

Finally, the research has been partially funded by Overseas Research Student (ORS) award, and by Toho High School in Tokyo. For the latter funding, I especially thank Yoshihisa Kobayashi and Mitsuo Onishi.

Contents

It is as if you explained subtraction, '$b - a$' , only in the context of addition, '$(b - a) + x$', so that you explained two operators at once. If you are a 'holist', probably you will not care; but then there is not much about which holists much care.

Nuel Belnap

Chapter 1

Introduction: QNP scope and Type Logical Grammar

1.1 Introduction

The main linguistic goal of this thesis is to provide an analysis of the scope of quantificational noun phrases (QNPs) which naturally explains the characteristic locality constraints on QNP scope. A theoretical goal of this thesis is to set up a Type Logical Grammar system in such a way that it respects the division lines between QNP scope, A-movement and A-bar movement phenomena which I argue are linguistically different from one another. While pursuing these goals, I consider what is the linguistically most insightful way of extending the expressive power of a logical grammar, starting with a grammar system that is empirically too restrictive.

Scope ambiguity has been studied extensively by linguists. Consider (1).

(1) A boy loves every girl.

The sentence in (1) has two scope readings as is shown in (2).

(2) a. Surface scope reading ($\exists > \forall$):
'The same boy loves all the girls.'

b. Inverse scope reading ($\forall > \exists$):
'Each girl is loved by a (potentially) different boy.'

In predicate logic, (2a) can be represented as (3a) and (2b) can be represented as (3b).

(3) a. $\exists x(boy'(x) \wedge \forall y(girl'(y) \rightarrow love'(y)(x)))$

b. $\forall y(girl'(y) \rightarrow \exists x(boy'(x) \wedge love'(y)(x)))$

Sub-terms of the predicate logic expressions in (3) do not correspond to sub-expressions of the English sentences in (1). Because of this, to show compositional derivations of the scope readings in syntactic derivations, I use higher order logic notation which I show in 1.5.2 in this chapter. But for the moment, (3) can approximate the two readings of (1). In (3a), the existential quantifier associated with the indefinite *a boy* takes wide scope over the universal quantifier that is encoded with the QNP *every girl*. Because the linear-left QNP in the PF string takes wide scope over the right QNP in the string, we call (2a)/(3a) the surface scope reading. In (3b), the (universal) quantifier encoded with the right QNP in the string takes wide scope over the (existential) quantifier encoded with the left QNP, and thus we call (2b)/(3b) the inverse scope reading. Abstracting away from the placements of the two QNPs in the PF string, I sometimes call (2a)/(3a) the indefinite wide scope reading (or the universal narrow scope reading), and (2b)/(3b) the indefinite narrow scope reading (or the universal wide scope reading).

May (1977) proposed that the syntactic derivation disambiguates scope readings at LF.[1] May argued that QNP scope should be explained in a similar way as Wh-movement is explained. Consider (4).

(4) a. Some boy$_i$ [every girl$_j$ [t$_i$ loves t$_j$]]

b. Every girl$_j$ [some boy$_i$ [t$_i$ loves t$_j$]]

c. cf. Who$_i$ [does the boy love t$_i$]?

Just as Wh-expressions in English (overtly) move from their positions as verbal arguments to the sentence initial positions, as shown in (4c), May assumed that QNPs covertly move and are adjoined to a sentential node, taking scope from that position, as shown in (4a) and (4b).

Though May's QR has been adopted widely, especially among Chomskian linguists,[2] there is a critical problem in explaining scope taken by QNPs in terms of covert A-bar movement. That is, as Johnson (2000) has shown in detail, the scope

[1]May (1985) uses one LF representation to represent two scope readings, by modifying the definition of c-command so that two QNPs that are successively adjoined to the same XP (say, IP) via Quantifier Raising (QR) can c-command one another (cf. May 1985: 34-35). I ignore this modification. Because he still needs to covertly move and adjoin the two QNPs to the same XPs, my arguments against an A-bar movement analysis of QR still hold.

[2]See Bruening (2001) for his support of A-bar movement analyses of QR, rather than A-movement analyses which we will see shortly. Heim and Kratzer (1998) provide some semantic arguments for QR, rather than in-situ treatments of QNP scope. Also in Type Logical Grammar, Bernardi (2002) instantiates the A-bar movement analysis by Beghelli and Stowell (1997) in TLG, and Vermaat (2006) also assimilates scope taking of QNPs to Wh-extraction, by treating QNPs in the same way as she deals with in-situ Wh-expressions.

of QNPs and Wh-movement are subject to different locality constraints.[3] I will briefly review Johnson's points with some other points that distinguish QNP scope from Wh-extraction in section 1.2.

If a syntactic movement analysis of QNP scope were independently well motivated irrespective of Wh-movement, it would be worth modifying the notion of A-bar movement so that it can accommodate different locality constraints that are applicable to QNP scope.[4] However, a more fundamental problem of seeing QR as a syntactic operation is that most of the arguments for QR are semantic ones, for instance, the argument that QNPs are semantically propositional operators, rather than being arguments of (verbal) functors, or the so-called infinite regress problem associated with Antecedent Contained Deletions.[5] But if we increase the expressive power of the grammar system to accommodate semantic factors which do not have clear effects on PF strings, the resultant grammar system may be less attractive as a candidate for a Chomskian Universal Grammar, which is meant to provide non-trivial constraints on PF-LF pairing by way of syntactic derivations. To have the consideration of QNP scope influence the syntactic structures, we would want some clear evidence from the PF side that shows that QR exists as a syntactic movement. As we briefly see in section 1.2, though Hungarian is claimed to show evidence of overt movement of QNPs, the data do not uncontroversially support an overt movement that is dedicated to the scope of QNPs. Also, though adjoining QNPs to local TP/IPs does capture the characteristic locality constraints on QNP scope, to the degree that it is assimilated to A-bar (or operator) movement in general, it is not clear why only QR (as opposed to Wh-movement or overt topicalization) is constrained that way.

Alternatively to A-bar movement analyses, Hornstein (1995) and Hornstein (1999a) have attempted to explain QNP scope in terms of a certain kind of A-movement (i.e., A-movement that licenses the argument status of each NP, by moving the subject NP to AgrS and moving the object QNP to AgrO, as in Chomsky (1995) for example). A merit of this analysis is that the tensed-clause bound locality constraint on QNP scope follows more naturally from it than with A-bar movement analyses, given the standard assumption that the argument status of each NP argument is resolved within each verbal projection line, which terminates

[3]Johnson (2000) has made this point with regard to overt topicalization movement, which is subject to more or less the same locality constraints that Wh-movement is constrained by.

[4]For example, QR generally adjoins QNPs to the local TPs/IPs, rather than moving them to (potentially non-local) spec CP as is the case with Wh movement.

[5]I do not review these arguments. For a good summary, see Heim and Kratzer (1998) chapter 7, Hornstein (1995) and Fox (2002a).

at the end of each TP in the Minimalist program (or stays within the minimal S in the categorial grammar framework which I adopt in this thesis). On the other hand, some problems arise from an attempt to use various A-copy positions as potential scope interpretation positions for QNPs, which I briefly discuss in section 3. There I suggest that QNP scope ambiguity cannot naturally be considered as a side effect of movement to case checking positions.

I conclude that postulating an additional syntactic operation dedicated to QNP scope is not well motivated for lack of clear evidence in the PF side, and also that syntactic operations that apply to (Q)NP arguments in general cannot naturally explain all the QNP scope phenomena (though they obviously interact with QNP scope). Thus, I choose to explain QNP scope in terms of syntax-semantics inconsistency. That is, syntactically, QNPs are arguments of the local functors, which are normally the local verbal functors, but which may be prepositional functors. Semantically, however, QNPs are interpreted as propositional operators. Rather than resolving this syntax-semantics inconsistency by postulating a syntactic movement that adjoins each QNP to some sentential node or a syntactic operation that has the same effect in a type logical grammar formalism, such as Montague's (1973) quantifying-in, or with Moortgat's (1991) analysis using his scope constructor, which Carpenter (1997) adopts in his chapter 7,[6] I choose to reformulate Argument Raising in Hendriks (1987) in Categorial calculus and apply the rule specifically to the local functor that takes the QNP in question as an argument. Because the rule only applies to the local functor, its effect cannot exceed the final output of this local functor, that is, the local category S which is the final value of the local functor in categorial grammar (or the local TP in Minimalism). Some A-movement phenomena may produce a complex predicate that is made out of lexical subcomponents, such as a control verb and a transitive verb, or an auxiliary verb and a transitive verb, and the above mentioned translation rule may apply to this functor as well, extending the scope of QNP beyond the strictly local area that the lexical verbal functor can cover. However, this cannot exceed the local S category (or the local TP, as in Minimalism), to the degree that linguistic observations support the assumption that complex predicates are formed only within each propositional

[6]More recent works by Moortgat (cf. Moortgat 2005, 2007) have derived his scope constructor from the basic connectives of his symmetric logical grammar. The analysis has a clear merit from a logical viewpoint, helping us to maintain nice properties of a logical grammar. On the other hand, it is not clear whether the characteristic locality constraints on QNP scope can be naturally explained in this analysis. Partly because my current grasp of this logically attractive system is not good enough, both from a logical viewpoint and in its application to linguistic phenomena, I leave the investigation of Moortgat's symmetric grammar for future research.

boundary (i.e. each S, or each TP). In this way, the characteristic tensed-clause locality constraint on QNP scope is naturally explained.

On the other hand, using some rule that applies only to specific items (that is, the local functors of QNP arguments) might not be ideal in terms of generality. Also, it turns out that the rule that I apply to the functor categories is not fully supported by the basic grammar system in a sense to be explained later. Postulating a semantically motivated lexical/phrasal rule that is not fully supported by the grammar system may be the last resort.[7] However, as I show in this thesis, my analysis does not generate more syntactic structures than are independently motivated by the standard syntactic phenomena. Given that there is not enough evidence that supports the incorporation of QNP scope phenomena into the syntax proper, I argue that this is a merit of my analysis.

Section 1.2 briefly shows that QNP scope is not reducible to A-bar movement phenomena. Section 1.3 does the same with regard to A-movement phenomena. The scope of the indefinite is claimed to be able to cross a tensed clause, but in section 1.4, I suggest that the exceptional scope of indefinites is not a matter of QNP scope. Section 1.5 provides the basics of Type Logical Grammar from linguistic viewpoints (the presentation of TLG from a more logical viewpoint is provided in chapter 3). Section 1.6 provides the basic organization of the thesis.

1.2 Double disassociation between A-bar movement and QNP scope

Johnson (2000) shows that A-bar movement and scope of QNPs are subject to different locality constraints.[8]

On the one hand, a QNP cannot easily take scope outside the minimal tensed clause in which it is embedded, whereas Wh-movement is not blocked by a mere tensed clause, as shown in (5).[9]

(5) a. Bill told some/a student (that) they would meet every linguist.
$\exists > \forall, ? * \forall > \exists$

[7]Because the rule may apply to complex predicates that the grammar generates, it is not literally a lexical rule, as we see later.

[8]Johnson compares QR with overt topicalization of NPs.

[9]Büring's B-accent/A-accent pattern allows scope switch across sentence boundaries more easily, for example, */SOME boy said that he likes E\very girl*, with which we can get the inverse scope *every* > *some* more easily ('/' informally indicates a rising tone contour and '\' indicates a falling tone contour). I abstract away from some effects from such special intonation contours. See Büring (1995) for some discussion.

b. Who$_i$ did Bill tell some/a student (that) they would meet t$_i$?

For the observation that QNP scope cannot cross a tensed clause, see Fodor & Sag (1982: 367-370), Reinhart (1995: 3-4) and Winter (2001: 82-85). In the case in (5), the locality constraint on QNP scope is stricter than the locality constraint on Wh-movement.

On the other hand, the constraint can be stricter for Wh-movement. (6) shows that Wh-movement is blocked by the subject island, whereas a QNP in the same syntactic position can take scope over the whole sentence.[10]

(6) a. Some/a student from every school gave a speech. $\exists > \forall; \forall > \exists$

b. *Which school$_i$ did some/a student from t$_i$ give a speech?

Some might try to explain the different locality constraints on Wh-movement and on QNP scope by moving Wh-expressions (overtly) and QNPs (covertly) to different types of A-bar positions. It might then be stipulated that only Wh-expressions can move across a tensed clause to the relevant type of landing site.[11] However, separating QNP scope from independently motivated A-bar movement phenomena, this explanation would require an independent motivation to deal with QNP scope as a syntactic phenomenon.

In the case of Wh-movement, there are languages in which at least one Wh-expression **obligatorily** moves to Spec CP (or in theory neutral terms, at least one Wh-expression is obligatorily placed in the sentence initial position in the phonological string, rather than in the PF position which the corresponding NP argument occupies. In contrast, it is not clear if there is a language in which QNPs obligatorily move in overt syntax. Some might argue that Hungarian is one such language. Consider (7).

(7) a. Sok ember mindenkit felhívott.
many man everyone-acc up-called
'Many men phoned everyone.'
many > every

b. Mindenkit sok ember felhívott.
everyone-acc many man up-called

[10]In the reading in question, the universal takes wide scope over the two indefinites. That is, for each school, the reading is about a different pair of a student and the speech that he/she gives.

[11]Wh-expressions move to Spec CP, whereas QR normally moves QNPs to Spec IP/TP. So it could be stipulated that only movements to Spec CP can cross a tensed clause. Beghelli and Stowell (1997) resorts to the same kind of stipulation to explain why the scope of indefinites can cross a tensed clause. I later argue that the scope of indefinites is actually blocked by a tensed clause, just like the scope of universal QNPs. See chapter 9.

'Many men phoned everyone.'

every > many Szabolcsi (1997: 118)

As Szabolcsi (1997) shows, when the QNPs are placed in front of the verb in the phonological string, the scopal order of quantifiers matches their left-to-right order. Moreover, she shows that positioning of (Q)NPs relative to sentential adverbials, placement of verbal affixes relative to the verbs etc. can distinguish the structural positions between QNPs and NPs, or between different kinds of QNPs.

The thesis does not aim to discuss Hungarian data even at the basic level, but it is at least questionable whether Hungarian provides evidence of obligatory movement of QNPs which is dedicated to QNP scope.

First of all, a QNP may appear to the right of the verb in the PF string, and if it does, depending on the phonological focus assignment to the QNP, we can have two scope readings. Consider (8).

(8) a. **Kevés filmet** látott mindent ember.
few film-acc saw every man-nom
'Few films were watched by everyone.'
few > every

b. **Kevés filmet** látott **mindent ember**.
few film-acc saw every man-nom
'Everyone watched few films.' (I.e. 'Nobody watched a lot of films.')
every > few Brody and Szabolcsi (2003), p.22

Szabolcsi (1997) indicates that when the post-verbal QNP is heavily stressed, it is an instance of stylistic post-posing in PF form (cf. Szabolcsi 1997: 118, footnote 9) (and thus, in order to take wide scope, QNPs must be obligatorily moved to a preverbal position in the syntax), but the consideration of intonation patterns weakens the argument for the existence of an overt movement of QNPs dedicated to scope. After all, the QNP *mindent ember* does appear post verbally in the PF string in (8b) with the wide scope interpretation. Just by looking at the overt PF string, we cannot tell whether there had been a syntactic movement of the QNP to a pre-verbal position and some PF movement has moved the QNP back to the post-verbal position again, or there has not been any movement at all in the first place.

Also, it is not the case that at least one QNP must be placed in front of the verb.

(9) Janos felhívott mindenkit.
Janos up-called everyone-acc

'Janos phoned everyone.'

Even when there is only one QNP in the sentence, that QNP can be left postverbally. If the postulated overt movement were to be triggered only for the QNP to take wide scope, (9) does not show any evidence either for or against the existence of such a movement beyond the data and the suggestions that I have provided above. On the other hand, (9) excludes the argument which says that at least one QNP must obligatorily move in Hungarian (in comparison to Wh-movement in English, in which one Wh-expression must move to Spec CP) which could provide some argument for postulating a dedicated operator movement to QNPs.

For lack of clear data that can be seen as decisive evidence for a dedicated operator movement for QNPs, I conclude that we do not have enough evidence to postulate an A-bar movement that is specifically dedicated to QNPs. Because QR behaves differently from independently motivated A-bar movement, such as Wh-movement or overt topicalization, I conclude that syntactic A-bar movement analysis of QNP scope is not well motivated.

In the next section, I briefly review the A-movement analysis by Hornstein.

1.3 Hornstein's A-movement analysis

Hornstein (1995, 1998 and 1999a) claims that scope ambiguity is a side effect of Case movement for NP arguments.[12] The basic idea is that all NP arguments, including QNPs in argument positions, move to certain positions to have their structural case checked by relevant heads.[13] Consider (10).

(10) a. Someone attended every seminar.

b. [$_{AgrSP}$ someone [$_{TP}$ T [$_{AgrOP}$ every seminar [$_{VP}$ (someone) [attended (every seminar)]]]]]

Hornstein (1999a: 49)

In (10), both the subject QNP *someone* and the object QNP *every seminar* move to the appropriate structural positions to have their cases checked, and for each QNP, either one of the two copy positions can count as its scope position.[14] A great

[12]The reasons why Hornstein does not regard QR as an A-bar movement are not only the characteristic locality constraints of QNPs as we have discussed here, but also various theory-internal considerations that come from the shift from GB to Minimalism. I abstract away from such theory-internal considerations. See the references suggested in the main text.

[13]In (10), the AgrS head checks the structural case of the subject NP and the AgrO head checks the case of the object NP. Today's Minimalism does not use AgrS and AgrO anymore. However, the exact identities of the case checking heads are not relevant to the current discussion.

[14]I represent all the copy positions other than the head positions by using parentheses.

merit of Hornstein's analysis is that the tensed-clause locality constraints on QNP scope fall out naturally. Without going into the exact structural configurations of case checking, based on the assumption that grammatical statuses of arguments (again, such as subjecthood, objecthood etc.) are resolved within each (tensed) TP, the scope of a QNP is predicted to stay there. Moreover, unlike QR as an A-bar movement, Hornstein's analysis can naturally explain why QNPs and NPs occupy the same positions in the PF string at the observation level. In this analysis, all the argument NPs, whether they are names, definite descriptions, or QNPs, move to the same structural positions to have their cases checked.

However, Hornstein's A-movement analysis has some fundamental problems. I do not review Hornstein's analysis in detail in this thesis, but I briefly mention two problems which indicate that QNP scope is not a mere side effect of some syntactic operation that applies to NP arguments in general.[15]

First, there is a semantic compositionality problem that has been pointed out in Cormack and Smith (2002). First, I repeat Hornstein's structure for a control sentence from Hornstein (1999b). (11) below is his (19) on p. 79.

(11) a. John hopes to leave.

b. [$_{IP}$ John [$_{VP}$ John [hopes [$_{IP}$ John to [$_{VP}$ John leave]]]]]

(11b) is Hornstein's structure for the subject control sentence in (11a). The important copy is the tail copy inside the embedded VP headed by the verb *leave*. In order to explain the inverse scope reading in the sentence in (12a), Hornstein would need to interpret the copy which is in the same VP internal position.

(12) a. A student tried to stand near every visitor. $a > every$; $every > a$

b. A student [$_{VP}$ (a student) tried to [$_{VP}$ [$_{VP}$ (a student) stand [near every visitor]]]]

In the structure in (12b), I ignore the copy of the indefinite in the Spec of the embedded IP, which is present in Hornstein's structure in (11b). I also put parentheses around all the copies other than the head copy which corresponds to the PF position.

To explain the inverse scope reading $every > a$ in (12), Hornstein would need to use the tail copy of *a student* as the scope position of the indefinite. The PP *near every visitor* modifies the embedded verb *stand* and thus, cannot take scope over the matrix VP headed by *try*. Let us assume that this means that for the inverse

[15]See Johnson (2000) for several problems with Hornstein (1995), though it is not clear if all of his arguments apply to a later version of Hornstein's analysis, such as Hornstein (1999b).

scope reading, the tail copy is interpreted as the existential quantifier. Now, the question is how to interpret the other, higher copies. Hornstein cannot interpret all the copies as QNPs. If he did so, he would get the wrong reading. To make the point clearer, let me ignore the copy just to the left of the main verb *try*, and interpret the head copy and the tail copy of the indefinite, both as existential QNP. Then the sentence would roughly mean that a student tried so that a (different) student would stand near every visitor, which is obviously not what (12a) means. Copy theory of movement normally stipulates that we interpret only one copy as a quantifier, while inserting a variable in the other copy position(s) which would then be bound by the quantifier which is interpreted at the place of the chosen copy. But this would not work either. Note that to explain the inverse scope reading, we have to choose the tail copy inside the embedded VP as the interpretation site for the QNP. Then we would need to insert a variable in the position of the external argument of the matrix verb *try*. But this variable cannot be bound by the QNP which is lower than this position. If we interpreted the head copy (that is, the copy in the PF position) as a QNP and let the quantifier in that position bind this variable inside the main clause VP, then we would again get the wrong meaning as I sketched above.

Another problem with Hornstein's account concerns pronominal binding. Consider (13).

(13) a. A student met every English lecturer. $\exists > \forall$; $\forall > \exists$

b. Every teacher$_i$ seemed to his$_i$ students t$_i$ to be good enough.

c. *[$_{DP}$ A student that he$_i$ once supervised] met every lecturer$_i$.

d. [$_{AgrS}$ a student [that he$_i$ once supervised]
[$_{TP}$ T [$_{AgrO}$ **every English lecturer$_i$**
[$_{VP}$ (**a student [that he$_i$ once supervised]**)
met (every English lecturer$_i$)]]]]

Generally, A-movement feeds the binding relation, as is shown by (13b). In Hornstein's analysis, the two QNPs in (13c) A-move to Spec AgrS and Spec AgrO respectively. In order to generate the object wide-scope reading, $\forall > \exists$, the tail copy of the subject indefinite (inside the VP) and the head copy of the object universal would have to be interpreted. Because both the movements are A-movement, we might naturally assume that the movement of the object universal feeds the binding of the pronoun in the VP internal subject indefinite. However, a bound pronoun reading is not acceptable with (13c). Considering that the object wide

scope reading is available in (13a) and even in (13c) if the pronoun *he* is interpreted referentially, such as denoting John, it is not clear why the inverse scope reading does not license the binding relation. Unlike an A-bar movement analysis as in May, we cannot resort to Weak Cross Over violation which applies to A-bar movement, not to A-movement. Thus, Hornstein would need an extra stipulation such that a pronoun in the VP internal copy of the subject QNP cannot be a candidate for binding.

Though A-movement analysis of QNPs is attractive in that it can explain the characteristic locality constraints on QNP scope, the rigid ordering produced by the operation that fixes the argument status of each NP is not a good way of explaining the flexibility associated with scope relations. It is more natural to assume that A-movement feeds the QNP scope which is computed using an independent mechanism.

In the next section, I briefly discuss the indefinite, whose scope allegedly can cross the local tensed clause. I argue that the existential scope of the indefinite is also clause-bound, and it is a certain kind of specificity that gives the impression of the exceptional wide scope.

1.4 Indefinites

Whereas the scope of universal QNPs is roughly clause bound, the scope of indefinites can allegedly cross a tensed clause, as is shown in (14)~(15).[16] Compare (14) and (15).

(14) a. Two teachers reported that *every student* smoked at school.

b. An impossible reading of (14a): For each student x, a different pair of teachers reported that x smoked at school.

c. A possible reading of (14a): The same pair of teachers reported that every student (in the relevant domain) smoked at school.

(15) a. Two teachers reported that *a student* smoked at school.

b. A possible reading A of (15a): For one and the same student, two teachers reported that that student smoked at school.

c. A possible reading B of (15a): For each of the two teachers, there is a different student about whom the teacher reported that the student smoked.

[16]See Carpenter (1997:255) for a summary of typical scope islands for quantifiers in general and the exceptional behaviors of indefinites in this regard.

(14a) has only the narrow scope reading of the universal QNP *every student* relative to the indefinite *two teachers* in the main clause. The wide scope reading of the universal, as in (14b), is not easy to get with this string. In contrast, if we place an indefinite *a student* in the position of the universal QNP, as is shown in (15a), apparently, we can get the 'wide scope reading' of *a student* relative to the main clause indefinite *two teachers*, as in (15b). If (15b) were really the wide scope reading of the embedded indefinite *a student*, then we would need to modify our algorithm to compute QNP scope.

However, as Ruys (1992) and Winter (2001) show, there is an important difference between the (impossible) reading in (14b) and the reading in (15b). In (14b), the pair of teachers covary with each student. In other words, for each member of the set of students, we think of a different pair of teachers. In contrast, this kind of covariation is not involved in (15b). In fact, (15b) is not an appropriate example to test the availability of this inverse covariation,[17] because the embedded indefinite is singular. To test the 'scope' of the indefinite in this way, we have to use a plural indefinite, as in (16).

(16) a. Two teachers reported that *three students* smoked at school.

b. An impossible reading A of (16a): For each of the three students, a different pair of teachers reported that he (or she) smoked at school.

c. A possible reading B of (16a): For each of the two teachers, there is a different triplet of students about whom the teacher reported that they smoked.

In (16a), we have placed a plural indefinite *three students* instead of the singular indefinite *a student* in (15a). With (16a), we cannot get the inverse covariation reading as I explained above. That is, (16b) is not a possible reading of (16a). If we identify this inverse covariation reading with the wide scope reading of the indefinite *three students* in (16a), then the lack of the inverse covariation reading in (16a) suggests that (15a) does not provide evidence for the wide scope reading of the indefinite either. Just as the inverse covariation or the inverse scope reading of *every student* is not available with the string in (14a), such a reading is not available in (16a) with regard to the plural indefinite *three students.*

In chapter 9, I explain the alleged wide scope reading of the indefinite (15a) in terms of a domain restriction that applies to the nominal restriction set of the indefinite, following Schwarzschild (2002). Informally, when the set of students is

[17] 'Inverse' in the sense that the linear left element, *the pair of teachers*, covary with (each member of) the linear right element *three students*)

pragmatically restricted to a singleton set in (15a), the sentence can only be about one student (for both the teachers), even if the scope of the existential quantifier stays inside the tensed clause. In the same way, when the pragmatics restricts the set of students so that it has only three members in (16a), then the sentence can only be about the same triplet of students, even if the existential scope of the indefinite stays inside the tensed clause. We find out that sometimes (though not every time) we have to restrict the domain in a different way for some other element in the sentence. To accommodate such data, I argue that indefinites are equipped with extra argument slots that can be bound by some other operator in the sentence. As we see in chapter 9, the binding of this extra argument slot can be explained by using the same mechanism that is used for *bound* pronouns.

1.5 Type Logical Grammar

This section introduces Type Logical Grammar (TLG) with some linguistic motivations. I do not provide a historical development of TLG from Categorial Grammar, which is normally assumed to have been founded by Ajdukiewicz (1935). See Versmissen (1996) and Bernardi (2002) for the history of Type Logical Grammar.

Categorial Grammar assigns categories to lexical items and we can read the derivation off the lexically specified categorial information by way of a set of syntactic rules which tell us how to combine the categories. Categorial Grammar is sometimes seen as an algorithm for reading the model theoretic semantic denotations directly off the phonological strings of natural language.[18] However, Type Logical Grammar, which is an instantiation of Categorial Grammar as a logical inference system, sees the categorial calculus in a neutral way between the phonological side and the semantic side, which I call PF and LF for convenience. In TLG, syntactic categories are seen as (categorial) formulas, which are made out of **atomic formulas** such as NP and S, and **complex formulas** such as $NP\backslash S$ and $S/(NP \bullet NP)$, which contain (formula) connectives such as '$\backslash$' and '$\bullet$.' The logical grammar system then explains the grammaticality in terms of the derivability of categorial formulas from structured configurations of formulas. Phonological strings and semantic representations are paired by way of interpreting such syntactic proofs as phonological and semantic objects. Lambek Calculi, such as L and NL (cf. Lambek (1958), Lambek (1961)), are the basis of TLG.

Though the exact rules of the grammar are provided later, consider (17) as an

[18] See Montague (1970) or Steedman (2000b), for example.

informal example.

(17) a. Sequent: Antecedent $\vdash$ Succedent

b. $(A, A\backslash B) \vdash_{TLG} B$

c. $(B/A, A) \vdash_{TLG} B$

d. $(A, B) \vdash (A \bullet B)$

Each sequent in the form of (17a) expresses the derivability of a categorial formula in the succedent from the given structural configuration of formulas in the antecedent.[19] Informally, (17b) means that by using each of the two categorial formulas A and $A\backslash B$ in the antecedent exactly once in the suggested structural configuration (which is explained later), the grammar can infer/derive exactly one occurrence of the formula B in the succedent. (17c) informally means that by using each of the two categorial formulas B/A and A exactly once in the suggested structural configuration, the grammar can infer/derive exactly one occurrence of the category B. (17d) informally means that given exactly one occurrence of each of the two categories A and B in the suggested configuration, the grammar can infer/derive exactly one occurrence of the complex category $(A \bullet B)$. The exact interpretations of the commas and other configurational information are explained later, but an important point of this system is that it is inherently symmetric. That is, for each of the connectives, $\backslash, /$ and $\bullet$ in (17), the grammar can either introduce it or eliminate it during the inference. This symmetry allows us to derive from the basic rules of the categorial connectives certain syntactic rules that are often stipulated as axioms, such as 'type raising' and 'function composition,' which are defined separately from the basic rule '(directional) function application' in a grammar such as Combinatoric Categorial Grammar in Steedman (2000b).

As we see shortly, the grammar defines successively more complex categorial formulas from a limited number of basic categories, and the derivability relations between these formulas are inferred by using a limited number of syntactic rules. These categorial formulas and syntactic derivations/proofs are then interpreted in the formal models of both phonological objects and semantic objects. In this way, we can see Categorial Grammar as an instantiation of Chomskian Universal Grammar which pairs PF and LF representations. We also see that TLG can maintain the Chomskian assumption of Lexical Inclusiveness as in Chomsky (1995).

A particular version of Lambek Calculus which I adopt as the basic logical grammar system is NL, which is non-associative and non-commutative. Though

[19]The interpretation of the formulas in the antecedent depends on which variant of Lambek Calculus we choose.

NL provides significant constraints on PF-LF mapping, it is too restrictive to explain the whole set of natural language data. We need to introduce some structural association and permutation rules to accommodate various language data. However, introducing such structural rules in the initial uni-modal setting of NL collapses the logical grammar into the associative and commutative Lambek Calculus, LP, which overgenerates, as described in chapter 3. Because of this, I extend the grammar by adding structural rules under the control of modality in that chapter. There, I adopt Moortgat's Multi-modal Type Logical Grammar as in Moortgat (1997).

In this chapter, after providing definitions of the essential theoretical building blocks of Categorial Grammar, I provide a Natural Deduction (ND) presentation of Type Logical Grammar. ND is less convenient for showing the structural rules compared with an inherently symmetric proof presentation system, Gentzen Sequent Presentation, which I explain in chapter 3. However, ND is more convenient when we show compositional derivation of concrete semantic terms from the lexical levels, using the syntactic derivations. ND presentation is also convenient because of its apparent similarity to standard syntactic tree representations.

Because I do not use structural rules until chapter 3, I wait for that chapter before I explain Gentzen Sequent presentations.

1.5.1 Categories and Types

Type Logical Grammar is based on syntactic categories as are recursively defined in (18) as the set of categorial formulas, F. The reasons why categories are called formulas will become apparent as we proceed.

(18) F: the set of (categorial) formulas.

a. The set of atomic formulas, At, is a proper subset of F,
where $At = \{NP, S, N, \ldots\}$

b. For all A, B.
If $A \in F$ and $B \in F$, then $(A\backslash B) \in F$, $(B/A) \in F$ and $(A \bullet B) \in F$.

c. The set of categorial formulas F is the smallest set obtained by means of (18a) and (18b).

At this stage, the atomic formulas (that is, the atomic categories) are NP, S and N. Informally, NP corresponds to DP in current Minimalism. That is, NP is the category for expressions such as *Jack*, *that boy* and *those two cakes*. S is

for sentential expressions, such as *Jack smokes* and N is for common nouns such as *boy* and *girl*. Atomic categories are not the same as lexical categories. The reason why atomic formulas are deemed 'basic' is partly for semantic reasons. Expressions of category NP denote individual terms (of type e) in the semantics, as we see later, and expressions of category S denote propositional formulas (of type t) in the semantics. Individuals are building blocks of the semantic models (i.e. each semantic model comes with its domain of individuals, D_e, as we see shortly). Truth values are 'goals' of the model theoretic semantics. By treating other expressions as functors that map individuals to truth values (possibly by way of intermediate functors), model theoretic semantics achieves compositional derivations of truth values. Assuming a tight correspondence between the basic categories in the syntax and the basic types of the semantics, we can read off the compositional model theoretic semantics from the natural language strings, as in Montague's system.[20] The atomic formula N (for *boy* and *girl*) is a compromise in this regard, because its semantic type, as we see later, is type (e,t) which is for semantic functors that map individuals to truth values. N could be replaced by $(NP\backslash S)$, but this would make syntactic derivations look more complex. It would also overgenerate the grammatical expressions without further constraints. Thus, I maintain N as a basic category.

In (18b), the **functor categories** in the forms of $(A\backslash B)$ and (B/A) select **argument/input category** A directly to the left and to the right respectively. In other words, the functor category $(A\backslash B)$ or (B/A) has an **argument selection/slot** of category A. This argument slot is **saturated** or **filled out** by an argument A, when the functor and the argument is merged in the required structural configuration, as in $(A, A\backslash B) \vdash B$, or in $(B/A, A) \vdash B$ (cf. (17) above), which we formulate as a syntactic rule later. B in the functor categories (B/A) or $(A\backslash B)$ is called the **value category**, though I often informally call it the **output category** of the functor.

The binary connective $\bullet$ concatenates two formulas, preserving the linear order between the two. The syntax and semantics of the binary connective $\bullet$ are provided later. (18c) means that nothing other than those obtained by means of (18a) and (18b) is a (categorial) formula. Given a finite set of atomic formulas At as in (18a), we can derive a denumerably infinite set F of categorial formulas by way of recursive applications of the rules in (18b), creating successively more complex formulas, such as $(NP\backslash S)$ and $((((NP\backslash S)/NP)/NP)\ldots/NP))$. In application, not all of these

[20] Cf. Montague (1970).

potential complex categorial formulas are instantiated in a particular language, but this recursive definition of complex categories explains the productivity of the syntax at the level of the building blocks of the grammar system.

The parentheses around each complex formula indicate that the binary connectives in the logical grammar that I have adopted are non-associative. That is, $(B\backslash C)/A \neq B\backslash(C/A)$. The grammar is also non-commutative, so $(A \bullet B) \neq (B \bullet A)$.

Corresponding to the syntactic categories as defined in (18), semantic types are recursively defined as in (19).

(19) *Sem*: the set of semantic types.

a. The set of basic types, BS, is a proper subset of *Sem*, where $BS = \{e, t\}$

b. If a and b are members of *Sem*, then (a, b) and $(a \times b)$ are semantic types.

c. The set of types *Sem* is the smallest set of types obtained by means of (19a) and (19b).

The basic types are type e for expressions denoting individuals, and type t for expressions denoting truth values. The functor type (a, b) would normally be represented as $< a, b >$, but for space reasons, I use the former notation. As with functor categories, functor expressions of type (a, b) have an **argument slot/selection** of type a and have b as the **value** type. The binary connective $\times$ as in $(a \times b)$ binds more strongly than (a, b). Thus, $(e \times e, t) = ((e \times e), t)$, and I omit the parentheses around $(a \times b)$ when it is not misleading. I also omit commas when the types are obvious, such as (et) and (e(et)).

Given (18a) and (19a), we provide a function *Type* which maps categorial formulas to semantic types as in (20).

(20) *Type*: $F \rightarrow Sem$.

a. Atomic Mapping. $Type(NP) = e$; $Type(S) = t$; $Type(N) = (e, t)$

b. For all $A, B \in F$, $Type(A\backslash B) = Type(B/A) = (Type(A), Type(B))$

c. For all $A, B \in F$, $Type(A \bullet B) = (Type(A) \times Type(B))$

As we see later, the syntactic calculus is solely based on categorial formulas as provided in (18). However, given the functional mapping from categorial formulas to semantic types as in (20), whenever the syntactic derivation converges, type compositionality holds at LF.

Given the semantic types as above, I explain the typed lambda terms that I use to describe the model theoretic meanings of natural language expressions and how they can be derived by way of categorial calculus.

1.5.2 Typed Lambda Terms

As LF representations, I use typed **lambda terms** (which I call typed **logical expressions** as well), as defined in (21). The presentation is based on Girard's in Girard et al. (1990: 15-16).

(21) a. For each $a \in Sem$ as in (19), there is a possibly empty set Con_a of constant terms, $c'_1 \ldots c'_n$ of type a.

b. For each $a \in Sem$, there is a denumerably infinite set of variables, $x_1, \ldots, x_n$, of type a. All variables are terms.

c. If α is a term of type (a, b) and β is a term of type a, then $(\alpha\beta)$ is a term of type b.

d. If x is a variable of type a, and ϕ is a term of type b, then $(\lambda x.\phi)$ is a term of type (a, b).

e. If u and v are terms of type a and b respectively, then $(u \bullet v)$ is a term of type $(a \times b)$.

f. If α is a term of type $(a \times b)$, then $(\pi_1\alpha)$ is a term of type a and $(\pi_2\alpha)$ is a term of type b.

I attach "′" (prime) to constant expressions, as opposed to variable expressions. For convenience, I use certain lower case letters as variables of certain types. For example, x, y, z, u, v are normally used as type e expressions, whereas higher case letters are assigned various types, such as A, B as type (et) variables and V as type (e(et)). But I sometimes break the patterns, so I specify the types of constant and variable expressions when they are used as expressions of different types than the above ones.

Though the logical language might be lacking in constant expressions for some of the semantic types, for the semantic types that we discuss in the thesis, we can always represent the semantics of language expressions as constants in the lambda language in (21). For example, the language has meg', $jack'$ of type e for *Meg* and *Jack*. And it has variously typed functor expressions such as $smoke'$ of type (et) for *smoke* or $like'$ of type (e(et)) for *like*. We can also assign functor constants to quantificational determiners *some* and *every*, as we see shortly.

Sometimes, it is convenient to explicitly spell out the argument selections of functors, and (21d) allows us to do that. Thus, the above two functors can be expressed as $(\lambda x.(smoke'x))$ and $(\lambda x.(\lambda y.((like'x)y)))$, respectively, though I later abbreviate some of the parentheses for readability.

Corresponding to categorial formulas in the form of $(A \bullet B)$ where A and B can be any categories, the lambda language has expressions in the form of $(u \bullet v)$, where u and v can be any terms. To get back each sub-component of this complex term, we have the two operators, π_1, which selects the left member u out of $(u \cdot v)$, and π_2, which selects the right subcomponent v out of $(u \cdot v)$.

Though (21) provides the exact definition of the well formed lambda terms, for notational convenience, I often omit parentheses when it is not confusing. For example, $(\alpha\beta)$ may be represented as $\alpha\beta$, $(\lambda x.\phi)$ as $\lambda x.\phi$, $(a \bullet b)$ as $a \bullet b$, and $(\pi_n \alpha)$ as $\pi_n \alpha$. When I drop the parentheses, I assume the left associativity of the internal structures. So $\alpha\beta_1\beta_2\beta_3$ represents $(((\alpha\beta_1)\beta_2)\beta_3)$.

On the other hand, I sometimes add extra pairs of parentheses to explicitly show the functor-argument (or operator-operand) relations between an n-ary functor and its n number of arguments, as in $\alpha^n(\beta_1)(\beta_2)\cdots(\beta_n)$. Taking π_1 as an operator with α being its operand, this notation will give us $\pi_1(\alpha)$ (cf. Carpenter, 1997). In this notation, a ternary functor α^3 will be represented as $\lambda x.\lambda y.\lambda z.(\alpha^3(x)(y)(z))$, which would be $(\lambda x.(\lambda y.(\lambda z.(((\alpha^3 x)y)z))))$, if we follow (21) strictly. When the functor-argument relations are obvious, I sometimes omit the parentheses even when left associativity could generate a wrong structure, as in $\lambda x.\lambda y.like'(x)(y)$ or, if I do not add parentheses around the argument slots, $\lambda x.\lambda y.like'xy$, with the constant $like'$ of type (e(et)) and variables x, y of type e. Normally, this does not cause a problem because my analysis does not involve vacuous binding of a variable by a lambda operator, as in $(\lambda x.(\lambda y.like'x))y$, though the rule in (21d) does allow $(\lambda y.like'x)$ as a lambda term. I normally either insert variables into all the argument slots of the functor and bind them by lambda operators, as in $\lambda x.\lambda y.like'xy$ or otherwise use completely η reduced terms (see (22) below for η reduction) in which no lambda operators appear, such as $like'$.

One lambda operator may bind more than one variable in its scope, as in $\lambda x.like'xx$, which would represent the meaning of the VP *like himself* in *Tom likes himself*. Generating this lambda term in the syntax, however, requires either a special (higher order) functor category which identifies more than one argument slot of the lexical verbal functor $\lambda x.\lambda y.like'xy$ by taking this verbal functor as its argument, or application of a modally controlled contraction rule which we see an example of in chapter 3.

Given (21), there are primary and secondary conversion rules between terms as in (22a).

(22) a. Primary conversions:

i) β reduction (β red): $(\lambda x.\phi)a \Rightarrow_{\beta red} \phi[x :\rightarrow a]$,

where a is free for x in ϕ.

ii) Projection: $\pi_1(u \bullet v) = u$ $\pi_2(u \bullet v) = v$

b. Secondary conversions:

i) η reduction: $\lambda x.Px = P$

where x is not free in P.

ii) Pairing: $(\pi_1\alpha \bullet \pi_2\alpha) = \alpha$

In (22a), $\phi[x :\rightarrow a]$ means that the occurrence(s) of x inside ϕ is/are replaced by a. The statement that a is free for x in ϕ means that replacing x with a does not lead to binding of a by an operator in ϕ. η conversion in (22b) means that provided we do not use free variables in our semantic representations, we can represent all the semantic compositions by using only constant terms which have direct correspondents in model theoretic objects, as we see later.

Typed lambda terms are expressions of a (higher order) logical language with its own syntax and semantics. The expressions of this logical language are interpreted as model theoretic objects as in the following. For more details, see Gamut (1991). For (23e) and (23f), Girard (1990: 15-16).

(23) a. For each term a of type e, $[\![a]\!]^{M,g} \in D_e$

b. For each propositional formula ϕ of type t, $[\![\phi]\!]^{M,g} \in D_t$

c. For each term α of type (a, b), $[\![\alpha]\!]^{M,g} \in {D_b}^{D^a}$

d. For each α of type (a, b) and for each term β of type a, $[\![\alpha\beta]\!]^{M,g} = [\![\alpha]\!]^{M,g}([\![\beta]\!]^{M,g})$, where $[\![\alpha\beta]\!]^{M,g} \in D_b$.

e. For each term u of type a and for each term v of type b, $[\![(u \bullet v)]\!]^{M,g} =< [\![u]\!]^{M,g}, [\![v]\!]^{M,g} >$, where $[\![(u \bullet v)]\!]^{M,g} \in D_{a\times b}$

f. For each term α of type $(a \times b)$, $[\![\pi_1\alpha]\!]^{M,g}$ is the first member (or first projection) in the ordered pair $[\![\alpha]\!]^{M,g}$ and $[\![\pi_2\alpha]\!]^{M,g}$ is the second member/projection of the ordered pair, $[\![\alpha]\!]^{M,g}$. Thus, $[\![\pi_1\alpha]\!]^{M,g} \in D_a$ and $[\![\pi_2\alpha]\!]^{M,g} \in D_b$

$[\![\cdot]\!]^{M,g}$ is an interpretation function that maps lambda terms as in (21) to model theoretic objects. The denotations of lambda terms may vary with each model M, and this variability is implemented by way of the interpretation function I_i which comes with each model M_i (where the index i indicates the correspondence between the model M_i and the interpretation function I_i) and which maps all the

constant terms to certain denotations in the model. Normally, the interpretation rules of constant terms are provided in two steps, as in $[\![jack']\!]^{M,g} = I(jack') = JACK$, where $JACK$ is the meta-representation of the individual Jack himself. Denotations of (free) variables may vary within the same model and we need to use variable assignment functions $g_1, \ldots, g_n$, which map variables to model theoretic objects. Within the same semantic model, the denotation of a variable may vary depending on which variable assignment function to choose. For example, in a model M_1 whose domain includes JACK, BILL and MEG, among others, the denotation of a free variable x may vary as in $[\![x]\!]^{M_1,g_1} = g_1(x) = JACK$, $[\![x]\!]^{M_1,g_2} = g_2(x) = BILL$ and $[\![x]\!]^{M_1,g_3} = g_3(x) = MEG$. As I indicate below, because of this variability of the interpretation of free variables within the same model, use of free variables in the semantic representations makes it difficult to use lambda terms as meta representations of model theoretic objects themselves. For this and for other reasons (see Jacobson (1999)), I do not use 'free' variables (as opposed to variables bound by operators which can be mostly eliminated from our LF representations[21]) in the LF representations.

I omit the details of the semantic rules. See Gamut (1991) for such details.

Coming back to (23), the basic domains of interpretation in Model Theoretic denotations are D_e, the set of individuals, and D_t, the set of truth values, $\{1, 0\}$ (1=True, 0=False). Type e expressions denote members of D_e and type t expressions denote members of D_t.

In (23c), each functor expression α of type (a, b) is interpreted as a function that maps members of D_a to members of D_b. If the argument slot of this functor is saturated by an argument β of type a, then we interpret the result of this merge as $[\![\alpha]\!]^{M,g}([\![\beta]\!]^{M,g})$ (which is of type b), as in (23d). In the model, this corresponds to the application of the functor $[\![\alpha]\!]^{M,g}$ to its argument $[\![\beta]\!]$, and thus, I call the merge of the functor term α with its argument term β in the derivation of $\alpha\beta$ a **function application** for convenience. Here, I have put a pair of parentheses around the argument to explicitly show the functor-argument relation, such as $Functor(Argument)$. I sometimes use this notational convention in typed lambda terms as well, as in $\alpha(\beta)$.

In (23e), the binary operator $\bullet$ is interpreted in such a way that it pairs two model theoretic objects a, b to form an ordered pair $< a, b >$. (23f) is provided so that the model theoretic objects can completely cover the logical language ex-

[21] Without variables, it is not easy to represent multiple scope readings, unless we use 'combinators' in the semantic representations. But I do not go into details about this relation between 'variable-free' semantics and Combinatory Categorial Grammar which Jacobson uses.

pressions in (21). That is, just as the operators π_1 and π_2 select the left and the right member of the complex expression α of type (a, b) at the level of the logical language, at the level of denotations we can select the left member of the complex denotational object $[\![\alpha]\!]^{M,g}$, so that the selected element $[\![\pi_1\alpha]\!]^{M,g}$ will be a member of D_a and we can select the right member of the complex denotational object $[\![\alpha]\!]^{M,g}$ so that the selected element $[\![\pi_2\alpha]\!]^{M,g}$ will be a member of D_b.

Other than the variable expressions in the lambda language in (21), we can postulate a one-to-one mapping between typed lambda terms as in (21) and model theoretic objects as in (23).[22] Variables, on the other hand, do not have their direct correspondents at the level of denotations, and thus, must be interpreted by way of variable assignment functions. Other than conceptual and technical problems that Jacobson has mentioned as problems of using free variables in the semantic representation (see Jacobson (1999) for example), this lack of direct correspondents of variables in semantic models makes it difficult to use the typed lambda terms as direct representations of model theoretic objects. Thus, I assume that LF representations do not use 'free variables' (as opposed to bound variables, most of which can be omitted via η reduction as in $\lambda x.\lambda y.like'xy \Rightarrow_{\eta red} like'$[23]). In this way, we can abstract away from the question of whether the lambda language represents an intermediate level of language representations between the (structured) phonological objects and the model theoretic denotations.

According to the way I see the grammar system, the syntactic representations (i.e. categorial formulas and binary structures made out of them) which we use to show categorial calculus are only notational tools to explain how the categorial proof works. Interpretations of the syntactic categories and proofs at PF and LF are different from such syntactic representations; they are supposed to approximate the real objects that the syntax pairs by way of syntactic proofs/derivations. In this view, the meta variables that I use to decorate Gentzen Sequent proofs in chapter 2 onwards to show the derivations of concrete (closed) lambda terms (as LF representations) have the same status as the tools used in the categorial proof presentations. That is, they are the tools we use to show how the concrete lambda terms can be derived by using Gentzen Sequent proofs, and thus, use of free (meta) variables in the proof presentations does not mean that the actual lambda terms that we derive at LF contains free variables, as I come back to in chapter 3.

[22] I abstract away from the treatment of necessarily co-extensive expressions

[23] As I wrote above, multiple scope readings are harder to represent without using variables. See the logical forms in (24d)~(24e) below, for example. This means that without variables the compositional derivation of multiple scope readings from a phonological string becomes harder to represent.

As an example of the use of typed lambda terms to represent the semantics of natural language expressions, I show that the lambda expressions as in (21) can represent the two scope readings of an English sentence *Some boy loves every girl.*

(24) a. Some boy loves every girl.

b. *some*: $\lambda A.\lambda B.some'AB$ (or $\lambda A.\lambda B.some'(A)(B)$)

c. *every*: $\lambda A.\lambda B.every'AB$ (or $\lambda A.\lambda B.every'(A)(B)$)

d. Surface scope: $\exists > \forall$

$some'boy'(\lambda y.(every'girl'\lambda x.(love'xy)))$

$=some'(boy')(\lambda y.every'(girl')(\lambda x.love'xy))$

e. Inverse scope: $\forall > \exists$

$every'girl'(\lambda x.(some'boy'\lambda y.(love'xy)))$

$=every'(girl')(\lambda x.some'(boy')(\lambda y.love'xy))$

Cf. Carpenter (1997: 237)

Types.

$some', every' : ((et),((et)t))$; $boy', girl' : (et)$; $love' : (e(et))$;

$A, B : (et)$; $x, y : e$

(24b) and (24c) show the lambda terms for the quantificational determiners *some* and *every*. For each of the scope readings in (24d) and (24e), the upper logical form uses omission of parentheses with the left association of the structure being the default. But this upper logical form does not clearly show the functor-argument relation. Thus the lower logical form for each reading adds an extra pair of parentheses, as in the schema $some'(A)(B)$ or in $every'(A)(B)$, where A, B are inserted as some type (et) expressions which denote sets of individuals. In (24), the determiners *some* and *every* are interpreted as two place functors. In this way, we can derive the generalized quantifier (GQ) interpretations of quantificational determiners as in Barwise and Cooper (1981) in compositional manners.[24]

(25) For all A, B of type (e,t).

a. $[\![some'(A)(B)]\!]^{M,g} = 1$ iff

$\{a \in D_e \mid [\![A]\!]^{M,g}(a) = 1\} \cap \{b \in D_e \mid [\![B]\!]^{M,g}(b) = 1\} \neq \emptyset$

[24]A major motivation for GQ and the higher order logical notation that is associated with it is that the logical language can then represent sentences such as *Most boys smoke* with the right model theoretic semantics, which is more difficult to do with the predicate logic language. But the issue is not directly relevant in this thesis.

b. $[\![every'(A)(B)]\!]^M = 1$ iff for all $a, b \in D_e$.
$\{a \in D_e \mid [\![A]\!]^{M,g}(a) = 1\} \subseteq \{b \in D_e \mid [\![B]\!]^{M,g}(b) = 1\}$
Cf. Barwise & Cooper (1981: 84), Chierchia and McConnell-Ginet (2000: 505).

Because I interpret type (et) expressions as functions that map individuals to truth values, rather than as sets of individuals as Barwise and Cooper (1981) does, the GQ interpretations are represented in an indirect way in (25).

Given the typed lambda expressions as in (24), and the model theoretic interpretations of the quantificational determiners *some* and *every* as in (25), we can assign a model theoretic interpretation to each sub-expression in the English sentence *Some boy loves every girl.* Though there is still an issue of interpreting a QNP object as type (et)t expression, which we will see later, the typed lambda expressions interpreted as model theoretic objects form the basis of compositional analysis of QNP scope readings by way of categorial calculus.

Being equipped with categorial formulas which are functionally mapped to semantic types which in turn specify the kinds of semantic/model theoretic objects that the grammar pairs with PF items, I now give an informal presentation of Type Logical Grammar as in Moortgat (1997).

1.5.3 TLG as PF-LF pairing algorithm

In this section, I provide an informal introduction to Type Logical Grammar (TLG), a deductive variant of categorial grammar. The basic grammar system I adopt is non-associative, non-commutative Lambek Calculus NL, as in Lambek (1961), which I adopt for its restrictiveness. NL is a good choice to express the characteristic locality constraints applicable to QNP scope. However, it turns out that NL is too rigid to explain the whole set of natural language data, which leads to the introduction of a Multi-Modal TLG as in Moortgat (1997) in chapter 3. In this section, however, I stick to NL. I use the Natural Deduction presentation of proofs/derivations, which has superficial similarity to tree representations, and thus, provides a good introduction of TLG to linguists.

Just like Classical Propositional Logic (CPL) explains our truth based inferences in terms of derivability relations between (sets of) formulas, TLG explains the mapping between phonological strings and semantic objects in terms of derivability of categorial formulas and their intended interpretations. Compare (26) with (27).

(26) a. Syntax: $p, p \rightarrow q \vdash_{CPL} q$

b. Semantics: $[\![p]\!]^M, [\![p \rightarrow q]\!]^M \models [\![q]\!]$

(27) a. Syntax: $NP, NP\backslash S \vdash_{NL} S$

b. Semantics: $[\![NP]\!]^{M_s}, [\![NP\backslash S]\!]^{M_s} \models [\![S]\!]^{M_s}$

c. PF interpretations: $[\![NP]\!]^{M_p}, [\![NP\backslash S]\!]^{M_p} \models [\![S]\!]^{M_p}$

Classical Propositional Logic in (26) is intended to explain our truth based spontaneous inference. To do this, CPL defines syntactic objects (i.e. propositional formulas) with their intended interpretations. The syntactic proof, as in (26a), is solely dependent on the syntactic rules, such as MPP (Modus Ponendo Ponens) that is used in (26a). However, because the syntactic objects/rules and their interpretations are set up in such a way that the syntax is sound and complete with regard to the intended semantics, whenever the syntactic derivation converges, it implies that the semantics comes out right. For example, the provable syntactic sequent in (26a) corresponds to the valid semantic argument in (26b).[25] Type Logical Grammar works in basically the same way as CPL. We set up the syntactic rules and their interpretations as phonological strings and as typed lambda expressions (as LF representations) in such a way that the syntactic derivation (which is solely based on categorial formulas and syntactic rules) converges if and only if the right PF strings are paired with the right LF representations. By doing this at a general level, abstracted away from particular category names, particular PF strings, or particular LF terms, we can "prove" that the grammar can always pair the right PF string with the right LF representation without showing it for all the possible pairs. This might not be especially impressive from the viewpoint of empirical linguistics, because the PF and LF representations that we are talking about here are the intended PF/LF models that the grammar system provides as the 'intended interpretations' of the grammar system[26] which are meant to match up with the categorial calculus in the first place. However, being able to show that the grammar actually works in the intended way at the level of the denotations without using particular LF expressions or PF terms makes TLG a good candidate for instantiating the Chomskian idea of Universal Grammar. If Universal Grammar is actually part of the innate knowledge of language users, as opposed

[25]The exact identity of the intended semantics for CPL is not relevant here. For convenience, we can interpret the semantic argument in (26b) as saying that whenever $[\![p]\!]^M$ and $[\![(p \rightarrow q)]\!]^M$ are both true, $[\![q]\!]^M$ is also true.

[26]For NL, the PF algebra needs to represent non-associative and non-commutative structures. NL is incomplete with respect to free groupoids. The LF semantics will be represented by the typed lambda terms that we have provided above.

to something that language users have acquired by way of exposure to the external language data, it is unlikely that the principles of UG are stated relative to particular PF strings and LF objects. In the same way, if the grammar rules are stated in terms of particular category names and constructions, such as NP, VP, S or complex NPs and adjunct PPs, then, such a grammar becomes less attractive from the viewpoint of UG. Thus, the presentation of rules using categorial meta variables, such as A, B, C as we see soon, and presentation of PF-LF pairing using meta variables for these PF-LF objects, such as a and b for PF and α and β for LF, is a-priori preferable.

Having said that, throughout the thesis, I do not aim to prove the correspondence of syntactic proofs and the semantics at a general level. Rather, I show the derivations of the PF and LF representations for particular sentences to check whether they are appropriate for explaining the natural language data. This is mostly for convenience of presentation, and because of the more empirically oriented nature of the thesis.

There are several differences between CPL and TLG. As I have already indicated above, TLG proofs are interpreted as two kinds of objects, PF and LF, rather than at one level of model theoretic structures. Because the PF and LF objects are both resource sensitive, the categorial calculus itself is set up as a resource sensitive system. Thus, whether there is one occurrence or two occurrences of a formula in the premise structure makes a difference to derivability/provability. The particular grammar system which I adopt, NL, is non-associative and non-commutative. In NL, we derive categorial formulas from binary configurations of categorial formulas. If we represent the derivability of a categorial formula in the form of a sequent, as in *Antecedent* $\vdash$ *Succedent*, then the Antecedent in NL is a structure as is defined in (28), whereas the Succedent is made out of a categorial formula. That is, NL is a variant of intuitionistic linear logic. Finally, in each inference, $\Gamma \vdash_{NL} A$, the Antecedent (which is a binary configuration of premise categorial formulas) is defined to be non-empty.

(28) The set of structures S for NL.

a. For all $A \in F$, $A \in S$.

b. For all $\Gamma, \Delta \in S$, $(\Gamma, \Delta) \in S$.

c. S is the smallest set that satisfies (28a) and (28b).

As I have indicated above, syntactic derivations in the form $\Gamma \vdash A$ are dependent on the syntactic rules, which are composed of logical rules (the rules about how to introduce and eliminate each connective) and structural rules (rules which explicitly

show the structure management properties (such as associativity and commutativity) of the grammar. I present the rules in such a way that the non-associativity and non-commutativity of the grammar system are incorporated into the rules of the connectives.[27]

First, the 'Premise introduction' rule in (29) allows us to introduce categories as premises of inference at the top of a Natural Deduction Proof.

(29) Premise/Hypothesis Introduction:

$$\underline{A} \qquad\qquad \underline{[A]_1}$$
$$\vdots \qquad\qquad \vdots$$

As far as the proof system is concerned, we can put any formula at the top of the derivation as a premise, but for linguistic applications, all of these premise formulas, except for those that are discharged later in the derivation, correspond to the categories of the lexical items, as we see shortly. Sometimes, we can introduce a premise formula by way of (29), and then discharge it. For convenience, I call such a premise formula a 'hypothetical formula.' When a hypothetical formula is discharged, we somehow have to keep track of which hypothesis is discharged at what stage of derivation. Thus, when I discharge a hypothesis, I put a square bracket with an index when I introduce it by (29), where the index shows which hypothetical formula is discharged with which rule, as we see with the $\backslash I$ and $/I$ rules below.

Next, I show the logical rules, that is, the rules about the categorial connectives. I first show the rules for $/, \backslash$. In Combinatory Categorial Grammar, $/E$ is forward application, and $\backslash E$ is backward application.

(30) NL: Eliminations rules for $\backslash, /$.

a. Syntax: For all $A, B \in F$ (the set of formulas)

$$\frac{\begin{array}{cc}\vdots & \vdots \\ A & A\backslash B\end{array}}{B}\ \backslash E \qquad\qquad \frac{\begin{array}{cc}\vdots & \vdots \\ B/A & A\end{array}}{B}\ /E$$

[27]This goes against the so-called 'Dŏsen principle,' which says, "[T]he rules for the logical operations are never changed: all changes are made in the structural rules" (Dŏsen 1988: 353). The name of the principle is from Wansing (1998: 11), which quotes exactly the same part.

b. Semantics:

$$\frac{\begin{array}{cc}\vdots & \vdots \\ \beta & \alpha\end{array}}{(\alpha\beta)}\ \backslash E \qquad\qquad \frac{\begin{array}{cc}\vdots & \vdots \\ \alpha & \beta\end{array}}{(\alpha\beta)}\ /E$$

c. Phonology:

$$\frac{\begin{array}{cc}\vdots & \vdots \\ a & b\end{array}}{(a \cdot b)}\ \backslash E \qquad\qquad \frac{\begin{array}{cc}\vdots & \vdots \\ b & a\end{array}}{(b \cdot a)}\ /E$$

In Natural Deduction presentations, all the premises of a categorial inference are placed above the horizontal line and the conclusion is placed under it. In sequents, $\backslash E$ in (30a) will be $(A, A\backslash B) \vdash_{\backslash E} B$ and $/E$ will be $(B/A, A) \vdash_{/E} B$. As we have indicated, the categories in the Antecedent of the sequents are configured into binary structures, in the form of (A, B). Structures that are made out of categorial formulas are defined recursively as in (28). Some might wonder why we do not represent the structure of the antecedent $(A, A\backslash B)$ above the bar, but as it becomes clear in application, this is not necessary, because the ND proofs normally look like binary syntactic tree representations upside down.

The functor category in the form of (B/A) requires its argument category A directly to the right, and $(A\backslash B)$ requires its argument category A directly to the left. Semantically, both cases correspond to function application. Because left-to-right directionality is not represented in the LF lambda terms, both $/E$ and $\backslash E$ lead to the same LF semantics. PF is successive non-associative merge of two items, where the left-to-right linear order between the two items is preserved at each step. I do not provide a formal definition of the intended PF structure, but PF expressions are made out of PF lexical items configured into binary structures by the PF structural connective '$\cdot$,' such as a, $(a \cdot b)$, $(c \cdot (a \cdot b))$, etc. The binary connective '$\cdot$' is non-associative and non-commutative, corresponding to the non-associativity and non-commutativity of the grammar system NL.

In order to prove soundness and completeness of NL with regard to the intended interpretations, it is more convenient to interpret each categorial formula, and each binary structure made out of formulas, as a set of model theoretic objects, as I briefly show in chapter 3. However, the exact identification of the intended interpretations of NL and the soundness and completeness proof are beyond the scope of this thesis, whose main object is to provide an initial implementation of

the grammar system which respects the linguistic data observation. For descriptive purposes, the presentation as in (30) is more useful.

As I indicated above, all the premise formulas above the top bars of the derivation other than the ones that will be later discharged correspond to lexical items (e.g., see ND Normal in (31a) below, the 'premises' are NP, $(NP\backslash S)/NP$ and NP). Thus, we can insert the lexical items at the top of the hypothesis, assuming that for each lexical category $A \in F$ which is for the lexical item Lex_A: $Lex_A \vdash A$. Because I use the corresponding English words to represent the lexical items, we can read the PF string off the derivation simply by reading the lexical items at the top of the natural deduction proof left to right, as we can see below in ND with Lex in (31b). In this notation, the ND proof would look almost like standard syntactic trees upside down, as we can see below, though the comparison may be misleading because the introduction rules which I soon explain will be represented in a different way in an asymmetrical tree presentation.

(31) a. ND Normal (with premise categories at the top):

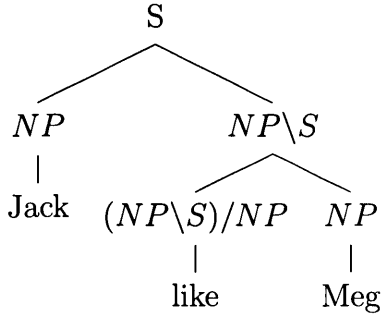
$$\dfrac{NP \qquad \dfrac{(NP\backslash S)/NP \quad NP}{NP\backslash S}\,\backslash E}{S}\,/E$$

b. ND with Lex (with lexical items at the top)

$$\dfrac{\dfrac{Jack}{NP} \qquad \dfrac{\dfrac{like}{(NP\backslash S)/NP} \quad \dfrac{Meg}{NP}}{NP\backslash S}\,\backslash E}{S}\,/E$$

The ND derivation in (31b) will be represented as the following tree.

Tree for ND in (31b)



Because we can read off the PF string by reading the 'names' of the lexical items at the top of each ND proof left to right, I normally do not explicitly show the derivation of PF strings in Natural Deduction proofs.

A fundamental difference between Type Logical Grammar and Combinatory Categorial Grammar as in Steedman (2000b) is that Type Logical Grammar is even-handed in that for each connective, we can both introduce and eliminate it during the derivation.[28] In other words, for each connective, there are both elimination and introduction rules.

Now, I present introduction rules for '/' and '\' in NL using ND presentation, followed by the introduction and elimination rules for '•.' These rules involve non-standard, speculative elements which remain to be proved.

(32) \, / Introduction

a. Syntax: For all $A, B, C \in F$:

$$\frac{\begin{array}{cc}\vdots & \vdots \\ [A]_1 & B\end{array}}{\dfrac{C}{A\backslash C}\backslash I_1} \qquad \frac{\begin{array}{cc}\vdots & \vdots \\ A & [B]_1\end{array}}{\dfrac{C}{C/B}/I_1}$$

[N.B] B is not empty for $\backslash I$ and A is not empty for $/I$.

b. Semantics:

$$\frac{\begin{array}{cc}\vdots & \vdots \\ [\beta]_1 & \dot{\alpha}\end{array}}{\dfrac{\phi}{\lambda x.\phi[\beta :\rightarrow x]}\backslash I_1} \qquad \frac{\begin{array}{cc}\vdots & \vdots \\ \dot{\alpha} & [\beta]_1\end{array}}{\dfrac{\phi}{\lambda x.\phi[\beta :\rightarrow x]}/I_1}$$

c. Phonology:

$$\frac{\begin{array}{cc}\vdots & \vdots \\ [a]_1 & b\end{array}}{\dfrac{(a \cdot b)}{b}\backslash I_1} \qquad \frac{\begin{array}{cc}\vdots & \vdots \\ \dot{a} & [b]_1\end{array}}{\dfrac{(a \cdot b)}{a}/I_1}$$

[28] In Gentezen Sequent presentation which I explain in chapter 3, even-handedness means that we can introduce each connective both in the antecedent and in the succedent of a sequent. The even-handedness between the premise side and the conclusion side of the grammar (in ND presentation) is not complete, though, because Lambek Calculus is intuitionistic, and therefore does not allow us to have more than one formula in the conclusion. See Moortgat (2007) for a grammar which pursues symmetry in a more complete way.

In (32), I have aimed to express the non-associativity of NL by limiting the applications of \I and /I to immediate sisters. In other words, we can abstract away from only one of the two elements that have just been merged in the previous stage. Whether this intuitive formulation produces the intended effects in a sound and complete way remains to be proved.

Informally, $\backslash I$ in (32a) means that if we can derive the category C from the binary configuration of the formulas (A, B), then from B alone, we can derive $A\backslash C$. $/I$ shows that we can discharge the right sister B in the premise (A, B) as a hypothetical category in the same way, deriving C/B from A alone. In (32a), the 'non-hypothetical formula' (i.e. the formula without the square brackets) is stipulated to be present whenever the rule applies, prohibiting the rule from applying just after introducing only one premise category A into the proof, for example. In (32b), $\phi[\beta :\rightarrow x]$ means that the occurrence of the sub-term β inside the term ϕ is replaced by x (which will then be bound by the lambda operator λx). In NL, $\backslash I$ and $/I$ only allow us to abstract away from one of the two sister categories, as the phonology in (32c) indicates (that is, we can abstract away either from a in $(a \cdot b)$ or from b in $(a \cdot b)$, but not from c in $(a \cdot (b \cdot c))$, to generate the string $(a \cdot b)$). In order to explain the extraction/movement phenomena, we need to introduce structural rules, as we do in chapter 3. Also, because $\backslash I$ and $/I$ take place between the sisters, they do not modify the binary (tree) configurations.

Introduction of the connective '$\bullet$' allows us to generate more structures while preserving the non-associativity and non-commutativity of the grammar system NL. '$\bullet E$' in (33b) represents the projectivity of the complex formula in the from of '$(A \bullet B)$' and introduces more than formulas in the conclusion.[29] This multiple conclusion rule might be problematic because it violates the 'maximally one conclusion' constraint on NL, which is a variant of intuitionistic logic. In (33b), the inference derives the first projection and the second projection from the complex formula, '$(A \bullet B)$' and the same applies in LF and PF. Whether the rule formulation as in (33) does the intended job in the grammar remains to be proved. This is related to the incompleteness of NL with regard to free groupoids which has been mentioned above.[30] Ideally, I should present the rule in a different way, but such a treatment involves a more complex rule formulation (see Carpenter 1997: 167). Because I do not use '$\bullet E$' anywhere in this thesis, I keep the rule presentation as

[29]The term 'projectivity' was suggested by Morrill.

[30]That is, given some assumptions about tree models, such as unique splittability of each node and seeing tree building as total functions, '$A, A\backslash(B \bullet C) \vdash A \bullet C$' becomes valid in tree models, whereas this sequent is not provable in NL. See Venema (1996) for further discussion.

in (33b), which informally presents what the connective '•' does.

(33) $\bullet I$ and $\bullet E$

a. $\bullet I$ Syntax LF PF

$$\frac{\vdots\ A \quad \vdots\ B}{(A \bullet B)}\bullet I \qquad \frac{\vdots\ \alpha \quad \vdots\ \beta}{(\alpha \bullet \beta)}\bullet I \qquad \frac{\vdots\ a \quad \vdots\ b}{(a \cdot b)}\bullet I$$

b. $\bullet E$ Syntax LF PF

$$\frac{\vdots\ (A \bullet B)}{A \quad B}\bullet E \qquad \frac{\vdots\ \gamma}{\pi_1\gamma \quad \pi_2\gamma}\bullet E \qquad \frac{\vdots\ (a \cdot b)}{a \quad b}\bullet E$$

For $\gamma = (\alpha \bullet \beta)$, we have $\pi_1\gamma = \alpha$ and $\pi_2\gamma = \beta$. PF simplified.

c. E.g.

$$\frac{C/(A \bullet B) \quad \dfrac{\dfrac{\dfrac{\vdots\ A \quad \vdots\ B}{(A \bullet B)}\bullet I}{A \quad B}\bullet E}{(A \bullet B)}\bullet I}{C}/E \quad \Rightarrow \quad \frac{C/(A \bullet B) \quad \dfrac{\vdots\ A \quad \vdots\ B}{(A \bullet B)}\bullet I}{C}/E$$

As I have explained above, • combines two formulas into a complex formula, preserving the structural configuration between the two formulas in the internal structure of the resultant complex formula. Thus, the structural configuration of the premise formulas is preserved in the output, as we can see in its successive application to four formulas.

(34)

$$\frac{\dfrac{B \quad \dfrac{C \quad D}{(C \bullet D)}\bullet I}{(B \bullet (C \bullet D))}\bullet I}{(A \bullet (B \bullet (C \bullet D)))}\bullet I$$

Because the basic grammar system is non-associative and non-commutative, which does not allow us to re-bracket the binary tree structures, the elimination rule does not play an essential role in application, beyond the preservation of the symmetry in

the grammatical inference. This is especially so if we assume that no phonological lexical items are assigned categories in the form of $(A \bullet B)$. In LF of $\bullet E$, π_1 and π_2 operators pick up the first and the second members of the complex expression α (i.e., 'projectivity' of the sign with a 'product type'). In other words, π_i maps the complex sign *alpha* into its i-th 'projection.' It is basically the same in the PF in (33b), it is only that we do not postulate the operator 'π_i' there. Assuming that the PF string always takes the form $(a \cdot b)$, whenever $\bullet E$ may apply, I leave the PF for '$\bullet E$' as in (33b), where the first projection is a and the second projection is b. Again, NL is not complete with respect to free groupoid, but I mostly ignore this complication in this thesis.

In (33c), assuming again that no lexical item has the category in the form of $(A \bullet B)$, application of $\bullet I$ followed by $\bullet E$ followed by another $\bullet I$ can be replaced by a single application of $\bullet I$, normalizing the proof. Thus, in applications, this application of $\bullet E$ is superfluous, and not necessary.

To show how the grammar pairs PF objects with LF objects relative to specific sentences, I often use tripartite lexical entries as in the following for convenience.

(35) *Jack likes Meg. Meg smokes.*

a. lexical item:
<*phonological form*; syntactic category; logical expression>

b. Meg: <*Meg*; NP; meg'>

c. Jack: $< Jack;\ NP;\ jack' >$

d. smoke: $<smoke;$ NP\S; $\lambda x.smoke'x >$

e. like: $< like; (NP \setminus S)/NP; \lambda x.\lambda y.like'xy >$
Types: $like' : (e(et));\ smoke' : (et);\ meg', jack' : e;\ , x, y : e$

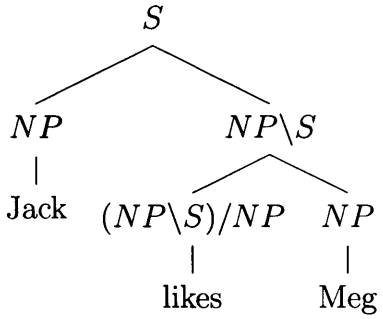
As I said above, to represent the functor argument relations explicitly, I often represent $\lambda x.\lambda y.like'xy$ as $\lambda x.\lambda y.like'(x)(y)$. In either case, the sub-terms $like'xy$ or $like'(x)(y)$ should be read in a left-associative way, that is, the former is the same as $((like'x)y)$ and the latter is the same as $((like'(x))(y))$.

Given $/E$ and $\backslash E$ above, the derivation for *Jack likes Meg* is as in (36), where Tree EX corresponding to the ND proof is placed to the right.

(36) Natural Deduction Proof

$$\dfrac{\dfrac{Jack}{NP} \quad \dfrac{\dfrac{likes}{(NP\backslash S)/NP} \quad \dfrac{Meg}{NP}}{NP\backslash S}\,/E}{S}\,\backslash E$$

Tree EX.

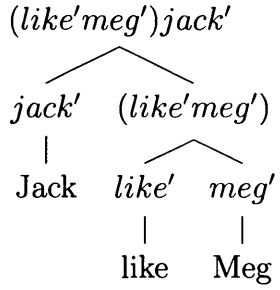


As I explained above, I have inserted phonological words above the hypothesis categories, $NP, (NP\backslash S)/NP$ and NP, left to right, but this is for presentational convenience to omit the PF derivation. Note that the binary structure as in Tree EX, which corresponds to the Natural Deduction proof above, can be read off the functor category $(NP\backslash S)/NP$. In other words, we can encode the binary structure as in Tree EX with the transitive category $(NP\backslash S)/NP$ which in turn is encoded with the lexical item *like*. Thus, Categorial Grammar can be regarded as a possible instantiation of the principle of Lexical Inclusiveness as in Chomsky (1995), which informally states that all the information that is used in syntactic derivation can be read off lexical information.

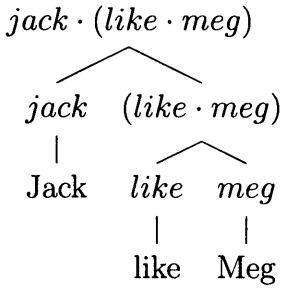
The interpretation at LF can be read off the syntactic derivation as above. I show the interpretation at PF as well, though as I wrote above, I normally omit the PF interpretation.

The interpretations of Tree EX at LF and at PF, given (35)

Semantics



PF



As I have explained informally above, the PF connective '·' configures PF units into a binary structure, such as $(jack \cdot (like \cdot meg))$, and this structure is read off the categorial calculus in the syntax. The LF terms are as defined in (21). A good

thing about ND derivation is that we can decorate each terminal node (or each premise category) with a constant logical expression for the lexical item (given my assumption that the semantic entry of no lexical item is a free variable x) and then show the derivation of concrete LF representations without using any free variables at any stage of derivation, as is shown in the semantics above. As I have indicated above, given the essential use of bound variables in a logical form such as $some'boy'(\lambda y.(every'girl'\lambda x.(love'xy)))$ for *some boy loves every girl*, we cannot eliminate the use of bound variables in the presentation of the semantics even in the ND presentation.

Both at LF and at PF, hierarchical structures in the syntax are preserved, but in the normalized logical forms as the semantic representations, left to right ordering is lost, where for each pair of a functor and its argument, the functor is placed to the left. As we briefly see later, the model theoretic structures that are well-motivated at LF and PF are different. To what degree we increase the expressive power of the LF representations and phonological representations to preserve systematic mapping at the two interfaces is an important question. As we see in the next chapter, my analysis of QNP scope compromises the tight correspondence between the syntactic proofs and the typed lambda terms as the semantic representations. From a formal viewpoint, this is a demerit. But from a linguistic viewpoint, the general proposal is not to incorporate (mostly) semantic phenomena into the syntactic system without some evidence from the PF side. After seeing that an operation like QR is mostly motivated by semantic considerations, without an obvious mark in the PF string, I introduce a type shifting operation that does not influence the syntactic structure.

1.5.4 Problems of interpreting object QNPs as of type ((et)t)

Though QNPs apparently occupy the same surface positions as NPs, they cannot be easily interpreted as type e expressions.[31] Suppose that Generalized Quantifier interpretation of QNPs as of type ((et)t), as was shown in (24) $\sim$ (25), is well motivated. According to the mapping function *Type* in (20), the categorial formulas for type ((et)t) expressions would be either $S/(NP\backslash S)$ or $(S/NP)\backslash S$. Because of the PF directionality, the first one is the category for subject QNPs (in English) and the second one will be the one for object QNPs, but in the non-associative NL,

[31] Though Kempson, Meyer-Viol, and Gabbay (2001) interprets QNPs as type e, it comes with an additional complexity of the semantic terms, by using epsilon calculus.

the derivation does not converge for a basic sentence such as *Some boy likes every girl* with the GQ categories as in (37).

(37) a. some: $< some; (S/(NP\backslash S))/N; some' >$
where, $some' := \lambda A_{et}.\lambda B_{et}.some'(A)(B)$

b. every: $< every; ((S/NP)\backslash S)/N; every' >$
where, $every' := \lambda A_{et}.\lambda B_{et}.every'(A)(B)$

c. like: $< like; (NP\backslash S)/NP; like' >$

d. boy: $< boy; N; boy' >$

e. girl: $< girl; N; girl' >$
Types: $some, every : (et)((et)t)$; $like : (e(et))$;
$boy, girl : (et)$; $A, B : (et)$

Semantic types can be inferred from the syntactic categories by using *Type* in (20), though I put them at the bottom, and also as subscripts for some of them, in (37).

(38) a. Syntax:

$$\dfrac{\dfrac{\dfrac{\dfrac{some}{(S/(NP\backslash S))/N}\quad\dfrac{boy}{N}}{S/(NP\backslash S)}\,/E\quad\dfrac{like}{(NP\backslash S)/NP}}{\cdots}\quad\dfrac{\dfrac{every}{((S/NP)\backslash S)/N}\quad\dfrac{girl}{N}}{(S/NP)\backslash S}\,/E}{\cdots}\,!!\,/E$$

b. Semantics:

$$\dfrac{\dfrac{\dfrac{\dfrac{some}{\lambda A.\lambda B.some'(A)(B)}\quad\dfrac{boy}{boy'}}{\lambda B.some'(boy')(B)}\,/E\quad\dfrac{like}{like'}}{\cdots}\quad\dfrac{\dfrac{every}{\lambda A.\lambda B.every'(A)(B)}\quad\dfrac{girl}{girl'}}{\lambda B.every'(girl')(B)}\,/E}{\cdots}\,!!\,/E$$

The derivation in (38) does not converge because no rule may merge the transitive verb and the object QNP at the step marked as !!.[32] Semantically, the transitive verb functor *like'* of (e(et)) is not the proper argument type for the object QNP of type ((et)t). If we assumed that the object QNP had the category and semantic type in (39a), which are different from the GQ categories/type above, then the derivation converges, but the scope reading can only be the surface reading.

[32] '...' means that no categorial formula can be derived in that position.

(39) a. every:

$< every; (((NP\backslash S)/NP)\backslash(NP\backslash S))/N; \lambda A.\lambda R.\lambda x.every'(A)(\lambda y.Rxy) >$

Types, $every' : (et)((e(et))(et)); R : (e(et))$

b. Syntax:

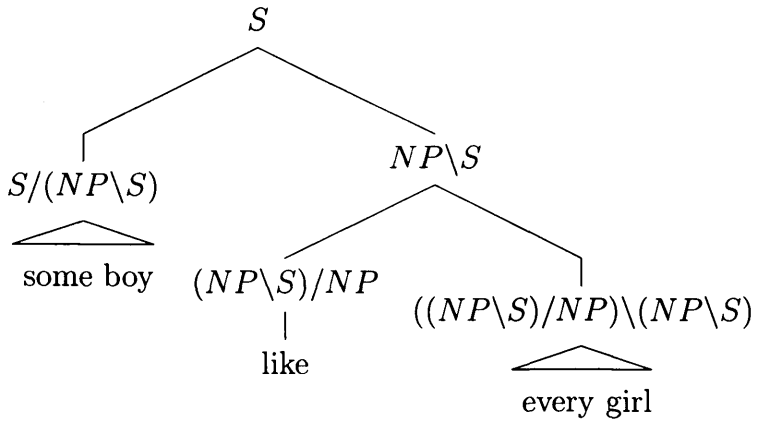
$$\dfrac{\dfrac{\dfrac{every}{(S/(NP\backslash S))/N}\quad\dfrac{boy}{N}}{S/(NP\backslash S)}\,/E\quad\dfrac{\dfrac{like}{(NP\backslash S)/NP}\quad\dfrac{\dfrac{\dfrac{every}{(((NP\backslash S)/NP)\backslash(NP\backslash S))/N}\quad\dfrac{boy}{N}}{((NP\backslash S)/NP)\backslash(NP\backslash S)}\,/E}{NP\backslash S}\,!!}{S}\,/E$$

c. Semantics:

$$\dfrac{\dfrac{\dfrac{some}{\lambda A.\lambda B.some'(A)(B)}\quad\dfrac{boy}{boy'}}{\lambda B.some'(boy')(B)}\,/E\quad\dfrac{\dfrac{like}{like'}\quad\dfrac{\dfrac{every}{\lambda A.\lambda B.every'(A)(B)}\quad\dfrac{girl}{girl'}}{\lambda R.\lambda y.every'(girl')(\lambda x.Rxy)}\,/E}{\lambda y.every'(girl')(\lambda x.like'xy)}\,/E}{some'(boy')(\lambda y.every'(girl')(\lambda x.like'xy))}$$

If we ignore the internal structures of *some boy* and *every girl*, the surface scope derivation in (39) would look like the following tree in (40). As we have seen in (38), we cannot replace the object QNP category with the GQ category, $(S/NP)\backslash S$.

(40) Derivation with object QNP of category $((NP\backslash S)/NP)\backslash(NP\backslash S)$



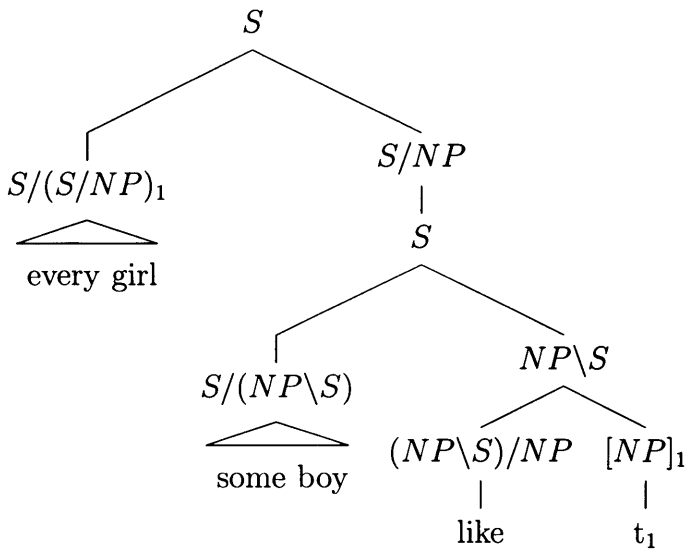
In chapter 2, I instantiate Hendriks' argument raising (AR) in categorial calculus so that we can merge the GQ category $(S/NP)\backslash S$ in the object position. Unlike Moortgat (1991), I do not incorporate Hendriks' idea into the basic rules of the grammar system, which would make the resultant grammar overgenerate the scope readings without additional locality constraints. Rather, I choose to instantiate it

as a specific phrasal rule that applies to the local functor which takes the QNP in question as an argument. This comes at the cost of postulating a rule that is not supported by the syntax at the basic level.

However, as we see in chapter 2, my analysis helps to prevent the grammar system from generating problematic syntactic structures just to accommodate QNP scope data. Also, the analysis allows us to treat QNPs with Generalized Quantifier categories as syntactic arguments of the local (verbal) functors. Thus, the same distribution of QNPs with NPs falls out naturally in this framework.

Before closing this subsection, note that postulating something similar to May's QR can solve the above problem of interpreting the object QNP as a generalized quantifier. In this derivation, both the subject and the object QNP are of category, $S/(NP\backslash S)$, which *Type* maps to the GQ type (et)t in the semantics. Consider (41).

(41) May's QR (simplified)[33]:



Because TLG cannot insert a 'trace' in the in-situ position of the object QNP, the derivation process would involve inserting a hypothetical category and then abstracting away from that category at a later stage of derivation, by way of the '/ Introduction' rule. However, as we have already seen in the rules $\backslash I$ and $/I$ in (32) above, in NL, we can only abstract away from one of the two items that have just been merged in the previous step. In order to abstract away from a category that was merged before the previous step of the derivation, we need to apply structural association rule to place the hypothetical category $[NP]_1$ in the left/right periphery

[33]For presentation reasons, I do not quantifier raise the subject QNP, because it can be interpreted as type (et)t in-situ.

position in the binary configuration of categorial formulas before we apply \ or /. Because NL is non-associative, this is impossible.

To accommodate Wh-movement and A-bar movement, we need to introduce structural association to the grammar system independently of QNP scope. And as we see in chapter 3, there is a way of doing this without collapsing NL to an associative grammar. Informally, we attach a unary operator $\Diamond$ to the hypothetical category to be discharged later, as in $\Diamond A$, or more specifically, $\Diamond\Box^{\downarrow}NP$ (instead of $[NP]_1$ above). We then use this category marked with $\Diamond$ to license an application of a structural association rule. However, the default setting for introducing structural association in this way is lack of locality constraints, as we see in chapter 3. Thus, we would need to add locality constraints to the application of this structural rule later to explain the characteristic locality of QNP scope. If we could use independently motivated island constraints on Wh-movement for QNP scope, that might justify treating QNP scope in the same way as we treat Wh-movement, but as we have seen briefly above, there is a double disassociation between QNP scope and Wh-movement in terms of the locality, and thus, setting up the lack of locality as the default status for QNP would end up adding rather ad-hoc locality measures to the grammar system just to accommodate scope data.

Thus, rather than using basically the same mechanism that we would use for Wh/A-bar movement to explain QNP scope, I choose to apply some rule specifically to the local (verbal) functor so that it can take QNP as type (et)t arguments. As we see in next chapter, this allows us to switch scope only between two QNPs which are co-arguments of this functor. Characteristic locality constraints on QNP scope can be explained in a more natural way in this analysis.

1.6 Organization of the thesis

Chapter 2 provides my basic analysis of QNP scope, which is based on Hendriks' argument raising. Keeping the scope of a QNP within the final output of the local functor turns out to be too restrictive. However, if we can form a complex predicate within each TP and apply argument slot raising to the complex predicate, then the analysis leads to the correct locality constraints on QNP scope. To generate such complex predicates requires controlled introduction of structural rules to NL. Because complex predicates should be formed only out of verbal elements within each verbal projection line (i.e. T-v-V in Minimalism), I introduce structural rules in a way that is sensitive to the merge modes. As we see in chapter 3 (and also

in chapter 7), this can naturally explain why QNP scope stays within each tensed TP.

In order to represent structural rules and their applications in such a way that we can see how the structure is modified at each stage, Gentzen Sequent (GS) presentation of proofs is more convenient than Natural Deduction presentation. Gentzen Sequent presentation also more directly represents the symmetry of the logical grammar system. Thus, in chapter 3, I first introduce Gentzen Sequent presentation, and then reconsider the proposal in chapter 2 from a more general viewpoint in GS presentation. After that, I introduce Multi-Modal Type Logical Grammar (MMTLG) as in Moortgat (1997) to introduce structural rules in modally controlled manners. After informally showing the different ways of introducing structural rules for complex predicate formation and for inherently long distance A-bar movement phenomena, such as Wh-movement, I briefly show how binding can be expressed in MMTLG. The binding mechanism that is introduced there is used in some of the later chapters. After introducing MMTLG, I first discuss two kinds of ditransitive constructions with regard to binding asymmetry (chapter 4) and the so called 'frozen scope' phenomena (chapter 5). In my analysis, scope and pronoun/reflexive binding do not depend on the same structural configuration, which I argue is empirically correct. In chapter 6, I discuss cases in which QNPs appear in PP, either PP complement or PP adjuncts. Chapter 7 deals with scope switch in infinitival constructions. To alternate QNP scope in infinitival constructions, we need to introduce controlled structural rules, but we do so in such a way that best represents the formation of complex predicates that are made out of verbal projection line elements, such as T-v-V.

In chapter 8, I briefly explain Wh-movement and show the different way of introducing structural association/permutation rules from the ones that I have used for clause internal cases, incorporating the different locality constraints that apply to A-bar movement on the one hand, and QNP scope on the other in the formalism.

Chapter 9 shows that 'exceptional scope' of QNPs that apparently crosses tensed clauses is not a matter of QNP scope. I analyze the exceptional scope of indefinites in terms of the domain restriction that can be dependent on some other element such as a universal QNP, in the sentence. Chapter 9 provides concluding remarks, and summary of important loose ends.

Chapter 2

Basic proposal

2.1 Organization of the chapter

This chapter explains how to compute QNP scope in the proposed analysis for simple English sentences. Section 2.2 explains the basic QNP scope algorithm. In section 2.2.1, I make an informal proposal about how to compute QNP scope. I show that the informal analysis leads to the correct locality constraints on QNP scope. To turn the informal idea into a proper theory, I introduce Hendriks' argument raising in Hendriks (1987) in section 2.2.2. Though Hendriks' idea limits the scope of QNP within the maximal projection of the local functor, the associative grammar system that he adopts generates a complex functor of an arbitrary size, which spoils the strict locality constraints that his analysis could have predicted otherwise. To deal with this problem, I reformulate Hendriks' argument raising in a non-associative and non-commutative grammar system NL in section 2.2.3. I call the reformulated mechanism **argument slot raising** or ASR in short. I then apply the reformulated scope system to a single clause English sentence.

In section 2.3, I show that argument slot raising for an object is not derivable in NL, which I have adopted as the base grammar system. However, if we incorporate Hendriks' original argument raising into the basic principles of the grammar system, then the resultant grammar system overgenerates without further restrictions. Because of this, I choose to define argument slot raising as a special rule that applies specifically to certain functor categories. Given the output of this special rule, we can maintain the structural asymmetry between the subject and object positions supported by the binding asymmetry between subject and object. In other words, my analysis does not generate additional syntactic structures on top of the ones which are motivated independently of QNP scope resolution. QNP

scope ambiguity is generated because of certain operations that influence lambda terms but that are not completely reflected in the categorical calculus. In this regard, my analysis loosens the tight syntax-semantics correspondence which is an important merit of Type Logical Grammar. However, the problem of incorporating QNP scope into the basic property of the grammar system is that, as we have briefly seen in chapter 1, it is not clear whether merging QNPs instead of non-quantificational NPs influences the PF structures, either in terms of overt extraction of QNPs from the PF positions of NPs (i.e. the case which corresponds to A-bar movement phenomena) or in terms of modifying c-command configurations among arguments (i.e. the case which corresponds to A-movement phenomena which put NP arguments to varying structural positions and as a result, feed or bleed binding possibilities). By incorporating QNP scope ambiguity into the basic axioms of the grammar, we would generate the extra structures that are not necessary to explain the PF strings/structures. It is at least questionable whether it is better to preserve isomorphic mapping between the syntactic structures (as generated as a result of syntactic proofs) and the LF semantics at the cost of generating all those unnecessary syntactic structures, or it is better to generate the syntactic structures which are well motivated from the viewpoint of using the syntax as PF-string generating mechanism paired with the rudimental argument structures. Thus, based on a methodological choice in which I aim to keep the number of syntactic structures to the ones which are independently motivated in terms of the generation of surface structures (which corresponds to the intended PF structures), I choose to define ASR as a special rule which is half independent of the syntactic calculus, and thus, in which we can do some more operation on the lambda terms beyond the level which is supported in the categorial calculus. Normally, such a special rule applies to 'lexical items,' but as it becomes clear when we deal with infinitive structures, we cannot limit it to lexical items. Because of this, for the moment, I treat the argument slot raising as I use in this chapter as a special 'lexical/phrasal rule,' as is distinguished from the basic axioms and the theorems that are derivable from these axioms of NL.

As it turns out, the lexical/phrasal argument slot raising together with NL is too restrictive in its locality constraints. It does not allow scope switch between the subject and the object in sentences such as *A girl tried to talk to every boy*, *A student seemed to review every paper* or *A student must have reviewed every paper*, that is, Control/Raising/Auxiliary constructions. To solve the problem would require a controlled introduction of structural rules. In section 2.4, I informally show how we can extend the basic grammar system with controlled introduction of

structural rules in such a way that the extension does not allow QNPs to take scope across tensed clauses. Introducing structural rules in different ways for complex predicate formation on the one hand (which explain Control/Raising/Auxiliary constructions) and for Wh/A-bar extraction phenomena on the other hand, the proposed system can still explain the different locality constraints that apply to QNP scope and A-bar movement. We can show exactly how we can introduce such structural rules only after the exposition of Multi-Modal Type Logical Grammar in chapter 3, but this chapter informally shows how we can extend our basic grammar NL in such a way that it can algorithmically distinguish the three linguistically different phenomena, that is, QNP scope, (certain) A-movement phenomena (plus Control phenomena) and Wh-extraction. Section 2.5 provides some concluding remarks.

2.2 Proposal

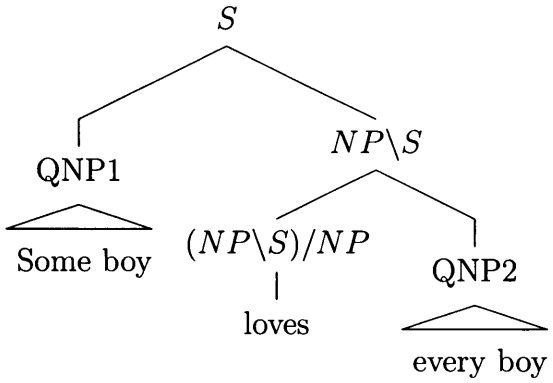
2.2.1 Informal proposal

The informal analysis that I propose for the QNP scope is as in (42).

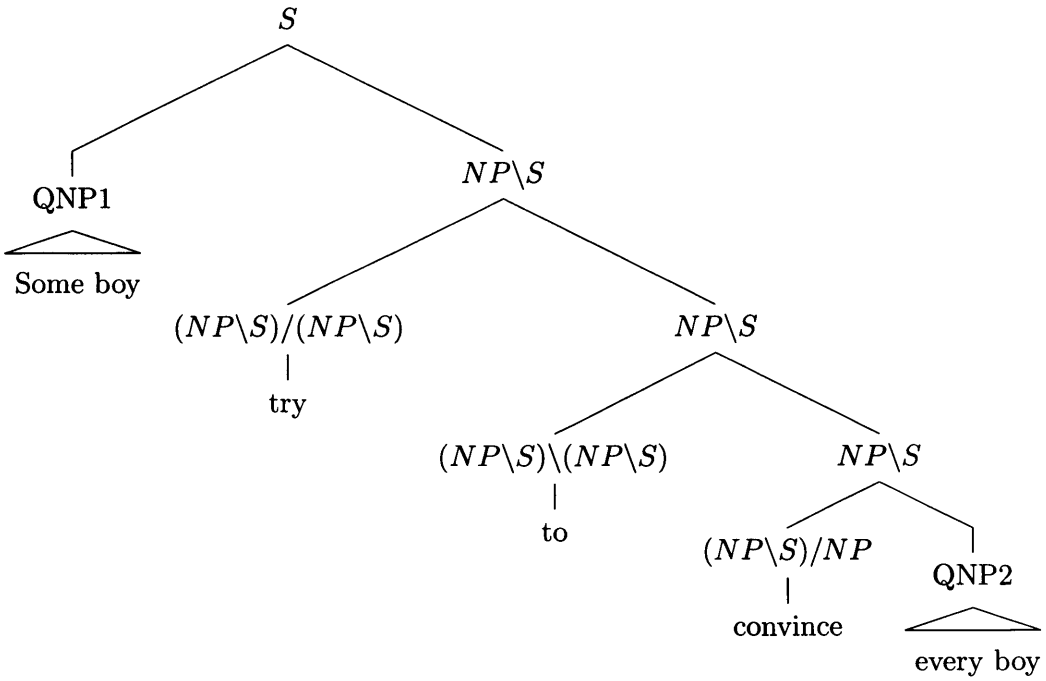
(42) QNPs are interpreted in-situ as arguments of the local verbal functor and thus, their scope stays inside the final output category of the functor, which, in categorial grammar, is the local S. In the semantics, the local S corresponds to the local type t expression. Thus, it follows that semantically, the QNP scope stays inside the minimal propositional element that contains it.

If we apply the informal idea in (42) by pretending as if QNPs could saturate the NP argument slots of the verbs, as in the following three trees, then, the proposal would naturally lead to the tensed-clause locality constraints on QNP scope. That is, it would exclude the scope switch only for (45) in the following three cases:

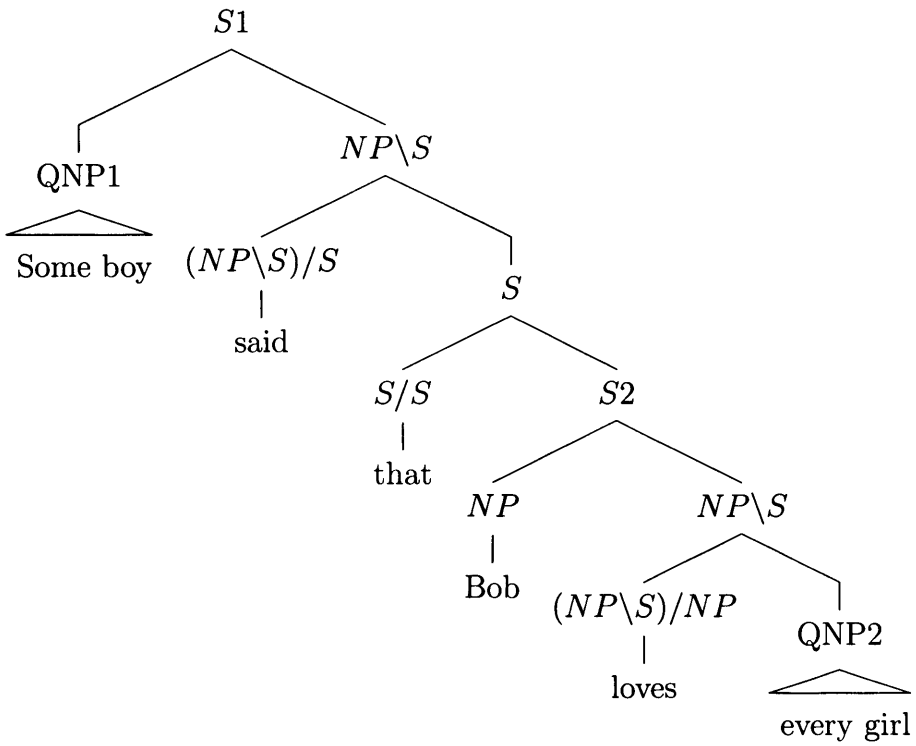
(43) Tree A. Single clausal structure: *Some boy loves every girl.*



(44) Tree B. Infinitival structure: *Some boy tried to convince every girl.*



(45) Tree C. Two tensed clauses: *(Some boy said) that Jack loves every girl.*



In Tree A in (43), the final output category of the local functor *love* is S, and thus, the proposal implies that the scope of a QNP argument of this functor stays within this S. In Tree B in (44), the local functor for the object QNP *every girl* is *convince*. However, the saturation of the external NP argument slot of this

functor is 'postponed' by way of the categories of the higher functors, *try* and *to*, which both select $NP\backslash S$ as their first arguments. This means that the minimal S containing the object QNP would become the root node S, which contains the subject QNP *Some boy.* Thus, the proposal in (42) would intuitively allow us to switch scope between the two QNPs for the structure in (44). In contrast, in Tree C in (45), the minimal S that contains QNP2 *every girl* is $S2$ and the scope of this QNP2 would stay inside $S2$. Thus, this QNP would not take scope over the matrix subject position, $NP1$.

The informal proposal is simple, but instantiating it in Type Logical Grammar is not as straightforward as stated in (42). First, we must find some means to treat $QNPs$ as 'arguments' of the verb in the syntax. In order to do this, I adopt Hendriks' idea of argument raising with some modification. As we have seen in chapter 1, I assume that QNPs are interpreted as Generalized Quantifiers as in Barwise and Cooper (1981). That is, all the QNPs are interpreted as type (et)t operators in the semantics. Together with my interpretation of Hendriks' argument raising as a special lexical/phrasal rule, this creates a certain degree of mismatch between the syntax and the semantics. I argue that this is a merit of my analysis. That is, expecting a perfect match between the syntax and the semantics incorporates too many non-structural factors into the syntax, and thus makes the syntactic system more expressive than is minimally required for dealing with less controversially syntactic phenomena, such as overt extraction phenomena.

2.2.2 Hendriks' argument raising

Basics

In order to treat QNPs as syntactic arguments of a (verbal) functor, we need to modify the type of the functor. To do so, I adopt Hendriks' proposal in Hendriks (1987). Hendriks' argument raising is defined over semantic types, as in (46).

(46) a. Type Shift: $(\vec{a},(b,(\vec{c},t))) \Rightarrow (\vec{a},(((b,t),t),(\vec{c},t)))$

b. Semantics: $P \Rightarrow \lambda\vec{x}.\lambda Q.\lambda\vec{y}.Q(\lambda z.P(\vec{x})(z)(\vec{y}))$

Types of variables. P: $(\vec{a},(b,(\vec{c},t))), Q:((b,t),t), \vec{x}:\vec{a}, \vec{y}:\vec{c}, z:b$

N.B. $\vec{a}, b, \vec{c}$ are meta-variables for some types, where $\vec{a}$ and $\vec{c}$ are such that

$(\vec{a},(b..)) = (b..),(e,(b..)),(e,(e,(b..))),(e,(e,(e,(b..)))))$, etc.

and $(\vec{c},t) = t,(e,t),(e,(e,t)),(e,(e,(e,t)))$, etc.

(cf. Hendriks 1987: 109)

If a functor has an argument slot of type b, then we can raise it to type $((b,t),t)$, so that the functor now can be applied to type $((b,t),t)$ argument, rather than to type b argument. The meta-variables $\vec{x}$ of type $\vec{a}$ and $\vec{y}$ of type $\vec{b}$ mean that the functor P can have any number of arguments (including zero arguments) before and after the argument z of type b, which is type-raised by the rule.

Problems of Hendriks' argument raising

Hendriks' system allows scope alternation between any co-arguments of a functor. However, some data suggest that scope alternation is not always free between co-arguments of a verbal functor. The so-called frozen scope in the double object construction provides one such case, as Bruening (2001) reports.

(47) a. The teacher showed a (*different) student every book.

$*every > a$, $a > every$

b. The teacher showed a (different) book to every student.

$a > every$, $every > a$

Bruening (2001: 235)

Between the two kinds of ditransitive constructions, Bruening reports that the scope relation between the two object QNPs is fixed for the double object (DO) construction, as is shown in (47a), whereas the prepositional ditransitive construction (PP) allows the two object QNPs to switch scope, as (47b) shows. If we apply the rule in (46) to the functor *show'* of type (e(e(et))) (the standard semantic type for ditransitive verbs), then it cannot block the unattested inverse scope reading with the DO construction in (47a). Given the functor *show'* of type (e(e(et))), we can apply argument raising to it with regard to its two object argument slots in two orders, deriving the two scope readings.

(48) a. Type Shift:

$(e,(e,(e,t))) \Rightarrow (((e,t),t),(e,(e,t))) \Rightarrow (((e,t),t),(((e,t),t),(e,t)))$

b. Semantics:

$show' \Rightarrow \lambda Q1.\lambda y.\lambda z.Q1(\lambda x.show'(x)(y)(z))$

$\Rightarrow \lambda Q1.\lambda Q2.\lambda z.Q2(\lambda y.Q1(\lambda x.show'(x)(y)(z)))$

(49) a. Type Shift:

$(e,(e,(e,t))) \Rightarrow (e,(((e,t),t),(e,t))) \Rightarrow (((e,t),t),(((e,t),t),(e,t)))$

b. Semantics:

$show' \Rightarrow \lambda Q2.\lambda y.\lambda z.Q2(\lambda y.show'(x)(y)(z))$

$\Rightarrow \lambda Q1.\lambda Q2.\lambda z.Q1(\lambda x.Q2(\lambda y.show'(x)(y)(z)))$,

Types, $Q1, Q2 : (et)t$, and $x, y, z : e$

Applying the outputs of (48) and (49) to two QNP objects of type ((et)t) and to a normal type e subject such as *Tom* in (50a) derives the two scope readings shown in (50b) and (50c).

(50) a. Tom showed some student every book.

b. $Some'(student')_{(et)t}(\lambda x.Every'(book')_{(et)t}(\lambda y.give'(x)(y)(tom')))$

Some > *Every*

c. $Every'(book')(\lambda y.Some'(student')(\lambda x.give'(x)(y)(tom')))$

Every > *Some*

To prevent this over-generation of scope readings, I reformulate Hendriks' argument raising in categorial calculus. I assign uncurried category $(NP\backslash S)/(NP \bullet NP)$ to ditransitive verbs.[1]

Another problem of Hendriks' analysis is its lack of locality constraints. As I have suggested before, argument raising should naturally lead to certain locality constraints on scope-taking; scope of a QNP should stay within the final output of the 'local functor' which takes the QNP as an argument. However, because Hendriks uses a fully associative grammar system (i.e. L) with an abstraction rule, a QNP can take scope beyond the final output of its local functor, by treating an indefinitely long phonological string as a complex functor.

(51) a. Some student [*said that Tom read*] every book.

b. Undesirable logical form:

$Every'(book')(\lambda y.Some'(student')_{(et)t}(\lambda x.say'_{(t(et))}(read'(e(et))(y)(tom'))(x)))$

Given (51a), association (re-bracketing) and λ abstraction allow us to treat the string *said that Tom read* as a complex functor of type (e(et)). We can then apply argument raising to this complex functor in terms of its two type e argument slots.[2] This derives the unacceptable scope reading '*every* > *some*,' as shown in (51b).

In order to solve these problems, I modify Hendriks' argument raising in a non-associative and non-commutative grammar system NL.

[1] I deal with ditransitive constructions in chapter 4 and chapter 5, rather than in this chapter, because explaining the binding asymmetry in the ditransitive constructions requires some use of modally controlled structural rules.

[2] See Hendriks (1987:112-115) for details.

2.2.3 Reformulating argument raising in TLG

I make the following two assumptions in deriving the scope of QNPs in categorial calculus.

(52) a. In the lexicon, the number of **syntactic arguments** of a functor is maximally one on both sides, i.e.,
OK: $NP\backslash S, (NP\backslash S)/NP, (NP\backslash S)/(NP \bullet NP)$
Not OK: $((NP\backslash S)/NP)/NP$ Cf. Morrill (1994:128)

b. Argument raising applies to any of the functors satisfying (52a) with regard to its maximally two NP argument slots.

The assumption in (52a) implies that scope switch between two QNPs 'normally' occurs in the following configuration.[3]

(53) $QNP1 + (Functor + QNP2)$,
where $QNP1$ and $QNP2$ are co-arguments of $Functor$.

(52b) is the same in algorithm as in Hendrik's argument raising, but it is reformulated in categorial calculus so that directionality of categorial selection is incorporated. I call this special rule **argument slot raising**, or ASR. Given the requirements as in (52), ASR in syntactic categories is as in (54).

(54) Argument slot raising (ASR), syntax:

a. $(NP2\backslash T)/NP1$, where T is normally S.

b. $(NP2\backslash T)/NP1 \Rightarrow (NP1\backslash T)/((S/NP2)\backslash S)$
$\Rightarrow ((S/(NP1\backslash S))\backslash T)/((S/NP2)\backslash S)$

c. $(NP2\backslash T)/NP1 \Rightarrow ((S/(NP2\backslash S))\backslash T)/NP1$
$\Rightarrow ((S/(NP2\backslash S))\backslash T)/((S/NP1)\backslash S)$

This special rule applies to all the items that have the categorial form of (54a).[4] Note that in (54), the final output categories are the same, in whichever order we apply ASR.

As we see later, the lifting of the object argument slot is not provable in the non-associative Lambek calculus NL which I adopt as the basic syntactic system.

[3] I have added the word 'normally' here, because there are exceptional cases that do not fit into this schema, as we see in chapter 5 and chapter 6. But these cases are limited in number and the requirement of an intervening functor between the two QNPs as its arguments provides significant constraints to QNP scope switch.

[4] I discuss how argument slot raising is applied to a functor that has a complex NP argument slot, for example, a ditransitive verb of category $(NP\backslash S)/(NP \bullet NP)$, in chapter 5.

However, rather than increasing the expressive power of the base grammar system to make ASR provable using its basic axioms, I chose to define the argument slot raising as a special lexical/phrasal operation that applies only to certain items.

Though the two orders of applications in (54) leads to the same output category, the different orders lead to different scope readings in the semantics. If we apply ASR to the functor with regard to its first argument slot (i.e. the argument slot for the NP to the right of the functor) first, and then to the second argument slot (i.e. the slot for the NP to the left of the functor), then we derive a functor which generates the surface scope reading between the two QNPs, as in (55).

(55) ASR, semantics, Surface Scope:

a. Type Shift: (e(et)) $\Rightarrow$ (((et)t), (e, t)) $\Rightarrow$ (((et)t), (((et)t), t))

b. Semantics: $P \Rightarrow \lambda Q1.\lambda y.Q1(\lambda x.P(x)(y)) \Rightarrow$
$\lambda Q1.\lambda Q2.Q2(\lambda y.Q1(\lambda x.(P(x)(y))))$
Variable types, $Q1, Q2$: (et)t; x, y: e

If we apply ASR to the functor with regard to its two argument slots in the opposite order, then we derive a functor that generates the inverse scope reading, as shown in (56).

(56) ASR, semantics, Inverse Scope:

a. Type Shift: (e(et)) $\Rightarrow$ (e, ((et)t), t) $\Rightarrow$ (((et)t), (((et)t), t))

b. Semantics: $P \Rightarrow \lambda x.\lambda Q2.Q2(\lambda y.P(x)(y)) \Rightarrow$
$\lambda Q1.\lambda Q2.Q1(\lambda x.Q2(\lambda y.(P(x)(y))))$

For example, application of ASR to a transitive verb *like* derives (57).

(57) Transitive verb with two QNPs.

a. A boy **likes** every girl. ($a > every$; $every > a$)

b. like: $(NP\backslash S)/NP; \lambda x.\lambda y.like'(x)(y)$

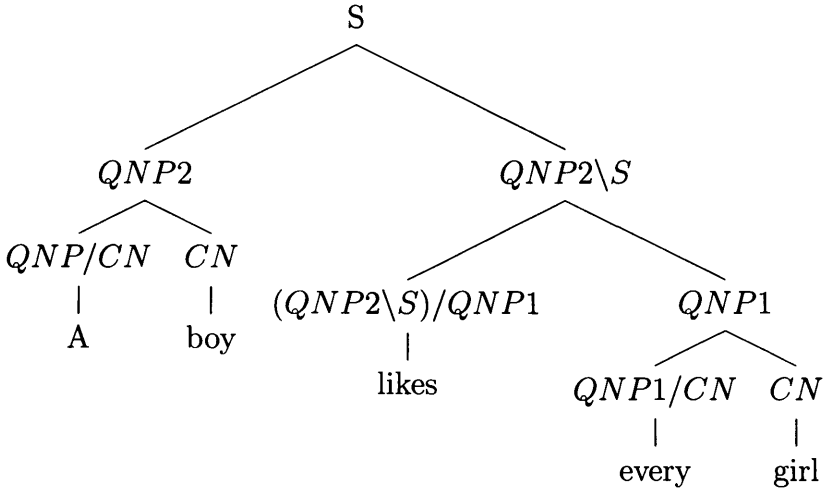
c. Surface scope:
$((S/(NP2\backslash S))\backslash T)/((S/NP1)\backslash S); \lambda Q1.\lambda Q2.Q2(\lambda y.Q1(\lambda x.like'(x)(y)))$

d. Inverse scope:
$((S/(NP2\backslash S))\backslash T)/((S/NP1)\backslash S); \lambda Q1.\lambda Q2.Q1(\lambda x.Q2(\lambda y.like'(x)(y)))$

Because (57c) and (57d) lead to the same output syntactic category, the syntactic structure is the same for the two scope readings, that is the structure in (58).

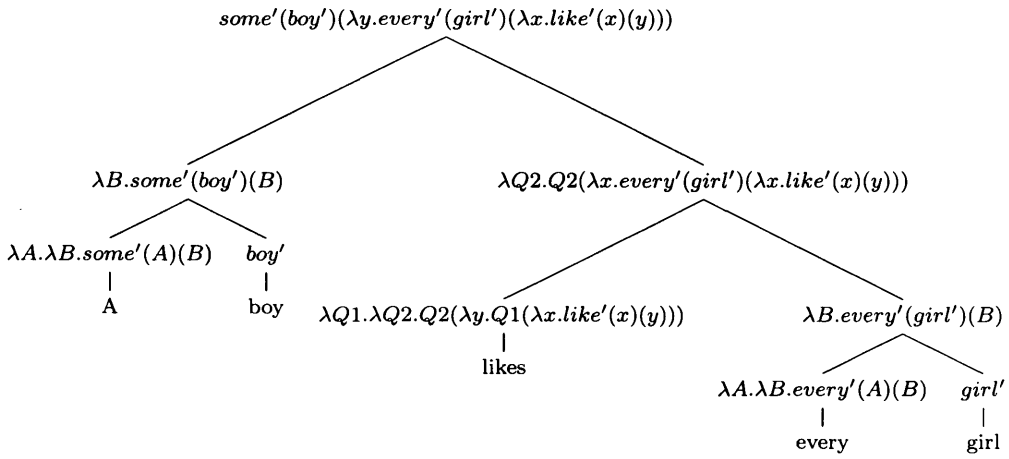
(58) **Tree D**

Abbreviation: $QNP2 = S/(NP2\backslash S)$; $QNP1 = (S/NP1)\backslash S$

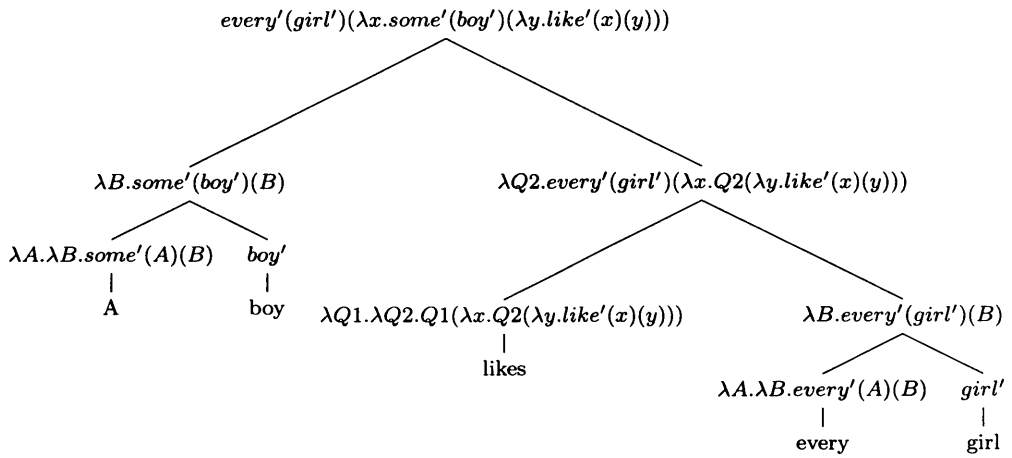


From this uniform syntactic structure, we can derive the two scope readings, as shown in (59)∼(60).

(59) Surface Scope reading, $\exists > \forall$



(60) Inverse scope reading, $\forall > \exists$



Compare this with an alternative analysis that uses function composition to generate the inverse scope reading, $\forall > \exists$. For example, Steedman (2000b).[5] In (61), '+' represents the merge of two items, both in the syntax and in the semantics, whereas '⇒' represents the derivability relation.

(61) a. Directional function composition (Forward composition):
Syntax: $C/B + B/A \Rightarrow C/A$
Semantics: $g_{(b,c)} + f_{(a,b)} \Rightarrow (g \bullet f)$, where $(g \bullet f)_{(a,c)} = \lambda x_a.g(f(x))$

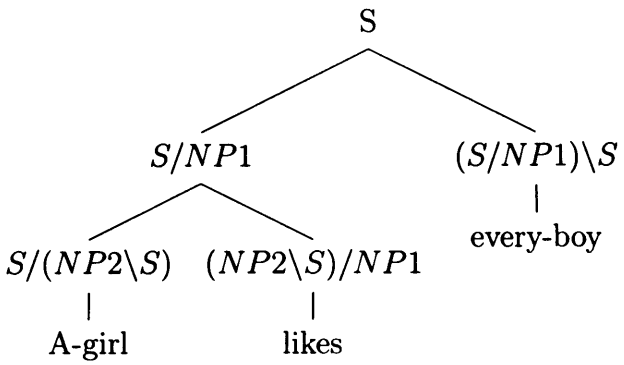
b. (61a) applied to *A girl likes (every boy).*
Syntax: $S/(NP\backslash S) + (NP\backslash S)/NP \Rightarrow S/NP$
Semantics:
$\lambda B.some'(girl')(B) + \lambda x.\lambda y.like'(x)(y)$
$\rightarrow \lambda x.some'(girl')(\lambda y.like(x)(y))$

Cf. Steedman (2000: 40)

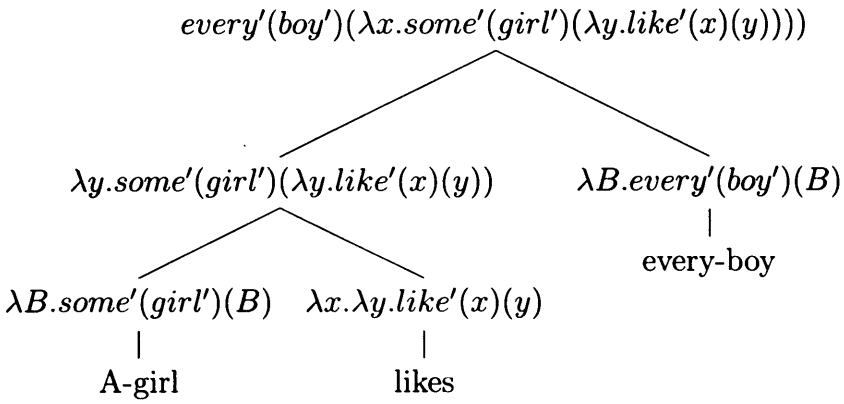
This function composition with directional GQ categories for the subject and the object QNPs leads to the derivation in (62)~(63) for the inverse scope reading, '$\forall > \exists$.'

[5]Steedman uses Combinatory Categorial Grammar (CCG), and the composition is achieved in terms of B combinator, such as '$(B(g(f))) = \lambda x_a.g(f(x))$' for (61a). In the main text, however, I do not represent the combinator explicitly for presentation reasons.

(62) Tree E for the alternative analysis, Syntax:



(63) Tree E, Semantics:



I argue that the following binding asymmetry between the subject and the object positions suggests that the structure in Tree E in (62)~(63) is ill motivated.[6]

(64) a. *Her$_1$ mother met every girl$_1$.

b. *A boy who she$_1$ liked met every girl$_1$.

If we used an analysis that generates the alternative tree structure Tree E, we should have an option of merging the object QNP later than the subject QNP. Thus, whatever binding mechanism that licenses the binding of the pronoun in the surface scope structure, as in *[[Every girl]$_1$ [met [her$_1$ mother]]]*, should license the binding for the sentences in (64) as well, contrary to our data judgments.[7]

[6]As for Steedman's analysis, though Steedman (2000b)'s scope switch system leads to the alternative syntactic structure as in Tree E above for the inverse scope, he does not define his binding conditions on the syntactic tree. Instead, he defines his binding conditions in terms of the logical expressions which are independent of the syntactic structures to a certain degree. See Steedman 2000: 66-69. However, this analysis introduces extra complexity to the level of logical forms, as we briefly see in chapter 6.

[7]Though it is not directly related to the point I am making here, Minimalists might argue that the sentences in (64) can be excluded in terms of Weak Cross Over violation with the assumption that covertly moving the object QNP across the co-indexed pronoun leads to WCO violation. However, this counts as an explanation only if *QR* is postulated. Note that the object QNP

Some might argue that the existence of a bound pronoun inside the subject (Q)NP in (64) somehow precludes the availability of the structure Tree E in (62)~(63). However, such an explanation should be accompanied by some well-stated binding mechanism that stops the function composition from applying between the subject QNP and the transitive verb functor only when the subject QNP contains a pronoun which is bound by the object QNP. Stating such mechanism would complicate the categorial calculus beyond necessity. Note that the inverse scope reading is available when the pronoun is interpreted referentially, as in *A boy who she (e.g. Meg) liked met every girl* ($\exists > \forall$ or $\forall > \exists$). Note also that the inverse scope reading is also available if the pronoun is bound by a higher operator, as in *An editor$_i$ said that a linguist who she$_{i/*j}$ knew interviewed every candidate$_j$* ($\exists > \forall$; $\forall > \exists$). In the latter sentence, the pronoun inside the subject would be in a right structural position in Tree E so that it can be bound by the main clause subject *an editor*, but we would have to prohibit the potential binding by the object which would generate the unattested binding relation marked by the index j.

Finally, after arguing that the structure as in Tree E is ill-motivated for deriving the inverse scope, I do not have a particular view of the usefulness of such a structure to explain coordination data, to explain a sentence such as *Every boy loved, but every girl hated, a philosophy professor.* I am not arguing that the merge of the subject and the transitive verb before we merge the result with the object never occurs. In this case, we might use the lexical entry of *and* as a trigger of some structural association rule. But the interaction of such association and scope data should be carefully investigated and I leave such investigation for future research.[8]

2.3 Argument-slot raising as a 'lexical/phrasal' rule

As I indicated above, argument slot raising of a functor category with regard to its object NP argument slot is not provable/derivable in NL. Without going into formal details (see Chapter 3 for such details), this is because NL does not accept structural association (i.e. 're-bracketing' of the structures).

---

does not c-command the pronoun for eitehr sentence in (64) and thus, without QR, the standard c-command binding condition is not satisfied anyway.

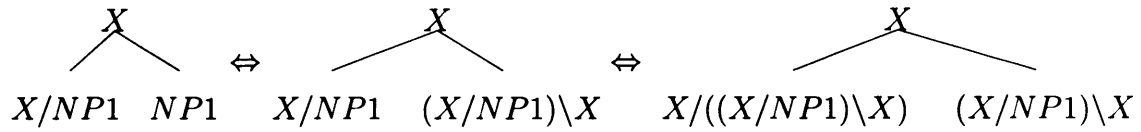
[8]However, I come back to this issue very briefly in chapter 10. Thanks to Alex Lascarides for reminding me of the relevance of the coordination structures to the issue of QNP scope.

(65) Argument-slot raising wrt the object NP slot is **not provable** in NL.

a. $(NP2\backslash S)/NP1 \vdash_{NL} ((S/(NP_2\backslash S))\backslash S)/NP_1$

$\not\vdash_{NL} ((S/(NP_2\backslash S))\backslash S)/((S/NP_1)\backslash S)$

b. Functor-argument status alternation between the sisters (derivable in NL):



NL does not structurally distinguish the functor and the argument that are merged. Thus, changing the functor-argument relation between the sisters is free. This licenses argument slot raising with regard to the subject NP. On the other hand, raising the internal NP argument to $(NP/S)\backslash S$ would require re-bracketing. Without association, the object NP argument slot of $(NP\backslash S)/NP$ can only be raised to $((NP\backslash S)/NP)\backslash(NP\backslash S)$, rather than the desired $(S/NP)\backslash S$.

I could introduce structural association in a modally controlled way, but then I would have problem explaining binding data as we have seen with (64).[9]

Although using a special lexical/phrasal rule that is not fully supported by the basic algorithms of the grammar system poses a question about what we can do by postulating such special rules, at the moment, I take the current proposal as a provisional way of differentiating QNP scope switch from the overt extraction phenomena in the syntax, such as A-bar movement, or from structural modification involved in A-movement phenomena, for which I use modally controlled structural rules in different ways, as see later. Again, as a benefit of postulating such a special rule, the analysis only uses one syntactic structure such as Tree D in (58). In other words, the special rule helps support my linguistic claim that scope ambiguity is basically a semantic phenomenon and we do not have enough syntactic motivation to generate two syntactic structures for generating two scope readings.

[9]I analyse the binding data in terms of Jacobsonian argument identification in the syntactic derivation, as in Jacobson (1999), and thus cannot resort to the LF representations to explain the binding data, as in Steedman (2000b). See chapter 4.

2.4 QNP scope in Control/raising/auxiliary constructions

As we have seen already, the subject control construction as in (66a) exhibits scope ambiguity.[10] The ambiguity is also observed in the raising and the modal auxiliary constructions, as is shown in (66b) and (66c).

(66) a. A student *tried to review* every paper. $a > every$; $every > a$

b. A student *seems to review* every paper. $a > every$; $every > a$

c. A student *must review* every paper. $a > every$; $every > a$

On the other hand, scope switch across a tensed clause (or across a more complex island which contains a tensed clause) is impossible, or at least significantly more difficult to get.

(67) a. Some/a boy said that Bob likes every girl.

$some > every; *?every > some$

b. Some/a boy heard the rumor that Bob likes every girl.

$some > every; *every > some$

c. Some/a will be happy if every girl comes to the party.

$some > every; *every > some$

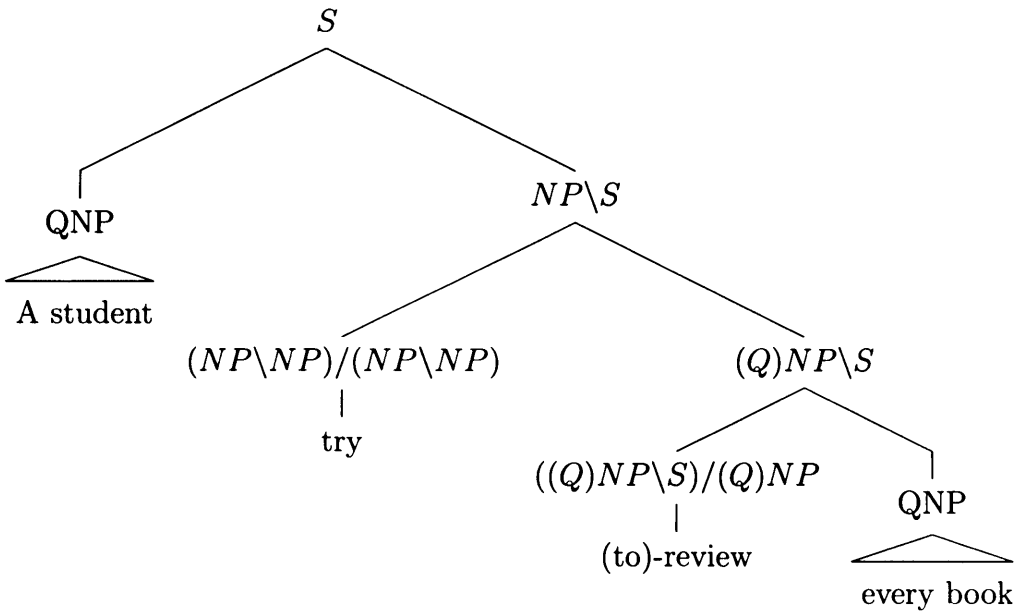
I have attributed this contrast to the fact that for (66a), both the QNPs are contained in the same minimal S category in the derived syntactic tree, whereas in the tree for (67a), the minimal S expression containing the embedded object QNP *every boy* does not contain the matrix subject QNP *some/a boy*. Given the standard categorial grammar category assignments, the raising construction in (66b) and the auxiliary construction as in (66c) are assigned structures analogous to (66a), whereas (67a)~(67c) lead to the structures in which the root-node S categories contain an embedded S category that dominates only the universal QNP. Thus, my informal analysis predicts that the scope ambiguity is available only with (66) and not with (67).

On the other hand, in order to formulate this informal analysis to generate scope ambiguity for the cases in (66) by way of the proposed argument slot raising, we should be able to treat the expressions in italics in (66) as complex predicates with category $(NP\backslash S)/NP$, so that we can apply argument slot raising to the

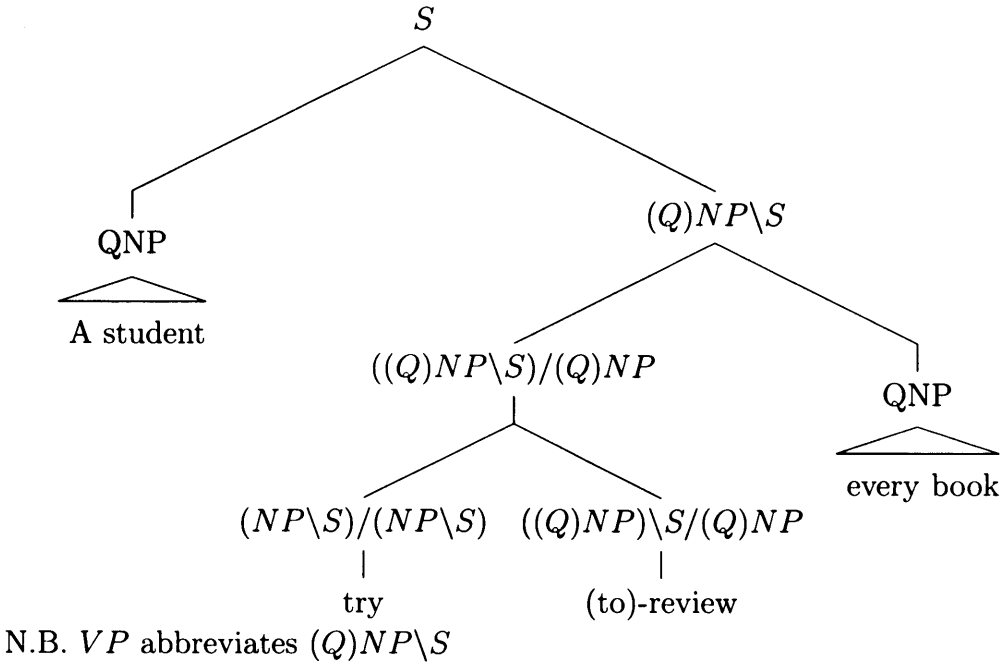
[10]Though object control constructions seem to block the scope switch between the main clause subject QNP and the embedded object QNP, as we see in chapter 7

derived predicates in two different orders, deriving the scope ambiguity. As we can see informally by comparing the two structures in (68) and (69), this would require use of structural association.

(68) **Tree F** (derivable with the standard lexical entries and the basic rules of NL).[11]



(69) **Tree G** (which we want for switching scope in (66a)).



N.B. VP abbreviates $(Q)NP\backslash S$

[11] I ignore the contribution of the infinitival *to* here, for convenience.

The tree structure in (69) requires function composition between the higher functor *try* and the lower functor *(to) review.* In Type Logical Grammar, this means we need to introduce structural association rule to NL. However, in order to do so without collapsing the whole system NL to an associative system such as L, we must discard a unimodal system and adopt a multi-modal deductive system, such as Multi-Modal Type Logical Grammar in Moortgat (1997).

Also, we want to introduce structural rules in such a way that it does not license scope switch across a tensed clause, given the data in (67). This means that we want to introduce structural association in a different way from how we use such a rule to explain Wh-extraction phenomena, where Wh-movement is known to cross a tensed clause, as we have seen in chapter 1. Without going into formal details, I provide the rough picture of the proposed grammar system as in (70).

(70) a. The grammar system: $NL_{\Diamond+ASR}$, Non-associative Lambek Calculus enriched by a family of connectives that include residuated unary operators, $\Diamond, \Box^{\downarrow}$, and with argument slot raising (ASR) as a special rule. Cf. Moortgat (1997), Bernardi (2002) and Vermaat (2006).

b. QNP scope is explained in terms of argument slot raising (ASR) that is applied specifically to the local functor that takes the QNP in question as an argument. Thus, the scope of a QNP cannot exceed the final output of this local functor. Cf. Hendriks (1987).

c. Control/raising/modal construction may produce a complex predicate via structural rules constrained by a specific pair of modes of combination (e.g., the pair of $\circ_i, \circ_j$, as in, $will \circ_i (play \circ_j tennis) \Rightarrow (will \circ_i play) \circ_j tennis$). This structural rule can go only as far as a (verbal) projection line, such as,

$$T \circ_i (v \circ_i (V \circ_j NP)) \Rightarrow (T \circ_i (v \circ_i V)) \circ i \ \mathbf{NP}.$$

Such complex predicates can be local functors of QNPs.

d. A-bar movement is explained in terms of structural rules constrained by unary operators, $\Diamond, \Box^{\downarrow}$, whose use is required by the category encoded with the Wh-expression. The association is not constrained by the modes of merge between the 'Wh operator' position and the 'trace' position. That is,

$$What \circ_x (did \circ_y (\cdots \circ_z (Meg \circ_j (like \circ_j \Diamond\Box^{\downarrow} t))))$$
$$\Rightarrow What \circ_x ((did \circ_y (\cdots \circ_z (Meg \circ_j like))) \circ_j \Diamond\Box^{\downarrow}\mathbf{t})$$

The base grammar system is NL, which is non-associative and non commutative, but to explain certain natural language syntax phenomena which we cannot ignore by any means, which correspond roughly to A-movement and A-bar movement phenomena, I adopt the enriched system $NL_{\diamond}$, which introduces certain degree of structural association/permutation under modal control. Together with the special semantically motivated rule argument slot raising for QNP scope, which does not influence the syntactic structure but still remains to be a rule of the proposed system, I call the total system $NL_{\diamond+ASR}$.

For A-movement, I introduce structural rules under the control of merge mode specification, as is indicated in (70c). The indices on the structural connective $\circ$ indicate the modes of merge, and structural rules apply only if we have certain combinations of such merge modes. In chapter 7, I show how we can use this system to introduce association/permutation within each verbal projection line, which roughly corresponds to the projection line indicated by each sequence of T-v-V in Minimalism. The mechanism allows us to create a complex functor within each projection line. Because argument slot raising can apply to such a complex functor, it allows a QNP to take scope within the final output of each verbal projection line, as I show in chapter 7. For Wh-extraction, which can cross a tensed boundary, I introduce structural rules under the control of unary operators $\diamond$ and $\Box^{\downarrow}$, as is indicated in (70d). These structural rules do not expand the scope of QNPs and thus the tensed clause boundary remains valid after the addition of these structural rules, as we see in chapter 8. Both the ways of introducing structural rules have already been used by Moortgat and his followers, as in Versmissen (1996), Moortgat (1997), Bernardi (2002) and Vermaat (2006).

2.5 Conclusion

In this section, I have provided the basic algorithm to compute QNP scope together with the general picture of the grammar system that I use. In the next chapter, I review the proposal in this chapter from a more deductive viewpoint. I then introduce a Multi-modal Type Logical Grammar (MMTLG) as in Moortgat (1997). The particular system that I adopt is $NL_{\diamond}$, the system that is non-associative at the base level, but can introduce structural rules with modal control. The added expressive power of the grammar is used in two different ways to capture A-movement and A-bar movement phenomena in linguistics in later chapters. Together with argument slot raising as a special rule, I call the total proposed system $NL_{\diamond+ASR}$ for

convenience.

Chapter 3

Type Logical Grammar

3.1 Deductive views of Categorial Grammar

This chapter re-introduces Type Logical Grammar in Gentzen Sequent (GS) presentation, which represents the symmetry of the categorial calculus in a better way than Natural Deduction proofs. GS presentation is also more convenient for showing structural rules, such as an association rule. In section 3.2, I re-introduce the Lambek Calculus NL in Gentzen Sequent presentation. I also show how we can derive the interface representations at LF and PF by using Gentzen Sequent proofs. In section 3.2.5, I show in GS proofs that argument slot raising of a transitive verb functor with regard to its subject position is provable whereas argument slot raising of such a functor with regard to its object argument slot is not provable in NL. I then show that value raising of such a functor is generally provable in NL, where value raising is used in some of the following chapters to switch scope in an exceptional manner. Section 3.3 provides the basics of Multi-Modal Type Logical Grammar (MMTLG), as in Moortgat (1997), in which we can introduce various structural rules in a modally controlled manner. I then informally show that the enriched formalism allows us to introduce structural rules in two different ways, as are appropriate for explaining two linguistically different phenomena, that is, A-movement and A-bar movement phenomena. Lastly, in section 3.4, I show how structural rules can be introduced in MMTLG to deal with pronominal and reflexive binding. The rules introduced in this section are used in chapter 4 when I explain the binding asymmetry in ditransitive constructions, and in chapter 9 when I explain the dependency of the indefinites to some other elements. Section 3.5 provides concluding remarks.

3.2 TLG in Gentzen Sequent Presentation

3.2.1 Symmetry of Inference

Type Logical Grammar (TLG) has been investigated by Morrill (1994), Moortgat (1997) and others. TLG has developed from Lambek Calculus in Lambek (1958), which reformulated Classical Categorial Grammar as in Ajdukiewicz (1935) in a way that is more suitable for the analysis of natural language, while interpreting Categorial Grammar as a deductive proof system that explains derivability of (categorial) formulas. It sets up the connectives /, \, and • as a residuated triple, as shown in (71). As we can see, we can move categorial formula components between the antecedent and the succedent of sequents while preserving the provability of the sequents, where each sequent has the form: **Antecedent** $\vdash$ **Succedent**.[1]

(71) $B \vdash A\backslash C \Leftrightarrow A \bullet B \vdash C \Leftrightarrow A \vdash C/B$

Intuitively, • on the one hand, and \, / on the other, represent operations in opposite directions. To see the point, compare • to '+' as in numerical addition on the one hand, and compare \, / to '−' (subtraction) on the other. If we see the addition/subtraction operation in terms of accessibility relations between numbers, then '+' and '−' express the opposite accessibility relations. Thus, by adding 2 to 1, we can reach the number 3. In the opposite direction, by subtracting 2 from 3, we can reach the number 1. We can see the same point by comparing • to '×' as in multiplication of numbers and \, / to '÷' as in division of numbers. Having operators that represent the opposite accessibility relations such as this enhances the deductive nature of the inference system. Compare (71) with (72) in that regard.

(72) $\forall a, b, c \in Q$ (where Q is the set of positive rational numbers).

a. $a \leq c - b \Leftrightarrow a + b \leq c \Leftrightarrow b \leq c - a$

b. $a \leq c \div b \Leftrightarrow a \times b \leq c \Leftrightarrow b \leq c \div a$

Having both '+' and '−' as operators allows us to modify the forms of inequalities while preserving their truth, as shown in (72a). The same applies for the pair '×' and '÷' in (72b). In a similar way, having a residuated triple of binary connectives in Type Logical Grammar allows us to express various derivability relations between

[1] Areces and Bernardi (2004) (p.130) suggests that the pair of / and \ alone (i.e. without •) already exhibits a certain kind of residuation, supported by the provability of the sequent, $A \vdash (B/A)\backslash B$.

formulas by way of inferences which preserve the truth of sequents.[2] In fact, the structural connective '(,)' as we have seen in chapter 1 (cf. (75) below) has the same interpretation as • (informally, both are interpreted as 'AND') in the antecedent of a sequent. Thus, we could express the derivability relations as in (71) without this formula connective •.

(73) $B \vdash A\backslash C \Leftrightarrow (A, B) \vdash C \Leftrightarrow A \vdash C/B$

However, as an intuitionistic logic, NL cannot use the structural connective '(,)' in the succedent (even if it did, it would mean a different thing, that is, a comma between structures means 'OR' in the succedent of a sequent). In contrast, a formula of the form of $(A \bullet B)$ may appear in the succedent in NL. Also, as a formula connective, • can appear in lexical categories, and it increases the expressive power of our grammar in a deductive manner. For example, assigning the category $(NP\backslash S)/(NP \bullet NP)$ to ditransitive verbs allows us to derive a structure different from the structure which we get with the standard ditransitive verb category $((NP\backslash S)/NP)/NP$, as we see in detail in chapter 4 and chapter 5.

To show categorial derivation, I have used natural deduction proof presentations so far, which I refer to as ND. ND is useful for showing step-wise composition of interface representations (such as typed lambda expressions at LF) from lexical levels. On the other hand, ND is less convenient for showing provability/derivability preserving relations between sequents as in (71). For that purpose, using sequents, rather than formulas, as premises and conclusions is more convenient, as in Gentzen Sequent (GS) presentation which I show shortly. For a similar reason, Gentzen Sequent presentation is useful for showing various structural rules which we introduce later.

Before I show the axioms in GS presentation, I briefly repeat the definitions of the set of (categorial) formulas and the set of structures from chapter 1 in different formats.

The set F of (categorial) formulas is defined as in (74).

(74) $F ::= At \mid (F/F) \mid (F\backslash F) \mid (F \bullet F)$

At is the set of atomic formulas, such as NP, N and S. Given At, we close the set F by successively applying binary connectives /, \ and • to any two formulas, either atomic or derived, creating complex formulas such as $NP\backslash S$, $S/(NP\backslash S)$ and

[2]In order to be able to add and subtract formulas freely both in the antecedent and in the succedent while preserving the validity of the sequent, we need binary operators that correspond to •, \, / in the succedent. See Wansing (1998) and Moortgat (1997).

$(NP\backslash S)/(NP \bullet NP)$. I omit the outer-most parentheses around complex formulas unless it is misleading somehow. Not all the members of F have to be included in the lexicon of natural languages. At might be presumed to be universal across languages, but each language typically has a different set of complex categories in its lexicon. For the presentation of the axioms and proofs, I use formula meta-variables $A, B, C \in F$.

Given F, S represents the set of structures, which configure formulas into certain structures.

(75) $S ::= F \mid (S, S)$

Example structures are $(NP, NP\backslash S)$ and $(NP, ((NP\backslash S)/NP, NP))$. In NL, which is non-associative and non-commutative, the bracketing and the left-to-right order are taken seriously. That is, $(A, (B, C)) \not\Leftrightarrow ((A, B), C)$, and $(A, B) \not\Leftrightarrow (B, A)$. I use meta variables $\Gamma, \Gamma', \Delta, \Delta' \ldots \in S$ for structures when I present the syntactic rules in GS presentation.

3.2.2 Gentzen Sequent Presentation

The rules are made out of the identity axiom and the so-called logical rules, which are pairs of Left and Right rules for /, \ and •. First, I show the identity axiom and Cut in (76), neither of which involves a connective.

(76) For all $A, B \in F$ and for all $\Gamma, \Delta \in S$:

a. Identity Axiom: $A \vdash A$

b. Cut Axiom

$$\frac{\Gamma \vdash A \quad \Delta[A] \vdash B}{\Delta[\Gamma] \vdash B}\ Cut$$

In GS presentation, every rule of NL other than the identity axiom in (76a) derives one sequent as a conclusion below the horizontal bar from one or two premise sequents that are placed above the horizontal bar. Each sequent has the form: 'Antecedent ⊢ Succedent,' where Antecedent is a structure as in (75) and the Succedent is a formula as in (74). In Lambek Calculus, the antecedent structure cannot be empty. The succedent is exactly one formula, such as S or S/NP, but not a complex structure, such as $(NP,\ NP\backslash S)$.

The identity Axiom is the only rule that does not require any premise sequent. Thus, for any formula A, we can derive A from A itself (note that any formula is a structure on its own, so we can see A in the antecedent of a sequent as a structure).

The Cut rule in (76b) informally means the following. For all $A, B \in F$ and for all $\Gamma, \Delta \in S$, if we can prove/derive the formula A from the structure Γ, and if we can prove/derive the formula B from the structure Δ in which the formula A occupies a particular structural position, then we can also prove/derive the formula B by placing the structure Γ instead of A in exactly the position which A occupied in Δ before replacement. In (76b), $\Delta[A]$ means that the formula A occupies a particular position in the binary structure Δ, and $\Delta[\Gamma]$ in the conclusion sequent means that Γ must replace A in exactly the same position in the structure Δ. An example may help.

(77) Cut (example)

$$\frac{(B, C) \vdash A \quad (D, (E, \mathbf{A})) \vdash G}{(D, (E, (\mathbf{B}, \mathbf{C}))) \vdash G}\ Cut$$

When we replace the so-called **cut formula** A in (77) with the structure (B, C) as we move from the top line to the bottom line, we have to place (B, C) exactly in the position which A occupied inside $(D, (E, A))$ before the replacement, as the boldface indicates.

Cut is convenient in presentation because we can connect two proofs into one without changing the structure of each proof. However, note that both in (76b) and in (77), the formula A disappears in the consequence sequent below the bar. This could be problematic if some sequent were provable only by using Cut, because then, we could not tell which formulas should appear in the premise sequents just by looking at the conclusion sequent (= the sequent to be proved). This could influence the decidability of the calculus, as we come back to later. However, it has been proved that any sequent that is provable by using Cut in GS presentation is provable without using Cut. Thus, we can use Cut as a convenient tool in presentation. For some cut elimination proofs in GS presentations, see Versmissen (1996: 9-11).

As a word of caution, Cut is admissible only with regard to the pure calculus NL. My analysis of QNP adds a non-logical axiom, ASR, to NL, and thus, Cut elimination is not proved in my system $NL_{\Diamond+ASR}$. Relatedly, the sub-formula property and decidability of Gentzen Sequent presentation only hold for the pure calculus NL, and not for $NL_{\Diamond+ASR}$.

In practice, I use Cut in this thesis only a few times in a non-essential way, but understanding the Cut rule is also important in that it represents the transitivity of the inference system. In other words, the basic idea behind Cut is that if we can prove B from A and if we can prove C from B, then we can prove C from A. That is, the provability relation expressed by $\vdash$ is transitive. The rule in (76b) looks more complex because the antecedent of each sequent is a structured configuration of formulas in NL, but the basic property of the transitivity of the derivability relation stays the same.

Next, I show the rules for the categorial formula connectives.

(78) Logical rules. For all $A, B, C \in F$ and for all $\Gamma, \Delta \in S$:

a.
$$\frac{\Delta[B] \vdash C \quad \Gamma \vdash A}{\Delta[(B/A, \Gamma)] \vdash C}\ /L \qquad \frac{(\Gamma, A) \vdash B}{\Gamma \vdash B/A}\ /R$$

b.
$$\frac{\Gamma \vdash A \quad \Delta[B] \vdash C}{\Delta[(\Gamma, A\backslash B)] \vdash C}\ \backslash L \qquad \frac{(A, \Gamma) \vdash B}{\Gamma \vdash A\backslash B}\ \backslash R$$

c.
$$\frac{\Delta[(A, B)] \vdash C}{\Delta[(A \bullet B)] \vdash C}\ \bullet L \qquad \frac{\Gamma \vdash A \quad \Delta \vdash B}{(\Gamma, \Delta) \vdash (A \bullet B)}\ \bullet R$$

As I explained above, each rule states that if every premise sequent above the horizontal bar holds, then the conclusion sequent below the bar also holds. Thus, when there are two sequents above the bar, the 'space' between the two premise sequents is interpreted as 'AND.' The left-to-right placement of the two premise sequents is not important, because how those premise sequents are consumed when we derive the conclusion sequent below the bar is specified in the antecedent of the conclusion sequent (i.e. we can read it off the structural configuration in the antecedent of the consequence sequent). To make each rule easier to read, however, I put the two premise sequents in the same left-to-right order as they are consumed in the antecedent of the consequence sequent. For each rule, each premise sequent is used exactly once.

For each connective, we have a pair of rules. Each 'Left' rule (marked with L at the right-end of the horizontal bar) introduces the corresponding connective in the antecedent of the sequent below the bar and each 'Right rule' (indicated by R

at the right-end of the horizontal bar) introduces the connective in the succedent of the sequent below the bar.

As I explained above, the notation $\Delta[\Gamma]$ is used to keep track of the structural position that the substructure Γ occupies inside the larger structure Δ in a rule application. Thus, in the consequence sequent for the rule $/L$ in (78a), the binary structure $(B/A, \Gamma)$ must replace the formula B exactly in the position that B occupied in the antecedent structure Δ of the left premise sequent. I provide an example use of $/L$ with formula variables.

(79) $/L$ (Example): $A, B, C, D, E, G, H \in F$

$$\frac{D, (\mathbf{B}, E) \vdash C \quad (G, H) \vdash A}{D, ((\mathbf{B/A}, \mathbf{(G, H)}), E) \vdash C}\ /L$$

As the bold-face letters indicate, when we replace B by the structure $(B/A, (G, H))$ in the conclusion sequent below the bar, we have to place the latter exactly in the place which B occupied in the antecedent of the left premise sequent. This requirement is necessary because NL is non-associative and non-commutative. Successive application of the rules in (78) generates a binary configuration of categorial formulas in the antecedent of the final sequent to be proved, as we see in the example proof in (81b) below. Unlike in Cut, the formula A in the right premise sequent is preserved as a sub-formula in B/A in the antecedent of the conclusion sequent. Every alphabet letter (representing a formula) which appears in the premise sequents also appears somewhere in the conclusion sequent below the bar. As we discuss after the example (81b) below, this **sub-formula property** of GS presentation confirms the decidability of the proof system.

Given the axioms in (76) and (78), each GS proof proves a goal sequent at the bottom line. (80) is an example of a goal sequent that is provable in NL.

(80) $(NP, ((NP\backslash S)/NP, NP)) \vdash S$

In (81b), I provide the GS proof of the sequent (80) for the sentence *Tom loves Meg*, with the lexical entries as in (81a).

(81) a. Lexical assignments:

Tom: $< Tom;\ NP;\ tom' >$

Meg: $< Meg;\ NP;\ meg' >$

love: $< love;\ (NP\backslash S)/NP;\ love' >$

Types, $tom' : e;\ meg' : e;\ love' : (e(et))$

b. GS proof for *Tom loves Meg.*

$$\dfrac{\dfrac{NP \vdash NP \quad S \vdash S}{NP, NP\backslash S \vdash S}\,\backslash L \qquad NP \vdash NP}{(NP, ((NP\backslash S)/NP, NP)) \vdash S}\,/L$$

We start each proof with atomic identity axioms, as the top line of the proof in (81b) shows.[3] From bottom to top, each step of the proof eliminates one occurrence of a connective. This is because the logical rules in (78) are set up in that way. Thus, any formula appearing in any premise sequent above the horizontal bar above the sequent in the bottom line of the proof is a subformula of a formula appearing in this bottom sequent (which is the goal formula to be proved). Because of this sub-formula property, Gentzen sequent proof of NL is decidable.[4] In other words, for any sequent to be proved, there are only a finite number of formulas and a finite number of connectives appearing in that sequent. Thus, as we have seen in (81b), we can successively reduce the number of the connectives bottom to top by using GS rules, and if the sequent is provable at all, we can reach in a finite number of steps the identity axioms of the atomic formulas (where all the atomic formulas appear as sub-formulas of the sequent to be proved). Note that the Cut rule in (76b) does not reduce the number of logical connectives bottom to top. Also, the subformula property does not apply in Cut, because the Cut formula A does not appear in the consequence sequent below the bar. Again, this could spoil the decidability of GS proofs, but as I have indicated above, Cut is admissible for GS presentation of the pure calculus, and therefore does not influence the decidability.

The next question is how we read off the PF-LF pairing from a Gentzen sequent proof. I only provide some informal rules here. I provide more formal interpretation rules in sections 3.2.3 and 3.2.4.

For each proof, we look at the sequent at the bottom line, that is, the sequent to be proved. In (81b), the goal sequent is $(NP, ((NP\backslash S)/NP, NP)) \vdash S$. Replacing each categorial formula with the corresponding PF and LF expressions (the lexical assignments as in (81a) show which categories correspond to which PF-LF items), we can derive the PF and the LF sequents as in (82).

(82) a. PF: $(Tom, (love, Meg)) \vdash (Tom \cdot (love \cdot Meg))$

[3]For convenience, I start some proofs with non-atomic identity axioms (e.g. $NP\backslash S \vdash NP\backslash S$). But the complex identity axioms that I use can be derived from atomic identity axioms.

[4]As I have mentioned above, the total grammar system $NL_{\diamond+ASR}$ is not decidable because of the non-logical axiom ASR.

b. LF: $(tom', (love', meg')) \vdash love'(meg')(tom')$

(82a) means that out of the three PF items, *Tom*, *love*, *Meg*, we can derive the sentential PF expression $(Tom \cdot (love \cdot Meg))$. As I have informally explained in chapter 1, the intended PF strings reflect the binary structures in the syntax, with the binary connective '$\cdot$' over PF strings being non-associative and non-commutative, though I generally abstract away from the relevance of the binary structure in phonological strings. Thus, ignoring the brackets, we get the string *Tom love(s) Meg*. The LF sequent in (82b) represents the same kind of derivability relation at the semantic level.

Though this informal PF-LF pairing works in linguistic analyses, it is not very satisfactory in several ways. For one thing, seeing the natural language syntax as a deductive proof system, we want to set up the interpretation rules in such a way that we can prove the soundness and completeness of the deductive system with regard to its intended interpretations at the two interfaces. Though the current grammar system that I use in this thesis, that is, $NL_{\Diamond+ASR}$, is not complete, it is still important to set up the interpretation rules as in the semantics of logical languages. Secondly, we want to know how the syntactic proofs based on categorial formulas derive the two interface sequents as in (82a) and (82b) step by step. Because of these, I show the interpretation rules in a slightly more formal way in the next sections.

3.2.3 Model Theoretic Interpretation at PF

In the Model Theoretic interpretation at PF, formulas (i.e. syntactic categories) are interpreted as sets of phonological expressions.[5] For atomic formulas, the valuation function ν_0 maps members of At in (74) to the corresponding sets of phonological expressions, as shown in (83).

(83) a. $\nu_0(NP) = \{Tom, Meg, (the \cdot book) \ldots\}$

b. $\nu_0(S) =$
$\{(Tom \cdot smoke), (Tom \cdot (love \cdot Meg)), (Tom \cdot (give \cdot ((the \cdot book) \cdot (to \cdot Meg)))) \ldots\}$

For readability, I omit the parentheses in PF strings from now on, unless they are important in some ways. Given the interpretations of the atomic categories at PF

[5]We could interpret categorial formulas as sets of ordered pairs of phonological and semantic objects. However, for convenience, I deal with the interpretations at the two interfaces separately.

as in (83), the atomic interpretation function ν_0 is then extended to the general interpretation function ν as in (84).

(84) a. $\nu(A \bullet B) = \{(a \cdot b) \mid a \in \nu(A) \;\&\; b \in \nu(B)\}$

b. $\nu(A\backslash C) = \{b \mid \forall a \in \nu(A) : (a \cdot b) \in \nu(C)\}$

c. $\nu(C/A) = \{b \mid \forall a \in \nu(A) : (b \cdot a) \in \nu(C)\}$

For example, by (84b), $\nu(NP\backslash S) = \{b \mid \forall a \in \nu(NP) : (a \cdot b) \in \nu(S)\}$. If the PF item *smoke* is a member of $\nu(NP\backslash S)$, then for each $a \in \nu(NP)$, the PF expression $(a \cdot smoke)$ must be a member of $\nu(S)$. As I have explained above, the binary connective '$\cdot$' over phonological expressions is non-associative and non-commutative. That is, we interpret categorial formulas as sub-sets of a groupoid.[6]

The (syntactic) derivability relation expressed by $\vdash$ is interpreted as the set-inclusion (or sub-set) relation, as shown in (85).

(85) a. Syntax: Antecedent $\vdash$ Succedent

b. Interpretation: $\nu(Antecedent) \subseteq \nu(Succedent)$

3.2.4 Model Theoretic Interpretation of NL at LF

As in the phonological interpretation, we interpret categorial formulas as sets of semantic objects, represented by typed lambda expressions, as we have seen in chapter 1. I characterize the semantic interpretations as structured objects, regarding the semantic representations as some sort of psychological structured meaning representations such as Language of Thought representations. However, nothing hinges on this formulation, and if a purely denotational characterization of the semantics is preferred, we may alternatively interpret each propositional expression 'ϕ' as a set of information states in which 'ϕ' is true.[7] In any case, the characterization of LF in terms of structure is a cautious suggestion.

(86) a. $\nu_0(NP) = \{tom', meg', \iota(book') \ldots\}$

b. $\nu_0(S) =$

$\{smoke'(tom'), love'(tom')(meg'), give'(meg')(\iota(book'))(tom') \ldots\}$

The atomic function ν_0 is extended to the general interpretation function ν in (87).

[6]As I have indicated before, NL is not complete with regard to free groupoid or binary tree structures with tree adjunction as addition. See Morrill (1994: 66-67) for this point, which refers to Dŏsen (1992) and Venema (1995). See these references, and Versmissen (1996) for some soundness and completeness proofs.

[7]See Wansing (1992), for example, for a characterization of information structures as interpretation of intuitionistic propositional logic.

(87) a. $\nu(A \bullet B) = \{(a \bullet b) \mid a \in \nu(A) \& b \in \nu(B)\}$

b. $\nu(A\backslash C) = \nu(C/A) = \{b \mid \forall a \in \nu(A) : b(a) \in \nu(C)\}$

By (87b), if *smoke'* is a member of $\nu(NP\backslash S)$, then for each $a \in \nu(NP)$, $smoke'(a)$ must be a member of $\nu(S)$.

(87b) interprets directional syntactic functors as sets of non-directional logical functors. However, the multiplicative connective $\bullet$ over lambda terms is non-commutative and non-associative, which corresponds to the non-commutativity and non-associativity of the syntactic system. We do β reduction in two directions, as in (88a) and (88b), but this does not mean that the lambda language system is commutative. The argument should appear in the direction that the syntactic functor category requires it. That is, the type e argument should appear to the left of the functor in $(NP \bullet (NP\backslash S))$ for (88a) and to the right of the functor in $((NP\backslash S)/NP \bullet NP)$ for (88b) in order to derive the expressions of the adequate types (that is, type t with category S for the former and type (et) with category $NP\backslash S$ for the latter) after the conversion steps.[8] Having said that, I do not identify the exact expressive power of the typed lambda language that I use in this thesis. This is partly because I leave for further investigation how much of the full expressive power of the typed lambda language that is required for systematic mappings between the syntactic proofs and the typed lambda terms should be preserved in the LF representations, as opposed to the lambda terms that are used to show the compositional derivations of such LF representations in the syntactic calculus. As an indication of the weaker expressive power of the LF logical expressions, it may be the case that the product '$\bullet$' does not appear in the LF representations, as some operation eliminates all the occurrences of '$\bullet$' by using π_i operators before the LF representations are derived, but this is only a speculation at this stage.

(88) a. $(tom' \bullet \lambda x.smoke'(x)) \Rightarrow (\lambda x.smoke'(x))(tom') \Rightarrow smoke'(tom')$

b. $(\lambda x.\lambda y.love'(x)(y) \bullet meg') \Rightarrow (\lambda x.\lambda y.love'(x)(y))(meg')$
$\Rightarrow \lambda y.love'(meg')(y)$

As I mentioned above, we want to show how the typed lambda expressions as meaning representations are derived in the syntactic proofs. This is easier to do in the Natural Deduction (ND) proofs that I used in section 3. ND proofs look like linguists' syntactic trees up-side down. Thus, we can decorate the leaf nodes with lexical expressions, such as tom', $\lambda x.\lambda y.like'(x)$ and meg', and then successively

[8]However, note that '$((NP\backslash S) \bullet NP)$' itself is a derivable formula in NL. It is only that we cannot derive the category 'S' from that complex formula.

merge them to derive the result expression, such as **like'(meg')(tom')**, at the root node. In contrast, the Gentzen Sequent (GS) presentation is not convenient for showing the step-wise derivation of the concrete lambda expressions in this way. To show how we derive the LF interface representations in GS proofs, we decorate the proofs with a meta-language of typed lambda expressions.[9] The basic idea is as in (89) (cf. Morrill, 1994: 16, 92-95).

(89) $(A_1 : x_1, (A_2 : x_2, \ldots, (A_{n-1} : x_{n-1}, A_n : x_n))) \vdash B : \phi$

For the formulas $A_1, \ldots, A_n$ in the antecedent of each sequent in the syntactic proof, we provide typed variables $x_1, x_2, \ldots, x_n$. The typed variables $x_1, x_2, \ldots, x_n$ represent any lambda expressions of the semantic types corresponding to the syntactic categories $A_1, \ldots, A_n$, according to the mapping function *Type* in (20) in chapter 1. The sequent in (89) says that with variables $x_1, \ldots, x_n$ of the semantic types, $Type(A_1), \ldots, Type(A_n)$, placed in the binary structural configuration, we can derive the logical expression ϕ, whose semantic type is $Type(B)$. In the antecedent side of each sequent, each x_i (for $1 \leq i \leq n$) is unique. This corresponds to the observation that Lambek Calculus is a variant of (intuitionistic) linear logic, which is resource sensitive. In the succedent side, each x_i (for $1 \leq i \leq n$) occurs exactly once as a sub-term of ϕ if the logical expression is unreduced. Thus, unreduced lambda terms can maintain the history of the proof. However, normalization of the lambda expressions may change this situation in the right-hand side of the sequent. Also, after we extend the system to a Multi-modal system and introduce structural rules under modal control, some term replacement/identification may occur in the succedent side, which may lead to situations in which a unique x_i in the antecedent side occurs twice (or zero times) in ϕ in the succedent side. But it is important to remember that either the duplication or apparent disappearance of some x_i in the succedent term occurs either as a result of lambda conversion or as a result of explicitly defined structural rules (i.e. after the introduction of structural rules to the grammar system, see section 3.3 and 3.4 in this chapter).

Now if we replace the variables $x_1, \ldots, x_n$ in the antecedent in (89) with concrete lambda expressions $\alpha_1, \ldots, \alpha_n$ of the right types, then in the succedent of the sequent, we derive the logical expression ϕ', which is ϕ, except that $\alpha_1, \ldots, \alpha_n$

[9]Labelled Deductive Systems, proposed by Gabbay (1991) and discussed by Kurtonina (1994) in TLG, decorate categorial formulas with extra elements such as typed lambda terms and phonological expressions. To them, lambda terms and PF expressions are part of the syntactic structures and therefore, they need to define interpretation rules for them. I use typed meta-variables as decorations just to show how we interpret the syntactic proofs step by step and then produce concrete lambda expressions as a result. The proofs themselves are solely based on the syntactic categories. I do not discuss the different implications of these two assumptions.

replace $x_1, \ldots, x_n$ inside ϕ. (90) provides an example of the use of this meta lambda-language in the goal sequent.

(90) a. $(NP : x_1, ((NP\backslash S)/NP : x_2, NP : x_3)) \vdash S : x_2(x_3)(x_1)$

b. $(NP : tom', ((NP\backslash S)/NP : love', NP : meg')) \vdash S : love'(meg')(tom')$

In (90a), we label the syntactic categories with typed meta variables. Replacing the variables with concrete lambda expressions, as in (90b), we can show the specific semantic sequent that we associate with the phonological sequent in (82a) via the syntactic proof. Note that, in (90a), even though I used concrete categorial formulas such as NP, $(NP\backslash S)/NP$, NP and S, rather than formula meta-variables such as B, $(B\backslash C)/A$, A and C, we still use the meta-variables x_1, x_2, x_3 and $x_2(x_3)(x_1)$ for the lambda expressions. This is because x_1, for example, can potentially be any member of $[\![NP]\!]$, such as *tom'* for *Tom* or *Jack'* for *Jack*.

Now we decorate the axioms in Gentzen Sequent presentation with meta variables for typed lambda expressions. Meta variables that are used in the rule presentation need some explanation.

(91) a. $A, B, C \in F$ (= the set of categorial formulas).

b. $u, v, x, X, \alpha, \beta, \gamma, \phi$ are meta variable lambda terms of the types corresponding to the categories with which they are paired in the form, $A : x$.

c. Γ, Δ are binary configurations of pairs of categorial formulas and typed variable terms. For example, Γ may be $(NP : x, NP\backslash S : \alpha)$ which is made up of two pairs, $NP : x$ and $NP\backslash S : \alpha$.

(91a) and (91b) follow the conventions which I have already explained. (91c) requires some further comment. In this notation, each structure is a maximally binary configuration of pairs of categorial formulas and meta variable terms, rather than a binary structure of categorial formulas as in the categorial sequent calculus. In practice, I always show the categorial calculus separately from the derivation of the logical expressions, and thus, in the derivation of the lambda expressions, I omit the categorial formula in each pair, which increases the readability. But in the rule presentations in (92) and (93), and also in the example derivation in (94), I use pairs of formulas and variable terms as building blocks of structures, in order to show the correspondence between the syntax and the semantics in a clearer way.

I first decorate Identity Axiom and Cut with meta lambda variables in (92). By convention, all the meta variables are universally quantified in the rule statements

and the rule applies to any structures/formulas/lambda terms, as long as each lambda term has the right type for the category it is paired with.

(92) a. Identity Axiom $A : u \vdash A : u$

b. Cut: (σ is free for u in ϕ)

$$\frac{\Gamma \vdash A : \sigma \quad \Delta[A : u] \vdash B : \phi}{\Delta[\Gamma] \vdash B : \phi[u :\rightarrow \sigma]}\ Cut$$

In (92b), '$\phi[u :\rightarrow \sigma]$' means that the term σ replaces the free variable u inside the term ϕ. 'σ is free for u in ϕ' means that when σ replaces the free variable u inside ϕ, no free variable in σ gets accidentally bound by an operator in ϕ.

(93) Logical rules

a.

$$\frac{\Delta[B : X] \vdash C : \gamma \quad \Gamma \vdash A : \beta}{\Delta[(B/A : \alpha, \Gamma)] \vdash C : \gamma[X :\rightarrow \alpha(\beta)]}\ /L \qquad \frac{(\Gamma, A : x) \vdash B : \phi}{\Gamma \vdash B/A : \lambda x.\phi}\ /R$$

b.

$$\frac{\Gamma \vdash A : \beta \quad \Delta[B : X] \vdash C : \gamma}{\Delta[\Gamma, A\backslash B : \alpha] \vdash C : \gamma[X :\rightarrow \alpha(\beta)]}\ \backslash L \qquad \frac{(A : x, \Gamma) \vdash B : \phi}{\Gamma \vdash A\backslash B : \lambda x.\phi}\ \backslash R$$

c.

$$\frac{\Gamma[(A : u, B : v)] \vdash C : \gamma}{\Delta[(A \bullet B) : (u \bullet v)] \vdash C : \gamma[(u, v) \rightarrow (u \bullet v)]}\ \bullet L \qquad \frac{\Gamma \vdash A : u \quad \Delta \vdash B : v}{(\Gamma, \Delta) \vdash (A \bullet B) : (u \bullet v)}\ \bullet R$$

$\alpha(\beta)$ is free for X in γ for $/L$, $\backslash L$. x is fresh for $/R$, $\backslash R$.

Categorial formulas such as NP and $NP\backslash S$ are paired with lambda variables of the corresponding types. Again, the variables are assigned appropriate types according to the mapping function, *Type*. For example, $Type(NP) = e$ for x in the pair $NP : x$ and $Type(NP\backslash S) = (et)$ for P in the pair $NP\backslash S : P$. In (93a) and (93b), $\gamma[X :\rightarrow \alpha(\beta)]$ means that we replace the free variable X inside the expression γ by the expression $\alpha(\beta)$. In order to prevent an accidental binding in $/L$ and $\backslash L$, $\alpha(\beta)$

has to be free for X in γ. That is, it is required that no free variable in $\alpha(\beta)$ ends up being bound by an operator in γ when $\alpha(\beta)$ replaces X.

In $/R$ and $\backslash R$, by saying that x is fresh, we mean that there is no free occurrence of x inside Γ. Thus, in each application of $/R$ or $\backslash R$, exactly one free variable x is bound.

As an example, I decorate the proof in (81b) with typed lambda expressions.

(94)

$$\dfrac{\dfrac{\dfrac{\dfrac{NP:x \vdash NP:x \quad S:X \vdash S:X}{NP:x, NP\backslash S:P \vdash S:X[X :\rightarrow P(x)]}\backslash L \quad NP:y \vdash NP:y}{NP:x, ((NP\backslash S)/NP:R, NP:y) \vdash S:P(x)[P :\rightarrow R(y)]}/L}{NP:x, ((NP\backslash S)/NP:R, NP:y) \vdash S:R(y)(x)}}{NP:tom', ((NP\backslash S)/NP:like', NP:meg') \vdash S:like'(meg')(tom')}\ Inst$$

Term replacements, $X[X :\rightarrow P(x)] = P(x)$; $P(x)[P :\rightarrow R(y)] = R(y)(x)$
Types, $R:(e(et))$; $P:(et)$; $x, y:e$

The sequent in the second line from the bottom is the one that is proved in (94). Instantiating x as tom', R as $like'(= \lambda u.\lambda v.like'(u)(v))$ and y as meg', we have the sequent in the bottom line. In the process that is marked as 'Inst,' the meta-variables are all replaced by constant logical expressions. This means that use of (meta) free variables decorating Gentzen Sequent proofs does not necessarily lead to the problems that Jacobson (1999) mentions with regard to the use of free variables in the logical expressions that represent the meanings of natural language expressions. We can assign the variable free requirements at the level of the derived LF lambda expressions as in the bottom line in (94).

Though I have shown the derivations of logical expressions paired with categories in (94), in the following chapters, categorial proofs/derivations are always presented separately as a 'Syntactic proof,' as in the proof in (81b). Thus, in the semantics of the proofs, I only show the derivation of the logical expressions, omitting the category part A in each pair $A:x$.

In the next section, I discuss the provability of some of the higher order rules that I have used in my proposal.

3.2.5 Argument slot raising as a special rule

In this section, I check the provability/unprovability of the main rules that I use for QNP scope in GS presentation.

As we have seen in chapters 1 and 2, argument slot raising is provable only for the subject argument slot, not for an object argument slot in NL. (95) shows it in Gentzen Sequent proofs.[10]

(95) a. Subject-slot raising: provable b. Object-slot raising: not provable

$$\dfrac{\dfrac{S \vdash S \quad NP\backslash S \vdash NP\backslash S}{S/(NP\backslash S), NP\backslash S \vdash S}\,/L}{NP\backslash S \vdash (S/NP\backslash S))\backslash S}\,\backslash R$$

$$\dfrac{\dfrac{NP\backslash S \vdash NP\backslash S \quad \dfrac{\vdots}{(S/NP)\backslash S \vdash NP}\,?}{(NP\backslash S)/NP, (S/NP)\backslash S \vdash NP\backslash S}\,/L}{(NP\backslash S)/NP \vdash (NP\backslash S)/((S/NP)\backslash S)}\,/R$$

Given the goal sequent in the bottom row in (95b), we can use $/R$ to move up one row in the GS proof. If we could derive the sequent '$(S/NP)\backslash S \vdash NP$' (which is for Type Lowering, instead of Type Raising which is provable in NL) one more row up as a premise sequent, then we could continue the proof, but this Type Lowering sequent is not provable in NL, as the two question marks in the proof indicate. Thus, the argument slot raising with regard to the object position is not provable, either.[11]

Though NL can prove argument slot raising only in terms of the subject argument slot, it allows us to merge the argument slot raised functors with their QNP arguments both for subject and object QNPs. To show this, I first show the proofs of the identity sequents for two Generalized Quantifier (GQ) categories for subject QNPs and object QNPs, respectively.

[10]Argument slot lowering is provable for all the argument positions. For example, NL can prove the sequent, $(NP\backslash S)/((S/NP)\backslash S) \vdash (NP\backslash S)/NP$, with regard to the object argument slot of transitive verbs. Thus, so far as scope switch in terms of the lexical functors such as *love* is concerned, we could encode the higher order category $((S/(NP\backslash S))\backslash S)/((S/NP)\backslash S)$ and the two lambda expressions that would lead to the two scope readings with *love* (i.e. lexical polymorphisms in the semantics). Then, when the arguments of *love* are normal NPs, we can lower the corresponding argument slots to the NP argument slots as above. However, though this alternative analysis lets us stay within NL, this does not work when the functor intervening the two QNPs is a non-lexical, complex predicate that has been derived in a syntactic derivation, as we see in chapter 7.

[11]This backward proof search may seem strange for those who are not used to GS proofs. We start with the conclusion sequent which we want to prove, and then we consider which rule we can use to derive that conclusion sequent from some premise sequent(s). This backward search is normally easy, because, other than Cut and structural rules which we have not yet introduced, and unless the goal sequent is already an identity axiom, we always eliminate one connective from some complex formula in the conclusion sequent, either in the antecedent side or in the succedent side.

(96)

$$\dfrac{S \vdash S \quad \dfrac{\dfrac{NP \vdash NP \quad S \vdash S}{NP, NP\backslash S \vdash S}\backslash L}{NP\backslash S \vdash NP\backslash S}\backslash R}{\dfrac{S/(NP\backslash S), NP\backslash S \vdash S}{S/(NP\backslash S) \vdash S/(NP\backslash S)}/R}/L \qquad \dfrac{\dfrac{\dfrac{S \vdash S \quad NP \vdash NP}{S/NP, NP \vdash S}/L}{S/NP \vdash S/NP}/R \quad S \vdash S}{S/NP, (S/NP)\backslash S \vdash S}\backslash L}{(S/NP)\backslash S \vdash (S/NP)\backslash S}\backslash R$$

Given the identity theorems in (96), we can prove (97).

(97)

$$\dfrac{\dfrac{S/(NP\backslash S) \vdash S/(NP\backslash S) \quad S \vdash S}{S/(NP\backslash S), (S/(NP\backslash S))\backslash S \vdash S}\backslash L \quad (S/NP)\backslash S \vdash (S/NP)\backslash S}{S/(NP\backslash S), (((S/(NP\backslash S))\backslash S)/((S/NP)\backslash S), (S/NP)\backslash S) \vdash S}/L}{(every, boy), (loves, (some, girl)) \vdash (every \cdot boy) \cdot (loves \cdot (some \cdot girl))}PF$$

(97) proves that we can use the argument slot raised transitive verb category with two GQ categories as its arguments without an association rule. In fact, we do not even have to use different GQ categories for the subject and the object QNPs. I could have replaced the identity axiom for the object QNP, $(S/NP)\backslash S \vdash (S/NP)\backslash S$, with the one for the subject QNP, that is, $S/(NP\backslash S) \vdash S/(NP\backslash S)$. But I stick to $(S/NP)\backslash S$ for object QNPs, to respect the left-to-right directionality of merges.

I show how the syntactic proof in (97) derives the semantic sequent. I omit the categorial formulas and only show the semantic meta-variables in (98), for readability.

(98)

$$\dfrac{\dfrac{\dfrac{Q1 \vdash Q1 \quad X \vdash X}{Q1, P \vdash X[X :\rightarrow P(Q1)]}\backslash L \quad Q2 \vdash Q2}{Q1, (R, Q2) \vdash P(Q1)[P :\rightarrow R(Q2)]}/L}{Q1, (R, Q2) \vdash R(Q2)(Q1)}$$

Term replacements:

$X[X :\rightarrow P(Q1)] = P(Q1)$; $P(Q1)[P :\rightarrow R(Q2)] = R(Q2)(Q1)$

Meta-variable types, $Q1, Q2 : (et)t, P : ((et)t)t, R : ((et)t), (((et)t), t)$

Variables $Q1$ and $Q2$ are for GQ type expressions (of type (et)t). R is for the

argument slot raised logical expression for *love*. If we apply argument slot raising to *love* (with its lexical entry, $(NP\backslash S)/NP; \lambda x.\lambda y.love'(x)(y)$, in the syntax and the semantics), then R can be instantiated as two differently argument-slot raised functors, representing the two orders of raising its internal and external argument slots. The two functor expressions in (99b) and (99c) are the two versions, representing the two scope readings, the subject wide scope and the object wide scope readings respectively.

(99) a. $Q1, (R, Q2)$

b. Surface scope:

$\lambda A.every'(boy')(A), (\lambda Q2.\lambda Q1.Q1(\lambda y.Q2(\lambda x.love'(x)(y))), \lambda B.some'(girl')(B))$

c. Inverse scope:

$\lambda A.every'(boy')(A), (\lambda Q2.\lambda Q1.Q2(\lambda x.Q1(\lambda y.love'(x)(y))), \lambda B.some'(girl')(B))$

As we have seen in (95), NL cannot derive the argument slot raised functors in (99b) and (99c) from the lexical entries for the verb *love*. Because of that, (99) at the moment can be derived only by using a special rule which operates on functors of certain categorial forms. On the other hand, once we are equipped with the argument slot raising in (99) as a special rule, then the syntactic proof in (97), and the accompanying derivation of the meta-lambda sequent in (98) do not require any structural association to merge the raised functor with its QNP arguments. The order of the merges of the three expressions is the same both for (99b) and (99c), though their scope readings are different. As I have suggested in chapter 2, this analysis prevents the syntax from generating two different syntactic structures for the two scope readings. Given lack of clear linguistic data that support that the two scope readings are linked to different syntactic structures, I argue that this analysis is more explanatory. At the cost of postulating a limited number of (semantically motivated) special rules, my analysis helps prevent the grammar from generating more structures than are independently motivated in natural language syntax.

In this section, GS proofs have confirmed that argument slot raising with regard to an object argument slot is not provable in NL. However, we have also seen that, given an argument slot raised functor by way of a special rule, we can merge this functor with its QNP arguments in the non-associative grammar NL, avoiding an un-necessary structural ambiguity for QNP scope reasons.

3.2.6 Value raising

Other than argument raising, Hendriks (1987) also mentions value raising. I use value raising in some of the later chapters to switch scope, and thus, I show here that value raising is generally provable in NL. To show the provability of value raising in a general sense, I show the raising of the value category C in the two place functor category $(A\backslash C)/B$, though the proof is easily turned into one for a one-place functor category $A\backslash C$ or C/A.

(100) Value raising of $(A\backslash C)/B$.

$$\dfrac{\dfrac{\dfrac{\dfrac{S \vdash S \quad C \vdash C}{S/C, C \vdash S}\,/L}{C \vdash (S/C)\backslash S}\,\backslash R}{A, A\backslash C \vdash (S/C)\backslash S}\,\backslash L \qquad B \vdash B}{A, ((A\backslash C)/B, B) \vdash (S/C)\backslash S}\,/L}{(A\backslash C)/B, B \vdash A\backslash((S/C)\backslash S)}\,\backslash R}{(A\backslash C)/B \vdash (A\backslash((S/C)\backslash S))/B}\,/R$$

Because the value raising of the functor category $(A\backslash C)/B$ with regard to its final output category C only changes the functor-argument relation between this output C and the next category to be merged (which is, in (100), stipulated as S/C for convenience, but S in this functor category S/C can be any category), the value raising does not require re-bracketing of the binary structures, and thus, is generally provable in NL.

Value raising is used in chapter 6 for switching scope when a QNP appears in a PP which is inside the nominal restriction of another QNP.

3.3 Multi-Modal Lambek Calculus, $NL\Diamond$

3.3.1 Syntax

I have adopted the non-associative and non-commutative Lambek Calculus NL as the base grammar system. Though NL is attractive because of its restrictiveness, it does not accommodate any extraction phenomena. It does not allow us to form complex predicates either, as we have informally seen in chapter 2.

(101) a. No complex predicate for *can play* in *Tom [can play] but [will not play]*

tennis:

$$\frac{(NP\backslash S)/(NP\backslash S), (NP\backslash S)/NP \not\vdash_{NL} (NP\backslash S)/NP}{can, play \not\vdash_{NL} (can \cdot play)} PF$$

b. No A-bar extraction for *Broccoli, Ad likes*:

$$\frac{S/(S/NP), (NP, (NP\backslash S)/NP) \not\vdash_{NL} S}{Broccoli, (Ad, likes) \not\vdash_{NL} Broccoli \cdot (Ad \cdot likes)} PF$$

Combinatory Categorial Grammar (CCG), as in Steedman (2000b) merges categories in non-standard ways by using Combinators. For example, remember the directional function composition Combinator that we have briefly seen in chapter 2, repeated here as (102a).

(102) a. Syntax: $C/B + B/A \Rightarrow C/A$
Semantics: $g_{(b,c)} + f_{(a,b)} \Rightarrow (g \bullet f)$, where $(g \bullet f)_{(a,c)} = \lambda x_a . g(f(x))$
Cf. Steedman (2000:40)

b. Forward composition Combinator:
Category: $((C/A)/(B/A))/(C/B)$ for any $A, B, C \in F$
PF: ϵ (that is, PF null), LF: omitted.

$$\frac{(((C/A)/(B/A))/(C/B), (NP\backslash S)/(NP\backslash S)), (NP\backslash S)/NP \vdash_{NL} (NP\backslash S)/NP}{(Combinator, can), play \vdash_{NL} (\epsilon \cdot can) \cdot play} PF$$

where $A = NP$; $B = (NP\backslash S)$; $C = (NP\backslash S)$ in this particular derivation.

There are two ways of understanding Combinators. First, as is the case with Steedman and most CCG practitioners, we can see the addition of Combinators as a modification of the basic algorithm of the grammar system. In this interpretation, we simply cannot use Combinators in Type Logical Grammar. Because of the way in which Type Logical Grammar works, we have to achieve the effects of introducing Combinators either by defining new formula connectives, as in Morrill (1994) or, as we do in this chapter, we have to decompose the effects of Combinators into structural rules such as structural association and permutation.

Though it is not a common practice, we could still instantiate Combinators as PF null operators in TLG. For example, as in (102b), we could insert a PF null function composition Combinator of the suggested category as the left sister of the auxiliary verb functor of category $(NP\backslash S)/(NP\backslash S)$ (where ϵ indicates the PF null status of this Combinator). If we instantiated Combinators in this way, then, the

syntactic behaviors of the connectives / and \ would stay the same (that is, they stay as non-associative and non-commutative connectives).

As we see in application chapters, I do use a limited number of PF null items, such as an operator that concatenates two object NPs in a ditransitive verb construction in chapters 4 and 5. Thus, some might wonder if we can instantiate Combinators in the same way. However, there is a significant difference between the roles Combinators play in CCG, and the roles some PF null items play. CCG uses Combinators to describe how the categorial calculus works. In other words, adding Combinators to the grammar is meant to change the generative power of the grammar system abstracted away from which categories are assigned to which lexical items. In contrast, though using PF null categories changes the expressive power of the grammar system in terms of which PF strings/structures it can generate, it does not change the categorial calculus itself. That is, as I implied above, if we achieved the effect of forward function composition by inserting a PF null operator in a relevant position, then the syntactic calculus itself would stay non-associative (and non-commutative). Some might argue that this is a good thing, but postulating a PF null item to achieve the effect of a 'syntactic operation' misses the point, because what we want to know is the exact identity of the syntactic system which can explain the structures that are relevant to the natural language phenomena. According to our general data observations, underlying syntactic structures of natural language strings are not totally non-associative (though they are not totally associative either). Then what we want to know is what degree of associativity can capture the structures that apply to natural language strings. Insisting that natural language syntax is non-associative by inserting as many PF null items as we want in the positions that achieve our descriptive goal totally misses the point of our investigation; it makes it difficult to find out what degree of associativity or commutativity is supported by natural language phenomena.

This should not be taken as a criticism of any theory which uses PF null items. Note that PF null items in a theory such as Minimalism are strictly restricted to cases where we can find some justification at the data level. For example, T for 'tense' may host a lexical item (such as an auxiliary verb, or even a verbal head as in French), though T may not be encoded with a particular lexical item, and the PF null functor that I postulate in the double object construction is partly supported by the existence of an overt preposition between the two objects in the PP ditransitive construction. In contrast, Combinators in CCG are postulated so that they can merge lexical categories in a way that is different from function application (i.e. $/E$, $\backslash E$ in ND presentation and $/L, \backslash L$ in GS presentation). Thus, any effect of

adding Combinators to the grammar should be reflected as some modification of the syntactic calculus itself, not in terms of inserting PF null items without modifying the grammar system itself.

As I have indicated above, if we do not use Combinators as PF null functors, then in order to explain complex predicate formation, scrambling or overt extraction phenomena which are well attested in natural language data, we either have to introduce new categorial connectives to the grammar, or we need to introduce structural rules. However, simplistic introduction of structural rules as in (103) in a uni-modal grammar system such as NL would collapse the total grammar system to an associative one or a commutative one.

(103) a. Simple association (bad)

$$\frac{X_1, (X_2, X_3) \vdash A}{(X_1, X_2), X_3 \vdash A}$$

b. Simple permutation (bad)

$$\frac{X_1, X_2 \vdash A}{X_2, X_1 \vdash A}$$

where $X_1, X_2, X_3 \in S$, $A \in F$.[12]

The rules as in (103) would spoil the restrictiveness of the non-associative and non-commutative grammar system NL. The problem is that because NL is uni-modal, all the rules interact with one another with full generality. Thus, the additional structural rules as in (103) would influence the whole grammar system and turn it into a fully associative/commutative grammar which would overgenerate natural language strings.

In order to introduce just the right degree of structural associativity and commutativity, I adopt a Multi-Modal Type Logical Grammar system as in Moortgat (1997). The particular grammar that I use is $NL_{\diamond_v}$, which is based on NL, but enriched with a certain degree of structural association and permutation. If we add argument slot raising as a special rule, then the total grammar system becomes $NL_{\diamond_v+ASR}$, but I abstract away from argument slot raising in the rest of this chapter.

[12]When I present structural rules, such as structural association and structural permutation, I use X, Y, Z as structural variables, instead of using Γ, Δ as I do when I present logical rules. This is because I need to use more structural variables when I show structural rules. Number subscripts as in X_n are for presentation reasons only, that is, for distinguishing variables from one another.

We start by defining the set of categorial formulas again. The definitions are as in Moortgat (1997).

(104) $F_{\diamond x,y} ::= At \mid F/_x F \mid F\backslash_x F \mid F \bullet_x F \mid \diamond_y F \mid \square^{\downarrow}_y F$
where $x, y \in I$ (= the set of mode indices).

In (104), we have two unary categorial operators, $\diamond$ and $\square^{\downarrow}$, whose semantics are provided shortly. These operators add extra expressive power to the grammar system. Also, each connective comes with a 'mode index' which specifies the mode of (binary) merge for the binary operators, and the mode of unary projection for unary operators. The revised grammar system uses families of connectives, $\{\backslash_x, /_x, \bullet_x\}$ (as well as $\diamond_y, \square^{\downarrow}_y$) for the set of mode indices I, rather than the single set of connectives $\{\backslash, /, \bullet\}$. This also adds extra expressive power to the grammar system, as we see shortly.

Just like the residuation laws that apply between the binary connectives $\backslash, /$ and $\bullet$, the unary connectives $\diamond$ and $\square^{\downarrow}$ enter into a residuation law as in (105).

(105) Laws of residuation (derived properties of $NL_{\diamond_y}$)

a. $A \vdash C/_x B \;\Leftrightarrow\; A \bullet_x B \vdash C \Leftrightarrow\; B \vdash A\backslash_x C$

b. $\diamond_y A \vdash B \;\Leftrightarrow\; A \vdash \square^{\downarrow}_y B$

Given the set of categorial formulas as in (104), the set of structures is provided as in (106). For representing structural rules, I explicitly specify the binary structural connective $\circ_x$, as in (106).

(106) The set of structures, $S_{\diamond x}$.
$S_{x,y} ::= F \mid (S \circ_x S) \mid (S)^{\diamond_y}$

For comparison, $(S \circ_x S)$ would be $(S, S)^x$ if I adopted the structural notation that I used for the uni-modal NL in (75), but this alternative notation is less convenient when we want to specify the mode of each binary merge, especially with the mode specification of the unary mode. Thus, I use the notation in (106) when I show a derivation that involves structural rule applications.

We redefine the logical rules in (78) in a way that is sensitive to the mode specification.[13] As usual, all the formula/structure variables are universally quantified, so each rule applies with full generality to any formula/structure, as long as the condition stated in each rule is satisfied.

[13]Identity axioms and Cuts do not influence the number of connectives, and so the mode specification is not relevant.

(107) Logical rules:

a.
$$\frac{\Delta[B] \vdash C \quad \Gamma \vdash A}{\Delta[(B/_x A \circ_x \Gamma)] \vdash C} /L \qquad \frac{(\Gamma \circ_x A) \vdash B}{\Gamma \vdash B/_x A} /R$$

b.
$$\frac{\Gamma \vdash A \quad \Delta[B] \vdash C}{\Delta[(\Gamma \circ_x A\backslash_x B)] \vdash C} \backslash L \qquad \frac{(A \circ_x \Gamma) \vdash B}{\Gamma \vdash A\backslash_x B} \backslash R$$

c.
$$\frac{\Delta[(A \circ_x B)] \vdash C}{\Delta[(A \bullet_x B)] \vdash C} \bullet L \qquad \frac{\Gamma \vdash A \quad \Delta \vdash B}{(\Gamma \circ_x \Delta) \vdash (A \bullet_x B)} \bullet R$$

Meta-variables: $A, B, C \in F$ and $\Gamma, \Delta \in S$

Note that in (107), the mode index x must be the same between the (categorial) formula connective and the structural connective for each rule. Thus, we can lexically control the merge mode by encoding the mode information with the lexical functor expression, as in $(NP\backslash_j S)/_j NP$ for transitive verbs or in $(NP\backslash_j S)/_j(NP\bullet NP)$ for ditransitive verbs, as we see in the application chapters.[14] Thus, the grammar system is still compatible with Chomskian idea of Inclusiveness, in which everything used in syntactic derivations comes from lexical specifications. The syntactic proof can be read off the lexical categorial information.

The logical rules for the two unary operators are as in (108).[15]

(108) a.
$$\frac{\Delta[(A)^{\diamond_\nu}] \vdash B}{\Delta[\diamond_y A] \vdash B} \diamond_y L \qquad \frac{\Gamma \vdash A}{(\Gamma)^{\diamond_\nu} \vdash \diamond_y A} \diamond_y R$$

b.
$$\frac{\Delta[A] \vdash B}{\Delta[(\Box^{\downarrow}_y A)^{\diamond_\nu}] \vdash B} \Box^{\downarrow}_y L \qquad \frac{(\Gamma)^{\diamond_\nu} \vdash A}{\Gamma \vdash \Box^{\downarrow}_y A} \Box^{\downarrow}_y R$$

Cf. Versmissen (1996: 56), Moortgat (1997)

[14]The index x on the product $\bullet_x$ does not normally play essential roles, and thus, I omit it from time to time.

[15]I adopt the notations as in Versmissen (1996) for the rules of MMTLG for presentational convenience. For the original rule presentations of Moortgat's MMTLG, see Moortgat (1997).

By using the rules in (108), we can derive the residuation in (105b), though I do not show the proofs here. Other important derived theorems in application are presented in (109a) and the proofs are provided in (109b).

(109) a. $\Diamond_y \Box^{\downarrow}_y A \vdash A \vdash \Box^{\downarrow}_y \Diamond_y A$

b.

$$\dfrac{\dfrac{A \vdash A}{(A)^{\Diamond_y} \vdash \Diamond_y A}\,\Diamond_y R}{A \vdash \Box^{\downarrow}_y \Diamond_y A}\,\Box^{\downarrow}_y R \qquad \dfrac{\dfrac{A \vdash A}{(\Box^{\downarrow}_y A)^{\Diamond_y} \vdash A}\,\Box^{\downarrow}_y L}{\Diamond_y \Box^{\downarrow}_y A \vdash A}\,\Diamond_y L$$

One way entailment relations as in (109a) are useful for various asymmetric linguistic phenomena, such as superiority effects of multiple Wh-movement (see Vermaat (2006)).

Note that the other direction is not provable, as indicated in (110).

(110) $\Box^{\downarrow}_y \Diamond_y A \nvdash A \nvdash \Diamond_y \Box^{\downarrow}_y A$

The sequent '$\Box^{\downarrow}_y \Diamond_y A \vdash A$' is not provable in $NL_{\Diamond_y}$ because in order to introduce the operator '$\Box^{\downarrow}_y$' in the antecedent of the conclusion sequent according to the rule $\Box^{\downarrow}_y L$ in (108b), we would have to embed the formula headed by '$\Box^{\downarrow}_y$' into a structure in the form of '$()^{\Diamond_y}$'. '$A \vdash \Diamond_y \Box^{\downarrow}_y A$' is not provable because in order to introduce the operator '$\Diamond_y$' in the succedent of this sequent, the antecedent structure of the sequent must have the from, '$()^{\Diamond_y}$', as we can see in the rule $\Diamond_y R$ in (108a). Seen from the other side, the proofs of both the sequents must start with the atomic identity axiom $A \vdash A$. The only rules we can apply with this identity axiom as a premise are $\Diamond_y R$ and $\Box^{\downarrow}_y L$, as we can see in (109). After reaching the second rows of the proofs in (109b), all we can do is either to proceed as in (109b), or apply $\Diamond_y L$ or $\Box^{\downarrow}_y R$ instead, proving the complex identity axioms as in (111).

(111)

$$\dfrac{\dfrac{A \vdash A}{(A)^{\Diamond_y} \vdash \Diamond_y A}\,\Diamond_y R}{\Diamond_y A \vdash \Diamond_y A}\,\Diamond_y L \qquad \dfrac{\dfrac{A \vdash A}{(\Box^{\downarrow}_y A)^{\Diamond_y} \vdash A}\,\Box^{\downarrow}_y L}{\Box^{\downarrow}_y A \vdash \Box^{\downarrow}_y A}\,\Box^{\downarrow}_y R$$

Unprovability in '$NP \nvdash \Diamond_y \Box^{\downarrow}_y NP$' is especially important in this thesis, because this means that we cannot insert the normal category NP instead of the hypothetical category $\Diamond\Box^{\downarrow}NP$, whereas the provability of '$\Diamond\Box^{\downarrow}NP \vdash NP$' means that we can insert $\Diamond\Box^{\downarrow}NP$ instead of NP (though this hypothetical category must be discharged later), as we see in section 3.3.4 and in chapters 9 and 10.

3.3.2 PF interpretation

The PF interpretations of the binary connectives are basically the same as before, but here the PF interpretations are defined in terms of Kripke frames.

A frame is made out of the set of PF strings Φ and a ternary relation R^x for each x and a binary relation R^{2y} for each y between PF strings. The set of PF strings is closed with regard to the relations R^x and R^{2y}.[16]

(112) a. Frame := $\langle \Phi, R^x, R^{2y} \rangle$, for $x, y \in Index$,
where $Index$ corresponds to the set of mode indices, I in (104).

b. PF_{lex} is the set of PF lexical items, i.e. $\{Jack, Bob, smoke, fast \ldots\}$.
where $PF_{lex} \subset \Phi$.

Abstracted away from the effect of the use of $\Diamond_y, \Box^{\downarrow}_y$ operators for introducing structural rules in the syntax, the binary connective in $(a \cdot b)$ is non-associative and non-commutative, corresponding to the non-associative and non-commutative $\bullet$ in the base grammar NL. Even after introducing different modes of merge, I assume that the binary merge $\circ_x$ (and therefore each set of logical connectives $\bullet_x, \backslash_x, /_x$) is non-associative and non-commutative within each mode.[17] Addition of structural rules across different merge modes would force us to introduce more than one ternary and binary relations in the PF frames, but I do not provide the exact identity of the PF models which correspond to $NL_{\Diamond_y}$ in this thesis. The interpretation of unary operators in the PF algebra is especially problematic and the exact identity of the PF objects is beyond the scope of my thesis.[18] The maximal expressive power of such grammar will be huge, and thus, we want to avoid using its full expressive power somehow.[19] Given the extra complexity that use of unary operators and merge modes introduces to the PF interpretations, I should make a thorough comparison of MMTLG with Morrill's uni-modal grammar system with wrapping connectives in terms of the complexity of the PF algebra, but I leave such comparison for future research.

Given the structured PF items as in (112), a PF model is defined as $\langle Frame, [\![\cdot]\!] \rangle$. The interpretation function $[\![\cdot]\!]$ interprets the binary connectives in

[16] I do not specify the exact identity of such relational structures. See Kurtonina (1994) for a thorough discussion of interpreting Moortgat's MMTLG in terms of Kripke Frames with ternary and binary relations.

[17] I have abstracted away from the effect of $\Diamond$ and $\Box^{\downarrow}$ within each mode.

[18] See Venema (1995) for his interpretation of a related unary operator, in comparison to Kurtonina (1994).

[19] Because of learnability, we should limit the number of modes to linguistically well-motivated ones.

the syntax as ternary relations in the PF structures, as in (113). The relational interpretation of connectives in Lambek systems is attributed to Moortgat (1994). Interpreting the product in terms of a ternary relation stems from modal logic (cf. Kripke (1963)). The notation in (113) is based on the presentation in Versmissen (1996).

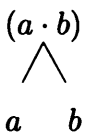
(113) a. $[A \bullet_x B] = \{c \in \Phi \mid \exists a \in [A], \exists b \in [B] : R^x\, cab\}$

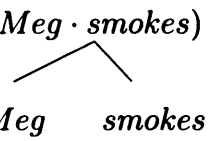
b. $[A \backslash_x C] = \{b \in \Phi \mid \forall a, c \in \Phi : ((a \in [A] \;\&\; R^x cab) \rightarrow c \in [C])\}$

c. $[C /_x B] = \{a \in \Phi \mid \forall b, c \in \Phi : ((b \in [B] \;\&\; R^x cab) \rightarrow c \in [C])\}$

Cf. Moortgat (1994), Versmissen (1996: 54)

To make some informal sense out of the presented relational structures, the ternary relation R^x can be loosely understood as in (114).[20] I abstract away from the effect of modalization of the syntax here and ignore the mode index x.

(114) a.


b.


The relation *Rcab* is as in (114a). That is, if we concatenate the two PF items a and b left-to-right, then we get $c = (a \cdot b)$, as a result. Again the binary connective '$\cdot$' in $(a \cdot b)$ is non-associative and non-commutative. Given this ternary relation R, and given the interpretation rules of categorial formulas as in (113), we can prove that NL is sound and complete with regard to the PF Model $< \Phi, [\cdot], R >$. Introducing modality makes the PF algebra more complex, if we maintain soundness and completeness of $NL_{\Diamond_y}$ with regard to the modified PF interpretations, but I leave the investigation of the matching PF structure for $NL_{\Diamond_y}$ for future research.[21] See Bernardi (2002), chapter 2, for an example of the PF algebra for MMTLG.

Ignoring the mode index, the PF interpretation of $\Diamond A$ and $\Box^{\downarrow} A$ are as in (115).

(115) a. $[\Diamond A] = \{b \mid \exists a \in \Phi : R^2 ab \;\&\; a \in [A]\}$

b. $[\Box^{\downarrow} A] = \{b \mid \forall a \in \Phi : (R^2 ba \rightarrow a \in [A])\}$

In analogy to modal logic, we can interpret $\Diamond$ and $\Box^{\downarrow}$ as directional accessibility relations between possible worlds, or information states. Note that $\Diamond$ and $\Box^{\downarrow}$ are

[20] The comparison between relational structures and bracketed strings is only for convenience. Remember that NL is incomplete with regard to bracketed strings or free groupoid, whereas NL is complete with regard to a specific class of relational models. See Kurtonina (1994) and Venema (1996) and some references therein for the definition of the class of relational models with regard to which NL is complete and also for some discussion about the comparison between relational structures and tree models.

[21] We would also consider the effect of the addition of identity elements to the syntax.

interpreted in opposite directions in terms of the binary (accessibility) relation R^2 in the PF models. This corresponds to the law of residuation in the syntax, as we have seen in (105b).

3.3.3 LF interpretation

From the viewpoint of using syntax to pair PF strings with their LF representations, many of the structural rules (such as association and permutation) are used to realize flexible PF linearization without changing the LF interpretations. For example, both $Jack \circ_j (can \circ_i (play \circ_j tennis))$ and $Jack \circ_j ((can \circ_i play) \circ_j tennis)$ are paired with the same LF term $can'(play')(tennis')(jack')$, where the second PF is the result of re-breacketing a sub-structure in the first PF. Things are not that simple once we try to make $NL_{\diamond_y}$ sound and complete with regard to the intended semantics, but I do not investigate the identities of such semantic representations. See Bernardi (2002), chapter 1 and chapter 2.

The informal assumption is that the lambda expressions that represent what I call the LF representations are the expressions of the same language that can also represent proper semantic objects, whether the proper semantic objects are some psychological semantic representations such as Language of Thought representations, or some model theoretic objects in the world. In contrast, the meta variables that decorate the syntactic proofs (such as Gentzen sequent proofs, but possibly even for ND derivations) to explain the derivation of such LF representations via the categorial calculus has a different status. Those meta lambda expressions are theoretical tools for representing how the proposed syntactic calculus can pair PF structures with their LF meanings step-by-step. The typed lambda language that is used as such meta language expressions decorating syntactic proofs should be in tight correspondence to the syntactic proofs, by way of Curry-Howard Isomorphism between syntactic proofs and typed lambda terms, for example, but there is no strong reason to assume that the lambda language that is used to represent the LF meanings has the same expressive power. For example, as I have indicated before, probably all the $\bullet$ connectives are eliminated by way of conversion using π_i operators by the time we derive the proper LF representations.

3.3.4 Two ways of introducing structural rules

Being equipped with the extra expressive power of $NL_{\diamond_y}$, I show how we can introduce structural rules without collapsing the basic grammar NL into an associative

and commutative grammar in the following chapters. In this subsection, though, I provide rough ideas about how I distinguish QNP scope, A-movement phenomena, and A-bar movement phenomena from one another and how these three interact or do not interact.

First off, argument slot raising in my analysis is a special rule and thus, it does not use any of the additional expressive power of $NL_{\diamond_v}$. Thus, without structural rules, QNP scope switch is possible only between two QNPs which are co-arguments of one functor, whose category is normally $(NP\backslash S)/NP$ (though there is still an exceptional case, in which scope may be switched by way of value raising when one QNP appears in the nominal restriction of another QNP. See chapter 5). However, the argument slot raising as a special rule may apply to complex predicate expressions that the grammar can generate by way of controlled structural rules. Thus, some kind of structural rules may extend the scope of QNPs.

Structural rules for A-movement.

Some of the A-movement phenomena are explained in terms of the structural rules controlled by the binary merge mode specification. Consider (116).[22]

(116) Mixed Association with the modes, i, j:

$$\frac{A \circ_i (B \circ_j C) \vdash D}{(A \circ_i B) \circ_j C \vdash D}\, MA_{i,j}$$

Cf. Moortgat (1997: 131)

From a formal viewpoint, the mode specification is just a way of avoiding the collapse of NL to an associative grammar by addition of the structural association rule. However, we can add some linguistic justification for the choice of the particular merge modes. That is, I argue that the mode of merge j is for merging functors with their NP arguments. The mode i is for merging higher order verbal functors such as control verbs, modal verbs or raising verbs with their VP complements, as in *try (to) run*, *seem (to) run* or *may swim*. With this stipulation, we can deal with A-movement phenomena in control/raising/modal constructions just as re-ordering of the merges, as is indicated in the VP structures in (117).

(117) a. Jack [$_{VP}$ tried to play tennis].

[22]In chapter 7, I present the rule in a slightly different way, and thus change the name of the rule, though the content of the rule stays the same.

b. $VP = (NP\backslash S)$:

$$\frac{\dfrac{VP/_iVP \circ_i (VP/_iVP \circ_i (VP/_jNP \circ_j NP)) \vdash VP}{(VP/_iVP \circ_i (VP/_iVP \circ_i VP/_jNP)) \circ_j NP \vdash VP}\, MA_{i,j} \times 2}{(try \circ_i (to \circ_i play)) \circ_j tennis \vdash (try \cdot (to \cdot play)) \cdot tennis}$$

c. VP:

$try \circ_i (to \circ_i (play \circ_j tennis)) \Rightarrow_{MA_{i,j}} try \circ_i ((to \circ_i play) \circ_j tennis)$

$\Rightarrow_{MA_{i,j}} (try \circ_i (to \circ_i play)) \circ_j tennis$

Versmissen has introduced structural rules in a similar way to deal with verb-raising in Dutch. See chapter 7 in Versmissen (1996). This way of introducing structural rules is different from another way of introducing structural rules which we see later in that we merge exactly the same categorial formulas only in different orders before and after the structure rule application. Thus, this way of introducing structural rules is convenient to postpone the saturation of an NP argument slot of the lower verb *play* until we merge the higher verb *try*, as in (117). After generating the complex functor *try (to play)*, we can apply argument slot raising to this complex functor and switch scope between its subject to the left and the object to the right, as we see in chapter 4. By using $MA_{i,j}$, we can only associate the structure as far as the mode i continues to the left. With an additional linguistic assumption that the i mode continues only until we generate the maximal verbal projection in each tensed clause, as is informally shown in $T \circ_i (v \circ_i (V \circ_j NP))$, we can naturally derive the tensed clause locality constraint on QNP scope.

Wh-extraction

A different way of introducing structural rules is to do so under the control of a unary operator, as in (118a).

(118) a.

$$\frac{A \circ (B \circ \Diamond C) \vdash D}{(A \circ B) \circ \Diamond C \vdash D}\, AR_{\Diamond}$$

b.

$$\dfrac{\dfrac{\dfrac{Y[A] \vdash B \quad \Diamond\Box^{\downarrow}A \vdash A}{Y[\Diamond\Box^{\downarrow}A] \vdash B}\, Cut}{\vdots}}{\dfrac{C \vdash C \qquad \dfrac{Y' \circ \Diamond\Box^{\downarrow}A \vdash B}{Y' \vdash B/\Diamond\Box^{\downarrow}A}\, /R}{C/(B/\Diamond\Box^{\downarrow}A) \circ Y' \vdash C}\, /L}$$

$A.B, C \in F$; $Y, Y' \in S$, where Y' is Y minus A in the designated position.

In (118a), the structural association is controlled by the unary operator $\Diamond$. Unless we further restrict the structural association in terms of some binary merge mode specification, as we did in (116), this association is not constrained by locality constraints, and thus is suitable for explaining Wh-movement, which is not blocked by tensed clauses.[23]

Also, as is shown in (118b), when we apply this method to deal with Wh-extraction, we merge the 'extracted item' as something other than the in-situ category A (that is, the category which would be assigned to the 'lowest trace' in the movement theory, or to the 'tail copy' in the copy theory in generative grammar). In the sequent in the bottom line in (118b), the extracted item is merged as a functor category $C/(B/\Diamond\Box^{\downarrow}A)$, which selects the argument category $B/\Diamond\Box^{\downarrow}A$. As we see in detail in chapter 8, this algorithm is suitable for dealing with A-bar movement such as Wh-movement, in which the extracted item is interpreted as a higher order operator, rather than, say, as a type e expression.[24]

Finally, though I use the word 'movement' to describe the linguistic phenomena, the terminology is for presentation only. Looking at the proved sequent in the bottom line in (118b), notice that the in-situ category A (or the hypothetical category $\Diamond\Box^{\downarrow}A$ which can be used as A in proofs, because of $\Diamond\Box^{\downarrow}A \vdash A$) does not

[23]On the other hand, we need to add some additional island constraints to Wh-movement. See chapter 8 on how we can assign locality constraints in a different way from specifying the modes of binary merges which we want to use only for complex predicate formation and some linguistically related phenomena.

[24]Though in overt topicalization as in *Broccoli, Ad likes*, the merge of *Broccoli* as a higher order category, such as $S/(S/\Diamond\Box^{\downarrow}NP)$ might be neutralized via normalization of the resultant logical form, depending on which lambda expression we assign to the topicalized NP *Broccoli*. I abstract away from this issue, because the exact semantics of topic/focus is beyond the scope of my thesis.

appear in the antecedent of this proved sequent. Because the categories appearing in the antecedent of the proved sequent are the only items that are merged (categories appearing in the higher rows are there just for showing how we can prove the final sequent by using the rules of the categorial calculus), this formalism is fundamentally different from movement theories or copy theories that actually merge either traces or tail copies at terminal/leaf nodes. In principle, the use of a hypothetical category marked with $\Diamond$ and its discharge later in the derivation is the same as in the conditional proof (CP) in classical propositional logic. Consider the natural deduction proof in (119) for classical propositional logic.

(119)

$$\dfrac{\begin{array}{c}[\phi]^1 \\ \vdots \\ \psi\end{array}}{\phi \rightarrow \psi}\, CP^1$$

What (119) shows is that, if we have proved ψ by using ϕ as a hypothetical premise, then it means that we have proved $\phi \rightarrow \psi$ without ϕ. Because the hypothetical premise ϕ is discharged, the proof of $\phi \rightarrow \psi$ does not depend on ϕ.

3.4 Reflexive/pronominal binding

Though this thesis does not discuss binding data as its main target, QNP scope and binding data interact with one another. Also, we need some basic algorithm for binding when we deal with ditransitive constructions in chapter 4 and when we explain the 'exceptional scope' of indefinites as an anaphoric dependency of nominal restriction sets to other elements. Thus, in the final section of this chapter, I show how we can use structural rules to generate pronominal and reflexive binding. The methods that I use are applications of the available tools of Multi-Modal Type Logical Grammar.

Pronominal binding is not constrained by obvious locality constraints. In this regard, it is natural to use the same mechanism that I use for Wh-movement. Wh-movement, of course, is known to be blocked by various islands, but compared with A-movement phenomena and obligatory control phenomena, both pronominal binding and Wh-extraction can form dependency relations that are longer in distance in the PF string. Algorithmically, the basic idea is that neither Wh-movement nor pronominal binding is sensitive to the modes of merge of the intervening items, that

is to say, the items that intervene between the 'in-situ position' and the 'landing site' for Wh-extraction, and the items that intervene between the bound pronoun and its binder for pronominal binding. Thus, lack of locality constraints is the default setting in the algorithms used for both of these phenomena. For Wh-movement, we can add further locality constraints by using the unary connective '$\Box^{\downarrow}$,' as we see in chapter 9. I do not have a definitive explanation as to why those additional locality constraints are applicable only to Wh-extraction. However, as we have discussed in the previous section, we do not merge any overt item in the in-situ position for Wh-extraction, whereas for pronominal binding, the in-situ position is occupied by a (bound) pronoun. The presence of this overt item for pronominal binding makes a difference in formalism, as we see shortly, and we may relate the different locality constraints applicable to the two phenomena to this difference. I leave the incorporation of such consideration into the formal algorithm for future research.

The particular linguistic analysis that I adopt for pronominal binding is Jacobson's analysis as in Jacobson (1992b) and Jacobson (1999). Jacobson regards pronouns as identity functions which introduce extra argument slots that can be percolated until later stages of derivation for a long distance and then can be bound by an element that is merged later. Jacobson's pronominal binding mechanism has also been used for explaining Wh-extraction (with some additional locality constraints) (cf. Dowty (1992)), and thus, it is an appropriate algorithm for us to adopt after we have assumed that the two phenomena have something crucial in common.

On the other hand, there is a technical problem with regard to instantiating Jacobson's 'super-script' categories in Type Logical Grammar. To deal with this, I use the basic mechanisms as in Dowty (1992) which itself is a development of Hepple (1990), but I do so in the MMTLG framework that I have explained in this chapter.

3.4.1 Jacobson's Binding analysis

To deal with binding phenomena in Combinatory Categorial Grammar (CCG), Jacobson (1999) uses 'super-script' categories as in (120).

(120) a. Syntax: B^A, e.g. NP^{NP}, S^{NP}, etc.

b. Semantic types:
$Type(B^A) = Type(A\backslash B) = Type(B/A) =< Type(A), Type(B) >$
E.g. $Type(NP^{NP}) =< e, e >$, $Type(S^{NP}) =< e, t >$, etc.

For example, pronouns are assigned category NP^{NP} with the logical expression $\lambda x.x$ of type $< e, e >$.[25] Though the semantic type of category B^A is the same as the 'functor' category B/A (or $A\backslash B$), we cannot merge this 'special functor category' with its argument category A in the same way as with B/A and $A\backslash B$. In fact, B^A is not merged as a functor. This special category with a superscript is meant to be merged as an argument with a functor which selects the category B. However, because of the superscript category A attached to it, we cannot merge B^A with, say, a functor category C/B by an application of $/E$ (or forward application in CCG). Thus, Jacobson defines the following Combinator which transforms the functor category C/B so that the resultant functor can be merged with B^A as its argument.

(121) Geach Combinator (generalized).

a. Syntax: $g_{C_0}(B/A) = B^C/A^C$, $g_{C_0}(A\backslash B) = A^C\backslash B^C$

Semantics: If f is a function of type $\langle a, b\rangle$, then $g_{C_0}(f)$ is a function of type $\langle\langle c, a\rangle, \langle c, b\rangle\rangle$, where $g_{C_0}(f) = \lambda V[\lambda C[f(V(C))]]$.

b. Syntax: $g_{C_n}(A^B) = g_{C_{n-1}}(A)^B$

If f is a function of type $\langle d, \langle a, b\rangle\rangle$, then $g_{c_n}(f)$ is a function of type $\langle d, \langle\langle c, a\rangle, \langle c, b\rangle\rangle$, where $g_{C_n}(f) = \lambda D.[g_{C_{n-1}}f(D)]$

c. Eg. $g_{C_1}((D/A)^B) = g_{C_0}(D/A)^B = (D^C/A^C)^B$, with appropriate semantics.

(Jacobson 1999:138)

The g_{C_0} Combinator in (121a) allows Jacobson to percolate the 'extra-argument slot' C from the argument category A^C to the value category B^C. (121b) is not directly relevant to us, and I ignore this here.[26] If I apply the idea to the determiner phrase *his mother* (of category NP^{NP} in Jacobson's analysis) and a transitive verb, the process is as in (122).[27]

[25] Jacobson treats both bound and free pronouns in this way. This assumption has a theoretical merit of eliminating use of free variables in meaning representations. On the other hand, her assumption implies that when the extra argument-slot which is introduced by a pronoun remains "unbound" in the syntactic derivation, the semantic representation for the sentence *He smokes*, for example, is of type (et), rather than type t. I abstract away from this bound/free pronoun issue in this thesis.

[26] But note that we can do (121c) by combining (121a) and (121b).

[27] I ignore the derivation of *his mother* from the lexical level. I treat definite descriptions as type e expressions of category NP, rather than as of type (et)t with a GQ category, but this is for presentational convenience only.

(122) Jack_1 *likes* his_1 mother. $(VP = NP\backslash S)$

a. $g_{C_o}(VP/NP) \circ NP^{NP} \vdash VP^{NP}/NP^{NP} \circ NP^{NP} \vdash VP^{NP}$

b. $g_{C_o}(\lambda x.\lambda y.like'(x)(y)) \circ \lambda u.Mof'(u)$
$\vdash [\lambda f.\lambda x.\lambda y.like'(f(x))(y)](\lambda u.Mof'(u))$
$\vdash \lambda x.\lambda y.like'(Mof'(x))(y)$

Because Jacobson's system is equipped with Type Lifting Combinator l (cf. Jacobson 1999: 131), she can always treat the other item which is merged with the category A^C as a functor, and thus, the percolation of the superscript category C (and the percolation of the corresponding argument slot in the semantic expression) can continue for as many steps as the syntactic derivation itself continues,[28] so that the super-script argument-slot can be bound by a (Q)NP long-distance, as in *Meg*$_1$ *told Bill that Nina liked him*$_1$.

In order to 'bind' percolated argument slots, Jacobson defines a 'binding' Combinator, Z_B.

(123) a. Syntax: $Z_B((NP\backslash B)/A) = (NP\backslash B)/A^{NP}$

b. Semantics: Given a function f of type $< a, < e, b >>$, $z_b(f)$ is a function of type $<< e, a >, < e, b >>$ where $z_b(f) = \lambda G.[\lambda x.[f(G(x))(x)]]$ where G is a variable of type $< e, a >$.

Jacobson (1999:132).

Note that the binding Combinator in (123) leads to the 'c-command' configuration in binding. That is, the derived functor category $(NP\backslash B)/A^{NP}$ leads to the structure in which the A^{NP} argument is c-commanded by the higher argument NP, and the super-script category in A^{NP} is 'bound' by this higher NP. Applying (123) to the verbal functor *like*, we can bind the pronoun by the subject (Q)NP, as in (124).

(124) Jack_1 *likes* his_1 *mother.*

a. $Z_B((NP\backslash S)/NP) \circ NP^{NP} \vdash \ (NP\backslash S)/NP^{NP} \circ NP^{NP} \vdash \ NP\backslash S$

b. $z_b(\lambda x.\lambda y.like'(x)(y)) \circ \lambda u.Mof'(u)$
$\vdash \ \lambda G.\lambda x.like'(G(x))(x) \circ \lambda u.Mof'(u)$
$\vdash_{LF} \ [\lambda G.\lambda x.like'(G(x))(x)](\lambda u.Mof'(u))$
$\Rightarrow_{\beta red.} \lambda x.like'(Mof'(x))(x)$

[28] Unless Jacobson adds an additional rule to constrain the percolation. Also, as Morrill (2000b) points out, she would have to incorporate anti-locality constraints on pronominal binding, but I ignore this extra complexity in my thesis.

Given the output in (124) for the string *likes his mother*, the next item to be merged, that is, the subject NP *Jack* can 'bind' the extra-argument slot.

Other than the apparent lack of anti-locality constraints on pronominal binding, Jacobson's analysis is empirically adequate, and I adopt the basic concept in this thesis. However, working in Type Logical Grammar, we have to either redefine the category A^C by using a new binary formula connective or instantiate Jacobson's percolation/binding mechanisms by way of modally controlled structural rules.

Jäger (2003) adopts the former strategy and defines a new binary connective '|' which corresponds to Jacobson's 'superscript connective.'[29] Addressing a comment in Morrill (2000b) to a previous version of his analysis with regard to the lack of rules for '|' in natural deduction presentation, Jäger (2003) provides both an introduction rule and an elimination rule for the connective '|' in ND presentation. However, the exact interpretation of the connective in the PF algebra is still not clear. Also, having one connective do almost the whole job of forming the long distance dependency relation spoils the deductive nature of TLG to a certain degree.

In contrast, Morrill (2003) explains reflexive/pronominal binding in terms of his independently motivated wrapping and infix connectives '$\uparrow$' and '$\downarrow$' in the categorial entries of bound pronouns. Morrill's analysis has several merits. For example, as I briefly mention in section 3.4.3, if we treat reflexives as functors which take predicates of arity n as arguments and give back predicates of arity $n-1$ as outputs, then we can naturally explain the strict locality constraint on reflexive binding. Also, Morrill's analysis is more insightful than Jäger's in that the connectives and the algorithms have more general motivations. However, use of wrapping connectives which merge PF items in fundamentally different ways from the standard connectives '$\backslash, /, \bullet$' means that he needs to define at least two sorted PF algebra. Morrill, Fadda, and Valentin (2007) has done that with a generalized wrapping with functor expressions which have n PF holes inside, and this system is complete in that regard. However, it is not only that using sorted PF algebra makes the PF structure more complex, it is also not clear if the kind of PF discontinuity that motivates wrapping is general enough to justify an additional sort of algebra that matches up with wrapping. I argue that the linear adjacency associated with the basic connectives '$\backslash, /, \bullet$' does capture the majority of the linguistic data, and thus I deal with the kind of PF discontinuity phenomena that are analyzed in terms of wrapping connectives either by explicitly applying structural rules to show how the

[29]That is, NP^{NP} would be $NP|NP$ with this connective, though Jäger uses the category N, rather than NP, for DPs.

PF strings (which respect linear adjacency) are modified (e.g. structural permutation with association can achieve the effect of wrapping), or for some other cases such as double object constructions, simply by not using syntactic structures that require wrapping.

The different properties that the two sets of logical connectives (that is, '$\backslash, /, \bullet$' on the one hand, and '$\uparrow, \downarrow$,' on the other) introduce to the grammar system as a whole become even more obvious if we use NL as the base system, because use of wrapping connectives would spoil both the non-associativity and non-commutativity of NL when they are used. The problem with this alternative way of extending the grammar system is that the discontinuity effect in this system is achieved at a totally different level by using a different set of connectives '$\uparrow$' and '$\downarrow$,' and it is hard to see how the initial non-associativity/non-commutativity of '$/, \backslash, \bullet$' have been affected by this addition.[30] The methodological preference in this thesis is that we start with the set of logical connectives (i.e. '$\backslash, /, \bullet$') that merges distinct linear adjacent PF items with restrictive structure management properties (that is, non-associative and non-commutative intuitionistic linear logic), and when we have to introduce different structure management properties, we do so by way of explicitly and separately defined structural rules (which are controlled by modes of combination on the one hand, and unary 'modal' connectives '$\Diamond$' and '$\Box^{\downarrow}$' on the other, and so we do not totally lose the restrictiveness of the base system). Thus, we instantiate Jacobson's analysis by defining structural rules explicitly in Multi-Modal Type Logical Grammar.

3.4.2 Binding of pronouns in MMTLG

Binding of one pronoun

In this subsection, I reformulate Jacobson's binding analysis in Multi-Modal Type Logical Grammar. The analysis uses more or less the same set of structural rules that Vermaat (2006) used to deal with Wh-movement, though there are some differences with regard to how we control structural rules. Abstracted away from a particular grammar formalism that I am adopting here, the analysis is basically the

[30]Though from a different perspective, keeping the interpretation of '$/, \backslash, \bullet$' totally untouched may count as a merit of Morrill's analysis, as Morrill (1994) indicates. Answering a related criticism to the introduction of structural rules which might modify the initial interpretation of '$/, \backslash, \bullet$,' we could pursue some rule presentations in which the rules of logical connectives are provided in such a way that they are independent of the structure management property of the grammar system. See Wansing (1998).

same as in Dowty (1992) which in turn is based on Hepple (1990).[31] The pronominal binding algorithm that I sketch in this section is applied straightforwardly when we derive the anaphoric dependency of indefinites in chapter 9, except for the well-known 'anti-locality' constraints on pronominal binding which do not apply to the 'binding' of the extra-argument slot of an indefinite. As I have indicated before, I ignore the anti-locality constraints on pronominal binding in the thesis. I speculate that it is some kind of blocking effect. That is, as we see shortly, reflexives require stricter locality in the binding by their antecedents. In other words, reflexives are specifically used only for 'clause internal' binding. Thus, though the formal system does not assign any restriction to the use of pronouns for the clause internal binding as well, at the level of language use, if there is an expression that fits more specifically to the given environment, then, the more general item, that is, the pronoun, is not used in that environment. See Reuland (2001) for the details in a different syntactic theory of such a system-external explanation of the anti-locality effect. Morrill (2003) also takes the observed anti-locality constraint on pronominal binding as something external to the type-logical binding mechanism. The implication of such an analysis is that my pronominal binding algorithm recognizes the binding of a pronoun by a local antecedent as well-formed. Its ill-formedness will be explained by some system external constraint.

(125) Jack_1 said that he_1 likes Meg.

Use of a NP, rather than a QNP, as the binder/antecedent in (125) is for presentation reasons. It works basically in the same way with a QNP subject of category $S/(NP\backslash S)$ (though in my analysis, I apply argument slot raising to the verbal functor *say*, which makes the proof harder to parse).[32] Now I provide some structural association and mixed association/permutation rules in (126).

(126) Gentzen sequent.

a. Standard association rule (generally available for A-bar extraction).

$$\frac{X \circ (Y \circ \Diamond A) \vdash B}{(X \circ Y) \circ \Diamond A \vdash B}\ AR$$

[31] They also use basically the same mechanism to explain relative pronouns. Thus, my analysis here is similar to theirs in that I am assimilating pronominal binding to Wh-phenomena in terms of how to form long distance dependency.

[32] Morrill (2000b), following Lambek (1958), assigns $S/(NP\backslash S)$ to the subject pronoun *he* so that it cannot be merged in an object position. I could adopt such an analysis, but here I ignore such complication for presentation reasons.

b. Mix of permutation and association (limited to the (bound)-pronoun phenomena by the specific mode requirement).

$$\frac{(X \circ \Diamond_p A) \circ Y \vdash B}{(X \circ Y) \circ \Diamond_p A \vdash B}\ PR$$

$A, B \in F$ (i.e. the set of formulas), and $X, Y \in S$ (i.e. the set of structures).

(126a) is as in Moortgat (1997) or in Vermaat (2006). For (126b), which combines permutation with association, I define it separately from an analogous rule that I use in Wh-movement (see chapter 8). The permutation rule for A-bar movement is not sensitive to any mode indices, whereas (126b) refers to the unary mode p. This is meant to capture the general observation that pronominal binding is less structurally constrained than A-bar movement, though from the viewpoint of the generality of rules, we should ideally reduce the two rules into one, given that their behaviors are basically the same.

The category for pronouns, as shown in (127), requires the use of the hypothetical category $\Diamond_p \Box^{\downarrow} NP$ in the derivation, which licenses the application of the structural rules as in (126a) and (126b). The logical expression for pronouns is the same as in Jacobson's analysis.

(127) he

a. Category: $NP/\Diamond_p \Box^{\downarrow} NP$

b. Logical expression: $\lambda x.x$

Cf. Jacobson (1999)

First, I show how the pronoun entries in (127) together with the structural rules in (126) allow us to 'percolate' the extra argument slot that corresponds to the hypothetical category $\Diamond_p \Box^{\downarrow} NP$.

(128) Syntax

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{NP \circ ((NP\backslash S)/S \circ (S/S \circ ((NP/\Diamond_p \Box^{\downarrow} NP \circ \Diamond_p \Box^{\downarrow} NP) \circ ((NP\backslash S)/NP \circ NP)))) \vdash S}{NP \circ ((NP\backslash S)/S \circ (S/S \circ ((NP/\Diamond_p \Box^{\downarrow} NP \circ ((NP\backslash S)/NP \circ NP)) \circ \Diamond_p \Box^{\downarrow} NP))) \vdash S}\ PR}{NP \circ ((NP\backslash S)/S \circ ((S/S \circ (NP/\Diamond_p \Box^{\downarrow} NP \circ ((NP\backslash S)/NP \circ NP))) \circ \Diamond_p \Box^{\downarrow} NP)) \vdash S}\ AR}{NP \circ (((NP\backslash S)/S \circ (S/S \circ (NP/\Diamond_p \Box^{\downarrow} NP \circ ((NP\backslash S)/NP \circ NP)))) \circ \Diamond_p \Box^{\downarrow} NP) \vdash S}\ AR}{((NP\backslash S)/S \circ (S/S \circ (NP/\Diamond_p \Box^{\downarrow} NP \circ (NP\backslash S)/NP \circ NP)))) \circ \Diamond_p \Box^{\downarrow} NP \vdash NP\backslash S}\ \backslash R}{(NP\backslash S)/S \circ (S/S \circ (NP/\Diamond_p \Box^{\downarrow} NP \circ ((NP\backslash S)/NP \circ NP)))) \vdash (NP\backslash S)/\Diamond_p \Box^{\downarrow} NP}\ /R}{said \circ (that \circ (he \circ (likes \circ Meg))) \vdash (NP\backslash S)/\Diamond_p \Box^{\downarrow} NP}\ PF$$

I show only the result of the LF derivation for the derivation in (128).

(129) Semantics

$$\frac{(NP\backslash S)/S \circ (S/S \circ (NP/\Diamond_p\Box^{\downarrow}NP \circ ((NP\backslash S)/NP \circ NP))) \vdash (NP\backslash S)/\Diamond_p\Box^{\downarrow}NP}{say' \circ (\lambda p_t.p \circ (\lambda x_e.x \circ (like' \circ meg'))) \vdash \lambda x.\lambda y.say'(like'(meg')(x))(y)}\ LF$$

(128)~(129) pair the string *said that he likes Meg* with the logical expression $\lambda x.\lambda y.say'(like'(meg')(x))(y)$ of type (e(et)) by way of the derived category $(NP\backslash S)/\Diamond_p\Box^{\downarrow}NP$. We can now use this derived category in further steps of the derivation. However, just as Jacobson needs the 'binding' Combinator Z which allows her to bind two argument slots by one lambda operator, we need some way of binding the two argument slots that are bound separately by λx and λy in (129) by one lambda operator. One way of achieving this might be to postulate a PF null item which corresponds to Jacobson's Z, which takes $(NP\backslash S)/\Diamond_p\Box^{\downarrow}NP$ as the right argument. The result of this function application would then be merged with the binder at the next step of the derivation, that is, with the main clause subject *Jack* in (125).

(130) (Z)

a. Category $Z : (NP\backslash S)/((NP\backslash S)/\Diamond_p\Box^{\downarrow}NP)$

b. Semantics: $Z := \lambda R_{e(et)}.\lambda z_e.R(z)(z)$

However, remember that we have decided not to have PF null items do the job of Combinators which are postulated to explain the syntactic algorithms. Thus, we define Jacobson's Z as a modally controlled structural contraction rule instead, with appropriate semantics corresponding to (130b).

(131) Contraction with regard to p (i.e. pronouns).

a. Syntax:

$$\frac{A \circ (X \circ \Diamond_p A) \vdash C}{A \circ X \vdash C}\ Z$$

b. Semantics:

$$\frac{x \circ (R \circ y) \vdash \phi}{x \circ R \vdash \phi[y :\rightarrow x]}\ Z$$

In (131), $\phi[y :\rightarrow x]$ means that x replaces the occurrence of y inside the expression ϕ. The context sensitivity of the contraction rule in (131b) (or even, AR and PR) is worrying in terms of the expressive power of the grammar system, but the rule application is controlled by the unary operators and the modality specification. Thus, it at least does not collapse the whole grammar system to an associative system or a commutative system.

The structural contraction rule in (131) might be too restrictive for the binding of pronouns, as the data in (132) suggest.[33]

(132) a. The mother of every boy_1 came to meet him_1.

b. Every boy_1's mother came to see him_1.

The data in (132) suggest that the bound pronoun reading is available both for (132a) and (132b). However, (131) (together with AR and PR) does not allow us to bind the pronoun *him* with the universal QNP for either sentence. The item corresponding to the 'binder' category A to the left in (131a) is not the universal quantifier in either sentence in (132). It is *the mother of every boy* in (132a) and *every boy's mother* in (132b). One way of accommodating such data would be to change the category A for the 'binder' into $\Diamond\Box^{\downarrow}A$, with the semantics staying the same, and with specific additional structural association/permutation rules.

(133) Contraction with regard to p (i.e. pronouns).

a. Syntax:

$$\frac{\Diamond_q A \circ (X \circ \Diamond_p A) \vdash C}{\Diamond_q A \circ X \vdash C}\ Z$$

b. Semantics:

$$\frac{x \circ (R \circ y) \vdash \phi}{x \circ R \vdash \phi[y :\rightarrow x]}\ Z$$

We could then restrict the additional structural rules that we would define in terms of the index q, which would be associated with quantifiers.[34] However, using $\Diamond_q$

[33]In GB/Minimalist terms, such data suggest that bound pronouns do not have to be c-commanded by their binders in the standard definition of c-command. See Brody and Szabolcsi (2003) for further data and their analysis which percolates quantifier's features up to the DP (= NP in our TLG notation) level.

[34]Though quantifiers in this case should be interpreted in a broader sense, considering that we can replace *every boy* with *Tom* and can still derive a bound pronoun reading, as opposed to co-reference reading, as some linguistic tests can show.

that would be encoded with QNPs to percolate the quantificational feature (by way of additional structural rule) would make my analysis quite close to Moortgat's analysis which uses scope constructors. Thus, it does not go well with the basic assumption that QNPs themselves do not trigger any use of structural rules.

On the other hand, the percolation of (quantificational) operator feature seems to be related to the availability of a functional reading associated with genitive/possessive constructions in a broad sense (such as *A's B* or *B of A*).[35] Thus, we might define adequate structural rules by using the relevant genitive constructions, rather than quantifiers, as a trigger. However, I leave the treatment of the sentences as in (132) for future research.

Getting back to our derivation for (125), when we apply Z in (131) to (128), we need to go back to the fourth line from the bottom in (128).

(134) a. Syntax (with PF):

$$\dfrac{\dfrac{NP \circ (((NP\backslash S)/S \circ (S/S \circ (NP/\Diamond_p\Box^{\downarrow}NP \circ ((NP\backslash S)/NP \circ NP)))) \circ \Diamond_p\Box^{\downarrow}NP) \vdash S}{NP \circ ((NP\backslash S)/S \circ (S/S \circ (NP/\Diamond_p\Box^{\downarrow}NP \circ ((NP\backslash S)/NP \circ NP)))) \vdash S}\, Z}{Jack \circ (said \circ ((that \circ (he \circ (likes \circ Meg))) \vdash S}\, PF$$

b. Semantics:

$$\dfrac{\dfrac{\dfrac{NP \circ (((NP\backslash S)/S \circ (S/S \circ (NP/\Diamond_p\Box^{\downarrow}NP \circ ((NP\backslash S)/NP \circ NP)))) \circ \Diamond_p\Box^{\downarrow}NP) \vdash S}{jack' \circ ((say' \circ (\lambda p_t.p \circ (\lambda x_e.x \circ (like' \circ meg')))) \circ y) \vdash say'(like'(meg')(y))(jack')}\, LF}{jack' \circ (say' \circ (\lambda p_t.p \circ (\lambda x_e.x \circ (like' \circ meg')))) \vdash say'(like'(meg')(y))(jack')[y :\rightarrow jack']}\, Z}{jack' \circ (say' \circ (\lambda p_t.p \circ (\lambda x_e.x \circ (like' \circ meg')))) \vdash say'(like'(meg')(jack'))(jack')}$$

Thus, we have compositionally derived the desired logical expression.

More complex cases

Next, we deal with a sentence in which two pronouns are bound by the same antecedent, and also a sentence in which two pronouns are bound by different antecedents, where one of the antecedents does not seem to be in a position that can bind the pronoun in question at first sight.

(135) a. Jack_1 said that he_1 liked his_1 room.

b. Jack_1 told Meg_2 that he_1 likes her_2.

The binding of the pronoun *her* by *Meg* in (135b) requires us to use a particular syntactic structure, as we see later. The lexical entry for the possessive pronoun *his* is as in (136).

[35] See Winter (2004) for various kinds of functional readings, and his analysis using choice functions.

(136) his:

a. $(NP/\Diamond_p\Box^{\downarrow}NP)/N$

b. his: $\lambda P_{et}.\lambda y_e.Gen'(P)(y) = \lambda P.\lambda y.\iota(\lambda x.P(x) \wedge poss'_{e(et)}(x)(y))$
where Gen' is of type (et)(et), and ι is of type (et)e.[36]

The detailed semantics is irrelevant here, but the informal interpretation of Gen' is that given a set of individuals P, and given an individual y, it specifies a unique member of P as the individual x for y. For example, given the set of rooms and given the individual Jack, Gen' chooses the unique room for Jack.

Now I show the derivation for (135). For presentation, I use the following abbreviations for categorial formulas.

(137) TV for $(NP\backslash S)/NP$, TVs for $(NP\backslash S)/S$, $\Diamond t$ for $\Diamond_p\Box^{\downarrow}NP$, and PRO for $NP/\Diamond_p\Box^{\downarrow}NP$.

The numbers in some of the rules, such as $AR1$ and $PR2$, indicate which (bound) pronoun licenses the application of which structural rule. Notice that AR and PR do not influence the logical expressions, in the sense that the logical expression in the succedent (i.e. the one to the right of the turnstile) stays the same before and after the rule application. Now, I show the derivation for (135a)

(138) a. Syntax

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{NP \circ (TVs \circ (S/S \circ ((PRO1 \circ \Diamond t1) \circ (TV \circ ((PRO2/N \circ N) \circ \Diamond t2))))) \vdash S}{NP \circ (TVs \circ (S/S \circ ((PRO1 \circ \Diamond t1) \circ ((TV \circ (PRO2/N \circ N)) \circ \Diamond t2)))) \vdash S}\,AR2}{NP \circ (TVs \circ (S/S \circ (((PRO1 \circ \Diamond t1) \circ (TV \circ (PRO2/N \circ N))) \circ \Diamond t2))) \vdash S}\,AR2}{NP \circ (TVs \circ (S/S \circ (((PRO1 \circ (TV \circ (PRO2/N \circ N))) \circ \Diamond t1) \circ \Diamond t2))) \vdash S}\,PR1}{NP \circ (((TVs \circ (S/S \circ (PRO1 \circ (TV \circ (PRO2/N \circ N))))) \circ \Diamond t1) \circ \Diamond t2) \vdash S}\,AR1,2}{NP \circ ((TVs \circ (S/S \circ (PRO1 \circ (TV \circ (PRO2/N \circ N))))) \circ \Diamond t1) \vdash S}\,Z}{NP \circ (TVs \circ (S/S \circ (PRO1 \circ (TV \circ (PRO2/N \circ N))))) \vdash S}\,Z}{Jack \circ (said \circ (that \circ (he \circ (likes \circ (his \circ room))))) \vdash S}\,PF$$

b. Semantics

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{jack' \circ (say' \circ (\lambda p_t.p \circ ((\lambda u_e.u \circ x) \circ (like' \circ ((Gen' \circ rm') \circ y))))) \vdash \phi_{yxj}}{jack' \circ (say' \circ (\lambda p.p \circ ((\lambda u.u \circ x) \circ ((like' \circ (Gen' \circ rm')) \circ y)))) \vdash \phi_{yxj}}\,AR2}{jack' \circ (say' \circ (\lambda p.p \circ (((\lambda u.u \circ x) \circ (like' \circ (Gen' \circ rm'))) \circ y))) \vdash \phi_{yxj}}\,AR2}{jack' \circ (say' \circ (\lambda p.p \circ (((\lambda u.u \circ (like' \circ (Gen' \circ rm'))) \circ x) \circ y))) \vdash \phi_{yxj}}\,PR1}{jack' \circ (((say' \circ (\lambda p.p \circ (\lambda u.u \circ (like' \circ (Gen' \circ rm'))))) \circ x) \circ y) \vdash \phi_{yxj}}\,AR1,2}{jack' \circ ((say' \circ (\lambda p.p \circ (\lambda u.u \circ (like' \circ (Gen' \circ rm'))))) \circ x) \vdash \phi_{yxj}[y :\rightarrow jack']}\,Z}{jack' \circ (say' \circ (\lambda p.p \circ (\lambda u.u \circ (like' \circ (Gen' \circ rm'))))) \vdash \phi_{jxj}[x :\rightarrow jack']}\,Z}{jack' \circ (say' \circ (\lambda p.p \circ (\lambda u.u \circ (like' \circ (Gen' \circ rm'))))) \vdash \phi_{jjj}}$$

[36]We could interpret ι as of type (et)((et)t) and make it compatible with Russellian quantificational analysis of definite descriptions (dds), but I simplify things by treating dds as type e expressions.

where

$\phi_{yxj} = (say'((like'((Gen'rm')y))x))jack' = say'(like'(Gen'(rm')(y))(x))(jack')$

$\phi_{jxj} = (say'((like'((Gen'rm')jack'))x))jack'$

$= say'(like'(Gen'(rm')(jack'))(x))(jack')$

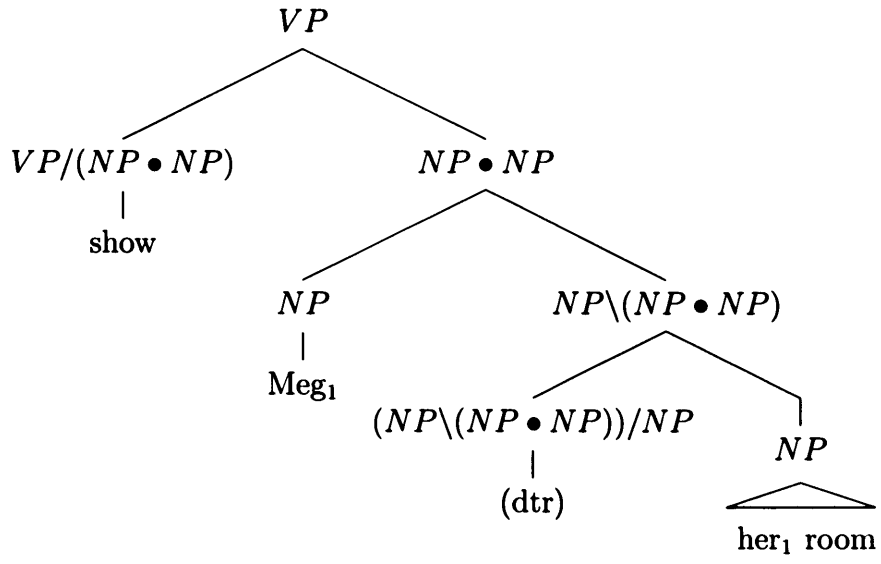
$\phi_{jjj} = (say'((like'((Gen'rm')jack'))jack'))jack'$

$= say'(like'(Gen'(rm')(jack'))(jack'))(jack')$

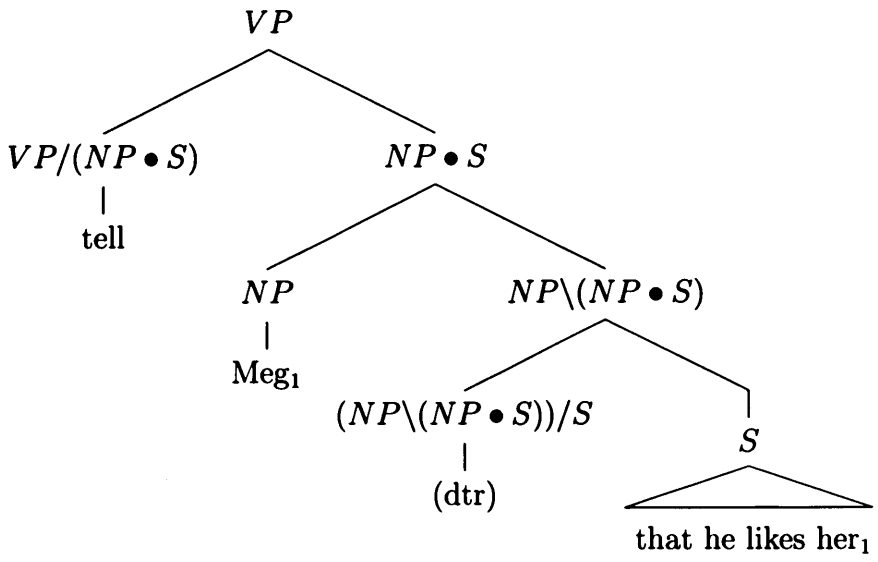
The 'binding' contraction rule Z in (131) is applied successively with regard to $\Diamond t2$ and then with regard to $\Diamond t1$ which have both been placed in the right positions via a few steps of AR and in the case of $\Diamond t1$, one application of PR. In this way, the two extra argument slots that have been originally introduced by the two pronouns are saturated by *jack'*.

To deal with (135b), I need to use the PF-null associate of the ditransitive verb that I introduce for the DO construction in chapter 4. This is the PF null item *dtr* which is associated with ditransitive verbs and which concatenates two internal NP arguments of a ditransitive verb into one complex argument, $(NP \bullet NP)$. I discuss the details of *dtr* in chapter 4 and chapter 5, but note that $\bullet$ may concatenate any two categories in NL. Thus, all we need to do is to assign an appropriate category to the functor, such as $VP/(NP \bullet S)$, so that the functor selects the concatenated complex argument $NP \bullet S$, as is shown in (140). I show the VP structure for the DO construction in (139) for comparison.

(139) DO: *show* Meg_1 her_1 *room*



(140) *tell Meg$_1$ that he likes her$_1$*



Note that because *AR* and *PR* in (126) may apply to the two hypothetical items that are introduced by two pronouns in (135b) in any orders, we do not need special mechanisms other than the ones we have already provided to deal with both the crossed binder-bindee relations as in (135b) and the nested binder-bindee relations as in *Jack$_1$ told Meg$_2$ that she$_2$ should ring him$_1$*. In contrast, in the case of Jacobson's binding algorithm, her superscript categories are stacked on top of one another according to the order of the merges of the two pronouns, such as $S^{NP_1 NP_2}$ for the embedded clause *(that) he likes her* in (135b), where $NP1$ represents the extra argument slot that is introduced by the first pronoun to be merged (i.e. *her* in (135b)) and $NP2$ is for the pronoun that is later merged (i.e. *he* in (135b)). Thus, she had to define a separate re-order operator to have the more deeply embedded superscript category to be 'bound' sooner than the one on top of it.

In the next section, I show how we deal with the reflexive binding.

3.4.3 Reflexive binding

Reflexive binding, as opposed to the pronominal binding, is subject to strict locality constraints. The strict locality constraints on reflexive binding could be neatly explained if we regarded a reflexive as a higher order functor which takes a (verbal) functor of arity n as its argument and then reduces the arity of this functor to $n-1$ by binding two argument slots of the selected functor by one lambda operator. For example, consider a derivation of *Meg sees herself* which uses the lexical entries for

see and *herself* in Szabolcsi (1992).[37]

(141) a. Syntax:

$$\dfrac{\dfrac{Meg}{NP1} \quad \dfrac{\dfrac{sees}{(NP1\backslash S)/NP2} \quad \dfrac{herself}{((NP1\backslash S)/NP2)\backslash(NP1\backslash S)}\backslash E}{NP1\backslash S}\backslash E}{S}$$

b. Semantics:

$$\dfrac{\dfrac{Meg}{meg'} \quad \dfrac{\dfrac{sees}{\lambda u.\lambda v.see'(u)(v)} \quad \dfrac{herself}{\lambda g.\lambda x.[g(x)(x)]}\backslash E}{\lambda x.see'(x)(x)}\backslash E}{see'(meg')(meg')}$$

cf. Szabolcsi (1992: 254)

In (141), the reflexive *herself* acts as a higher-order functor which applies to the verbal functor *see*. After the function application, the two argument slots of the verb, $NP1$ and $NP2$, are identified as $NP1$.[38] In the semantics, the two argument slots of the verbal functor see', that is, the u-slot and the v-slot, are identified as x. Thus, both the argument slots will be saturated by the same item at the next stage of the derivation.[39] Thus, the next expression to be merged, that is, meg' for the subject *Meg*, saturates both the external and the internal argument slots of the verbal predicate expression, as in $see'(meg')(meg')$.[40] Note that in this algorithm, we have to merge the verbal functor and the reflexive first, which binds the two argument slots of the verbal functor by one lambda operator and thus allows the saturation of the two argument slots by the same item in the next step. If the 'antecedent' of the reflexive were merged with the verb first, in a sentence such as **Herself sees Meg*, then the first-argument slot of the verb would be saturated by the object NP, as in $\lambda v.see'(meg')(v)$, before we merge the reflexive functor, and we could no longer identify the remaining argument slot of this verbal functor (i.e.

[37]Szabolcsi's example on p. 254 of her article is *Everyone sees herself*, but I change the subject QNP to an NP for simplicity. She has not provided the semantic expression for *see*, but it is easy to deduce an adequate functor term from the category she assigns to the transitive verb.

[38]As I explained in chapter 1, numbering on NPs is notational convenience for distinguishing different occurrences of the same category.

[39]Note that the number of type e arguments of the functor see' is preserved. What the reflexive does is to bind the two argument slots of the verbal functor by the same lambda operator, λx, rather than by two, as in the lexical entry for the verb *see*.

[40]If the subject is a quantifier, as in Szabolcsi's example, the two argument slots are bound by this quantifier, as in $every'(girl')(\lambda x.love'(x)(x))$ for *Every girl$_i$ loves herself$_i$*.

the v slot) with the object argument slot (i.e. the u-slot) which has already been saturated.

Though Szabolcsi's treatment of reflexives makes some linguistic sense, it requires polymorphic category assignments to deal with reflexive binding in different syntactic configurations.

(142) a. Bob_1 talked to himself_1.

b. Bob_1 showed himself_1 a nice picture.

c. Meg showed Bob_1 himself_1 in the mirror.

d. Bob_1 showed Meg himself_1 in the mirror.

Dealing with verbal functors of different arities as in (142a) and (142b) might not pose a serious problem, given the presence of some other functors whose argument categories are underspecified to certain degrees (e.g. *and*, *nice*, etc.). But to deal with (142c) by treating the reflexive as the functor which takes the verb as its argument, we would have to merge the reflexive with the ditransitive verb first, before we merge the result with the left object. Because of my methodological choice of not using any logical connective which merges non-adjacent items (such as wrapping connectives) and explaining long distance dependency by explicitly defining structural rules separately from logical rules, this causes some formal problem.[41] Also, a reflexive in the same position as in (142c) can be bound by a binder which appears in the subject, as shown in (142d).

It is possible to use structural rules together with a category that is similar to Szabolcsi's but with some use of $\Diamond$ and $\Box^{\downarrow}$, but considering that reflexive binding is not a main target of this thesis, and also, for the purpose of this thesis, deriving the c-command condition in Type Logical Grammar is enough to argue for one kind of structures against another, I deal with reflexives by using more or less the same algorithm that I have used with bound pronouns, only with additional locality constraints defined in terms of the modes of merges. From a linguistic viewpoint, this is a compromise, because in the context of Wh-extraction, I argued that structural rules triggered by $\Diamond$ should not be constrained by the modes of binary merges. But I leave a possible reformulation of my analysis of reflexives along the lines of Szabolcsi for future occasion.

I assign the following category/logical expression to the reflexives.

[41] For an analysis of binding that uses wrapping connectives, see Morrill (2000b).

(143) *himself* (*herself*)

a. Syntax:
$NP/_{rp}\Diamond\Box^{\downarrow}NP$

b. Semantics: $I'_{(e,e)} = \lambda x_e.x$

The entries are the same as pronouns except for the different mode index, which trigger slightly different structural rules. That is, structural rules that are sensitive to merge mode specifications.

(144) a. RA (Association):

$$\frac{X \circ_j (Y \circ_{rp} \Diamond_{rp} A) \vdash B}{(X \circ_j Y) \circ_{rp} \Diamond_{rp} A \vdash B}\ RA$$

b. RP (Association/permutation mixed):

$$\frac{(X \circ_{rp} \Diamond_{rp} A) \circ_j Y \vdash B}{(X \circ_j Y) \circ_{rp} \Diamond_{rp} A \vdash B}\ RP$$

The association rule for the reflexive pronoun, as in (144a) is restricted by requiring the unary mode *rp* which is for reflexive pronouns, and the binary merge mode j which merges (verbal) functors with their arguments (which are typically NPs). In normal transitive verb constructions, such as $Meg \circ_j (sees \circ_j (herself \circ_{rp} \Diamond_{rp} A))$ for (141), the rule AR allows us to merge the verb and the reflexive pronoun directly, with exclusion of the hypothetical 'trace' category $\Diamond_{rp} A$, which would then be placed outside the brackets to the right, as in $Meg \circ_j ((sees \circ_j herself) \circ_{rp} \Diamond_{rp} A)$, before the binding rule Z in (131) is applied to this output configuration.

RP in (144b) requires the same pairs of merge-modes, but j here in application is for merging the object NP arguments with a functor that concatenates the two objects into one complex object category ($NP \bullet NP$). As we see in detail in chapter 3, I insert such a concatenating functor between the two objects both for the double object ditransitive construction (= DO) such as *Tom*$_1$ *showed himself*$_1$ *the city* and in the prepositional ditransitive construction (= PP) such as *Tom showed himself to Meg.* Informally, use of RP followed by RA in (144) allows us first to extract the hypothetical category $\Diamond_p A$ to the right of the right object, and then re-bracket the structure so that we can apply the binding rule Z in (131) to identify the

hypothetical $\Diamond A$ with the subject of the sentence, that is, *Tom*, as is informally shown in (145).

(145) a. DO: $Tom \circ_j (showed \circ_j ((himself \circ_{rp} \Diamond_{rp} A) \circ_j (dtr \circ_j thecity)))$
$\Rightarrow_{RP} Tom \circ_j (showed \circ_j ((himself \circ_j (dtr \circ_j thecity)) \circ_{rp} \Diamond_{rp} A))$
$\Rightarrow_{RA} Tom \circ_j ((showed \circ_j (himself \circ_j (dtr \circ_j thecity))) \circ_{rp} \Diamond_{rp} A)$

b. PP: $Tom \circ_j (showed \circ_j ((himself \circ_{rp} \Diamond_{rp} A) \circ_j (to \circ_j Meg)))$
$\Rightarrow_{RP} Tom \circ_j (showed \circ_j ((himself \circ_j (to \circ_j Meg)) \circ_{rp} \Diamond_{rp} A))$
$\Rightarrow_{RA} Tom \circ_j ((showed \circ_j (himself \circ_j (to \circ_j thecity))) \circ_{rp} \Diamond_{rp} A)$

The mode specification in the rules in (144) at the moment is generally too descriptive (so that we can cover the empirical data in an adequate way), and thus it is not clear what insight we can get with regard to what the grammar system can do and cannot do. Especially the structural rule in (144b) looks stipulative, given that it is tailor made for allowing the left object reflexive to be bound by the subject (Q)NP. However, the general linguistic idea is clear: a reflexive identifies itself with a co-argument of its local functor and the co-argument has to be merged with the functor later than the reflexive (where this 'local functor' can be a complex predicate given certain restrictions, as we indicate in the infinitival chapter 7).[42] Given the clarity of the linguistic idea, and given that a more complete analysis of reflexive binding is beyond the scope of my thesis,[43] I leave a possible modification of the mode specification (into a more general formulation) and its use to control structural rules for future research.

Now, we need some 'binding' structural rule.

(146) Z_{ref}

a. Syntax:

$$\frac{A \circ_j (X \circ_j \Diamond \Box^{\downarrow} A) \vdash B}{A \circ_j X \vdash B} \, Z_{ref}$$

where $A, B \in F$. $X \in S$.

[42]Morrill (p.c.) mentioned the sentence, *Meg talked to John$_1$ about himself$_1$*. Because each 'talking' event involves some topic of the talk, we might treat *himself* as an argument of the complex predicate *talk...about*. But this would require some use of structural permutation, as well as structural association. Also, note that *Meg talked to John$_1$ about HIM$_1$* with some phonological focus on the pronoun *him* is acceptable, and thus, it is not clear if *himself* in this sentence is a normal reflexive pronoun. I leave this sentence for future research.

[43]For example, there is a strong tendency such that the binder of a reflexive pronoun is the grammatical subject, and thus, the binding of the right object by the left object as in English is exceptional in that regard. How to accommodate such a tendency may influence the ultimate formalization of reflexive binding.

b. Semantics:

$$\frac{x \circ_j (X \circ_j y) \vdash \phi}{x \circ_j X \vdash \phi[y :\rightarrow x]} \, Z_{ref}$$

The structural rule Z_{ref} allows us to replace all the argument slots that the hypothetical item y has saturated by an argument x which is merged later. $\phi[x :\rightarrow y]$ in (146) means that the logical expression is the same as ϕ except that the (unique) free occurrence of y inside ϕ is replaced by x. Because the association rule in (144a) only accepts association relative to the merge mode j, the reflexive cannot identify itself with an argument that is merged after the final output of the local functor (normally a verb), as we see in chapter 4.

3.5 Conclusion

In this chapter, I have first introduced Gentzen Sequent presentation of Type Logical Grammar. I have shown in GS proofs that argument slot raising to a verbal functor with regard to an object argument slot is not provable in NL. But I have also shown that, once argument slot raising as a special rule has transformed the verbal functor category/logical expression, then use of the transformed item in a derivation is supported in NL. I have also shown that value raising is generally supported in NL, which is used in some later chapters.

In order to explain various A-movement and A-bar movement phenomena, I have introduced structural rules in a modally controlled manner. For that purpose, I have adopted Multi-Modal Type Logical Grammar as in Moortgat (1997), Bernardi (2002) and Vermaat (2006).

I have briefly shown how we can introduce structural rules in different ways for A and A-bar phenomena so that we can reflect the linguistic differences between the two phenomena in the formal algorithms. In chapters 7 and 8, I spell out these algorithms in a more detailed manner, when I explain how they interact or do not interact with the basic scope algorithm using argument slot raising.

Finally, I have provided the basic analyses of pronominal/reflexive binding which are used in chapter 4 and 8.

Chapter 4

VP structures in double object constructions

4.1 Introduction

Chapter 4 and chapter 5 deal with two kinds of ditransitive verb constructions. Chapter 5 discusses scope relations between the two QNPs occupying the two object positions in these constructions, as in *Tom gave a student every book*, which has only the surface scope reading (i.e. $a > every$), and *Tom gave a book to every student*, which has both the scope readings (i.e. $a > every$ and $every > a$). In chapter 5, I consider why the scope of the first sentence is 'frozen' as the surface scope, as Bruening (2001) has pointed out, whereas the second sentence with a preposition *to* shows scope ambiguity.

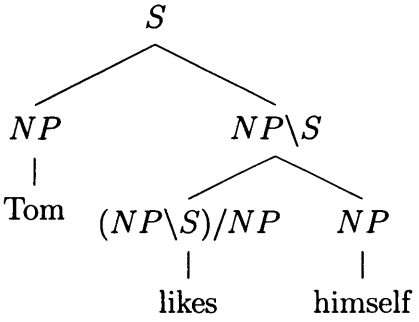
Before discussing QNP scope, however, I discuss binding in the two kinds of ditransitive constructions in this chapter. Reflexive/pronominal binding is sensitive to the relative structural positioning between the binder and the bindee. In Type Logical Grammar, the structural relation between the positions that two items occupy in the syntactic tree generally corresponds to the order of the merges of those two items in the corresponding Natural Deduction derivation. Thus, by investigating reflexive binding in the two kinds of ditransitive constructions, we can find out which order we should merge the (Q)NPs in question, which influences our QNP scope analysis in these constructions.

With that in mind, compare Tree DO and the Natural Deduction derivation in (147) for Tom_1 *likes himself*$_1$.[1] In (147), I assign 'NP' to *himself* for presenta-

[1]Though TLG does not use syntactic trees in an essential way, we can still informally explain the theory in terms of the trees corresponding to ND proofs. In (147), I assign the category 'NP' to

tional convenience, instead of '$NP/\Diamond_{rp}\Box^{\downarrow}NP$' which I use in my actual analysis of reflexive binding.

(147) Tree DO:



ND derivation:

$$\dfrac{\dfrac{Tom}{NP} \quad \dfrac{\dfrac{likes}{(NP\backslash S)/NP} \quad \dfrac{himself}{NP}}{NP\backslash S}\,/L}{S}\,\backslash L$$

The NP that is merged later (that is, *Tom*) in the ND derivation asymmetrically 'c-commands' the NP that is merged first (that is, *himself*) in the corresponding tree structure Tree DO above, according to the informal definition of 'c-command,' which states that a node A c-commands a node B if we can reach the node B by going one node up from the node A, and then going one or more node down in the other branch of the tree. The term 'c-command' is used for describing what our Type Logical theory aims to do in an intuitive way in tree representations. Though the notion of c-command does not have any official status in our grammar, remember from chapter 3 that the 'binding' structural rule Z roughly requires the c-command configuration as has just been sketched above between the binder and the bindee in the antecedent structure of the sequent to which the rule applies. In NL, antecedent structures are binary configurations of categorial formulas which can be represented as binary trees.

In Tree DO above, the object NP does not c-command the subject NP. Given the condition that the binder must c-command the reflexive in order to bind it, this informal analysis predicts the illegal binding in **Himself likes Tom*, which we will discuss again in (151) below. The analysis of reflexive binding in a TLG framework which I have shown in chapter 3 implies that only an NP that is merged sooner can be bound (or 'identified') by an NP that is merged later (in a certain structural configuration that the 'binding' structural rule Z requires, which I will use in this chapter).

himself for presentation reasons. The correct category in the proposed analysis is '$NP/\Diamond\Box^{\downarrow}NP$,' as we see later.

Thus, the binding asymmetry between NP argument positions tells us in which order we should merge the (Q)NP arguments with the verb. The order of the merges that we establish in this chapter (or the VP structure that is generated in the corresponding syntactic trees) plays a crucial role when we discuss the scope of two QNPs placed in the two object positions for the two kinds of ditransitive constructions.

In order to merge two objects with a ditransitive verb in the correct order, I will assign the category $(NP\backslash S)/(NP \bullet NP)$ to ditransitive verbs, with the semantics which I will explain in section 4.2. The proposed analysis preserves a uniform lexical entry for the ditransitive verb such as *give* and *show*, and a uniform thematic hierarchy among their arguments at LF. The use of the connective $\bullet$ in the ditransitive verb category delays a function application in the syntax, without influencing the normalized logical forms.

As Barss and Lasnik (1986) and Larson (1988) observed, binding relations in the double-object (DO) construction, as in (148), and in the PP ditransitive construction, as in (149), pose a problem for binding theory.[2]

(148) a. Meg showed Tom_1 himself_1 (in the mirror).

b. *Meg showed himself_1 Tom_1.

(149) a. Meg showed Tom_1 to himself_1 (in the mirror).

b. *?Meg showed himself_1 to Tom_1 (in the mirror).

If we merge the verb with the left object in the phonological string, and then merge the result with the right object, as the bracketing in (150) suggests, then the right object c-commands the left object in the syntactic structure, both in the double object construction and in the PP ditransitive construction.[3] But we want the left object to c-command the right object to syntactically explain the grammaticality judgments in (148) and (149).

(150) a. Meg [[showed Tom_1] himself_1]

b. Meg [[showed Tom_1] to himself_1]

[2]The sentences in (148a) and (149a), though grammatical according to our judgment, may sound pragmatically odd. In order to make these sentences sound more natural, assume that *Tom* is a chimpanzee whose reaction Mary is investigating by showing him his figure in the mirror.

[3]The right object in (150b) does not c-command the left object according to the informal definition of c-command above, because of the preposition *to*. However, considering that the right object counts as an argument of the ditransitive verb *show*, I assume here for the sake of argument that the PP *to Tom* as a whole may somehow count as a potential binder in the structure in (150b). In contrast, my analysis regards *to* as a functor that takes the two objects as its arguments, as we see in section 4.2.

The problem is the same in Type Logical Grammar, in which we normally merge adjacent lexical items successively by function application (that is, by $\backslash E$ and $/E$ in ND proofs). The functor *show* would be merged first with the object next to it in the string, and then with the more distant object to the right. As I have discussed above, an argument that is merged later may bind an argument that is merged earlier, but not vice versa, as the binding relation between the subject and the object suggests in (151).

(151) a. [Tom$_1$ [saw himself]]

b. *[Himself$_1$ [saw Tom$_1$]]

In this analysis, only the right object in the string should be able to bind the left object, contrary to the data in (148) and (149). To make matters more complex, in (148a), *Tom* is the goal argument and binds the reflexive as the theme argument, whereas in (149a), *Tom* is the theme argument and binds the reflexive as the goal argument.[4] This makes it difficult to define a uniform binding condition at LF, which is the level of representation at which we expect arguments with different thematic roles to enter into fixed hierarchical orders.[5]

Steedman (1996) solves the problem by modifying the hierarchical order between the two internal arguments of the functor *show* in the logical expression on the one hand, and by assigning two lexical entries to the functor on the other. In his Predicate-Argument Structure (= PA structure),[6] where logical expressions used as the semantic representations are fully normalized, the left object in the string, which is merged first with the verb in a syntactic derivation, c-commands the right object, which is merged later, both for the double object construction and for the PP ditransitive construction. By defining a binding condition in his PA structure, Steedman explains why the object that is merged first with the verb in the syntax can bind a pronoun in the object position that is merged later. He has simply modified the hierarchical order between the first two arguments of the logical expressions for *show* in the two constructions, so that the first argument to be merged with the verb c-commands the second argument in the PA structure for both constructions.

[4]See the end of section 4.2 for an alternative theta role assignment possibility to the left object in the DO construction, which I reject by arguing that it is an unnecessary multiplication of theta roles.

[5]This problem remains even if we use a wrapping operator (cf. Bach 1981) to merge the verb and the right object first. I also mention another problem for a wrapping analysis later.

[6]Steedman's PA structure roughly corresponds to my LF in that lambda expressions are fully normalized (see Steedman 1996: 88). There is some difference between the two, but both are the levels where we expect a fixed hierarchy to hold among the type e arguments of verbal functors.

Steedman's analysis is unsatisfactory in several ways. First, he postulates multiple lexical entries for the verb *show*. But this lexical ambiguity can be avoided. As we see in my analysis, we can assign a uniform entry to the verb. We can explain the seemingly different binding relations by assining a specific category/logical expression to the preposition *to* in the PP ditransitive construction.

Also, Steedman's analysis makes dubious the idea of a fixed structural hierarchy holding among different argument slots of a verb. Compare Steedman's polymorphic assignments to the verb *show* with a standard lexical entry for the same verb in (152c).[7]

(152) a. show_1: $< show;\ ((NP\backslash S)/NP)/NP;\ \lambda x.\lambda y.\lambda z.show'_1(y)(x)(z) >$

b. show_2: $< show;\ ((NP\backslash S)/PP)/NP;\ \lambda x.\lambda y.\lambda z.show'_2(y)(x)(z) >$

(Steedman, 1996: 21)

c. cf. show $< show;\ ((NP\backslash S)/NP)/NP;\ \lambda x.\lambda y.\lambda z.show'(x)(y)(z) >$

(152a) is his entry for the double object construction and (152b) is for the PP construction.[8] I allocate the lexical items $show_1$ and $show_2$ to represent the two entries. In both the entries, the functor's first argument ($\boldsymbol{x}$ for the left object NP in the string) c-commands the second argument ($\boldsymbol{y}$ for the right object) in the normalized form. However, the functor's third argument ($\boldsymbol{z}$ for the subject NP in the string) c-commands the first and the second arguments in the normalized form. Thus, he loses a uniform relation between the order of the function applications to the three arguments, and the hierarchy among them in the normalized lambda expressions. Missing this relation at the lexical level is risky, if we do assume a fixed hierarchy among the verbal arguments in the normalized logical forms. The standard category and logical expression for ditransitive verbs in (152c) are better than Steedman's in this respect, because it is always the case that in the normalized logical form, an argument that is merged with the verbal functor later c-commands an argument that is merged earlier.

Lastly, his analysis implies that we will not have a fixed thematic hierarchy between the theme argument and the goal argument in the PA structure (or LF in my analysis). In (152a), the goal argument $\boldsymbol{x}$ (for *Tom* in (148a)) c-commands the theme argument $\boldsymbol{y}$ (for *himself* in (148a)) in the PA structure. But in (152b), $\boldsymbol{x}$ is

[7] I do not show an entry for *show* which selects a PP as the right object, that is, $((NP\backslash S)/PP)/NP$.

[8] To preserve notational consistency, I have changed Steedman's *output-to-the-left* category notation, and use the categorial notations that I am using throughout this thesis. That is, for Steedman, the English intransitive verb category is represented as $S\backslash NP$, which I represent as $NP\backslash S$.

the theme argument (for *Tom* in (149a)), which c-commands the goal argument ***y*** (for *himself* in (149a)). As I briefly discuss at the end of section 4.2, Steedman could assign different theta roles to *Tom* as the left object in (148a) and *Tom* in the PP in (149a), which would help him maintain a uniform thematic hierarchy. However, a conceivable independent motivation for this otherwise stipulatory multiplication of theta roles is challenged in section 4.2, and I argue that for lack of a convincing independent motivation, it is a-priori preferable to assign the same theta role to *Tom* in both constructions. By assigning a novel entry to the prepositional functor *to*, my analysis maintains a fixed thematic hierarchy between the theme and the goal arguments in the normalized logical forms for both the constructions, while forming the binding relation somewhere other than LF.

I take an approach in which the reflexives in (148a) and in (149a) are bound in a syntactic derivation, without defining a binding condition in normalized logical forms at LF. The syntax merges the two object NPs first, before it merges the result with the verb *show*. In order to make this possible, I adopt Morrill's use of the connective '•' as in '$NP \bullet NP$' (Morrill 1994: 128), which informally means the two NPs are available in this (linear) order. The verb *show* is given the category: $(NP\backslash S)/(NP \bullet NP)$. Although this 'uncurried' category changes the order of the syntactic merges from the one that the 'curried' standard category in (152c) dictates, semantically speaking, the logical expression *show'* of type ((e×e),(e,t)) is equivalent to the standard expression *show'* of type (e,(e,(e,t))) (= (152c)), if we use the π_1 and π_2 operators in the way that we have seen in chapter 1. Because of this equivalence, Morrill assumes that the two categories $((NP\backslash S)/NP)/NP$ and $(NP\backslash S)/(NP \bullet NP)$ are inter-derivable (in *L*) for *show*.[9] However, the binding asymmetries in (148) and (149) mean that we have to uniformly use the latter category for these sentences. Morrill forces the use of this category in the double-object string *John shows Mary$_1$ herself$_1$* by giving a special category to a reflexive pronoun *herself*, using the 'wrapping' connective '↑' and another new connective '>' which allows us to merge two categories across another category.[10]

(153) a. *herself*; $((NP\backslash S)/(NP \bullet NP)) > ((NP\backslash S) \uparrow \mathbf{NP})$

b. (*show, herself*); $(NP\backslash S) \uparrow \mathbf{NP}$

c. ((*show, herself*) *W Mary*) = *show Mary herself*; S\NP

[9]These categories are interderivable in L, which is associative.

[10]Morrill has revised the entry in (153a) to a category that uses his wrapping connective '↑' and his infixation connective '↓' (cf. Morrill 2000b: 9). Among other things, this allows him to assign a uniform category to *herself*, whether the reflexive in the right object position is bound by the left object QNP or the subject QNP.

cf. Morrill (1994:128-129)

Informally, the functor category '$Y > X$' requires the argument category 'Y' across another category, producing the output category 'X' after the application. In (153), the functor *herself* selects its argument category '$(NP\backslash S)/(NP \bullet NP)$' across another NP *Mary*, forcing the use of this category for the verb *show*. This merge produces a phonological string with a gap inside: (*show*, *herself*) with the category '$(NP\backslash S)\uparrow$**NP**.' This string wraps itself around the argument *Mary*, as the wrapping connective W in (153c) dictates. As a result of the informal phonological computation rule given in (153c), we get the correct string and binding relation. I have omitted bracketing in PF strings because Morrill generally uses associative grammar L as the base system.

I will not review Morrill's analysis in this thesis,[11] but I have several reasons for not following it. First, the use of the wrapping connective '$\uparrow$' is known to make the categorial proof system incomplete relative to the intended interpretation of the syntactic system, unless we take special measures.[12] Fixing the problem within the wrapping analyses makes the PF algebra complex.[13] Thus, use of a wrapping operator requires a strong linguistic motivation with some means to control its use. A phrasal verb such as *give...up* or *throw...away* might provide some linguistic motivations. However, these verb-particle pairs would presumably be stored in the lexicon as such, suggesting that use of a wrapping operator can be limited to these special lexical pairs. In contrast, it is hard to assume that a ditransitive verb and its right object are stored as a pair in the lexicon.[14]

Secondly, because the reflexive *herself* as the right object can either be bound by the left object or the subject, Morrill (1994) needs to use two different categories for *herself*. In the case of a simple pronoun, we would need an even more complex category assignment, because a pronoun can be bound by any c-commanding operator without locality constraints (though pronominal binding is subject to

[11] Among other things, I should ideally compare my analysis with his in terms of passivization and extraction. Wrapping analyses generally work better than my analysis for the passive, but I show that my analysis can also deal with the passive of the ditransitive sentences in chapter 7.

[12] See Versmissen (1996: 22-23) for the details of this incompleteness.

[13] For example, see Morrill (1994: 101-106) for his solution to the problem. Morrill and Fadda (2005) generalize the idea of wrapping so that they can first derive a PF expression with n 'gaps' or 'holes' inside, which is then merged with n arguments in one go, where the n arguments can be inserted into the appropriate argument slots. He then defines a complicated n-way sorted algebra with regard to which his grammar system is sound and complete.

[14] In a normal wrapping analysis, the connective '$\uparrow$' would appear in the category for *show*. In Morrill (1994), the reflexive *himself* introduces the connective. This is a questionable move. Once we make use of the connective '$\bullet$,' we should be able to explain the sentences in question without the connective '$\uparrow$.'

'anti-locality constraints,' as we have briefly discussed in chapter 3). He also needs to use a polymorphic category for the NPs in bold in (153a) and (153b) to deal with the PP ditransitive construction in which the right object is in PP. Lastly, because the right object becomes the first argument of the logical functor expression for *show* both in the double object and the PP ditransitive constructions, whereas the left object becomes the second argument of this functor, we cannot easily preserve the same thematic hierarchy between the verb's theme and the goal arguments in the normalized logical forms in the two constructions, as we have discussed for Steedman's analysis above.

Unlike Morrill, I assume that '$(NP\backslash S)/$(NP•NP)' is the **uniform category** for the verb *show*. This forces us to merge the two objects first. In the double object (DO) construction, I insert a PF-null concatenating functor which I represent as *dtr* between the two objects in the PF string. This functor is merged with the right object first and then with the left object. In the PP construction, the preposition *to* as a syntactic functor is merged with the object NP to the right first and then with the NP to the left later. Only the NP argument to be merged later can bind a pronoun in the NP that is merged first, either with the PF null functor *dtr* or the prepositional functor *to*, explaining the binding asymmetry between the two object positions. In the PP construction, the preposition changes the order of the two NP arguments relative to the verbal functor *show*. This makes it possible to preserve a fixed thematic hierarchy between the theme argument and the goal argument in my LF interface representation.

Section 4.2 gives my analysis. Section 4.3 is the summary.

4.2 Proposal

4.2.1 VP structures

As we have seen in (150), successive merge of the left object and the right object with the ditransitive verb does not lead to the correct syntactic structure for the binding of the reflexives in (148) and (149). To solve this problem, we first merge the two objects, creating one complex object as a result, of category '$(NP \bullet NP)$.' This complex object is then merged with the ditransitive verb in one step. This means that in the syntax, there is only one complex NP argument for an English ditransitive verb. On the other hand, semantically speaking, we still want to preserve the verb's two internal argument slots at LF. We can achieve this syntax-semantic 'mismatch' by using the multiplicative category-forming connective '•' in NL, as

we have seen in chapter 1 and 3.

To analyse the binding in the double object construction in (148) and in the PP construction in (149), I assign the lexical entries as in (154).

(154) a. Meg: <*Meg*; NP; m′ {type e}>

b. Tom: <*Tom*; NP; t′ {type e}>

c. show: <*show*;(NP\ S)/(NP•NP); $show'_3$ >
where, $show'_3 \stackrel{def}{=} \lambda\alpha.\lambda z.[show'(\pi 1(\alpha))(\pi 2(\alpha))(z)]$ of type ((e×e), (et))

d. to: <*to*; ((NP2\(NP1•NP2))/NP1; $\lambda x.\lambda y.[x \bullet y]$ {type (e,(e,(e×e)))}>

The category and the semantics in (154c) are the same as in one of the two lexical entries that Morrill (1994) provides for ditransitive verbs (Morrill, 1994: 128). In order to preserve the deductive nature of the basic grammar system maximally, Morrill uses both the category $(NP\backslash S)/(NP \bullet NP)$ and the standard category $((NP\backslash S)/NP)/NP$ (again they are inter-derivable in L). But the latter category would lead to the wrong syntactic structure in (150a). As I have briefly explained in section 4.1, Morrill uses a special category as in (153a) for a reflexive such as *herself* to force the use of the category $(NP\backslash S)/(NP \bullet NP)$ when binding is involved. However, this involves the use of wrapping connectives in the syntax, which I would like to avoid, for reasons which I have briefly explained in section 4.1. Also, independently of the use of the wrapping connective in the grammar (and independently of the binding phenomena in the ditransitive constructions), it is not clear if the syntactic structure in (150) is well-motivated for ditransitive constructions in the first place.[15] Thus, I use the entry in (154c) as the uniform entry for *show* in both the ditransitive constructions. Because the base grammar system is non-associative, we cannot derive the category $((NP\backslash S)/NP)/NP$ from $(NP\backslash S)/(NP \bullet NP)$, as desired. A functor of category $(NP\backslash S)/(NP \bullet NP)$ expects its two objects to be merged into the category '$(NP \bullet NP)$' first. The verbal functor is then applied to this complex object by $/E$ (i.e. forward application).

In the double object construction, we could simply merge the two object NPs by using the logical rule $\bullet I$ (in Natural Deduction presentation, cf. chapter 1) in the grammar system NL (in Gentezen Sequent presentation, the corresponding rule is $\bullet R$). I repeat the ND rule in (157) for two NP objects.

[15]There are some opposing views in this regard, which I do not review. See Pesetsky (1995) which uses both the syntactic structures for different purposes, for example.

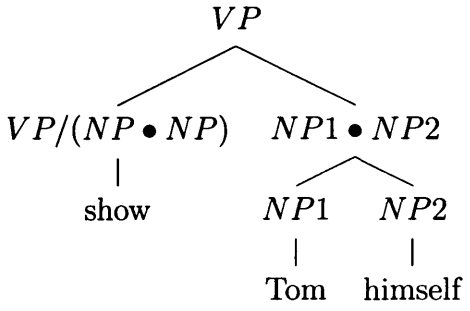
(155) $\bullet I$

a. Syntax: $(NP1, NP2) \vdash (NP1 \bullet NP2)$

b. Semantics: $(x, y) \vdash (x \bullet y)$

Using the logical rule $\bullet I$ to merge the two object NPs would be the most parsimonious derivation, but this would put the two object NPs into a symmetric relation to each other in the syntax, as is shown in the hypothetical VP structure in (156).

(156) An ill-motivated VP structure for the DO construction.



In (156), NP1 and NP2 mutually c-command each other and with this structure, we cannot explain the binding asymmetry in the DO construction in (148) in terms of c-command (or some mechanism in TLG which has the same effect). This analysis would require a rather stipulative binding mechanism which is sensitive to left-to-right linear order. To avoid this, we assume that there is a phonologically null functor *dtr* between the two object NPs. The functor *dtr*, which is mnemonic for 'ditransitive,' has the following lexical entry.[16]

(157) dtr

a. Syntax: dtr $\stackrel{\text{def}}{=}(NP1\backslash(NP1 \bullet NP2))/NP2$

b. Semantics: $dtr' \stackrel{\text{def}}{=}\lambda x.\lambda y.(y \bullet x)$

c. PF: ϵ (that is, it is phonologically null).

The syntactic structures that my analysis generates are roughly the same as Larson's VP shell structures, as in Larson (1988), though in different grammar systems. Pesetsky (1995) also uses the same structures for ditransitive constructions. In the

[16]At the moment, I do not force the use of dtr. Thus, we can still use '• I,' when we merge the two objects. Because the symmetrical structure as in (156) does not license either *Meg showed Tom himself* or **Meg showed himself Tom*, the availability of the structure in (156) as an option does not cause an immediate problem, but I leave the possibility to force the use of dtr as an open question.

DO construction, Pesetsky postulates a PF-null item G, which he regards as a PF-null prepositional head that introduces the theme argument, as in the structure, *Bill gave [$_{XP}$ Sue [$_{PP}$ G a book]]* (cf. Pesetsky 1995: 157). Unlike Pesetsky, I do not regard *dtr* as the head of PP. Also, I do not regard XP in Pesetsky's structure as a small clause. However, if we ignore these linguistic details and consider my analysis in terms of the binary tree structures that it generates, we could see my proposal here as an instantiation of Pesetsky's analysis using G in Type Logical Grammar.

On the other hand, though the use of the notation *dtr* for the PF-null concatenation functor might indicate that the functor is an item associated only with ditransitive verbs, such as *show*, we could use it more generally. That is, we might insert this PF-null functor between NPs whenever more than one NP argument appears on one side of the verbal functor in the phonological string. This extended use of *dtr* might be useful for explaining the fixed scope relation between NPs that appear on one-side of the verb in the PF string in some languages, such as Japanese. Japanese is a verb final language and thus all the NPs appear in front of the verb. When those NP arguments are canonically ordered, the scope relation between the NPs is apparently dictated by the PF linear order. That is, the linear left QNP apparently takes wide scope over the linear right QNP all the time.[17] Though I mention this observation briefly in the next chapter, this thesis mostly concentrates on English data, and I leave the exact identity of *dtr* in the proposed system as an open question.

During the syntactic derivation, the PF-null functor *dtr* is inserted between the two NP categories. The functor *dtr* is first applied to an NP category to the right (= the right object in the string) and then to the NP to the left (= the left object). The order of the two merges is important. In the derivational binding mechanism that I show below, only an NP that is merged later can bind a reflexive/pronominal argument that is merged earlier.

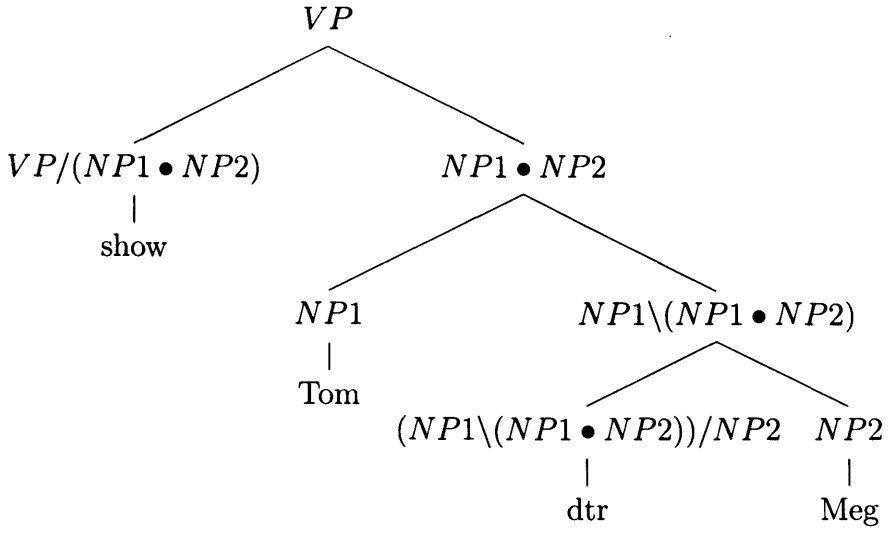
As we can see in its category $(NP1\backslash(NP1 \bullet NP2))/NP2$ in (157), the functor *dtr* puts the left object NP (which saturates the NP1 argument slot of the functor) as the left sub-category in the output category $(NP1 \bullet NP2)$. The functor places the right object NP (which saturates the NP2 argument slot) as the right component of the complex output category. Accordingly, NP1 corresponds to the semantic

[17]The difficulty is that I, like many, but not all native speakers of Japanese, find most of the sentences with canonical argument structures ambiguous in scope readings. Also, we would need to explain the effect of scrambling the QNP arguments somehow, because the so-called A-scrambling is claimed to produce scope ambiguity, even for those who do not get scope ambiguity in sentences with canonical word orders.

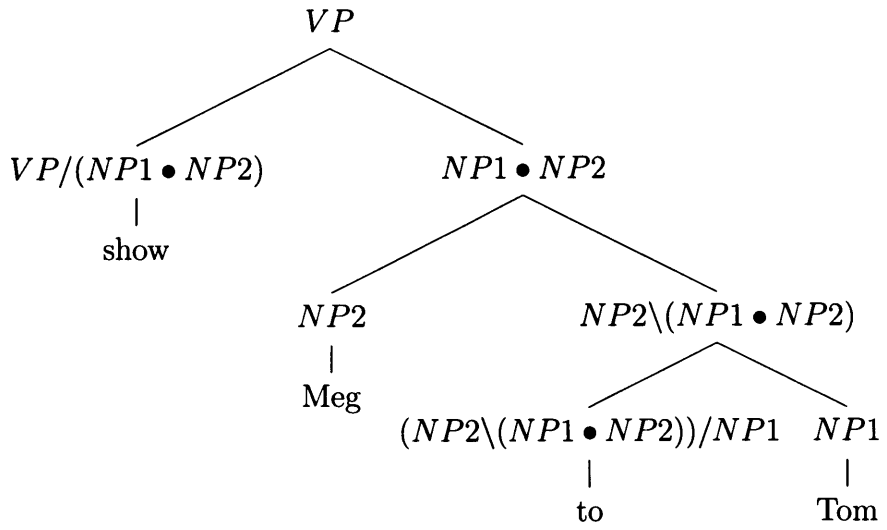
variable y, which is the left-member of the complex expression $(y \bullet x)$, and NP2 corresponds to the semantic variable x, which is the right-member of the complex expression $(y \bullet x)$.

In the PP construction, the prepositional functor *to* with the category $(NP2\backslash(NP1 \bullet NP2))/NP1$ as shown in (154d) places the left object and the right object in a different way from the way *dtr* does, relative to the left-to-right linear order between the two object NPs. For presentation reasons, I have used NP1 to mark the argument slot for the 'goal' argument for both the constructions, and NP2 for the argument slot for the 'theme' argument for both the constructions, instead of using the numbering according to the left-to-right linear order of the corresponding NP arguments. For the PP construction, in the output category: $(NP1 \bullet NP2)$, the prepositional functor puts the right object NP (which saturates the NP1 argument slot of the functor) as the left sub-component, and the functor places the linear left NP (which saturates the NP2 argument slot) as the right sub-component in the output. Accordingly, the logical term for the PF right object replaces the left subcomponent variable x in the complex logical expression $(x \bullet y)$ and the logical term for the PF left object NP replaces the right sub-component variable y in $(x \bullet y)$ in the output. Note that for both the constructions, the left sub-component of the output logical expression of type $(e \times e)$ is the goal argument and the right sub-component is the theme argument. This enables us to use a uniform entry for the verb *show*. The proposed analysis would generate the VP structure for the DO and the PP ditransitive constructions in binary tree representations as in (158)~(159):

(158) DO:



(159) PP:



Both the derivations generate the same normalized logical expression, $\lambda z.show'(tom')(meg')$ at LF, in contrast to the analysis in Steedman (1996), as we saw in (152), which would generate $\lambda z.show'_1(meg')(tom')(z)$ for the DO construction above and $\lambda z.show'_2(tom')(meg')(z)$ for the PP construction above, failing to get the uniform theta hierarchy between the theme argument and the goal argument.

In order to explain the binding asymmetry between the two object positions, both for the DO construction and the PP construction, I apply the reflexive binding mechanisms that I have explained in chapter 3.

4.2.2 Reflexive binding

I repeat the lexical entry for reflexive pronouns from chapter 3. From chapter 3, remember that there is a major problem with an analysis which treats reflexives as functors that take verbal functors as their arguments and identify two argument slots of the verbal functors. Such an analysis leads to polymorphic category assignments to reflexives in order to deal with reflexive binding in various structural configurations with regard to the binders. It also leads to the use of wrapping connectives or their equivalents. In contrast, I show that the uniform lexical entry in (160) can explain all the binding data in the two kinds of ditransitive constructions.

(160) *himself* (*herself, itself etc.*)

a. Syntax:

$NP/_{rp}\Diamond\Box^{\downarrow}NP$

b. Semantics: $I'_{(e,e)} = \lambda x_e.x$

I also repeat the structural rules from chapter 3. The presentation is Gentzen Sequent. As usual, for all $A, B \in F$ (= the set of formulas) and $X, Y \in S$ (= the set of structures):

(161) a. RA (Association):

$$\frac{X \circ_j (Y \circ_{rp} \Diamond_{rp} A) \vdash B}{(X \circ_j Y) \circ_{rp} \Diamond_{rp} A \vdash B}\ RA$$

b. RP (Association/permutation mixed):

$$\frac{(X \circ_{rp} \Diamond_{rp} A) \circ_j Y \vdash B}{(X \circ_j Y) \circ_{rp} \Diamond_{rp} A \vdash B}\ RP$$

Finally, I repeat the 'binding' structural rule Z_{ref}.

(162) Z_{ref}

a. Syntax:

$$\frac{A \circ_j (X \circ_j \Diamond_{rp} \Box^{\downarrow} A) \vdash B}{A \circ_j X \vdash B}\ Z_{ref}$$

b. Semantics:

$$\frac{x \circ_j (X \circ_j y) \vdash \phi}{x \circ_j X \vdash \phi[y :\rightarrow x]}\ Z_{ref}$$

According to this binding mechanism which is based on Jacobson's treatment of binding, a reflexive expression requires introduction of a hypothetical argument marked with '$\Diamond_{rp}$,' which can be identified only with an element that is merged later in the derivation (in a certain syntactic configuration). We will see shortly that this can explain the binding asymmetry between the two object positions, given the order of the merges of the NP arguments (or the resultant VP structures) which we have adopted above. I first show the derivation for *Meg showed Tom*$_1$ *himself*$_1$ in (148a). First, we merge the two objects in the DO construction, generating the PD string, $Tom \cdot (\epsilon \cdot himself)$, where ϵ for *dtr* is PF null. After that, we apply the structural rules, AR and Z_{ref}. The presentation is Gentzen Sequent. It is easy

to check that the sequent in the top line is provable in NL. After the top line, we apply structural rules in the order of going from top to bottom.

(163) a. Syntax:

$$\dfrac{\dfrac{\dfrac{NP \circ_j ((NP\backslash_j (NP \bullet NP))/_j NP \circ_j (NP/_{rp}\Diamond_{rp}\Box^{\downarrow}NP \circ_{rp} \Diamond_{rp}\Box^{\downarrow}NP)) \vdash NP \bullet NP}{NP \circ_j (((NP\backslash_j (NP \bullet NP))/_j NP \circ_j NP/_{rp}\Diamond_{rp}\Box^{\downarrow}NP) \circ_{rp} \Diamond_{rp}\Box^{\downarrow}NP) \vdash NP \bullet NP}\, Z_{ref}}{NP \circ_j ((NP\backslash_j (NP \bullet NP))/_j NP \circ_j NP/_{rp}\Diamond_{rp}\Box^{\downarrow}NP) \vdash NP \bullet NP}\, RA}{Tom \circ_j (\epsilon \circ_j himself) \vdash Tom \cdot (\epsilon \cdot himself)}\, PF$$

b. Semantics:

$$\dfrac{\dfrac{\dfrac{tom' \circ_j (dtr' \circ_j (I' \circ_{rp} y)) \vdash (dtr'(I'y))tom'}{tom' \circ_j ((dtr' \circ_j I') \circ_{rp} y) \vdash tom' \bullet y}\, RA}{tom' \circ_j (dtr' \circ_j I') \vdash tom' \bullet y[y :\rightarrow tom']}\, Z_{ref}}{tom' \circ_j (dtr' \circ_j I') \vdash tom' \bullet tom'}$$

Reduction:

$(dtr'(I'y))tom' \Rightarrow (dtr'((\lambda u.u)y))tom' \Rightarrow ((\lambda u \lambda v.v \bullet u)y)tom'$

$\Rightarrow (\lambda v.v \bullet y)tom' \Rightarrow tom' \bullet y$

If we change the order between the two object NPs, and place the reflexive in the left position, then the proposed algorithm cannot have the right object (Q)NP bind the reflexive, as is obvious in (164). I abbreviate $(NP\backslash(NP \bullet NP))/NP$ as Dtr.

(164) Syntax: **(Meg showed) himself Tom.*

$$\dfrac{(NP/\Diamond_{rp}\Box^{\downarrow}NP \circ_{rp} \Diamond_{rp}\Box^{\downarrow}NP) \circ_j (Dtr \circ_j NP) \vdash NP \bullet NP}{(NP/\Diamond_{rp}\Box^{\downarrow}NP \circ_j (Dtr \circ_j \mathbf{NP})) \circ_{rp} \Diamond_{rp}\Box^{\downarrow}NP \vdash NP \bullet NP}\, RP$$

We can move the hypothetical $\Diamond_{rp}\Box^{\downarrow}NP$ to the right peripheral position by way of PR, but this hypothetical argument cannot be merged with the NP in bold which has already been merged and is embedded in a lower position than $\Diamond_{rp}\Box^{\downarrow}NP$.

On the other hand, this hypothetical category introduced by the reflexive in the left object position can be identified with the (Q)NP argument that is merged later than this stage, within the verbal projection line. I merge the result of RP in (164) with the ditransitive verb category $(NP\backslash S)/(NP \bullet NP)$ and then with the subject NP in that order in the top line in (165a). After that, the usual structural rules lead to the binding of the reflexive in the left object position by the subject

NP. I attach numbers to NPs to show which NP argument slots are saturated by which NP arguments.

(165) *Bob$_1$ showed himself$_1$ Meg*

a. Syntax:

$$\dfrac{\dfrac{\dfrac{NP1 \circ_j ((NP_1\backslash S)/(NP_2 \bullet NP_3) \circ_j ((NP_2/\Diamond_{rp}\Box^{\downarrow}NP \circ_j (Dtr \circ_j NP_3)) \circ_{rp} \Diamond\Box^{\downarrow}NP)) \vdash S}{NP1 \circ_j (((NP_1\backslash S)/(NP_2 \bullet NP_3) \circ_j (NP_2/\Diamond_{rp}\Box^{\downarrow}NP \circ_j (Dtr \circ_j NP_3))) \circ_{rp} \Diamond\Box^{\downarrow}NP) \vdash S}\, RA}{NP1 \circ_j ((NP_1\backslash S)/(NP_2 \bullet NP_3) \circ_j (NP_2/\Diamond_{rp}\Box^{\downarrow}NP \circ_j (Dtr \circ_j NP_3))) \vdash S}\, Z_{rp}}{Bob \circ_j (show \circ_j (himself \circ_j (\epsilon \circ_j Meg))) \vdash S}\, PF$$

b. Semantics:

$$\dfrac{\dfrac{bob' \circ_j (show'_3 \circ_j ((I' \circ_j (dtr' \circ_j meg')) \circ_{rp} x)) \vdash (show'_3((dtr'meg')(Ix)))bob'}{bob' \circ_j ((show'_3 \circ_j (I' \circ_j (dtr' \circ_j meg'))) \circ_{rp} x) \vdash (show'_3(x \bullet meg'))bob'}\, RA}{bob' \circ_j (show'_3 \circ_j (I' \circ_j (dtr' \circ_j meg'))) \vdash (show'_3(x \bullet meg'))bob'[x :\rightarrow bob']}\, Z_{rp}$$

$show_3 = \lambda\alpha.\lambda z.show'(\pi_1(\alpha))(\pi_2(\alpha))(z)$, $dtr' = \lambda u.\lambda v.v \bullet u$, $I' = \lambda x.x$

c. Reductions (with omission of some intermediate steps):

1: $(dtr'meg')(Ix) \Rightarrow (dtr'meg')x \Rightarrow x \bullet meg'$

2: $(show'_3((dtr'meg')(Ix)))bob' \Rightarrow (show'_3(x \bullet meg'))bob'$

3: $(show'_3(x \bullet meg'))bob'[x :\rightarrow bob'] = (show'_3(bob' \bullet meg'))bob$

$= ((show'bob')meg')bob' = show'(bob')(meg')(bob')$

I omit the binding of a reflexive as the right object by the subject, because it is straightforward by way of RA and Z_{ref}.

Now, I show the binding of a reflexive in the right object position by the left object NP in the PP ditransitive construction. The basic mechanism is the same as in the DO construction. I use NP1 for the right object NP and NP2 for the left object NP and the corresponding argument slots of the prepositional functor.

(166) *(Meg showed) Tom to himself*

a. Syntax:

$$\dfrac{\dfrac{\dfrac{NP2 \circ_j ((NP_2\backslash(NP_1 \bullet NP_2))/NP_1 \circ_j (NP_1/\Diamond_{rp}\Box^{\downarrow}NP \circ_{rp} \Diamond\Box^{\downarrow}NP)) \vdash NP_1 \bullet NP_2}{NP2 \circ_j (((NP_2\backslash(NP_1 \bullet NP_2))/NP_1 \circ_j NP_1/\Diamond_{rp}\Box^{\downarrow}NP) \circ_{rp} \Diamond\Box^{\downarrow}NP) \vdash NP_1 \bullet NP_2}\, RA}{NP2 \circ_j ((NP_2\backslash(NP_1 \bullet NP_2))/NP_1 \circ_j NP_1/\Diamond_{rp}\Box^{\downarrow}NP) \vdash NP_1 \bullet NP_2}\, Z_{ref}}{Tom \circ_j (to \circ_j himself) \vdash tom \cdot (to \cdot himself)}\, PF$$

b. Semantics:

$$\dfrac{\dfrac{\dfrac{tom' \circ_j (to' \circ_j (I' \circ_{rp} y)) \vdash (to'(I'y))tom'}{tom' \circ_j ((to' \circ_j I') \circ_{rp} y) \vdash y \bullet tom'}\, RA}{tom' \circ_j (to' \circ_j I') \vdash y \bullet tom'[y :\rightarrow tom']}\, Z_{ref}}{tom' \circ_j (to' \circ_j I') \vdash tom' \bullet tom'}$$

Reduction:

$(to'(I'y))tom' = (to'((\lambda v.v)y))tom' \Rightarrow ((\lambda u.\lambda v.u \bullet v)y)tom'$

$\Rightarrow (\lambda v.y \bullet v)tom' \Rightarrow y \bullet tom'$

The only difference between (163) and (166) is that the PF null functor dtr' places the left object as the left sub-component and the right object as the right subcomponent in the output, whereas the functor to' does the opposite. On the other hand, for both the constructions, we merge the functors with their two NP arguments in the same order. That is, each functor is merged with the NP to the right first, and then with the NP to the left. Thus, only the NP merged later (i.e. the left object NP) can bind the NP merged sooner (i.e. the right object NP) for both the constructions. As (164) and (167) show, the binding of a reflexive in the left object position by the right object is impossible for either construction, according to the proposed algorithm. In (167), I abbreviate the category $(NP3\backslash(NP2 \bullet NP3))/NP2$ as To, where NP2 is used to indicate the argument slot for the PF right object NP and NP3 marks the argument slot for the PF left object. Just as in (164), the hypothetical argument $\Diamond_{rp}\Box^{\downarrow}NP$ cannot be identified with the right object NP (i.e. the **NP2** in bold face) that is embedded deeper than $\Diamond_{rp}\Box^{\downarrow}NP$ in the structure in the bottom line in (167).

(167) Syntax: **(Meg showed) himself to Tom.*

$$\dfrac{(NP3/\Diamond_{rp}\Box^{\downarrow}NP \circ_{rp} \Diamond_{rp}\Box^{\downarrow}NP) \circ_j (To \circ_j NP2) \vdash NP2 \bullet NP3}{(NP3/\Diamond_{rp}\Box^{\downarrow}NP \circ_j (To \circ_j \mathbf{NP2})) \circ_{rp} \Diamond_{rp}\Box^{\downarrow}NP \vdash NP2 \bullet NP3}\, RP$$

(168) *Bob*$_1$ *showed himself*$_1$ *to Meg*

a. Syntax:

$$\dfrac{\dfrac{\dfrac{NP1 \circ_j ((NP_1\backslash S)/(NP_2 \bullet NP_3) \circ_j ((NP_3/\Diamond_{rp}\Box^{\downarrow}NP \circ_j (To \circ_j NP_2)) \circ_{rp} \Diamond\Box^{\downarrow}NP)) \vdash S}{NP1 \circ_j (((NP_1\backslash S)/(NP_2 \bullet NP_3) \circ_j (NP_3/\Diamond_{rp}\Box^{\downarrow}NP \circ_j (To \circ_j NP_2))) \circ_{rp} \Diamond\Box^{\downarrow}NP) \vdash S}\, RA}{NP1 \circ_j ((NP_1\backslash S)/(NP_2 \bullet NP_3) \circ_j (NP_3/\Diamond_{rp}\Box^{\downarrow}NP \circ_j (To \circ_j NP_2))) \vdash S}\, Z_{rp}}{Bob \circ_j (show \circ_j (himself \circ_j (to \circ_j Meg))) \vdash S}\, PF$$

b. Semantics:

$$\dfrac{\dfrac{bob' \circ_j (show'_3 \circ_j ((I' \circ_j (to' \circ_j meg')) \circ_{rp} x)) \vdash (show'_3((to'meg')(Ix)))bob'}{bob' \circ_j ((show'_3 \circ_j (I' \circ_j (to' \circ_j meg'))) \circ_{rp} x) \vdash (show'_3(meg' \bullet x))bob'}\, RA}{bob' \circ_j (show'_3 \circ_j (I' \circ_j (to' \circ_j meg'))) \circ_{rp} \vdash (show'_3(meg' \bullet x))bob'[x :\rightarrow bob']}\, Z_{rp}$$

$show_3 = \lambda\alpha.\lambda z.show'(\pi_1(\alpha))(\pi_2(\alpha))(z)$, $dtr' = \lambda u.\lambda v.v \bullet u$, $I' = \lambda x.x$

c. Reductions (with omission of some intermediate steps):

1: $(to'meg')(Ix) \Rightarrow (to'meg')x \Rightarrow meg \bullet x'$

2: $(show'_3((to'meg')(Ix)))bob' \Rightarrow (show'_3(meg \bullet x'))bob'$

3: $(show'_3(meg' \bullet x))bob'[x :\rightarrow bob'] = (show'_3(meg' \bullet bob'))bob$

$= ((show'meg')bob')bob' = show'(meg')(bob')(bob')$

As a merit of my analysis, we can preserve the uniform lexical entry to ditransitive verbs, while maintaining a uniform thematic hierarchy among their type e arguments at LF. For example, my binding analysis can generate the logical forms in (170a) and (170b) for the sentences in (169a) and (169b). I do not show the derivation, but *tom'* is for *Tom*, *meg'* is for *Meg* and $\lambda x.Sof'(x)$ (of type (e, e)) is for *his students*.

(169) a. DO: Meg showed Tom$_1$ his$_1$ students.

b. PP: Meg showed Tom$_1$ to his$_1$ students.

(170) Thematic hierarchy (TH) preserved in the normalized logical expressions.

a. DO: **show'(tom')(Sof'(tom'))(meg')**

b. PP: **show'(Sof'(tom'))(tom')(meg')**

c. TH: **show'(Goal)(Theme)(Agent)**

Note that the thematic hierarchy among the three arguments of the verbal functor *show'* is uniformly as in (170c) in my analysis.

However, some might argue that the theta roles that the two objects receive are different between the two constructions, based on the alleged semantic differences between (171) and (172).

(171) a. Bob sent Meg the letter.

b. The letter reached Meg.

c. 'Recipient' for *Meg*

(172) a. Bob sent the letter to Meg.

b. The letter went to Meg.

c. 'Goal' for *Meg*

The claim is that in order for (171a) to be true, *Meg* must have received the letter. In contrast, (172a) will be true if Bob sent off the letter, whether Meg has received it or not. If this alleged difference of the interpretations between (171a) and (172a) were to be explained by way of different logical forms at LF, then the preservation of the uniform logical form after normalization in my analysis would not be explanatory in that regard. To explain the contrast between (171a) and (172a), an alternative analysis might assign 'Recipient' to *Meg* in (171a) as its theta role whereas assigning 'Goal' to *Meg* in (172a). In this way, they might maintain a uniform thematic hierarchy, say, 1 Recipient 2 Theme and 3 Goal, with Steedman's analysis which we have discussed in section 4.1.

However, I doubt that the alleged semantic difference above is real, and even if it were, it is not clear if the LF logical forms need to represent such subtle semantic difference. Consider (173a).

(173) a. ? Bob sent me a letter but I did not receive it.

b. Bob sent a letter to me but I did not receive it.

A majority of the native speakers that I have contacted have said that (173a) is acceptable, though some of them can see some contrast between (173a) and (173b). I argue that representing such subtle semantic difference in the LF logical representations (as opposed to some meaning representations after contextual/pragmatic factors have been incorporated) is not well-motivated. Given that we can naturally assign the Goal theta role to *me* for both the constructions in (173), it is a-priori preferable to avoid the unnecessary multiplication of thematic role assignments. To conclude, I argue that the uniformity of the normalized logical form is a merit of my analysis.

4.3 Summary

Explaining the binding asymmetries in the double object and the dative constructions, I use the syntactic category: $(NP\backslash S)/(NP \bullet NP)$ for *show*. In the preposition-less double object construction, a phonologically null binary functor *dtr* is inserted between the two object NPs. This functor concatenates the two object NPs into a complex object $NP1 \bullet NP2$, in which the linear order between the two NPs is preserved, such that $NP1$ is the left object and $NP2$ is the right object. The preposition *to* of category: $(NP2\backslash(NP2 \bullet NP1))/NP2$, in contrast, changes the order of the two object NPs before we merge them with the verb, enabling us to use $(NP\backslash S)/(NP \bullet NP)$ as the uniform category for *show*. On the other hand, this reordering in the case of the PP construction, as opposed to the DO contraction, does not influence reflexive binding, because reflexive binding is explained according to the order of the merges of the two NPs. That is, according to the analysis of reflexive binding that I adopt, the reflexive can identify two argument slots of the local functor with which it merges only if those two arguments have not yet been saturated when the reflexive is merged. In my analysis, both for the DO construction, and for the PP construction, the order of the merge with either the PF null functor *dtr* or the prepositional functor *to* is the right object NP first and the left object NP second. Thus, only a reflexive as the right object NP can be bound by a left object (Q)NP, which is merged later with the intervening functor. As for the syntax-semantics mapping, use of the connective $\bullet$ can delay a function application in the syntax, while preserving equivalence in the logical forms. The analysis suggests that not all the derivational history in syntax is represented in normalized logical forms at LF.

In the next chapter, I consider how the suggested analysis of the two kinds of ditransitive constructions can explain QNP scope in these constructions.

Chapter 5

Frozen scope in double object constructions

5.1 Introduction

Given the VP structures in the ditransitive constructions that are supported by the binding data in chapter 4, this chapter shows how my QNP scope analysis can explain the so-called frozen scope phenomena in the double object construction. Bruening (2001: 235) reports that the double object (DO) construction in (174a) has only the surface scope reading, $a > every$, whereas the prepositional (PP) construction in (174b) has both the surface and the inverse scope readings.

(174) a. The teacher showed a (# different) student every book.

$*every > a, a > every$

b. The teacher showed a (different) book to every student.

$a > every, every > a$

If we merge the verb *show* successively with its two objects in (174a), as the bracketing in (175a) suggests, then the universal QNP c-commands the indefinite and we should get the inverse scope reading. If we adopt an equivalent of May's QR and assume that the two quantifiers as higher order functors are applied to the verbal expression in two different orders, as (175b) and (175c) suggest, then we should get both the surface and the inverse scope reading with (174a). Neither is the case in Bruening's judgment.

(175) a. The teacher [[showed a student] every book]

b. [Some-student$_{((et)t)}$[λx.[Every-book$_{((et)t)}$[λy.[the teacher showed x y]]]]]

c. [Every-book$_{((et)t)}$[λy.[Some-student$_{((et)t)}$[λx.[the teacher showed x y]]]]]

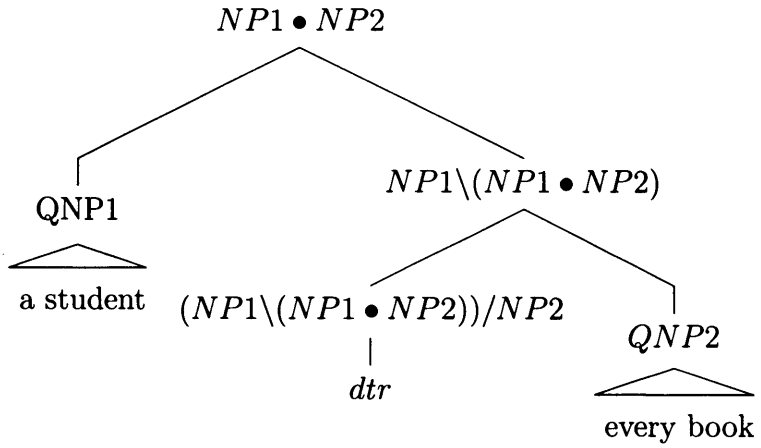
With this in mind, consider possible lexical assignments to ditransitive verbs.

(176) a. *show* : $(NP1\backslash S)/(NP2 \bullet NP3)$;
$\lambda\alpha.\lambda x.[show'(\pi 1(\alpha))(\pi 2(\alpha))(x)]$ type:$((e \times e), (e,t))$

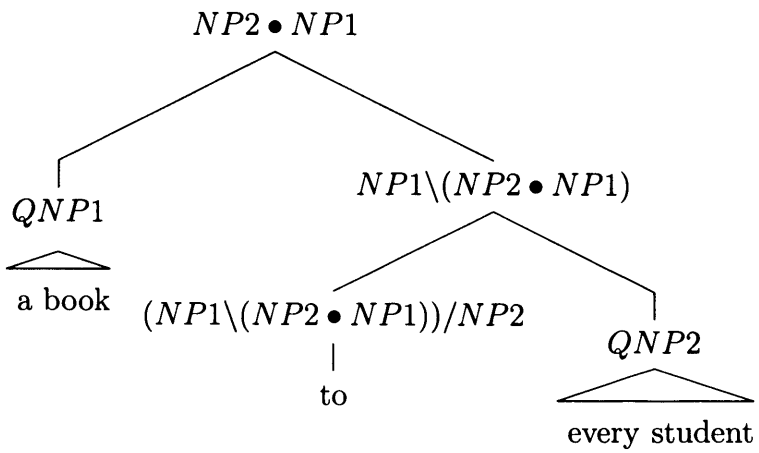
b. *show* : $((NP1\backslash S)/NP3)/NP2$;
$\lambda x.\lambda y.\lambda z.[show'(x)(y)(z)]$ type: $(e,(e,(e,t)))$

If we use the standard curried ditransitive category in (176b), we would generate the wrong structure as in (175a). In contrast, the un-curried ditransitive verb category *show* in (176a), together with the lexical entries for the PF-null functor *dtr* in the DO construction and for the prepositional functor *to* in the PP construction that I have used in chapter 4, generate more promising structural configurations, as shown in (177)~(178).

(177) DO: *(Meg showed) a student every book. a > every; *every > a*



(178) PP: *(Meg showed) a book to every student. a > every; every > a*



If we could merge the two QNPs with either the PF-null *dtr* or the prepositional functor *to* just as if the QNPs were two NP arguments of the functor, then the structures that we would generate for the two constructions are as above. For each structure, the left QNP is merged later than the right QNP, leading to the hierarchical structure that can readily explain the surface scope reading $a > every$.

We still need some algorithm to generate the inverse scope reading $every > a$ for the PP construction, without generating the inverse scope for the DO construction. In chapter 2, I reformulated Hendriks' argument slot raising in categorial calculus, so that we can merge QNPs as arguments of verbs, rather than as functors of the verbs. Thus, we might consider applying the argument slot raising to the functors *dtr* and *to* with regard to their NP argument slots. However, if we applied argument slot raising to these functors with regard to their two NP argument slots in two different ways, then we would generate the scope ambiguity for both the sentences in (174), contrary to Bruening's judgments.

To solve this problem, I apply argument slot raising to the PF-null functor *dtr* only in one way, so that it will generate only the surface scope reading. In contrast, argument slot raising is applied to the prepositional functor *to* in two different orders, so that we can generate the two scope readings. This is apparently a stipulation, but there is some linguistic justification.

First, I argue that in the basic cases, argument slot raising is applied only to functor categories which have 'S' as the final value (or the final output) category before any unary categorial transformation rules apply to the functor categories, such as value raising. I call a functor category that has S as its final value a **Boolean category**. Now, this means that normally, argument slot raising applies to the functor of the category $(NP\backslash_j S)/_j NP$, such as transitive verbs or complex predicates such as *try to review*, which the grammar can treat as of category $(NP\backslash_j S)/_j NP$, as we discuss in a more detailed manner in chapter 7. As we see later, ditransitive verbs with the category $(NP\backslash S)/(NP \bullet NP)$ have also S as the final output, and thus, we can freely apply argument slot raising to such functors. Though this additional constraint on the application of argument slot raising is a stipulation, it is based on some linguistic assumption. That is, though QNPs are syntactically arguments of the verbs, they are semantically propositional operators and thus, to maintain the syntax-semantic correspondence as much as we can, we apply the argument slot raising to the functor that generates propositional expressions (of category S) as the final value.

Now, if we think of the functors that concatenate the two object NPs into a complex object in the ditransitive constructions, neither the PF-null functor

dtr nor the prepositional functor *to* has the category 'S' as the final value. Thus, basically, we might say that argument slot raising is not readily applicable to either of them. However, if we do not apply argument slot raising, then the derivation does not converge. We have to somehow consume the QNPs with GQ categories as arguments of these functors. Thus, in the case of the PF-null functor *dtr*, we apply argument slot raising to this functor as the last resort measure for helping *dtr* to concatenate the two QNP arguments into a complex QNP object, which can then be consumed by the ditransitive verb as a complex QNP argument, as we see shortly in application. However, because the only job that this PF-null functor *dtr* is supposed to do is to concatenate the two objects into one complex object while preserving the initial configuration between the two objects, when we apply argument slot raising to this functor with regard to its two NP argument slots, we only do so in a way that preserves the surface scope order between the two QNPs. In other words, *dtr* only concatenates the two object QNPs in the way that the left QNP takes scope over the right QNP, as is represented in the PF linear order. Remember that dealing with two object NPs, *dtr* simply places the left object as the left sub-component of the output category, and puts the right object as the right sub-component of the output category. I propose that application of argument slot raising is to respect this weak expressive power of the PF-null concatenating functor *dtr*.

Now, what about the prepositional functor *to*? In the PP construction, the prepositional functor *to* reverses the order between the two NP arguments in the input and in the output, as we have seen in chapter 4. That is, the left object NP in the PF string appears as the right NP sub-component and the right object NP in the PF string appears as the left NP sub-component in the output category $(NP2 \bullet NP1)$ of the functor category $(NP1\backslash(NP2 \bullet NP1))/NP2$. This extra expressive power of the functor *to* (compared to *dtr* which simply preserves the PF linear order) may provide justification for applying argument slot raising to *to* in two orders. Note also that the preposition *to* is sometimes translated as a two place predicate in Predicate Calculus, as in the propositional formula, $to'(london')(tom')$, which might be considered as truth evaluable on its own. Though the propositional status of the form $to'(a')(b')$ is partly because of the limited expressive power of Predicate Calculus, it is not impossible to see the propositional functor *to* as a certain kind of Boolean functor (that is a functor that produces a type t expression in the end) in the semantics, and this provides another justification for applying argument slot raising to this functor, just as we apply argument slot raising to verbal functors.

As I indicated above, in order to allow the two object QNPs together to take wide scope over the subject QNP for both DO and PP constructions, I apply argument slot raising to ditransitive verbs of category $(NP\backslash S)/(NP \bullet NP)$. This application is straightforward, because ditransitive verbs lexically have Boolean categories.

(179) a. A student showed every visitor to University College London.

$a > every$; $every > a$

b. A student showed UCL to every visitor.

$a > every$; $every > a$

Section 5.2 shows the application of argument slot raising to *dtr* and *to*. As I have indicated above, argument slot raising is applied with its full potential only to the prepositional functor *to*, because of the 'near' predicate-like nature of the preposition. We apply argument slot raising to *dtr* only in the way that it generates the surface scope reading between the two object QNPs. Section 5.3 shows how we can switch scope between the subject QNP and the two object QNPs together by applying argument slot raising to ditransitive verbs with regard to their two argument slots, one of which is a complex object argument slot. Section 5.4 provides concluding remarks. All the logical proofs in this section are presented in natural deduction presentation, rather than Gentzen Sequent, because I do not apply any structural rules in this chapter.

5.2 Analysis

The argument slot raising that I have presented in chapter 2 and the following chapters 'normally' applies only to Boolean functors, that is a functor category that has S as the final value. Typically, the functor category in question is $(NP\backslash S)/NP$, as in (180).

(180) a. $(NP1\backslash T)/NP2$ where T is $((\ldots\backslash S)/\ldots)$, $(\ldots(S/\ldots))$, etc.

b. $(NP1\backslash S)/NP2 \Rightarrow (NP1\backslash S)/((S/NP2)\backslash S)$
$\Rightarrow ((S/(NP1\backslash S))\backslash T)/((S/NP2)\backslash S)$

c. $(NP1\backslash S)/NP2 \Rightarrow ((S/(NP1\backslash S))\backslash T)/NP2$
$\Rightarrow ((S/(NP1\backslash S))\backslash T)/((S/NP2)\backslash S)$

Combined with the semantics in (181)~(182), we can switch scope between the two QNPs without creating structural ambiguity in the syntax. That is, as we have

seen in chapter 2, the derivation trees for the two scope readings are the same in the binary structure (cf. Tree D in (60) in chapter 2).[1]

(181) a. Type Shift: $(e,(e,t))) \Rightarrow (((e,t),t),(e,t))$
$\Rightarrow (((e,t),t),(((e,t),t),t))$

b. Semantics: $P \Rightarrow \lambda Q1.\lambda y.Q1(\lambda x.P(x)(y))$
$\Rightarrow \lambda Q1.\lambda Q2.Q2(\lambda y.Q1(\lambda x.(P(x)(y))))$
Variable types Q1, Q2: (et)t; x, y: e.

(182) a. Type Shift: $(e,(e,t)) \Rightarrow (e,(((e,t),t),t))$
$\Rightarrow (((e,t),t),(((e,t),t),t))$

b. Semantics: $P \Rightarrow \lambda Q2.\lambda y.Q2(\lambda y.P(x)(y))$
$\Rightarrow \lambda Q1.\lambda Q2.Q1(\lambda x.Q2(\lambda y.P(x)(y)))$

The question is whether we can apply the argument slot raising to *dtr* (for the DO construction) and *to* (for the PP construction).

(183) a. $dtr :< \epsilon;\ (NP1\backslash(NP1 \bullet NP2))/NP2;\ dtr' >$
where $dtr' \stackrel{\text{def}}{=} \lambda x.\lambda y.y \bullet x$

b. $to :< to;\ (NP2\backslash(NP1 \bullet NP2))/NP1;\ to' >$
where $to' \stackrel{\text{def}}{=} \lambda x.\lambda y.x \bullet y$

Formally, it is possible to extend the use of argument slot raising to both of these functors, as we can value raise $(NP1\backslash(NP1 \bullet NP2))/NP2$ to $(NP1\backslash((S/(NP1 \bullet NP2))\backslash S))/NP2$.[2] As I show in the next section, we can define the semantics accordingly and switch scope between the two QNPs, though just as in the application to transitive verbs, this creates some syntax-semantics mismatch in that we use a special rule which generates the readings that the syntactic calculus cannot generate.

Given the argument slot raising as a special rule, we could potentially switch scope between the two object QNPs for both the DO and the PP ditransitive constructions, but as I have indicated above, considering the minimal role that *dtr* plays in the derivation, I apply argument slot raising to this PF-null functor only in such a way that it preserves the surface scope reading. The preposition *to* contributes more to the categorial derivation, putting the two objects into different

[1]The structures for both the scope readings have '(subject, (verb, object))' bracketing. On the other hand, ASR is postulated as a non-logical axiom in the syntax, so QNP scope is syntactically represented in some sense.

[2]Remember that value raising is provable in NL.

left-to-right orders between the input and the output. To take into account this non-trivial contribution of the prepositional functor *to*, we see the preposition as a proper predicate and apply argument slot raising to it in two different orders, generating two scope readings.

5.2.1 Application 1: DO construction

First, I apply my analysis to the DO construction in (196). In my analysis, I merge the two QNPs by way of the intervening functor *dtr* (in the DO construction), which selects the left QNP object as an argument to the left. Thus, the left object is assigned the category '$S/(NP\backslash S)$,' as in (184c).

(184) a. Meg showed a (♯ different) student every book. $*every > a, a > every$

b. show:
$< show; (NP1\backslash S)/(NP2 \bullet NP3); show'_3 >$
where $show'_3 \stackrel{\text{def}}{=} \lambda\alpha.\lambda x.[show'(\pi 1(\alpha))(\pi 2(\alpha))(x)]$

c. a student: $< a \cdot student; S/(NP\backslash S); \lambda A.some'(St')(\lambda u.A(u)) >$

d. every book: $< every \cdot book; (S/NP)\backslash S; \lambda B.every'(Bk')(\lambda v.B(v)) >$
Types, $Every', Some' : (et)((et)t)$; $\alpha : e \times e$; $x, u, v : e$; $A, B : (et)$.

Given the functor category for the PF-null *dtr* in (185a), we first value-raise it. After that, we apply argument slot raising (= ASR) to its NP argument slots, but only in the order which preserves the surface scope reading, as I have indicated above. I abbreviate $S/(NP\backslash S)$ as QNP1 and $((S/NP)\backslash S)$ as QNP2.

(185) DO

a. *dtr*: $< \epsilon;\ (NP1\backslash(NP1 \bullet NP2))/NP2;\ \lambda x.\lambda y.y \bullet x >$

b. Syntax:
$(NP1\backslash(NP1 \bullet NP2))/NP2$
$\Rightarrow_{VR}\ (NP_1\backslash((S/(NP_1 \bullet NP_2))\backslash S))/NP_2$
$\Rightarrow_{ASR}\ (NP_1\backslash((S/(NP_1 \bullet NP_2))\backslash S))/QNP_2$
$\Rightarrow_{ASR}\ (QNP_1\backslash((S/(NP_1 \bullet NP_2))\backslash S))/QNP_2$

c. Semantics:
$\lambda x.\lambda y.y \bullet x \Rightarrow_{VR}\ \lambda x.\lambda y.\lambda R.R(y \bullet x)$
$\Rightarrow_{ASR}\ \lambda Q2.\lambda y.\lambda R.Q2(\lambda x.R(y \bullet x))$
$\Rightarrow_{ASR}\ \lambda Q2.\lambda Q1.\lambda R.Q1.(\lambda y.Q2(\lambda x.R(y \bullet x)))$
$= dtr_q'$
Types, $Q1, Q2 : (et)t$; $R : (e \times e), t$

Value raising is necessary in (185) to get the semantics right. When we merge two QNP arguments of type (et)t into one complex argument, the output should not be an expression of type $(e \times e)$, which is the type for two NPs concatenated into one complex argument, not for two QNPs. The output of merging two QNPs into one complex QNP should have the type in the form of $((a,t),t)$, in accordance with our Generalized Quantifier analysis of QNPs. Alternatively to my analysis, this output could be just a concatenation of two GQ types, such as $((et)t \times (et)t)$, but I combine the semantics of the two QNPs a little more than just juxtaposing them side by side. Thus, the output in my analysis is the complex QNP expression of type $(((e \times e),t),t)$, which corresponds to the value-raised output category '$(S/(NP \bullet NP))\backslash S$' and which is an adequate GQ type.

If I explain the same point in terms of the categorial calculus, if the output category of merging two QNPs after we apply argument slot raising to the functor *dtr* with regard to its two NP argument slots stayed as $(NP \bullet NP)$, then the merge of the argument slot raised functor with its two QNP arguments would produce the category $(NP2 \bullet NP1)$. Representing the category for the argument slot raised functor as TO for convenience, this process corresponds to the sequent, $(S/NP)\backslash S, (TO, (S/NP)\backslash S) \vdash (NP \bullet NP)$. This sequent has the same effect as type lowering (though it is by way of binary merges, rather than a unary operation). Type lowering, as in '$(S/(NP \bullet NP))\backslash S \vdash (NP \bullet NP)$,' or in '$S/(NP\backslash S) \vdash NP$' with regard to one QNP, is not provable in NL. And even if we decided to ignore the verdict of NL, interpreting a QNP as a type e expression would be very difficult, though not impossible.[3]

In (186), we use the final output in (185) to merge the two object QNPs. Derivation is presented in ND format.

(186) a. Syntax:

$$\dfrac{\dfrac{a \cdot student}{S/(NP1\backslash S)} \quad \dfrac{\dfrac{dtr}{((S/(NP1\backslash S))\backslash((S/(NP1 \bullet NP2))\backslash S))/((S/NP2)\backslash S)} \quad \dfrac{every \cdot book}{(S/NP2)\backslash S}\,/E}{((S/NP1)\backslash S)\backslash((S/(NP1 \bullet NP2))\backslash S)}\,\backslash E}{((S/(NP1 \bullet NP2))\backslash S)}$$

[3]See Kempson, Meyer-Viol, and Gabbay (2001) for type e treatment of QNPs using epsilon calculus, but even for them, the 'operator' interpretation of QNPs is incorporated inside the epsilon terms, so that we can translate the logical forms into standard predicate calculus representations in which QNPs are placed as propositional operators.

b. Semantics:

$$\dfrac{\dfrac{a \cdot student}{\lambda B.Some'(St')(\lambda u.B(u))}\,D \quad \dfrac{\dfrac{\epsilon}{dtr_q'}\quad \dfrac{every \cdot book}{\lambda A.Every'(Bk')(\lambda v.A(v))}\,D}{\lambda Q1.\lambda R.Q1(\lambda v.Every'(Bk')(\lambda y.R(v \bullet y)))}\,/E}{\lambda R.Some'(St')(\lambda v.(Every'(Bk')(\lambda u.R(v \bullet u)))}\,\backslash E$$

where $dtr_q' = \lambda Q2.\lambda Q1.\lambda R.Q1(\lambda y.Q2(\lambda x.R(y \bullet x)))$

The Natural Deduction presentation as in (186) is roughly like the standard syntactic tree notation upside down, with the lexical expressions aligned at the top in the PF left to right order. The output logical expression in the bottom line in (186b) has type $(((e \times e), t), t)$, where its argument R has type $((e \times e), t)$. The logical expression shows that the scope relation is the surface scope, *some* > *every*, between the two object QNPs.

In order to consume the resultant complex QNP object in (186) as the internal argument of the ditransitive verb *show*, we apply argument slot raising to *show*.

(187) a. $(NP1\backslash S)/(NP \bullet NP) \Rightarrow (NP\backslash S)/((S/(NP \bullet NP))\backslash S)$

b. $\lambda\alpha.\lambda z.[show'(\pi 1(\alpha))(\pi 2(\alpha))(z)]$
$\Rightarrow \lambda Q^2.\lambda z.Q^2(\lambda\alpha.show'(\pi 1(\alpha))(\pi 2(\alpha))(z))$
$= show_q'$
Types: $Q^2 : (((e \times e), t), t); \alpha : (e \times e); z : e.$

The output of (187) is a functor whose first argument Q^2 is the complex QNP type derived in (186) which is made out of the two object QNPs. The rest of the derivation is shown in (188).[4]

(188) a. Syntax:

$$\dfrac{\dfrac{Meg}{NP3} \quad \dfrac{\dfrac{show}{(NP3\backslash S)/((S/(NP1 \bullet NP2))\backslash S)} \quad \dfrac{a \cdot student \cdot every \cdot book}{(S/(NP1 \bullet NP2))\backslash S}\,D}{NP3\backslash S}\,/E}{S}\,\backslash L$$

[4] D in the derivation means that I have omitted the derivation from the lexical level to this level.

b. Semantics:

$$\frac{show}{show_q{}'} \qquad \frac{a \cdot student \cdot every \cdot book}{\lambda R.Some'(St')(\lambda v.Every'(Bk')(\lambda u.R(v \bullet u)))}\ D$$

$$\frac{}{show_q{}'(\lambda R.Some'(St')(\lambda v.Every'(Bk')(\lambda u.R(v \bullet u))))}\ /E$$

$$\frac{}{\lambda z.[\lambda R.[Some'(St')(\lambda v.Every'(Bk')(\lambda u.R(v \bullet u)))](\lambda \alpha.show'(\pi_1(\alpha))(\pi_2(\alpha))(z))]}\ \beta$$

$$\frac{}{\lambda z.[Some'(St')(\lambda v.Every'(Bk')(\lambda u.[\lambda \alpha.show'(\pi_1(\alpha))(\pi_2(\alpha))(z))](v \bullet u))]}\ \beta$$

$$\frac{Meg}{meg'} \qquad \frac{\lambda z.[Some'(St')(\lambda v.Every'(Bk')(\lambda u.show'(\pi_1(v \bullet u))(\pi_2(v \bullet u))(z)))]}{\lambda z.[Some'(St')(\lambda v.Every'(Bk')(\lambda u.show'(v)(u)(z)))]}\ \pi_1, \pi_2 \ (\beta)$$

$$\frac{}{\lambda z.[Some'(St')(\lambda v.Every'(Bk')(\lambda u.show'(v)(u)(z)))](meg')}\ \backslash E$$

$$\frac{}{Some'(St')(\lambda v.Every'(Bk')(\lambda v.show'(v)(u)(meg')))}\ \beta$$

where $show_q{}' = \lambda Q^2.\lambda z.Q^2(\lambda \alpha.show'(\pi_1(\alpha))(\pi_2(\alpha))(z)$

β: β reduction; π_1, π_2, conversion by π_i operators

$\pi 1$ and $\pi 2$ select the left and the right members of $(v \bullet u)$, which are put into the appropriate argument slots of *show'*. The generated scope reading is: *a student* $>$ *every book*.

Application 2: PP construction

Next, I apply my analysis to the PP construction in (189a). (189b) is the lexical assignment to the preposition *to*, repeated from (183b). The other items have similar entries as in (184), though the nominal restrictions of the indefinite and the universal must be swapped in comparison to the DO assignments in (184).

(189) a. Meg showed a (different book) to every student. $a > every, every > a$

b. to: $< to; (NP2\backslash(NP1 \bullet NP2))/NP1; \lambda x.\lambda y.(x \bullet y) >$

Note that (189b) reverses the linear order between the two object NPs in the output. This enables us to use the uniform entry for the verb *show* in both the constructions.

As for scope alternation, (189b) does not have the final value category S, as we discussed for (190). Its output category is $NP \bullet NP$. Because of this, we value-raise the output category $(NP \bullet NP)$ to $(S/(NP \bullet NP))\backslash S$. Then, we can apply argument slot raising to this functor in two different orders, deriving two scope readings.

Order 1. Argument slot and value raising to the preposition *to*. The surface scope reading.

(190) a. Syntax:

$(NP2\backslash(NP1 \bullet NP2))/NP1$

$\Rightarrow (NP2\backslash((S/(NP1 \bullet NP2))\backslash S))/((S/NP1)\backslash S)$

$\Rightarrow ((S/(NP2\backslash S))\backslash((S/(NP1 \bullet NP2))\backslash S))/((S/NP1)\backslash S)$

b. Semantics:

$\lambda x.\lambda y.x \bullet y \Rightarrow \lambda Q1.\lambda y.\lambda R.Q1(\lambda x.R(x \bullet y))$

$\Rightarrow \lambda Q1.\lambda Q2.\lambda R.Q2.(\lambda y.Q1(\lambda x.R(x \bullet y)))$

Types: $Q1, Q2 : (et)t; R : ((e \times e), t); x, y : e$

Order 2. Argument slot and value raising to the preposition *to*. The inverse scope reading.

(191) a. Syntax:

$(NP2\backslash(NP1 \bullet NP2))/NP1$

$\Rightarrow ((S/(NP2\backslash S))\backslash((S/(NP1 \bullet NP2))\backslash S))/NP1$

$\Rightarrow ((S/(NP2\backslash S))\backslash((S/(NP1 \bullet NP2))\backslash S))/((S/NP1)\backslash S)$

b. Semantics:

$\lambda x.\lambda y.x \bullet y \Rightarrow \lambda x.\lambda Q2.\lambda R.Q2(\lambda y.R(x \bullet y))$

$\Rightarrow \lambda Q1.\lambda Q2.\lambda R.Q1.(\lambda x.Q2(\lambda y.R(x \bullet y)))$

Note that in (190b), the argument slot for the left object QNP (= Q2) takes wide scope over the argument slot for the right object QNP (= Q1), representing the surface scope reading. The scope relation is the opposite in (191b), representing the inverse scope reading. I only show the inverse scope derivation by applying the output of (191) to its two QNP arguments.

(192) a. Syntax:

$$\dfrac{\dfrac{a \cdot book}{S/(NP2\backslash S)} \quad \dfrac{\dfrac{to}{(QNP2\backslash((S/(NP1 \bullet NP2))\backslash S))/QNP1} \quad \dfrac{every \cdot student}{(S/NP1)\backslash S}\ /E}{QNP2\backslash((S/(NP1 \bullet NP2))\backslash S)}\ \backslash E}{((S/(NP1 \bullet NP2))\backslash S)}$$

where $(QNP2\backslash((S/(NP1 \bullet NP2))\backslash S))/QNP1$

$=((S/(NP2\backslash S))\backslash((S/(NP1 \bullet NP2))\backslash S))/((S/NP1)\backslash S)$

b. Semantics:

$$\dfrac{\dfrac{a \cdot book}{\lambda B.Some'(Bk')(\lambda u.B(u))}\,D \quad \dfrac{\dfrac{to}{to_{q2}'} \quad \dfrac{every \cdot student}{\lambda A.Every'(St')(\lambda v.A(v))}\,D}{\lambda Q2.\lambda R.Every'(St')(\lambda v.Q2(\lambda y.R(v \bullet y)))}\,/E}{\lambda R.Every'(St')(\lambda v.Some'(Bk')(\lambda u.R(v \bullet u)))}\,\backslash E$$

where $to_{q2}' = \lambda Q1.\lambda Q2.\lambda R.Q1(\lambda x.Q2(\lambda y.R(x \bullet y)))$

The logical expression in the bottom line shows the scope relation: *every* $>$ *a*, which is the inverse scope reading.

If we use (190), instead of (191), we can derive the surface scope reading. In (193), I only show the result of merging the output functor in (190) with its two QNP arguments.

(193) a. Syntax: $(S/(NP1 \bullet NP2))\backslash S$

b. Semantics: $\lambda R.some'(bk')\lambda u.every'(st')(\lambda v.R(v \bullet u))$

I omit the derivation steps up to the sentential level, which is similar to (188).

5.3 Object wide scope over subject

5.3.1 Derivation

The data in (194) show that an object QNP can take wide scope over the subject QNP in both DO and PP constructions.

(194) a. A/some student showed every visitor UCL. *a* $>$ *every*; *every* $>$ *a*

b. A/some student showed UCL to every visitor. *a* $>$ *every*; *every* $>$ *a*

Both the sentences in (194) are scopally ambiguous, and we want to generate the two readings. Because our ditransitive category $(NP\backslash S)/(NP \bullet NP)$ is a Boolean category (that is, the final value is S) from the start, this is more straightforward, compared with the scope switch between the two object QNPs. We apply the argument slot raising in two orders to the ditransitive verb functor and generate the two scope readings. I only show the results of the operations. The categorial outputs are the same for both the readings, so I only show it once in (195a). (195b) and (195c) are for the surface and the inverse scope readings respectively.

(195) Argument slot raising (ASR) to *show*

a. Syntax:

$(NP\backslash S)/(NP \bullet NP) \Rightarrow_{ASR} ((S/(NP\backslash S))\backslash S)/(S/(NP \bullet NP))\backslash S)$

b. Semantics (surface scope):
$\lambda\alpha.\lambda z.show'(\pi 1(\alpha))(\pi 2(\alpha))(z)$
$\Rightarrow \lambda Q^2.\lambda Q_1.Q_1(\lambda z.Q^2(\lambda\alpha.show'(\pi 1(\alpha))(\pi 2\alpha)(z)))$

c. Semantics (inverse scope)
$\lambda\alpha.\lambda z.show'(\pi 1(\alpha))(\pi 2(\alpha))(z)$
$\Rightarrow \lambda Q^2.\lambda Q_1.Q^2(\lambda\alpha.Q_1(\lambda z.show'(\pi 1(\alpha))(\pi 2\alpha)(z)))$

When one of the objects is QNP and the other is NP, we 'type raise' the NP into '$(S/NP)\backslash S$' and then merge the two as if they were two QNPs (though this does not create scope ambiguity because the lower type NP expression of type e saturates the corresponding type e argument slot of the ditransitive verb after normalization).

(196) *(A student showed) every visitor UCL*

a. Syntax:

$$\dfrac{\dfrac{every \cdot visitor}{S/(NP\backslash S)} \quad \dfrac{\dfrac{dtr}{(QNP1\backslash((S/(NP1 \bullet NP2))\backslash S))/QNP2} \quad \dfrac{\dfrac{UCL}{NP2}}{(S/NP2)\backslash S}\,TR}{QNP1\backslash((S/(NP1 \bullet NP2))\backslash S)}\,/E}{((S/(NP1 \bullet NP2))\backslash S}\,\backslash E$$

where $(QNP1\backslash((S/(NP1 \bullet NP2))\backslash S))/QNP2$
$= ((S/(NP\backslash S))\backslash((S/(NP1 \bullet NP2))\backslash S))/((S/NP2)\backslash S)$

b. Semantics:

$$\dfrac{\dfrac{every \cdot visitor}{\lambda B.Every'(vtor')(\lambda u.B(u))}\,D \quad \dfrac{\dfrac{\epsilon}{dtr_q'} \quad \dfrac{\dfrac{UCL}{ucl'}}{\lambda A.A(ucl')}\,TR}{\lambda Q1.\lambda R.Q1(\lambda y.R(y \bullet ucl'))}\,/E}{\lambda R.Every'(vtor')(\lambda u.R(u \bullet ucl'))}\,\backslash E$$

where $dtr_q' = \lambda Q2.\lambda Q1.\lambda R.Q_1(\lambda y.Q_2(\lambda x.R(y \bullet x)))$
Reductions of sub-expressions for $/E$:
$(\lambda A.A(ucl'))(\lambda x.R(y \bullet x)) \Rightarrow_{\beta red} (\lambda x.R(y \bullet x))(ucl') \Rightarrow_{\beta red} R(y \bullet ucl')$

In (197), we merge the outputs in (196) with the outputs in (195). For the ditransitive verb, I use the derived logical expression for the inverse scope in the bottom line in (195c).

(197) *A student showed every visitor UCL.* (For the inverse scope: *every* > *a*).

a. Syntax:

$$\dfrac{\dfrac{a \cdot student}{S/(NP3\backslash S)}\,D \quad \dfrac{\dfrac{show}{(QNP3/(S/(NP1 \bullet NP2))\backslash S)} \quad \dfrac{every \cdot visitor \cdot UCL}{((S/(NP1 \bullet NP2))\backslash S}\,D}{QNP3\backslash S}\,/E}{S}\,\backslash E$$

where $(QNP3\backslash S)/(S/(NP1 \bullet NP2))\backslash S)$
$= ((S/(NP3\backslash S))\backslash S)/(S/(NP1 \bullet NP2))\backslash S)$

b. Semantics:

$$\dfrac{\dfrac{a \cdot student}{\lambda A.some'(student')(A)}\,D \quad \dfrac{\dfrac{show}{show'_{sw}} \quad \dfrac{every \cdot visitor \cdot UCL}{\lambda R.Every'(visitor')(\lambda u.R(u \bullet ucl'))}\,D}{\lambda Q1.Every'(visitor')(\lambda u.Q_1(\lambda z.show'(u)(ucl')(z)))}\,/E}{Every'(visitor')(\lambda u.some'(student')(\lambda z.show'(u)(ucl')(z)))}\,\backslash E$$

where $show'_{sw} = \lambda Q^2.\lambda Q_1.Q^2(\lambda\alpha.Q_1(\lambda z.show'(\pi_1(\alpha))(\pi_2(\alpha))(z)))$

Reductions (I put [ ] around the functor in question at each reduction for convenience):

For $/E$:

$[\lambda Q^2.\lambda Q_1.Q^2(\lambda\alpha.Q_1(\lambda z.show'(\pi_1(\alpha))(\pi_2(\alpha))(z)))](\lambda R.Every'(visitor')(\lambda u.R(u \bullet ucl')))$
$\Rightarrow_{\beta red} \lambda Q_1.[\lambda R.Every'(visitor')(\lambda u.R(u \bullet ucl'))](\lambda\alpha.Q_1(\lambda z.show'(\pi_1(\alpha))(\pi_2(\alpha))(z)))$
$\Rightarrow_{\beta red} \lambda Q_1.Every'(visitor')(\lambda u.[\lambda\alpha.Q_1(\lambda z.show'(\pi_1(\alpha))(\pi_2(\alpha))(z))](u \bullet ucl'))$
$\Rightarrow_{\beta red} \lambda Q_1.Every'(visitor')(\lambda u.Q_1(\lambda z.show'(\pi_1(u \bullet ucl'))(\pi_2(u \bullet ucl'))(z)))$
$\Rightarrow_{\beta red} \lambda Q_1.Every'(visitor')(\lambda u.Q_1(\lambda z.show'(u)(ucl')(z)))$

For $\backslash E$:

$[\lambda Q1.Every'(visitor')(\lambda u.Q_1(\lambda z.show'(u)(ucl')(z)))](\lambda A.some'(student')(A))$
$\Rightarrow_{\beta red} Every'(visitor')(\lambda u.[\lambda A.some'(student')(A)](\lambda z.show'(u)(ucl')(z)))$
$\Rightarrow Every'(visitor')(\lambda u.some'(student')(\lambda z.show'(u)(ucl')(z)))$

I omit the derivation of the surface scope reading, *a* > *every* which is straightforward, using (195b), rather than (195c) in the semantics in (197).

5.3.2 Loose ends

The proposed algorithm can switch scope between the two object QNPs by applying argument slot raising to the functor *to* in two orders in the PP construction. In the DO construction, the PF-null functor *dtr* cannot support argument slot raising

in two orders and thus the scope is frozen in the surface order. The algorithm can switch scope between the subject QNP and the two object QNPs together by applying argument slot raising to the ditransitive verb functor in two orders. If all the three arguments are QNPs, ASR on its own can generate two scope readings for the DO and four scope readings for the PP construction, as indicated in (198).[5]

(198) a. Every estate agent showed a customer two houses.
every > *a* > *two*; *a* > *two* > *every*.

b. Every estate agent showed a room to two customers.
every > *a* > *two*; *every* > *two* > *a*;
a > *two* > *every*; *two* > *a* > *every*.

However, the sentences in (198) seem to have more readings. For example, the scope reading *a* > *every* > *two* which roughly says that there is one customer to whom each agent showed a different set of two houses sounds possible. In other words, we can get the reading in which only the left object takes wide scope over the subject QNP, whereas the right object QNP takes narrow scope with regard to the subject. In the PP construction, there are more cases in which only one of the object QNPs takes wide scope over the subject. The suggested scope algorithm cannot generate any of these readings.

However, in chapter 9, I argue that the so-called 'exceptional scope taking' of indefinites is actually a matter of the domain restriction that applies to the nominal restriction set of the indefinites, following Schwarzschild (2002). I do not go into details here, but consider (199).

(199) Five students discussed two problems.

(199) has two readings (among others). That is, we can either pick up the same pair of problems for all the students, or pick up a different pair of problems for each of the five students. With pragmatic domain restriction, we can generate these two readings without resorting to any scope switch algorithm. The former reading is generated if we pragmatically restrict the set of problems into two-member set. If the set of problems includes only two members, we cannot pick up more than two problems for all the students combined, even if the existential scope of the object indefinite stays narrower than the subject indefinite. Assuming that domain restriction is an independently motivated (partially) pragmatic operation, some of the readings of the sentences in (198) do not pose a problem for my analysis.

[5]The numbers of the readings do not represent the contribution of the dependency of the indefinites on other operators which I discuss in chapter 9. I do not regard such dependency of indefinites as a matter of QNP scope. See also the discussion surrounding (199) below.

I divert a little now with regard to (199). That is, there are other readings available for the sentence. For example, (199) has a reading which says that there are two specific problems, and for each of these two problems, a different set of five students discussed it. The domain restriction analysis that I introduce in chapter 9 can make the indefinite *two problems* either be dependent on or independent of the higher indefinite *five students*. In the latter case, we can get a reading about the same pair of students for the five students by restricting the set of problems to a set that contains only two problems independent of the higher indefinite *five students*. However, the lower indefinite *two problems* being independent of the higher indefinite *five students* does not mean that the higher indefinite is dependent on the lower indefinite. In fact, in the current formulation of the analysis which I explain in more detail in chapter 9, the indefinite *five students* cannot be dependent on the lower indefinite *two problems* because only an indefinite that is merged earlier in the derivation can be dependent on an element that is merged later. Thus, to get the reading in which each of the two specific problems was discussed by a different set of five students, we need to use scope switch algorithm in terms of argument slot raising in my analysis. For (199), ASR may apply because the two indefinites are within the same S expression (i.e. within the same tensed clause), but the sentence *Five students said that Tom would discuss two problems* is predicted not to have the corresponding reading which says that for each of the two specific problems, a different set of five students said that Tom would discuss it. See chapter 9 for details.

Coming back to the ditransitive constructions, we can incorporate this additional property of indefinites into my analysis and test if the proposed scope switch algorithm is completely adequate. For convenience, I only consider the DO construction.

(200) a. Two students showed every teacher five problems. $every > two > five$

b. For each teacher, there is a different pair of students, and for each one of the pair, there is a different set of five students he or she showed to the teacher.

The reading in question in (200) is a little difficult to understand, but if there are five teachers, the reading means that there are maximally 5×2=10 students involved and the number of the problems can maximally be $10 \times 5 = 50$. This reading, if the reading is actually supported by native speaker judgment, cannot be generated even by the combination of my scope switch algorithm and the mechanism that explains the domain dependency of indefinites. I do not discuss this

issue any further and leave it for future research.

5.4 Conclusion

This chapter discusses QNP scope in two kinds of ditransitive constructions. To deal with the frozen scope between the two object QNPs in the DO construction as opposed to the PP construction which exhibits scope ambiguity, I apply argument slot raising to the two NP argument slots of the prepositional functor *to* in two orders, whereas I apply it to the PF null functor *dtr* for the DO construction only in one way, that is, the way which preserves the surface scope reading. Though this contrast is just a stipulation as far as the formalism is concerned, I suggested that it is related to the extra expressive power of the prepositional functor *to* in linguistic terms, which switches the order of the two arguments in the output (the extra power which may be potentially related to the ability of the prepositional head to assign a certain kind of semantic role to its arguments). With this extra assumption, the suggested scope mechanism can generate the readings which we want. There is a problem with regard to the scope relation between the subject QNP and the object QNPs in ditransitive constructions which my analysis cannot explain at the moment, and I have left it for future research.

Chapter 6

QNP in PP

6.1 Introduction

In the cases which we have seen so far, QNP scope has been explained in terms of argument slot raising that is applied to the functor that takes the QNP in question as an argument. The functor typically has a lexical category in the form of (201a), where T is a shorthand for some Boolean category in the form of $((\ldots\backslash S)/\ldots)$ or $(\ldots\backslash(S/\ldots))$.

(201) a. $(NP\backslash T)/NP$

b. $(NP\backslash T)/NP \Rightarrow (NP\backslash T)/((S/NP)\backslash S)$
$\Rightarrow ((S/(NP\backslash S))\backslash T)/((S/NP)\backslash S)$

c. $(NP\backslash T)/NP \Rightarrow ((S/(NP\backslash S))\backslash T)/NP$
$\Rightarrow ((S/(NP\backslash S))\backslash T)/((S/NP)\backslash S)$

In practice, argument slot raising is normally applied to verbal functors. A typical case is its application to verbal functors of category $(NP\backslash S)/NP$.

Two ways of applying argument slot raising lead to two scope readings, as the repeated semantic derivations in (202) and (203) indicate.

(202) a. Type Shift: (e(et)) $\Rightarrow$ (((et)t)(et)) $\Rightarrow$ (((et)t)(((et)t)t))

b. Semantics:
$P \Rightarrow \lambda Q1.\lambda y.Q1(\lambda x.P(x)(y)) \Rightarrow \lambda Q1.\lambda Q2.Q2(\lambda y.Q1(\lambda x.(P(x)(y))))$
Variable types $Q1, Q2 : (et)t$; $\quad x, y : e$

(203) a. Type Shift: (e(et)) $\Rightarrow$ (e(((et)t)t)) $\Rightarrow$ (((et)t)(((et)t)t))

b. Semantics:
$P \Rightarrow \lambda Q2.\lambda y.Q2(\lambda y.P(x)(y)) \Rightarrow \lambda Q1.\lambda Q2.Q1(\lambda x.Q2(\lambda y.(P(x)(y))))$

As we have discussed in chapter 5, however, the functor does not have to be a verb. In the prepositional ditransitive construction, we applied argument slot raising to the prepositional functor *to*.

In this chapter, I discuss some more cases in which a QNP is in a prepositional phrase (PP). There are three sub-cases I consider, though I identify the first one as the PP ditransitive construction that I have already discussed. I provide detailed analyses of the other two.

Firstly, there is a case as in (204a).

(204) a. Tom put ((an apple) (in (every box))). $\#a > every$, $every > a$

b. Tom showed ((a book) (to (every student))). $a > every$, $every > a$

In (204a), the surface scope reading, $a > every$, is odd (indicated by #). But I assume that this oddity is pragmatic in nature. That is, we cannot put the same apple in more than one box at one time because of some physical law that applies in our world. But from a theoretical viewpoint, I assume that (204a) is scopally ambiguous.

Unlike the other two cases below, the PP *in every box* in (204a) is obligatory. That is, English speakers find **/?Tom put an apple* either ill-formed or elliptical. Thus, I assume that the verb *put* obligatorily requires two internal arguments and treat (204a) in the same way as I have dealt with the PP ditransitive construction in (204b) in chapter 5.

Secondly, as in Carpenter (1997: 232-238), Bale and May (2006) and Heim and Kratzer (1998), we have to consider the case in which a QNP is inside a PP which in turn is inside the nominal restriction of another QNP, as shown in (205a) and (205b). I call this second case, the 'QNP in QNP' construction. As with (204a), I assume that the oddities indicated by # are pragmatic in nature, and thus assume that both the sentences are scopally ambiguous with regard to the two QNPs in question.

(205) a. (An (apple (in every box))) was mouldy. $every > a$, $\#a > every$

b. (A (student (from every school))) gave a speech.

$every\ school > a\ student$, $\#a\ student > every\ school$

Though for each sentence in (205), there is a prepositional functor (i.e. *in* or *from*) between the two QNPs in the PF string, we cannot apply argument slot raising in (201)~(203) to the functor in order to derive scope ambiguity. In order to do this, we would need a functor of category $(NP\backslash T)/NP$, so that we can apply argument slot raising with regard to the two NP argument slots. But in (205a), it is not easy

to treat the preposition *in* in this way. Corresponding to the internal syntactic structure of the subject QNP as is suggested by the parentheses in (205a), the logical expression for the prepositional functor *in* needs to be incorporated into the nominal restriction of the existential quantifier *some*. That is, the prepositional expression must occur within A in the form $some'(A)(B)$. This means that we have to merge the PP *in every box* first with the common noun *apple* before we merge the result with the determiner *a*, rather than to merge the PP *in every box* with the indefinite NP *an apple*, as in the structure $((an\ apple)(in\ (every\ box)))$. The same applies to (205b).

Lastly, I consider cases in which the PP is normally analysed as a VP adjunct, as in (206).

(206) a. Tom ((bought (an apple)) (in (every shop))). $\#a > every,\ every > a$

b. Tom ((met (a student)) (in (every shop))). $a > every,\ every > a$

The PPs in (206) are normally analysed as VP adjuncts, as is indicated by the informal bracketing. The type logical system that I have adopted does not structurally distinguish whether a categorial formula is merged as the head or as the complement for each merge. Thus, the use of the term 'adjunct category' in this thesis is for presentational convenience. Having said that, all the 'adjunct categories' have the form $(X\backslash X)$ or (X/X), though not all the items that have these categorial forms are 'adjuncts' in the GB/Minimalist sense. For presentational convenience, I call the structures which involve categories of the form '$(X\backslash X)$' **adjunct structures** in this thesis.

Coming back to (206), to represent this PP-as-adjunct structure in the lexical categories, the preposition *in* would have to be assigned the category, $(VP\backslash VP)/NP$ (where VP is a shorthand for $NP\backslash S$). Because this prepositional functor *in* can have only *every shop* to the right as its QNP argument by way of argument slot raising, and because the indefinite (*an apple* in (206a) or *a student* in (206b)) as the QNP argument of the transitive verb (i.e. *buy* in (206a) or *meet* in (206b)) is contained inside the VP that is the second argument of the prepositional functor, we cannot switch scope by applying argument slot raising (or value raising) to the prepositional functor.

To deal with these sentences, I adopt the complement analysis of PPs that Dowty proposes in Dowty (2003) with modifications. Unlike Dowty, however, I define the verbal functor in the PP-as-complement structure as a higher order version of the standard verbal functor expression (e.g. the standard category is $(NP\backslash S)/NP$ for transitive verbs). Thus, just as the effect of type raising of a type

e expression (such as *tom'* for *Tom*) to a type (et)t expression (i.e. $\lambda A.A(tom')$) can be cancelled by way of normalization of the resultant logical form, the use of the higher order verbal expression that takes the PP as an argument, instead of using the lexically assigned verbal functor that is merged with the PP as an adjunct, leads to the same normalized logical form, unless scope relation or binding relation is involved. Though this analysis fails to explain the semantic differences that Dowty has tried to capture by postulating two mutually irreducible verbal functor expressions for the adjunct analysis on the one hand and for the complement analysis on the other, I argue that my formulation has a merit in that it represents well the general property of the Type Logical Grammar system that does not distinguish the head item and the complement item when it merges two items at each step (and thus, it does not formally distinguish complements from adjuncts in this way, either). On the other hand, the complement re-analysis is not completely provable in the grammar system NL, as we see in section 6.3.3.

In section 6.2, I discuss the QNP in QNP case. In section 6.3, I deal with PPs that are normally analysed as VP adjuncts. Section 6.4 provides concluding remarks.

6.2 QNP in QNP

In this section, I consider some cases in which a QNP appears inside a PP which in turn is contained in the nominal restriction of another QNP.

6.2.1 Basics

Consider (207). The sentences show scope ambiguity between the two QNPs.

(207) (An (apple (in (every box)))) was mouldy. *every* > *a*, #*a* > *every*

As I said above, in this construction, the logical expression for the PP is incorporated in the nominal restriction of the left QNP in the PF string. Using (207) as an example, this may be represented in the logical expression for *in* given in Carpenter (1997: 233).[1]

[1]Providing compositional semantics to restrictive relative clause constructions across different languages, Bach and Cooper (1978) assumes that an NP such as *an apple* has an optional argument slot for property denoting (i.e. type (et)) expressions. In such an analysis, $\lambda R_{et}.\lambda P_{et}.\exists x[(R(x) \wedge apple'(x)) \wedge P(x)]$ might be the logical entry for the NP *an apple* (cf. Bach and Cooper (1978: 148)). When this NP functor is merged with a relative clause expression, such as $\lambda z.eat'(z)(tom')$ for *which Tom ate*, then the normalization of the logical expression will produce the expression $\lambda P.\exists x[(eat'(x)(tom') \wedge apple'(x)) \wedge P(x)]$ of type (et)t for *an apple which Tom ate*. In this way,

(208) in_1

a. Category: $(N\backslash N)/NP$

b. type: (e, ((et), (et)))

c. Logical expression: in'_1 of type (e, ((et), (et)))
where, $in'_1 \stackrel{\text{def}}{=} \lambda x.\lambda A.\lambda y.(A(y) \wedge in'_0(x)(y))$
Types, $in'_0 : e(et)$; $A : (et)$; $x, y : e$

in_1 given in (208c) can be explained as derived from in_0 by covert generalized conjunction.[2] I now show that we can derive the two scope readings for (207) by using the lexical entry in (208).

6.2.2 Surface Scope

The proposed analysis can relatively easily derive the surface scope reading, $a >$ *every*. We first apply argument slot raising (ASR) to the prepositional functor in (208) with regard to its unique NP argument slot.

(209) ASR applied to in_1 in (208).

a. $(N\backslash N)/NP \Rightarrow$
$(N\backslash N)/((S/NP)\backslash S)$

b. $\lambda x.\lambda A.\lambda y.(A(y) \wedge in'_0(x)(y)) \Rightarrow$
$\lambda Q.\lambda A.\lambda y.(A(y) \wedge Q(in'_0(x)(y)))$

(210) provides the entries for the other items in (207).

(210) a. a: $< a; (S/(NP\backslash S))/N; \lambda A.\lambda B.some'(A)(B) >$ [type (et)((et)t)]

b. apple: $< apple; N; \lambda x.apple'(x) >$ [type (et)]

c. every box: $< every{\cdot}box; (S/NP)\backslash S; \lambda B.every'(box')(B) >$ [type (et)t]

The derivation of the subject QNP *an apple in every box* in (207) is shown in (211).

---

they can merge the head NP with the relative clause as in the binary structure, '[$_{NP}$ [$_{NP}$an apple][$_{CP}$ that Tom ate]],' but they can still incorporate the logical expression for the relative clause inside the nominal restriction of the head NP. We could use the same entry for *an apple* and treat the preposition *in* as of lexically type (e(et)). Though this analysis might have some semantic merits (e.g. the PP *in the box* might better be assigned a type (et) expression in the copula construction, such as *The apple is in the box*), it is not obvious that we want the extra complexity in Bach and Cooper's analysis, especially for English sentences (as opposed to the Hittite relative clause construction, with which they motivate their alternative structure). Also, because the indefinite NP *an apple* would then be assigned the type (et)((et)t), rather than our GQ type ((et)t), it is not easy to switch scope in the proposed system. For these reasons, I do not consider their alternative structure.

[2]See Cormack and Smith (2005)

(211) *an apple in every box*

a. category: $S/(NP\backslash S)$

b. LF: $\lambda B.some'(\lambda y.(apple'(y) \wedge every'(box')(\lambda x.in'(x)(y))))(B)$

c. PF: $(an \cdot (apple \cdot (in \cdot (every \cdot box))))$

The PF bracketing in (211c) represents the order of the syntactic merges. The scope reading is fixed as the surface scope, $a > every$, at this stage, as we can see in (211b).

At the last step of this derivation, the determiner *a* is the functor and *apple in every box* is the argument. This normal functor-argument relation is reversed when we derive the inverse scope reading in the next sub-section. Because the functor-argument alternation is free in NL as long as we do not rebracket the structure, we can derive the inverse scope reading in NL, as we see in the next section.

After deriving the subject indefinite as in (211), we merge the result with the main clause predicate *was mouldy*, which I analyze as of category $NP\backslash S$ and of type (et). We first apply argument slot raising to this main clause predicate with regard to its unique NP argument slot. Then this predicate can take the derived indefinite in (211) as its argument. The result of this final step is shown in (212), which represents the surface scope reading of (207). Again, the bracketing in PF represents the order of the syntactic merges.

(212) *An apple in every box was mouldy.* ($some > every$)

a. LF: $some'(\lambda y.(apple'(y) \wedge every'(box')(\lambda x.in'(x)(y))))(\lambda y.mouldy'(y))$

b. PF: $(an \cdot (apple \cdot (in \cdot (every \cdot box)))) \cdot (be \cdot mouldy)$

6.2.3 Inverse Scope

To derive the inverse scope reading, we apply argument slot raising to the functor in_1 in (208) with regard to its unique NP argument slot, and apply value raising with regard to its output category N. There are two ways of doing this, but I only show the rule application that leads to the inverse scope, because the other form of application leads to exactly the same result as in (212) above.

The result of applying ASR and value raising to in_1 in (208) (for the inverse scope) is shown in (213). I abbreviate $S/(NP\backslash S)$ as $QNP1$ and $(S/NP)\backslash S$ as $QNP2$.

(213) Inverse scope:

a. $(N\backslash N)/NP \Rightarrow$
$(N\backslash((QNP1/N)\backslash QNP1))/QNP2$

b. $\lambda x.\lambda A.\lambda y.(A(y) \wedge in'_o(x)(y)) \Rightarrow$
$\lambda Q.\lambda A.\lambda D.\lambda B.Q(\lambda x.D(\lambda y.A(y) \wedge in'_o(x)(y))(B))$

Given the argument slot and value raised functor in (213) and the other lexical items in (210), we can derive the subject QNP *an apple in every box* as in (214). The PF bracketing in (214c) represents the order of the syntactic merges.

(214) a. Category: $S/(NP\backslash S)$

b. Logical expression:
$\lambda B.every'(box')(\lambda x.some'(\lambda y.(apple'(y) \wedge in'(x)(y)))(B))$

c. PF: $(an \cdot (apple \cdot (in \cdot (every \cdot box))))$

Note that the order of the merges stays the same both for the surface scope derivation in (211c) and the inverse scope derivation in (214c). The difference is that, in the last step, the determiner *a* is merged as the functor in (211c), but in (214c), it is merged as the argument. This functor-argument alternation is supported by NL, which does not distinguish the structural positions of functors and arguments.

At the sentential level, we can derive the category and the logical form as in (215) for the inverse scope reading.

(215) An apple in every box was mouldy.

a. Category: S

b. Logical form for inverse scope:
$every'(box')(\lambda x.some'(\lambda y.(apple'(y) \wedge in'(x)(y)))(\lambda y.mouldy'(y)))$

To deal with scope switch in the QNP in QNP construction, all we need as a special rule is argument slot raising which, again, is not fully supported by NL. Value raising is provable in NL and so given ASR, we can naturally switch scope between the embedded QNP and the containing QNP in *an apple in every box.*

6.2.4 Some other QNP embedded in QNP structures

QNP in a participial modifier

If a QNP appears in a participial phrase which modifies the nominal restriction of the containing QNP, the suggested scope switch analysis predicts scope ambiguity. Consider (216).

(216) (Some/a student (viewing (every school))) was Irish.

some > *every*, *every* > *some*

Given the lexical assignment to *view* as in (217), the scope ambiguity in (216) is expected.

(217) *viewing*

a. Category: $(N\backslash N)/NP$

b. logical expression:
$viewing'_1 := \lambda x.\lambda A.\lambda y.(A(y) \wedge view'_0(x)(y))$, (cf. in_1 in (208)).
where $viewing'_1$ is of type (e,((et),(et)))

Because (217) assigns the same syntactic category and semantic type to *viewing* as (208) assigns to the preposition *in*, we can apply the same procedures and derive two scope readings.

(218) a. Surface scope:
$some'(\lambda y.(student'(y) \wedge every'(school')(\lambda x.view'(x)(y))))(\lambda y.irish'(y))$

b. Inverse scope:
$every'(school')(\lambda x.some'(\lambda y.(student' \wedge view'(x)(y)))(\lambda y.irish'(y)))$

QNPs may be embedded further down, as in (219), but I do not show the analyses of these sentences.

(219) a. Some/a candidate chosen in every/each selection was English.
some > *every*, *every* > *some*

b. An athlete checked with every/each machine proved positive.
a > *every*; *every* > *a*

In the following subsection, I show that the combination of argument slot raising and value raising does not allow us to switch scope across a relative clause.

QNP inside a restrictive RC

The argument slot and value raising cannot switch scope if the right QNP is embedded in a relative clause that modifies the nominal restriction of the left QNP, as in (220).

(220) a. ♯ (An (apple (which (was (in (every box)))))) was mouldy.

a > *every*, **every* > *a*

b. $(N\backslash(\ldots\backslash\mathbf{N}))/NP$

In order to switch scope by argument slot and value raising as in (215), the prepositional functor *in* would have to have the categorial form as in (220b), where the NP argument slot and the output category **N** (in bold face for emphasis) would then be raised to switch scope. Technically, there can be some other arguments (marked as ... in (220b)) between the N argument for *apple* and the output category **N**, but all the items between the noun *apple* and the prepositional functor *in* would have to be merged as arguments of *in*, if there are any between the two. However, assignment of such a category to *in* in (220a) would be very difficult, because it is difficult to treat *which* and *was* as arguments of the preposition *in* in (220a). The relative pronoun *who* is normally assigned a functor category such as $Wh/(\Diamond\Box^{\downarrow}NP\backslash S)$, and is merged with the rest of the relative clause being its argument of category $\Diamond\Box^{\downarrow}NP\backslash S$.[3] Also, as the semantic contribution of the copula *be* is little, *be* is normally assigned polymorphic identity function categories in the form of X/X, which select various predicate categories as arguments and give back the same categories as the output of the function application.[4] Assuming that such assignments to the functor *in* are not linguistically sustainable, the proposed analysis correctly predicts that the embedded QNP in a relative clause inside the nominal restriction of another QNP cannot take wide scope over the containing QNP.

It might seem that there is another way of switching scope in (220a). That is, if we could regard the PF string *which was in* as a complex predicate, of category $(N\backslash N)/NP$, for example, then we could apply argument slot raising and value raising to this hypothetical complex predicate. But as we will see in chapter 7 and 8, our grammar system cannot derive this string as a complex predicate, roughly because it crosses a propositional boundary (or because we cross the minimal S category).[5] Thus, we cannot switch scope by this route, either.

---

[3] $\Diamond\Box^{\downarrow}NP$ marks the hypothetical 'trace' category inside the relative clause. See chapter 8 and Vermaat (2006) for the treatment of Wh movement in TLG.

[4] For example, *Tom is nice* can be analyzed as a sequent, $NP, ((NP\backslash S)/(NP\backslash S), NP\backslash S) \vdash S$, where $NP\backslash S$ for *nice* is merged as the argument of the copula *be*. Then, the PP *in the garden* in *Tom is in the garden* will have a similar categorial status to the adjective *nice*, that is, it is merged as the argument of *be* as well. For different kinds of category assignment to *be*, see Carpenter (1997: 193).

[5] Note that there is a tensed clause inside the relative clause, which corresponds to the minimal S in our analysis.

6.2.5 Binding from an embedded position

There seems to be some connection between the inverse scope taking of an embedded QNP over the containing QNP on the one hand, and the embedded QNP's scope for binding pronouns on the other. Consider (221).

(221) a. Some/a student from every school$_1$ liked it$_1$.

every > *some/a*; ??*some/a* > *every*

b. Some/a student studying every language$_1$ liked it$_1$.

every > *a*; ??*some/a* > *every*

c. *Some/a student who studied every language$_1$ liked it$_1$.

Both (221a) and (221b) seem to have the suggested bound pronoun interpretation. The universal QNPs are embedded inside the subject QNPs and hence do not c-command the pronoun *it*. Thus, it is not clear how the sentences have the bound pronoun interpretation.[6]

Interestingly, however, some speakers of English report that the bound pronoun interpretation is significantly more difficult to get with the surface scope reading, *some* > *every* than with the inverse scope reading *every* > *some*.[7] Also, in (221c), where the embedded universal QNP inside the relative clause cannot take scope over the containing indefinite, the bound pronoun reading is unavailable.

In the logical forms in (214b), (215b) and (218), which my scope-switch system generates for *An apple in every box was mouldy* and *A student visiting every school was Irish*, the matrix predicate is in the scope of the embedded universal quantifier only when the universal takes wide scope over the subject indefinite.[8] In other words, in the proposed scope switch system, the universal quantifiers in these embedded positions can take scope over the matrix clause predicates only in the inverse scope reading. Now, consider the logical forms in (222a) and (222b),

[6]We might apply QR as in May and adjoin the universal QNP to the matrix sentential node, or use Montague's quantifying-in or its type logical equivalent, in order to let the universal QNP have the sentential scope. However, as I have pointed out before, raising all the QNPs to a sentential node overgenerates the scope readings, even if we keep the raising within the smallest tensed clause that contains the QNP. QR would incorrectly generate the inverse scope reading in the DO ditransitive construction of chapter 5. QR would also generate the unattested reading, *every* > *two* > *some* for the string, *Some/a student from every school gave two presentations.*

[7]At least for (221b), the decreased acceptability with the reading ??*some/a* > *every* cannot be because of the difficulty of the surface scope reading itself, because the sentence without the bound pronoun does have the surface scope reading, as in *Some/a student studying every language might still not be a good linguist.* For (221a), the surface scope reading is odd because the same student normally does not belong to many schools.

[8]In the notation, *every*$'(A)(B)$, let me call A the nominal restriction, and B the scope, of *every*, following the standard terminology.

which are for the surface scope reading and the inverse scope reading of (221b) respectively.

(222) a. Surface scope (*some* > *every*):

$some'(\lambda y.(student' \wedge every'(language')(\lambda x.study'(x)(y)))(\lambda y.like'(\mathbf{x})(y))$

b. Inverse scope (*every* > *some*):

$every'(language')(\lambda x.some'(\lambda y.(student'(y) \wedge study'(x)(y)))(\lambda y.like'(\mathbf{x})(y)))$

In the higher order logical notations that I am using, in order for some type e argument slot to be 'bound' by a quantifier, such as the universal quantifier encoded by *every*, that argument slot has to be bound by one of the two lambda operators in the schematic form, $every'(\lambda x.\phi)(\lambda x.\psi)$, where the functor of the argument slot in question occurs somewhere in either ϕ or ψ of type t. The argument slot **x** in bold face is bound by λx in the appropriate position only in (222b) for the inverse scope reading. In (222a) for the surface scope reading of (221b), the argument slot **x** in question stays free.[9]

This means that if the proposed system can generate these two logical forms for (221b) without further stipulation, my system can naturally explain the binding data in the QNP in QNP constructions. As far as the proposed scope switch system is concerned, we might expect that the logical form in (222b) is easily derivable in my analysis, judging from (215b) and (218). However, the pronominal 'binding' system in terms of argument slot identification, which I explained in chapter 3, does not easily generate the logical form as in (222b) with regard to the argument slot x. This is because the structural binding rule Z which we have seen in chapter 3 requires strict c-command configuration between the binder and the bindee. Remember from chapter 3 that we have decided not to modify the rule Z to accommodate sentences such as *The mother of every child*$_1$ *came to pick him*$_1$ *up* or *Every child's*$_1$ *father defended him*$_1$. Thus, whatever modification we might make to Z to accommodate these sentences may allow us to explain the binding data in (221). But I leave for further research what kind of modifications are required in the binding mechanism that I use in order to generate (222b).

6.3 Adjunct PPs

In this section, I discuss PPs that are normally analyzed as adjuncts to VPs.

[9]For expository reasons only, I use a free variable in (222a).

6.3.1 Dowty's dual analysis of adjuncts

Consider (223).

(223) a. Tom $[_{VP}$ $[_{VP}$ met [a linguist]] $[_{PP}$ in every school]]

$a > every$; $every > a$

b. Tom_i $[_{VP}$ $[_{VP}$ bought [an apple]] $[_{PP}$ in every shop]]

$\#a > every$; $every > a$

PPs such as *in every shop* and *in every school* in (223a) and (223b) are optional elements of the sentence and they can generally be stacked on top of one another, as in *Tom bought an apple in a (student) shop in every school.* Because of this optionality, these PPs are normally analysed as VP adjuncts with the categorial assignment, $(NP\backslash S)\backslash(NP\backslash S)$. However, if we represent this adjunct status of the PP in (223a) or (223b) in the lexical category of the prepositional functor *in*, then the indefinite QNP will not be an argument of the prepositional functor that will be assigned the lexical category $((NP\backslash S)\backslash(NP\backslash S))/NP$. In (223b), by the time the PP is merged with the inner VP, the indefinite *an apple* has already been merged with the transitive verb *buy* and has formed part of the inner VP structure, leading only to the narrow scope reading of the indefinite. Thus, with this adjunct structure, we cannot switch scope in terms of the prepositional functor *in*.

However, it is not clear whether the structure in (223) is the only one that we need for linguistic analyses. For example, consider the following binding relation.

(224) a. Tom met every student_1 in his_1 high school.

b. Tom met every linguist_1 on his_1 birthday.

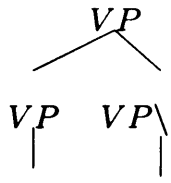
The binding relation as in (224) needs to be explained somehow and the VP adjunct structure in (223) does not lead to the correct c-command relation for this, because the PP is placed higher than the object QNP in that binary structure. In order to explain such a binding relation, we need an alternative structure.

For dealing with a different set of problems than pronominal binding, Dowty (2003) proposes that adjuncts can be re-analysed as complements.[10,11]

[10] I briefly mention a semantic motivation for Dowty's dual analysis below. For others, see Dowty (2003).

[11] The preposition *to* in this construction is different from *to* in the ditransitive construction. In the next sub-section, we can see the differences in terms of the categories and the logical expressions assigned to *to*.

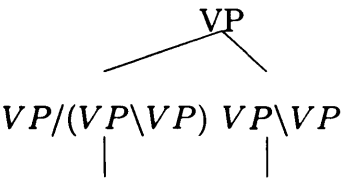
(225) a. adjunct analysis:



Sem: $(to'(mary'))(speak'_1)$

N.B. $VP = (NP\backslash S)$

b. complement reanalysis:



Sem: $speak'_2(to'(mary'))$

cf. Dowty (2003: 44)

The categorial re-analysis of *speak* from VP to $VP/(VP\backslash VP)$ (where VP is a shorthand for $NP\backslash S$) is just a special case of the re-analysis of an argument category as a functor category, which is generally provable in NL.[12]

Because this re-analysis is syntactically free, normally we define the corresponding semantic shift applied to the original argument expression in such a way that the merge of the sisters in question will lead to exactly the same logical expression as in the initial analysis by way of normalization of the logical expressions. More specifically, if we defined the functor $speak'_2$ in the complement analysis in Dowty as $speak'_3$ in (226c) below, which is just a lifted version of the normal intransitive logical expression $speak'_1$ (which is used for the adjunct analysis), then the result of the merge of the verb and the PP in the complement analysis would lead to the same normalized logical expression as in the adjunct analysis, as is shown in (226d).

(226) VP: *speak to Mary*

a. Adjunct analysis: $(to'(mary'))(speak'_1)$

b. Complement analysis: $speak'_2(to'(mary'))$

Type assignments (provided by the author):
$to'(mary') : ((et)(et))$; $speak'_1 : (et)$; $speak'_2 : (((et)(et)),(et))$

c. (Hypothetical alternative for $speak'_2$)
$speak'_3 := \lambda M.M(speak'_1)$

Type: $speak'_3 : (((et)(et)),(et))$; $M : ((et)(et))$

d. $speak'_3(to'(mary')) = (\lambda M.M(speak'_1))(to'(mary'))$
$\Rightarrow_{\beta red.} (to'(mary'))(speak'_1)$

[12]For Dowty's comments about the statuses of complement vs. adjunct in Categorial grammar, see Dowty (2003: 36-40)

The situation is the same in the type lifting of NP, such as *tom'*, as is shown in (227).

(227) a. Type Raising (syntax): $NP \vdash S/(NP\backslash S)$

b. Type Raising (semantics): $tom'_e \Rightarrow \lambda A_{et}.A(tom')$

c. $(tom'_e, speak'_{et}) \Rightarrow speak'(tom')$

d. $(tom'_e, speak'_{et}) \Rightarrow (\lambda A_{et}.A(tom'))(speak') \Rightarrow_{\beta red} speak'(tom')$

The effect of NP type lifting is nullified in the semantic output in (227d). By way of normalization, we derive the same form, $speak'(tom')$, as in the normal derivation (227c).

Because an important motivation for Dowty's dual analysis is to distinguish certain semantic differences among PPs, Dowty does not derive the complement version expression $speak'_2$ from the adjunct version expression $speak'_1$. He distinguishes the semantics of the PP in *Mary kicked the ball to the fence* on the one hand, and *Mary explained the memo to John* on the other. The former sentence is assigned the VP adjunct logical form, in which *to* denotes the new location at/near which the direct object referent ends up as a result of the action performed on it. The latter is provided with the complement analysis, in which the semantic contribution of *to* may vary with the choice of the main verb (cf. Dowty 2003: 41). My analysis in the next sub-section reduces the logical expressions in the adjunct and the complement analyses to the same normal form, unless scope ambiguity or binding is involved. Thus, my analysis cannot explain the semantic differences that Dowty attributes to the two cases. However, it is not clear to me whether we need to explain such subtle semantic differences as Dowty's at the level of LF interface representation (which I assume underspecifies the fully semantic interpretations that are relevant only at some post-LF level such as Language of Thought). Also, if Dowty insists that the PP expression $to'(mary')$ is exactly the same in the adjunct and the complement analyses, it is not clear if the expression $speak'_2(to'(mary'))$ in (225b) can be provided with some well-defined model theoretic interpretation. More specifically, if we do not define $speak'_2$ in terms of $speak'_1$ (cf. the lifted $speak'_3$ in (226c)), it is not clear whether we can either supply arguments of the appropriate types to the two argument slots of the functor expression $to'(mary')$ of type ((et)(et)) or bind those argument slots by some independently well-motivated operators.

Because of these, I define the semantic shift accompanying the complement reanalysis of a PP adjunct as in normal type lifting operations. Thus, in a case such as (225), the dual analysis does not lead to any difference either in the syntactic

tree structure or in the normalized logical expression. However, when the inner VP to which the PP is adjoined contains one or more internal NP arguments, as in *meet Tom in London* or *introduce Tom to Nancy in London*, then the analysis of the adjunct PP as a complement has to be constrained by my assumption which requires that verbal functors lexically have maximally one syntactic argument at each side. This means that I merge the original internal NP argument(s) with the newly created argument PP by using the connective '•,' creating a complex syntactic argument of the verb. This leads to some scope ambiguity when there is more than one QNP involved in the structure.

In chapter 8, the adjunct structure in (223b) above is used in explaining Wh-extraction in the PP ditransitive construction, such as *What*$_1$ *did you give* t_1 *to Tom*. Though the adjunct structure in (223b) is shared in the two analyses, the relation between this adjunct structure to the other structure is quite different in the two cases. See (311) and (312) and discussion surrounding them in section 8.4.1 of chapter 8.

Now, I show how the adjunct-complement alternation analysis works in the basic cases.

6.3.2 Adjunct structure derivation

I now explain how the adjunct structure leads to the 'inverse' scope reading ('inverse' in the sense that the PF left QNP takes narrow scope relative to the PF right QNP) and the complement structure leads to the surface scope reading.

(228) Tom met a boy in every shop.

I provide the lexical entries.

(229) a. in_v: $< in; ((NP\backslash S)\backslash(NP\backslash S))/NP; \lambda x.\lambda V.\lambda y.(in'_v(x))(V)(y) >$[13]
Type: $in'_v : (e,((et),(et)))$; $V(et)$

b. meet: $< meet; (NP\backslash S)/NP; \lambda x.\lambda y.meet'(x)(y) >$

c. a boy: $< a \cdot boy; (S/NP)\backslash S; \lambda A.some'(boy')(A) >$

d. every shop: $< every \cdot shop; (S/NP)\backslash S; \lambda A.every'(shop')(A) >$

Adjunct Structure

We first apply argument slot raising to the prepositional functor in'_v in (229a). For readability, I abbreviate $NP\backslash S$ as VP.

[13] $in_v \stackrel{\text{def}}{=} \lambda x.\lambda V.\lambda y.[V(y)\ \&\ in'_0(x)(y)]$, which is comparable to in'_1 in (208), but with a different category.

(230) in_v

a. Syntax: $(VP\backslash VP)/NP \vdash (VP\backslash VP)/((S/NP)\backslash S)$

b. Semantics: $\lambda x.\lambda V.\lambda y.(in'_v(x))(V)(y)$
$\Rightarrow \lambda Q.\lambda V.\lambda y.Q(\lambda x.(in'_v(x))(V)(y))$

Because the internal argument of the transitive verb *meet* is a QNP, we apply argument slot raising to the verbal functor as well.

(231) Arguement slot raising of *meet*

a. Syntax: $(NP\backslash S)/NP \Rightarrow (NP\backslash S)/((S/NP)\backslash S)$

b. Semantics: $\lambda u.\lambda v.meet'(u)(v) \Rightarrow \lambda Q1.\lambda v.Q1(\lambda u.meet'(u)(v))$

Using the argument slot raised transitive verb for *meet*, the derivation is straightforward. The presentation is Natural Deduction, and I split the semantics into three parts for space reasons.

(232) a. Syntax:

$$\dfrac{\dfrac{Tom}{NP} \quad \dfrac{\dfrac{\dfrac{meet}{VP/((S/NP)\backslash S)} \quad \dfrac{a \cdot boy}{(S/NP)\backslash S}\,/E}{VP} \quad \dfrac{\dfrac{\dfrac{in}{(VP\backslash VP)/((S/NP)\backslash S)} \quad \dfrac{every \cdot shop}{(S/NP)\backslash S}\,/E}{VP\backslash VP}\,\backslash E}{VP(=NP\backslash S)}\,\backslash E}{S}$$

b. D_1

$$\dfrac{\dfrac{meet}{\lambda Q1.\lambda v.Q1(\lambda u.meet'(u)(v))} \quad \dfrac{a \cdot boy}{\lambda A.some'(boy)(A)}\,/E}{\lambda v.some'(boy')(\lambda u.meet'(u)(v))}$$

c. D_2

$$\dfrac{\dfrac{in}{\lambda Q.\lambda V.\lambda z.Q(\lambda y.(in'_v(y))(V)(z))} \quad \dfrac{every \cdot shop}{\lambda A.every'(shop')(A)}\,/E}{\lambda V.\lambda z.every'(shop')(\lambda y.(in'_v(y))(V)(z))}$$

d.

$$\dfrac{tom' \quad \dfrac{\dfrac{meet \cdot (a \cdot boy)}{\lambda v.some'(boy')(\lambda u.meet'(u)(v))}\,D_1 \quad \dfrac{in \cdot (every \cdot shop)}{\lambda V.\lambda z.every'(shop')(\lambda y.(in'_v(y))(V)(z))}\,D_2}{\lambda z.every'(shop')(\lambda y.(in'_v(y))(\lambda v.some'(boy')(\lambda u.meet'(u)(v)))(z))}\,\backslash E}{every'(shop')(\lambda y.(in'_v(y))(\lambda v.some'(boy')(\lambda u.meet'(u)(v)))(tom'))}\,\backslash E$$

Because the internal argument *a boy* has already been merged with the transitive verb by the time the resultant VP is merged with the adjunct PP, the quantifier in the adjunct takes wide scope over the other QNP inside the VP.

6.3.3 PP complement structure

As in Dowty's analysis, I use the same lexical entry for the preposition, that is, in_v given in (229a) both in the adjunct analysis and in the complement re-analysis. The complement re-analysis is achieved by the following lexical entry for the transitive verb *meet*. For notation, let $meet_1$ represent the normal transitive verb lexical entry of category $(NP\backslash S)/NP$ (that is, $meet_1 = meet$ in (229b)) and define $meet_2$ as in (233).

(233) meet_2

a. syntactic category: $(NP\backslash S)/(NP \bullet ((NP\backslash S)\backslash(NP\backslash S)))$

b. logical expression: $meet'_2 \stackrel{\text{def}}{=} \lambda\sigma.\lambda z.(\pi 2(\sigma))(\lambda y.meet'_1(\pi 1(\sigma))(y))(z)$
Type: $meet'_2 : ((e \times ((et)(et))), (et))$; $\sigma : e \times ((et)(et))$; $meet'_1 : (e(et))$
$\pi 1 : ((e \times ((et)(et)), e)$; $\pi 2 : ((e \times ((et)(et)), ((et)(et)))$

Note that $meet'_2$ is defined in terms of $meet'_1$. If the base grammar system were associative, the category for $meet_2$ would become derivable/provable from the category of $meet_1$, as shown in (234). To show derivability, I use the Gentzen Sequent presentation. For readability, I abbreviate $NP\backslash S$ as VP.

(234)

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{VP \vdash VP \quad NP \vdash NP}{VP/NP, NP \vdash VP}\,/L \qquad VP \vdash VP}{(VP/NP, NP), VP\backslash VP \vdash VP}\,\backslash L}{VP/NP, (NP, VP\backslash VP) \vdash VP}\,Assoc}{VP/NP, (NP \bullet (VP\backslash VP)) \vdash VP}\,\bullet L}{VP/NP \vdash VP/(NP \bullet (VP\backslash VP))}\,/R}{meet_1 \vdash meet_2}\,PF$$

Thus, if the grammar system were associative, reanalysis of the standard transitive verb category $(NP\backslash S)/NP$ as the derived category $(NP\backslash S)/(NP \bullet (NP\backslash S)\backslash(NP\backslash S))$ for $meet_2$ in (233) would be supported by the grammar. However, because the base grammar system that I use is non-associative, the complement re-analysis, $VP/NP \vdash VP/(NP \bullet (VP\backslash VP))$, stays as a special 'lexical' rule in my system.[14]

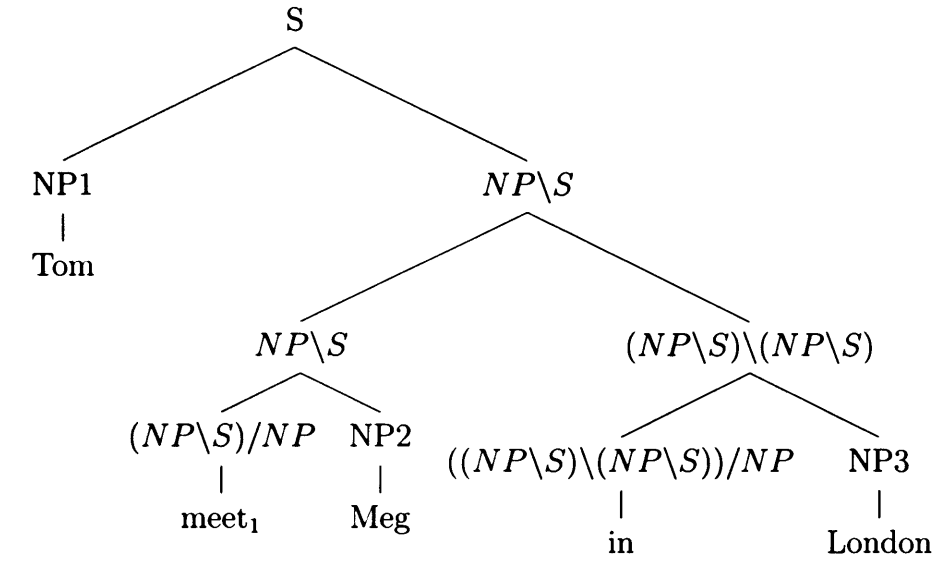
The reason why NL does not support the complement re-analysis when the verbal functor has one or more internal argument slots is that the re-analysis modifies

[14]In contrast to argument slot raising, it is easier to stick to this lexicalist claim, because we can assume that the operation applies only to lexical verbs, as long as we can count verb particle pairs as lexical, such as *take ... off* and *listen to*.

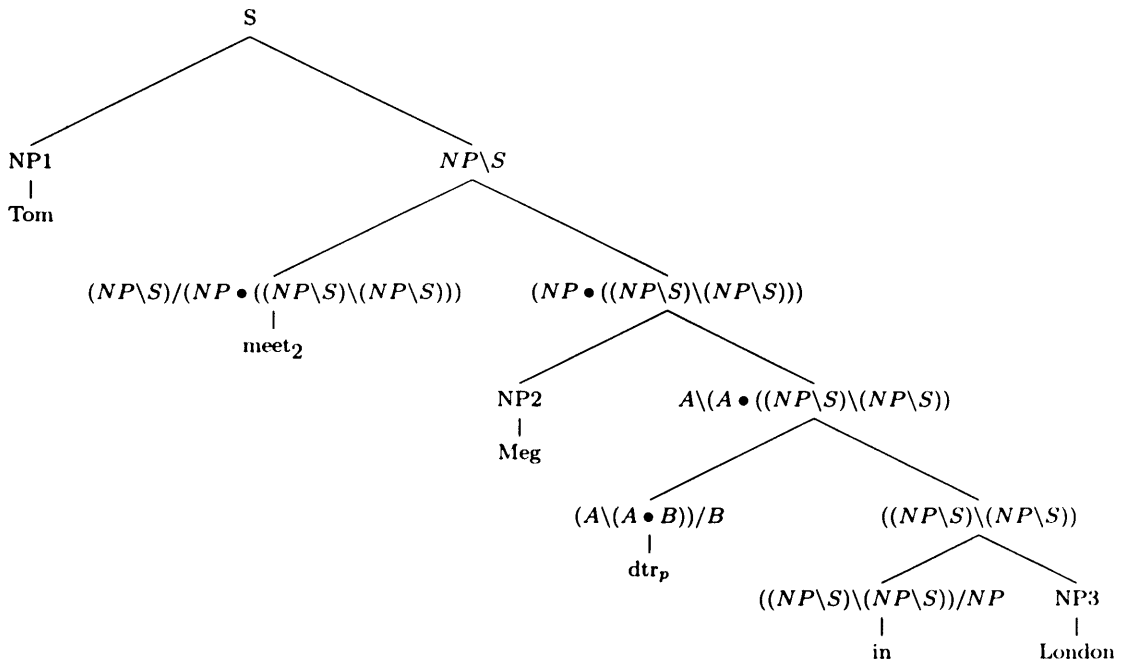
the binary syntactic structure in this case. I show the two structures for a transitive verb *meet*.

Structural difference with *meet*

(235) PP as Adjunct:



(236) PP as Complement:



In the Complement reanalysis, the unpronounced functor dtr_p is inserted to concatenate the two arguments of the verb $meet_2$ into a complex argument. dtr_p is the most general form of *dtr* which we have seen in the DO construction in chapter 5. As is the case with *dtr*, dtr_p preserves the identities of the two items to be merged,

and the left-to-right PF order between the two, in the output. Note that the complement structure in (236) represents the correct c-command relation between NP2 and NP3 for the binding data as in (224). As I show shortly, the two structures correspond to two scope readings when we replace NP2 and NP3 with QNPs.

As for the syntactic validity of the complement structure in (236) above, the coordination test seems to support the structure. That is, *Jack met Fiona in London, and Alison in Paris* is grammatical, and this suggests that *Fiona in London* and *Alison in Paris* are constituents in this sentence.

Though the structural difference as above influences the binding relation and scope readings, when the internal argument of the verb and the argument of the preposition are both NPs, rather than QNPs, the resultant logical forms become equivalent by way of normalization, as is shown below. This is because $meet_2$ is derived from $meet_1$ by way of type lifting and uncurrying of the internal arguments. Thus, just as type raising of the NP *Tom* in (227) does not influence the normalized logical form for *Tom speaks*, the normalized logical forms for the adjunct structure and the complement structure become equivalent in this case. We can see this point by comparing (237) and (238) for *Tom met Meg in London.*

(237) PP adjunct:

$(\lambda x.\lambda V.\lambda y.(in_v(x))(V)(y))(london')((\lambda u.\lambda v.meet'_1(u)(v))(meg'))(tom')$
$\Rightarrow_{\beta red.\times 4}\ (in_v(london'))(\lambda v.meet'_1(meg')(v))(tom')$

(238) PP complement:

$meet'_2(meg' \bullet (\lambda V.\lambda x.(in'_v(london'))(V)(x)))(tom')$
$= (\lambda\sigma.\lambda z.(\pi 2(\sigma))(\lambda y.meet'_1(\pi 1(\sigma))(y))(z)))(meg' \bullet (\lambda V.\lambda x.(in'_v(london'))(V)(x)))(tom')$
$\Rightarrow_{\beta red.,\ \pi 1,\ \pi 2}\ (\lambda z.(\lambda V.\lambda x.(in'_v(london'))(V)(x))(\lambda y.meet'_1(meg')(y))(z))(tom')$
$\Rightarrow_{\beta red.\times 2}\ (\lambda z.(in'_v(london'))(\lambda y.meet'_1(meg')(y))(z))(tom')$
$\Rightarrow_{\beta red.}\ (in'_v(london'))(\lambda y.meet'_1(meg')(y))(tom')$

With $meet_2$ in (233), I show the derivation of (228) (i.e. *Tom met a boy in every shop*) in its PP-as-complement structure.

The internal argument of the transitive verb is a QNP, i.e., the existential QNP in (239).

(239) *a boy*

a. $(S/NP)\backslash S$

b. $\lambda A.some'(boy')(A)$

Next, we type lift the PP expression in (240) (the initial entries are for the PP as a VP adjunct). Type lifting is optional but may freely apply in my analysis.

(240) *in every shop*

a. Category:
$((NP\backslash S)\backslash(NP\backslash S)) \Rightarrow_{TL} (S/((NP\backslash S)\backslash(NP\backslash S)))\backslash S$

b. logical expression:
$\lambda V.\lambda z.every'(shop')(\lambda y.(in'_v(y))(V)(z))$
$\Rightarrow_{TL} \lambda\nu.every'(shop')(\lambda x.\nu(in'_v(x)))$
Types $V : ((et)(et))$; $\nu : (((et)(et)),t)$; $x, y, z : e$

Now we merge (239) with the output of (240) by applying the concatenating functor dtr_p to them in succession. For presentation reasons, I decompose the operation of applying dtr_p to two quantificational arguments (as opposed to two non-quantificational arguments) into two parts. The first part is fully supported by NL, whereas the latter part is not supported by NL, being expressible only in terms of lambda term conversions.[15] Thus, this latter process as shown in (242) below is stated as a non-logical rule that is additional to the base grammar NL[16] The presentation of the proof is Natural Deduction.

(241) *a boy in every shop*

a. Syntax

$$\dfrac{\dfrac{a \cdot boy}{(S/NP)\backslash S} \quad \dfrac{\dfrac{dtr_p}{(A\backslash(B \bullet C))/C} \quad \dfrac{in \cdot (every \cdot shop)}{(S/((NP\backslash S)\backslash(NP\backslash S)))\backslash S}\,/E}{B\backslash(B \bullet ((S/((NP\backslash S)\backslash(NP\backslash S)))\backslash S))}\,\backslash E}{((S/NP)\backslash S) \bullet ((S/((NP\backslash S)\backslash(NP\backslash S)))\backslash S)}$$

b. Semantics

$$\dfrac{\dfrac{a \cdot boy}{\lambda A.some'(boy')(A)} \quad \dfrac{\dfrac{dtr_p}{\lambda X.\lambda Y.Y \bullet X} \quad \dfrac{in \cdot (every \cdot shop)}{\lambda\nu.every'(shop')(\lambda x.\nu(in'_v(x)))}\,/E}{\lambda Y.Y \bullet (\lambda\nu.every'(shop')(\lambda x.\nu(in'_v(x))))}\,\backslash E}{\lambda A.some'(boy')(A) \bullet (\lambda\nu.every'(shop')(\lambda x.\nu(in'_v(x))))}$$

[15]The operation in (242b) below is definable in terms of the input lambda term and the output lambda term, which are both well-formed, but I hesitate to argue that the operation is not problematic in the semantics, because I have not fully identified the syntactic rules that underpin the typed lambda calculus that is used in the semantics of my analysis. I leave further investigation of the formal properties of the operation in (242) for future research.

[16]Again, use of a non-logical axiom might be problematic from a logical viewpoint.

Types, $dtr_p : (a, (b, (b \times a)))$, for whatever types, a, b.

Now, the output of (241) is only a juxtaposition of the two arguments left to right. Thus, the two quantifiers stay as separate in the output. However, by hypothesis, dtr_p needs to turn the two objects into some syntactic object that counts as one argument of the verb. In that sense, the output in (241b) is still incomplete, and there is more for dtr_p to do when its arguments are quantificational. This final part of the job of the functor dtr_p is to create a complex quantificational object which linguistically counts as one argument of the verb. This latter part of the operation of dtr_p is spelt out in (242). It uses the syntactic and semantic outputs in (241) as inputs. Just as *dtr* has preserved the scope relation between the two QNPs as in the surface order, dtr_p preserves the surface scope reading as in the linear PF string.

(242) *a boy in every shop*

a. Syntax:
$((S/NP)\backslash S) \bullet ((S/((NP\backslash S)\backslash(NP\backslash S)))\backslash S)$
$\Rightarrow (S/(NP \bullet ((NP\backslash S)\backslash(NP\backslash S))))\backslash S$

b. Semantics:
$\lambda A.some'(boy')(A) \bullet (\lambda\nu.every'(shop')(\lambda x.\nu(in'_v(x))))$
$\Rightarrow \lambda V^p.some'(boy')(\lambda y.every'(shop')(\lambda x.V^p(y \bullet (in'_v(x)))))$
Type, $V^p : ((e \times ((et)(et))), t)$

As with *dtr* in the DO construction, this final bit of the operation of dtr_p is not supported in the non-associative system NL. The operation is motivated by still informal assumption that the two internal arguments of the $meet_2$ must go through some transformation so that the two of them together can count as one argument of this verb. How to formulate this requirement in the theory is not clear yet.

Now, in order to take the output of (242) as a complex QNP argument, the functor $meet_2$ in (233) has to undergo argument slot raising with regard to the internal argument slot. In (243), I name the output of this ASR process $meet_3$, where the semantic output is named $meet'_3$ accordingly.

(243) $meet_3$

a. Syntax
$(NP\backslash S)/(NP \bullet ((NP\backslash S)\backslash(NP\backslash S)))$
$\Rightarrow (NP\backslash S)/((S/(NP \bullet ((NP\backslash S)\backslash(NP\backslash S))))\backslash S)$

b. Semantics

$\lambda\sigma.\lambda z.(\pi 2(\sigma))(\lambda v.meet'_1(\pi 1(\sigma))(v))(z)$

$\Rightarrow \quad \lambda Q^2.\lambda z.Q^2(\lambda\sigma.meet_2(\sigma)(z))$

$= \lambda Q^2.\lambda z.Q^2(\lambda\sigma.(\pi 2(\sigma))(\lambda v.meet'_1(\pi 1(\sigma))(v))(z))$

$= meet'_3$

Types: $Q^2 : ((e \times ((et)(et))), t)$; $\sigma : (e \times ((et)(et)))$

Merging the outputs of (242) and (243), the rest of the derivation leads to the surface scope reading. First, up to the highest VP level.

(244) a. Syntax:

$$\frac{\dfrac{meet}{(NP\backslash S)/((S/(NP \bullet ((NP\backslash S)\backslash(NP\backslash S))))\backslash S)} \quad \dfrac{a \cdot boy \cdot in \cdot every \cdot shop}{(S/(NP \bullet ((NP\backslash S)\backslash(NP\backslash S))))\backslash S}\ /E}{NP\backslash S}\ /E$$

b. Semantics:

$$\dfrac{\dfrac{\dfrac{\dfrac{meet}{meet'_3} \quad \dfrac{a \cdot boy \cdot in \cdot every \cdot shop}{\lambda X.some'(boy')(\lambda y.every'(shop')(\lambda x.X(y \bullet (in'_v(x)))))}}{\lambda z.((\lambda X.some'(boy')(\lambda y.every'(shop')(\lambda x.X(y \bullet (in'_v(x))))))(\lambda\sigma.(\pi 2(\sigma))(\lambda v.meet'_1(\pi 1(\sigma))(v))(z)))}\ /E}{\lambda z.(some'(boy')(\lambda y.every'(shop')(\lambda x.(\lambda\sigma.(\pi 2(\sigma))(\lambda v.meet'_1(\pi 1(\sigma))(v))(z))(y \bullet (in'_v(x))))))}\ \beta}{\lambda z.(some'(boy')(\lambda y.every'(shop')(\lambda x.\pi 2(y \bullet (in'_v(x)))(\lambda v.meet'_1(\pi 1(y \bullet (in'_v(x))))(v))(z))}\ \beta}{\lambda z.(some'(boy')(\lambda y.every'(shop')(\lambda x.(in'_v(x))(\lambda v.meet'_1(y)(v))(z))}\ \pi_1, \pi_2$$

where $meet'_3 = \lambda Q^2.\lambda z.Q^2(\lambda\sigma(\pi 2(\sigma))(\lambda v.meet'_1(\pi 1(\sigma))(v))(z))$

At the bottom line, we can see that the scope reading is fixed as $a/some > every$. Merging the subject *Tom* derives the sentential logical form.

(245) a. Syntax

$$\frac{\dfrac{Tom}{NP} \quad \dfrac{meet \cdot a \cdot boy \cdot in \cdot every \cdot shop}{NP\backslash S}}{S}\ \backslash E$$

b. Semantics

$$\frac{\dfrac{Tom}{tom'} \quad \dfrac{meet \cdot a \cdot boy \cdot in \cdot every \cdot shop}{\lambda z.(some'(boy')(\lambda y.every'(shop')(\lambda x.(in'_v(x))(\lambda v.meet'_1(y)(v))(z))}}{some'(boy')(\lambda y.every'(shop')(\lambda x.(in'_v(x))(\lambda v.meet'_1(y)(v))(tom')}\ \backslash E$$

In this section, I have discussed cases in which QNPs appear in PPs which are normally analysed as adjuncts of VPs. The adjunct structure leads to the inverse scope reading. To generate the surface scope reading, I have adopted Dowty's complement reanalysis of PPs with modification.

6.4 Conclusion

This chapter has discussed some of the cases in which QNPs appear in prepositional phrases. The first case, such as *Jack put an apple into every box*, is identified as the PP ditransitive construction, and thus, we can use the same strategy as we have used in chapter 5 to derive the scope ambiguity. The QNP in QNP case, as in *A student from every school talked*, can be dealt with by applying value raising and argument slot raising to the prepositional functor *in*. The third case is the most difficult one, namely, a case as in *I met a boy in every shop*, which is also scopally ambiguous. The inverse scope is generated straightforwardly by analysing the PP *in every shop* as an adjunct of the VP. To generate the surface scope, I have adopted Dowty's reanalysis of the PP as an extra argument of the verb. Together with the assumption that a natural language functor can only have one syntactic argument at each side, I have managed to generate the surface scope reading, $a > every$. However, just as was the case with the ditransitive constructions, the operation of turning two separate quantificational arguments into one complex quantificational argument is not fully supported by NL, and this raises the question about when we can rely on such a special operation and when we cannot.

The next chapter discusses QNP scope in infinitival constructions.

Chapter 7

QNP scope in infinitival constructions

7.1 Introduction

According to the informal calculation of QNP scope which I sketched in chapter 2, the scope of a QNP stays inside the first S category (containing the QNP) that we derive in the syntactic derivation. Though the analysis that uses categorial argument slot raising is meant to realize this informal idea in Type Logical Grammar, it has turned out to be too restrictive in locality constraints when we deal with control/raising/auxiliary constructions, as we briefly saw in chapter 2. In the previous chapters, we saw exceptional cases in which the scope of a QNP can be extended in terms of value raising applied to the local functor of the QNP (cf. chapter 6) or, when the final output of the local functor is taken in as an argument of another functor, in terms of successive applications of argument slot raising to these two functors (such as two objects together taking wide scope over the subject QNP in the ditransitive constructions in chapter 5). But neither of the two methods can extend QNP scope in a desired way for control/raising/auxiliary constructions.

Consider the infinitival construction in (246a) as an example. If we treat *try* as of category $(NP\backslash S)/(NP\backslash S)$, and *(to) review* as of category $(NP\backslash S)/NP$, then we cannot derive the category $(NP\backslash S)/NP$ for *try to review* in the non-associative system NL. Thus, we cannot use the argument slot raising to switch scope between the indefinite subject and the universal QNP in (246a). Unfortunately, (246a) is scopally ambiguous, just as (246b) is.

(246) a. A student *tried to review* every paper. $a > every$; $every > a$

b. A student *reviewed* every paper. $a > every$; $every > a$

In this chapter, I discuss how we can treat *try to review* in (246a) as a complex functor of category $(NP\backslash S)/NP$ so that we can apply argument slot raising to it. I extend the analysis to some of the object control and raising constructions. Linguistically, for all of these phenomena, NPs are merged as NPs and are interpreted as type e arguments, rather than being merged as higher order categories and interpreted as higher order operators, as is the case with Wh-expressions. To capture this observation, I adopt a way of defining a structural association rule in a way that is sensitive to the merge mode between (verbal) functors and their NP arguments, as we see later.

Also, the merge mode specification is provided in such a way that we can form a complex predicate only when the lexical category of the higher verb selects VP (i.e. $(NP\backslash S)$ in categorial formulas), rather than S, as shown in (247)~(248) (see Carpenter (1997), chapter 11, for such category/type assignments to control/raising verbs).

(247) Categories for verbs that select VP, rather than S.

a. try: $< try;\ (NP\backslash_j S)/_i(NP\backslash_j S);\ try' >$
where $try' \stackrel{\text{def}}{=} \lambda V.\lambda x.try'_0(V(x))(x)$

b. persuade: $< persuade;\ ((NP\backslash_j S)/_i(NP\backslash_j S))/_j NP;\ persuade' >$
where $persuade' \stackrel{\text{def}}{=} \lambda x.\lambda V.\lambda y.persuade'_0(x)(V(x))(y)$
Cf. Carpenter (1997: 438)

c. must: $< must;\ (NP_j\backslash S)/_i(NP\backslash_j S);\ must' >$
$must' = \lambda V.\lambda x.must'(V)(x)$

The details of the semantics are explained later, but the main point is that the saturation of the external NP argument of the verbal head of the selected infinitival VP (e.g. *review* in *try to review every paper*) is postponed because the higher verbs such as *try* postpone the saturation of this argument slot by identifying this argument slot with one of their own NP argument slots (by binding the two argument slots by the same lambda operator, as in $\lambda x.try'_0(V(x))(x)$, for example). Thus, given these category assignments to the control verbs, the category S is not derived at the level of the embedded infinitives. The derivation is analogous to the merge of an auxiliary verb with the VP that it selects, as we can see in the entry for *must* in (247c), where *must* percolates the external argument slot of the selected VP.[1]

[1]In (247c), we could define *must'* as $must' \stackrel{\text{def}}{=} \lambda V.\lambda x.must'_1(V(x))$ as I do with subject raising verbs in (248a) below. But in this thesis, I represent the auxiliary status of such verbs in a direct

Treating control verbs such as *try* and *hate* in the same way as we treat auxiliary verbs (i.e. treating these as VP selecting functors) is less controversial because these verbs cannot select S, as shown in **Tom tried [$_S$ for these students to take his course]* or **Jack persuaded Meg [$_S$ for the students to meet their tutors]*.[2]

In contrast, assigning the same syntactic category to subject/object raising verbs, as shown in (248) may need more justification.

(248) a. seem$_2$: $< seem;\ (NP\backslash_j S)/_i(NP\backslash_j S);\ seem'_2 >$
where, $seem'_2 \stackrel{\text{def}}{=} \lambda V.\lambda x.seem'_1(V(x))$ Cf. Carpenter (1997: 439)

b. expect$_2$: expect: $< expect;\ ((NP\backslash_j S)/_i(NP\backslash_j S))/_j NP;\ expect_2' >$
where $expect'_2 :\stackrel{\text{def}}{=} \lambda x.\lambda V.\lambda y.expect'_1(V(x))(y)$
Types: $seem'_2 : (et)(et);\ seem_1' : (tt);$
$expect'_2 : e,((et),(et));\ expect'_1 : t(et)$

c. Jack seems to review the paper.

d. I expect Jack to review the paper.

With the categorial assignments in (248), the verbs *seem* and *expect* select VP as an argument, just as control verbs do. However, in (249b) and (249d), these verbs select a clausal expression S in which all the NP arguments of the embedded verb (that is, *review*) have been saturated by overt NPs. Also, for *seem*, we can insert an expletive *it* as its syntactic subject. Based on the assumption that expletives are not represented in the semantic terms, some might argue that the semantic expression for *seem* does not have an external type e argument slot. With *expect*, it is less clear whether the semantic functor for *expect* has an internal NP argument in its most natural meaning, but there is some contrast between *expect* and *persuade* with regard to the semantically essential status of the internal NP argument. Some might argue that then, given the semantically more basic statuses of the entries as in (249), we should assign the entries as in (249a) and (249c) to *seem* and *expect*

manner. Unlike raising verbs, auxiliary verbs on their own cannot select sentential expressions as their complements.

[2]This thesis does not discuss the verbs that select infinitive clauses with the complementizer *for*, as in *Jack aimed [$_S$ for these students to take his course]* or *Meg arranged [$_S$ for the students to meet their tutors]*. When these verbs select the clause headed by *for* (i.e. when they select S), the proposed analysis expects scope switch to be blocked as the minimal S category that contains the universal QNP is derived before we reach the matrix indefinite, as shown in *A student arranged [$_S$ for kids to come to every lecture]* ($a > every$; $*every > a$). Note that in this case, the minimal S category does not correspond to the minimal 'tensed-clause,' according to the intuitive interpretation of a 'tensed-clause.' But I do not investigate these sentences any further.

(the mode index 's' in the category '$S/_sS'$ is mnemonic for 'sentences').[3]

(249) a. seem$_1$: $<$ *seem*; $S/_s S$; *seem*$'_1$ $>$

b. It seems that Jack reviews the paper.

c. expect$_1$:$<$ *expect*; $(NP\backslash_j S)/_s S$; *expect*$'_1$ $>$

d. I expect that Jack will review the paper.

On the other hand, the semantically basic status of the entries in (249) does not mean that we do not need the categorial entries as in (248). Unless we postulate a phonologically null item such as PRO in Minimalism which has category NP and then have this PF-null item saturate the external argument slot of the embedded infinitival verb, as in *Jack seems [$_S$ PRO to review the paper]*, deriving the category S as a result, we need the entries in (248) so that the categorial derivations converge for the infinitival constructions. In that case, for those infinitival sentences in (248), we do not derive the category S until we reach the top of the whole sentence. Syntactically speaking, then, the verbs *seem* and *expect* still select VPs, rather than S as an argument. Whether those categories are categories derived from the basic ones in (249) or not is less important for our current concerns.

Based on these considerations, I assume that for control/raising/auxiliary constructions, the higher verb selects an expression in which the external NP argument slot of the main verb has not yet been saturated, that is, the verb selects VP (= $NP\backslash S$), rather than S.[4] To limit the structural rule application to forming a complex predicate for these infinitival constructions, as opposed to the cases when the higher verb selects a complete clause in which all the NP arguments have been saturated by overt NPs, I use a structural association rule licensed by the pair of the mode which merges the above mentioned higher verb with its VP argument and the mode which merges the component functors of the complex predicate with their NP arguments. In this way, we can limit the formation of a complex predicate within the minimal S expression that contains the sub-component expressions. This minimal S category corresponds to each minimal TP as in Minimalism.[5] Thus, together with the assumption that argument slot raising may apply to complex

[3]Also, technically speaking, it is easier to assume S/S as the basic category for *seem* and derive $(NP\backslash S)/(NP\backslash S)$ from there, using some kind of Geach rule as in Jacobson (1999).

[4]I am not arguing that (obligatory) control/raising/auxiliary verbs are syntactically the same. All that I have argued for is assigning a particular lexical entry to these verbs so that they will syntactically select VPs, rather than S, when they select infinitival clauses such as *to review each paper*.

[5]Though this correspondence is not strict. What is important in the theory is the minimal expression of category S that contains the QNP in question, where this expression 'normally' corresponds to the minimal tensed clause. See footnote 2 in this chapter.

verbal predicates that the grammar generates, it leads to the tensed clause locality constraints on QNP scope, as desired. After discussing QNP scope in control and raising constructions, I briefly discuss the passive, because the basic way of introducing structural rules for control/raising/auxiliary constructions can be used to deal with the passive as well. I briefly show that passivizing the two kinds of ditransitive constructions does not influence the number of scope readings between the two object QNPs either for the double object or the prepositional ditransitive constructions. This is because, in contrast to control/raising/auxiliary constructions, use of structural association for the passive does not lead to the formation of a complex predicate to which argument slot raising may newly apply. Though passivization allows us to merge an internal argument of the verbal functor in the syntactic position for the subject, the (Q)NP argument in question still saturates the initial internal argument slot of the verbal functor. The QNP argument does not end up saturating the external argument slot of the verbal functor via passivization (i.e. the external argument slot is either existentially closed, or saturated by an argument introduced by a *by*-phrase, as in *The issue was discussed by Bill*). In this analysis which I explain in section 7.4, the number of scope readings between the two QNPs in the passivized double object constructions stays the same as the number of scope readings between the two object QNPs in the active ditransitive sentences, if we ignore the potential presence of another QNP as the external argument.

Section 7.2 discusses QNP scope in typical control constructions. Section 7.3 deals with QNP scope in raising constructions. Section 7.4 discusses how we can use a similar way of introducing structural rules to deal with the passive.

Because the suggested way of introducing structural rules does not apply association to structures beyond the local S (or TP in Minimalism), we need a fundamentally different way of introducing structural association/permutation rules for linguistic phenomena that do cross the local S. Section 7.5 informally shows how we can introduce structural rules in different ways for Wh-extraction, which is subject to different locality constraints than QNP scope. This provides a transition to the contents of the next chapter. Section 7.6 provides concluding remarks.

7.2 Mixed-mode association in control constructions.

7.2.1 Basics

As we have seen in chapter 3, Multi-Modal Type Logical Grammar can manipulate different modes of binary merge. Thus, we can define structural rules across different modes of combination, as in (250) (cf. Moortgat (1996), Moortgat (1997)).

(250) Mixed-mode association for control/raising/auxiliary constructions.
For all $A, B, C \in F$ and for all $X \in S$

$$\frac{A \circ_i (X \circ_j B) \vdash C}{(A \circ_i X) \circ_j B \vdash C} A_{i,j}$$

As I have explained in chapter 3, we can specify the merge modes in categorial formulas assigned to functor categories, as in $(NP\backslash_j S)/_j NP$ for transitive verbs, or $(NP\backslash_j S)/_i(NP\backslash_j S)$ for subject control/raising/auxiliary verbs. In (250), $\circ_i$ linguistically represents the merge of a control/raising/auxiliary verb with a VP headed by a lexical verb. $\circ_j$ on the other hand represents the merge of a verb with its NP argument(s). The structural association rule in (250) means that we can merge a control verb with a transitive/ditransitive verb directly before we merge the transitive/ditransitive verb with its internal NP argument(s). The linguistic generalization here is that those verbs which select VP, such as control verbs, raising verbs and (modal) auxiliary verbs, do not care whether the head of the VP has already been merged with its internal NP argument(s) or not. According to the standard lexical category assignment to these verbs, that is, $(NP\backslash S)\backslash(NP\backslash S)$. these verbs 'inherit' only the external NP argument slot of the infinitival verb. In order to switch scope for (246a), we have to percolate the internal NP argument slot of *review* as well, and we do this by introducing structural association with the pair of merge modes, i and j.

Some might argue that rather than introducing a structural rule under the control of mode specification, we could percolate internal argument slots of infinitive verbs by way of polymorphic lexical assignments,[6] as in (251) (where (251a) is the basic assignments which I will use).

[6]Many thanks to Michael Moortgat for reminding me of this possibility.

(251) Polymorphic assignments to *try* (a hypothetical alternative analysis).

a. $(NP\backslash S)/(NP\backslash S)$; $\lambda V_{et}.\lambda z.try'(V)(z)$

b. $((NP\backslash S)/NP)/((NP\backslash S)/NP)$; $\lambda R^2.\lambda y.\lambda z.try'(R^2(y))(z)$

c. $((NP\backslash S)/(NP \bullet NP))/((NP\backslash S)/(NP \bullet NP))$;
$\lambda R^3.\lambda \alpha.\lambda z.try'(R^3(\alpha))(z)$

Variable types, $R^2 : (e(et))$; $R^3 : ((e \times e),(et))$; $\alpha : (e \times e)$; $y, z : e$

The entry in (251a) would be used for sentences such as (252a), (251b) would be for (252b) and (251c) would be for (252c), where the expressions in the square brackets in (252b) and (252c) could be analyzed as complex predicates by using these entries.

(252) a. A student tried to run away.

b. A student [*tried to review*] every paper.

c. A student [*tried to give*] Meg every book.

This alternative analysis has some merit in that no structural rules would then be involved in explaining QNP scope, drawing an even clearer line in the theory between QNP scope and Wh-extraction than in my analysis. That is, in this alternative analysis, they might argue that structural rules are involved only in truly long distance phenomena that may cross a tensed clause, such as Wh-movement.

However, this alternative analysis would lead to rather awkward lexical assignments to deal with object control sentences.

(253) Jack persuaded a student to review every paper. *a* > *every*; *every* > *a*

(253) has two scope readings that we need to explain. The problem is that the indefinite, *a student*, has to be an internal argument of the object-control verb *persuade*, as we can see by turning the sentence into the passive, *A student was persuaded to review every paper.*[7] Thus, we cannot merge the embedded VP directly with *a student* by applying argument slot raising to the embedded verb *review* with regard to its external argument slot. Remember from chapter 1 that if we literally merge the indefinite *a student* twice in (253), first as the external argument of *review* and secondly as the internal argument of the control verb *persuade*, we would have a compositionality problem, because (253) does not mean *Jack persuaded a student*

[7]Or from a semantic viewpoint, the event denoted by the predicate *persuade* involves an internal type e argument, i.e., the individual who is persuaded. Thus, we can paraphrase (253) with *Jack persuaded a student that he should review every paper.*

that a student should review every paper. If we applied argument slot raising to *review* with regard to its external argument slot and to *persuade* with regard to its internal NP argument slot and somehow managed to merge one QNP *a student* twice to saturate these two type lifted argument slots,[8] then we would generate the wrong reading above that (253) does not have.

In order to use the same lexical mechanism that is used in (251) to percolate the argument slots of the infinitive verb *(to) review* for (253), some might merge *persuade* and *(to) review* first, generating a PF discontinuous functor *persuade* $\cdots$ *to review.* They could then merge the indefinite *a student* in (253) with this complex functor only once, avoiding the above mentioned problem in the semantics. A straightforward way of deriving this complex predicate would be to use wrapping connectives. But as I pointed out several times, rather than defining a new set of logical connectives that help us deal with linguistic data that the basic connectives $\backslash, /, \bullet$ cannot deal with, a methodological preference in this thesis is to take the logical connectives that respect PF linear adjacency as the basic connectives and then add structural rules explicitly (under modal control) to derive the desired result.

As we see later, if we apply the structural association rule $A_{i,j}$ in (250), then we can deal with object control sentences as above without using either wrapping connectives or additional structural rules on top of $A_{i,j}$. In terms of the uniformity of the analysis, I use the structural rule $A_{i,j}$ both for subject control and object control constructions. As we see later, we can use the same mechanism to deal with raising constructions (and its application to modal auxiliaries is straightforward as well). Thus, we can see the basic algorithm that we have adopted here as a way of introducing certain degrees of variation to the PF positions where NP arguments are merged as NP arguments with their functors.[9]

7.2.2 Subject control constructions

In this subsection, I apply the structural association rule in (250) to the subject control construction first. The lexical entries of the two functors are as in (254a)$\sim$(254b).

[8]To merge one QNP twice in NL, which is a variant of linear logic, we would have to specify how many times the item is merged by way of multisets.

[9]As I mentioned in chapter 3, Versmissen (1996) uses a similar mechanism to explain verb raising and cross serial dependency phenomena in Dutch which also involves formation of complex predicates. Versmissen has introduced permutation in a similar way, which we might define with mode specification as well, if we take into account NP Argument scrambling phenomena in the future.

(254) *A student tried to review every paper.* Scope: $a > every; every > a$

a. try:$< try; (NP\backslash_j S)/_i(NP\backslash_j S); \lambda V.\lambda z.try'(V)(z) >$
where, $try' \stackrel{\text{def}}{=} \lambda V.\lambda z.try'_{\text{o}}(V(z))(z)$

b. review:$< review; (NP\backslash_j S)/_j NP; \lambda x.\lambda y.review'(x)(y) >$
Types. $try' : (et)(et)$; $try'_{\text{o}} : t(et)$; $review' : e(et)$; $V : (et)$; $x, y, z : e$

Now we apply (250) to the sentence in (254). Let VP represent $NP\backslash_j S$. In (255) and after, I omit the merge of the infinitival *to* with the embedded VP, ignoring its contributions for presentation purposes.[10]

(255) a. Syntactic proof:

$$\dfrac{\dfrac{VP/_iVP \circ_i (VP/_jNP \circ_j NP) \vdash VP}{(VP/_iVP \circ_i VP/_jNP) \circ_j NP \vdash VP}\, A_{i,j}}{VP/_iVP \circ_i VP/_jNP \vdash VP/_jNP}\, /_jR$$

b. Derived LF sequent (omitting the derivation steps)
$\lambda V.\lambda z.try'(V)(z) \circ_i \lambda x.\lambda y.review'(x)(y) \vdash \lambda x.\lambda z.try'(review'(x))(z)$

c. Derived PF sequent:
$try \circ_i (to \cdot review) \vdash try \cdot (to \cdot review)$

As we have pointed out, the indices have to match between the binary categorial connectives such as '$/_j$' and the structural connective '$\circ_j$.' Thus, '$R/_j$' at the last step in (255a) can only introduce '$VP/_jNP$,' not '$VP/_iNP$,' for example, to the right of the turnstile in the bottom sequent.

After deriving the functor *try to review* of category $(NP\backslash_j S)/_j NP$, we can apply argument slot raising to this complex functor and generate the two scope readings.

The analysis is basically the same when the embedded verb is a ditransitive verb as in (256). (257) is the lexical entry for *show*. The mode index 'c' in '$\bullet_c$' is mnemonic for 'combining,' but I abstract away from the exact identity of this merge mode in the thesis.

(256) An enthusiastic estate agent tried to show me every house.

$a > every$; $every > a$

[10]If we assign the lexical entry '$< to; VP/_iVP; \lambda V_{et}.\lambda x_e.V(x) >$' to *to*, and merge *to* with the embedded VP by using the mode '$\circ_i$,' then, the structural rule in (250) can be applied in more or less the same way as we do in (255) to derive *try to review* as a complex functor of category $VP/_jNP$.

(257) show: $< show; (NP\backslash_j S)/_j(NP \bullet_c NP); \lambda\alpha.\lambda y.show'(\pi 1(\alpha))(\pi 2(\alpha))(y) >$

(258) shows the derivation.

(258) a. Syntactic proof:

$$\dfrac{\dfrac{VP/_iVP \circ_i (VP/_j(NP \bullet_c NP) \circ_j NP \bullet_c NP) \vdash VP}{(VP/_iVP \circ_i VP/_j(NP \bullet_c NP)) \circ_j NP \bullet_c NP \vdash VP}\, A_{i,j}}{VP/_iVP \circ_i VP/_j(NP \bullet_c NP) \vdash VP/_j(NP \bullet_c NP)}\, /_jR$$

b. Derived LF sequent (omitting the derivation steps)

$\lambda V.\lambda z.try'(V)(z) \circ_i \lambda\alpha.\lambda y.show'(\pi 1(\alpha))(\pi 2(\alpha))(y)$

$\vdash \lambda\alpha.\lambda z.try'(show'(\pi 1(\alpha))(\pi 2(\alpha)))(z)$

Types, $\alpha : (e \times e)$; $V : (et)$; $\pi 1, \pi 2 : ((e \times e), e)$

c. Derived PF sequent:

$try \circ_i (to \cdot show) \vdash try \cdot ((to) \cdot show)$

We can apply argument slot raising to the derived complex functor *try to show* of category $(NP\backslash_j S)/_j(NP \bullet_c NP)$, with regard to its $(NP \bullet_c NP)$ argument slot and the NP argument slot in two orders and switch scope between the two double object NP arguments together and the matrix subject, as we have seen in chapter 4.

7.2.3 Object control constructions

(259) a. John persuaded a student to review every paper.

$a > every$; $every > a$

b. persuade:

$< persuade; ((NP\backslash_j S)/_i(NP2\backslash_j S))/_j NP2; persuade' >$

where, $persuade' \stackrel{\text{def}}{=} \lambda x.\lambda V.\lambda y.persuade'_o(x)(V(x))(y)$

Types, $persuade' : (e, ((et), (et)))$; $persuade'_o : (e(t(et)))$;

$V : et$; $x, y : e$

Informally, the logical form $persuade'(x)(V)(y)$ in (259b) means that y persuades x to do V. With the lexical entries in (259b), the indefinite *a student* in (259a) is merged only once, as the internal NP argument of *persuade*. I assume that the external type e argument slot of the infinitive verb *review* is identified as the internal type e argument slot of *persuade* because of the lexical meaning of *persuade*, as is

explicitly represented in the definition using the functor $persuade'_0$. In the category in (259b), the numbering in 'NP2' is used only for presentational convenience in order to show which the control verb *persuade* identifies the two NPs with the number two in the semantics.

We first apply argument slot raising to the internal NP argument slot of *persuade*.

(260) a. Syntax:

$((NP1\backslash_j S)/_i(NP2\backslash_j S))/_j NP2 \Rightarrow ((NP1\backslash_j S)/_i(NP2\backslash_j S))/_j((S/_j NP2)\backslash_j S)$

b. Semantics:

$\lambda x.\lambda V.\lambda y.persuade'(x)(V)(y) \Rightarrow \lambda Q1.\lambda V.\lambda x.Q1.(\lambda x.persuade'(x)(V)(y))$

In this analysis, the indefinite *a student* in (259a) does not directly saturate the external NP argument slot of the embedded verb *review*. Rather, the indefinite is taken in as an argument of the control verb *persuade* by way of argument slot raising applied to the control verb. If we applied argument slot raising to the embedded verb *review* in (259a), then the derivation would not converge because then the infinitival VP *(to) review every paper* would have the category $(S/_j(NP\backslash_j S))\backslash_j S$, which is not the argument category that the control verb *persuade* of category $((NP\backslash_j S)/_i(NP\backslash_j S))/_j NP$ selects as its second argument (i.e. the second argument of *persuade* is $NP\backslash_j S$). Because of this, the external argument slot of *review* cannot be raised to the QNP category in (259a).

When we raise the type of the internal NP argument slot of *persuade*, we can do this at two different stages of the syntactic derivation, leading to the scope ambiguity in question. To derive the surface scope reading for (259a), we apply argument slot raising to *review* with regard to its internal argument slot right away, without applying the structural association rule $A_{i,j}$. We only show the final sequent to prove in Gentzen sequent presentation, as in (261a), together with the corresponding LF and PF sequents in (261b) and in (261c). '$(S/_j NP)\backslash_j S$' is abbreviated as 'QNP.' Again, the contribution of the infinitival *to* is ignored for presentation reasons. The proof of the sequent in (261a) will be straightforward, using $/L$ three times. The semantics in (261b) includes some intermediate steps between the antecedent and the succedent, to show some of the term conversion steps.

(261) a. Syntax:

$(((NP\backslash_j S)/_i(NP\backslash_j S))/_j QNP2 \circ_j QNP2) \circ_i ((NP2\backslash S)/QNP3 \circ_j QNP3)$

$\vdash NP\backslash S$

b. Semantics:

$(\lambda Q1.\lambda V.\lambda y.Q1.(\lambda x.persuade'(x)(V)(y)) \circ_j \lambda B_{et}.some'(student')(B)) \circ_i$
$(\lambda Q2.\lambda v.Q2(\lambda z.review'(z)(v)) \circ_j \lambda B_{et}.every'(paper')(B))$
$\vdash \lambda V.\lambda y.some'(student')(\lambda x.persuade'(x)(V)(y)) \circ_i$
$(\lambda v.every'(paper')(\lambda z.review'(z)(v))$
$\vdash \lambda y.some'(st')(\lambda x.persuade'(x)(\lambda v.every'(pap')(\lambda z.review'(z)(v)))(y))$

c. Derived PF string

$(persuade \circ_j a \cdot student) \circ_i (to \cdot (review \circ_j every \cdot paper))$
$\vdash (persuade \cdot (a \cdot student)) \cdot (to \cdot (review \cdot (every \cdot paper)))$

In the succedent of the LF sequent in (261b), the lexical meaning of *persuade* identifies the external type e argument slot v of *review* with the internal type e argument slot x of *persuade* and then has both of these argument slots bound by the existential quantifier in *a student*.

Given the output in (261), we merge the subject *John* and derive the surface scope reading $a >$ *every* for *John persuaded a student to review every paper* in (259a).

(262) a. LF:

$some'(student')(\lambda x.persuade'(x)(\lambda v.every'(paper')(\lambda z.review'(z)(v)))(john'))$

b. PF:

$John \cdot ((persuade \cdot (a \cdot student)) \cdot (to \cdot (review \cdot (every \cdot paper))))$

To derive the inverse scope, *every* $>$ *a*, we merge the embedded verb *review* without applying argument slot raising. Also, we percolate the internal argument slot of the verb *review* by using the modally controlled association rule $A_{i,j}$ in (250), as is shown in (263).

(263) *persuade a student to review*

a. Syntax:

$$\dfrac{\dfrac{\dfrac{((((NP\backslash_j S)/_i(NP2\backslash_j S))/_j QNP2 \circ_j QNP2) \circ_i ((NP2\backslash_j S)/_j NP3 \circ_j NP3) \vdash NP\backslash_j S}{((((NP\backslash_j S)/_i(NP2\backslash_j S))/_j QNP2 \circ_j QNP2) \circ_i (NP2\backslash_j S)/_j NP3) \circ_j NP3 \vdash NP\backslash_j S}\, /_j R}{(((NP\backslash_j S)/_i(NP2\backslash_j S))/_j QNP2 \circ_j QNP2) \circ_i (NP2\backslash_j S)/_j NP3 \vdash (NP\backslash_j S)/_j NP3}\, A_{i,j}}{(persuade \circ_j a \cdot student) \circ_i to \cdot review \vdash (NP\backslash_j S)/NP3}\, PF$$

b. Derived LF sequent:

$(\lambda Q2.\lambda V.\lambda y.Q2(\lambda x.persuade'(x)(V)(y)) \circ_j$

$\lambda B.some'(student')(B))) \circ_i \lambda u.\lambda v.review'(u)(v)$

$\vdash \lambda u.\lambda y.some'(student')(\lambda x.persuade'(x)(\lambda v.review'(u)(v))(y))$

c. Derived PF sequent:

$(persuade \circ_j (a \cdot student)) \circ_i (to \cdot review)$

$\vdash (persuade \cdot (a \cdot student)) \cdot (to \cdot review)$

Now, we apply argument slot raising to the output in (263) with regard to the NP3 argument slot.

(264) a. $(NP\backslash_j S)/_j NP_3 \Rightarrow (NP\backslash_j S)/_j((S/_j NP_3)\backslash_j S)$

b. $\lambda u.\lambda y.some'(student')(\lambda x.persuade'(x)(\lambda v.review'(u)(v))(y))$

$\Rightarrow \lambda Q1.\lambda y.Q1(\lambda u.some'(student')(\lambda x.persuade'(x)(\lambda v.review'(u)(v))(y)))$

After merging the result with *every paper* and *John*, we derive the inverse scope reading, *every* > *a* for (259a).

(265) a. LF:

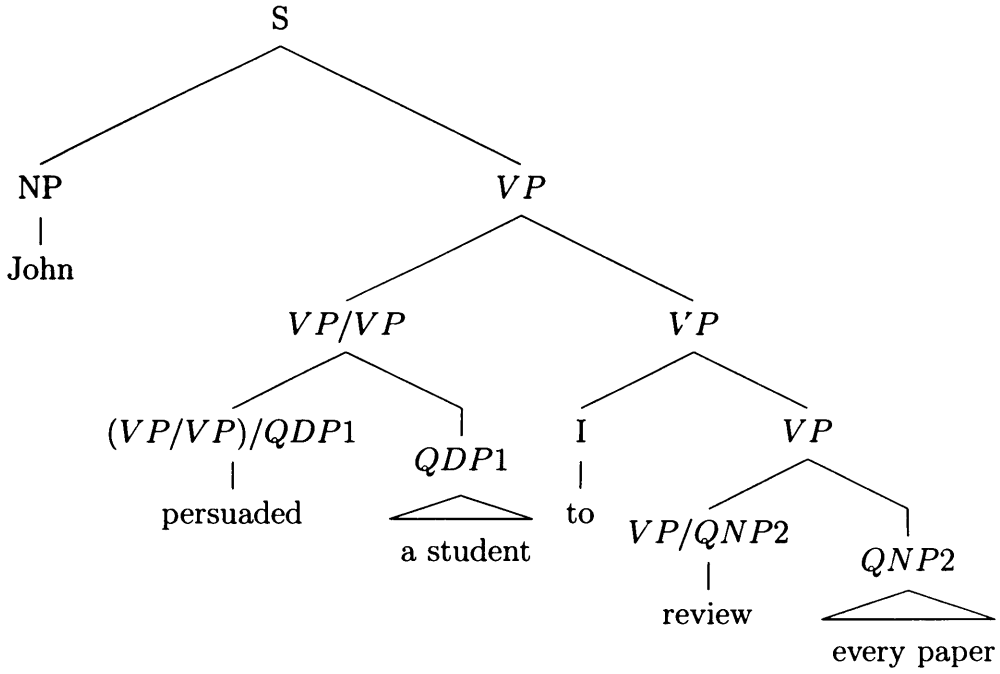
$every'(paper')(\lambda u.some'(student')(\lambda x.persuade'(x)(\lambda v.review'(u)(v))(john')))$

b. PF:

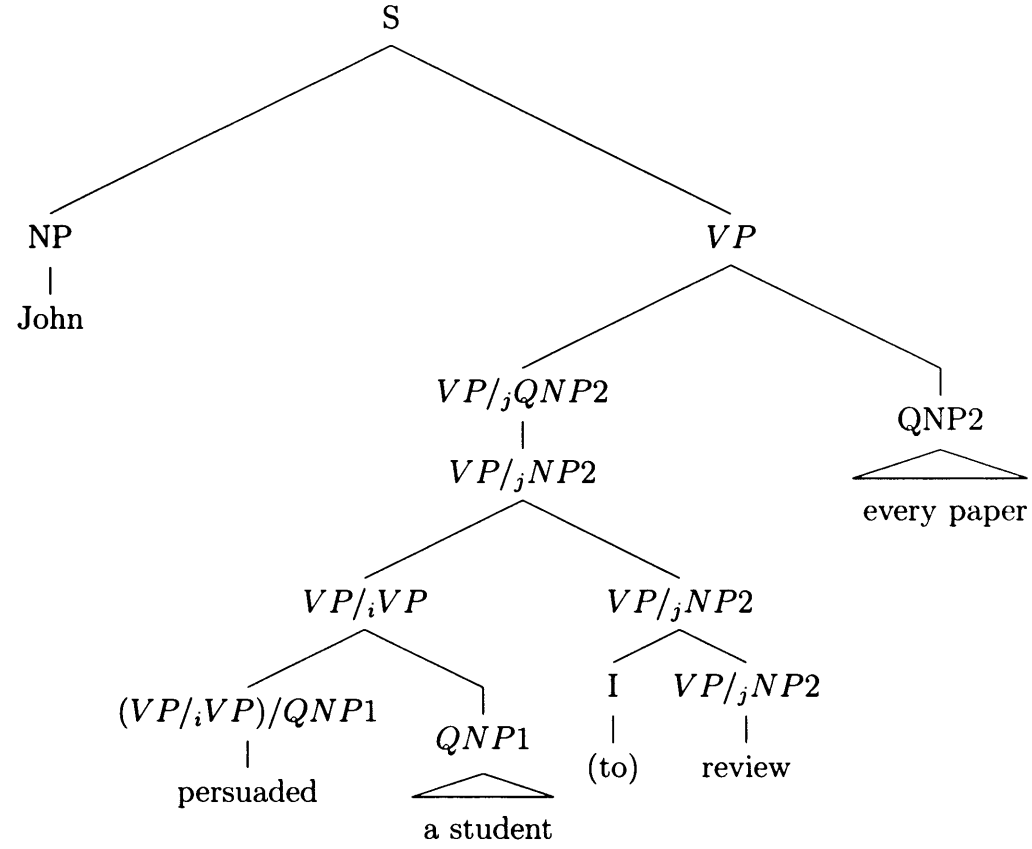
$John \cdot (((persuade \cdot (a \cdot student)) \cdot (to \cdot review)) \cdot (every \cdot paper))$

Comparing the two PF structures for the two scope readings in (262) and (265), we observe the following structural differences (VP represents $NP\backslash_j S$ and I omit all the indices on the formula connectives, other than the pair i, j which triggers the application of $A_{i,j}$).

Tree A: Surface scope, as in (262)



Tree B: Inverse scope, as in (265)



In Tree B above, the '$NP2$' argument slot is percolated and we apply argument slot raising at the highest '$VP/_jNP2$' node, turning it into '$VP/_jQNP2$,' as is

represented by an unbranching node in Tree B. Though the tree structures do not have an official status in the grammar system that we are using, the constituency indicated in the trees will be important to explain other phenomena such as coordination. I do not discuss the interaction of other factors such as coordination with scope readings in this thesis, but that is an interesting point to investigate in the future.[11]

Switching scope between the internal and external QNP arguments of a control verb is straightforward. We apply argument slot raising to the control verb *persuade* with regard to its internal and external NP argument slots in two orders.

(266) *persuade* (surface scope).

a. $((NP1\backslash_j S)/(NP2\backslash_j S))/NP2$
$\Rightarrow ((NP1\backslash_j S)/(NP2\backslash_j S))/((S/NP2)\backslash S)$
$\Rightarrow ((S/(NP1\backslash S)\backslash_j S)/(NP2\backslash_j S))/((S/NP2)\backslash S)$

b. $\lambda x.\lambda V.persuade'(x)(V)(y) \Rightarrow \lambda Q1.\lambda V.\lambda y.Q1(\lambda x.persuade'(x)(V)(y))$
$\Rightarrow \lambda Q1.\lambda V.\lambda Q2.Q2.(\lambda y.Q1(\lambda x.persuade'(x)(V)(y)))$

(267) *persuade* (inverse scope)

a. $((NP1\backslash_j S)/(NP2\backslash_j S))/NP2$
$\Rightarrow (((S/(NP1\backslash S))\backslash_j S)/(NP2\backslash_j S))/NP2$
$\Rightarrow (((S/(NP1\backslash S))\backslash_j S)/(NP2\backslash_j S))/((S/NP2)\backslash S)$

b. $\lambda x.\lambda V.\lambda y.persuade'(x)(A)(y)$
$\Rightarrow \lambda x.\lambda V.\lambda Q2.Q2(\lambda y.persuade'(x)(V)(y))$
$\Rightarrow \lambda Q1.\lambda x.\lambda Q2.Q1(\lambda x.Q2(\lambda y.persuade'(x)(V)(y)))$

The rest of the derivation is straightforward for either reading, so I only show the result. We do not need to apply any structural association, and thus the PF bracketing is as is suggested by the verbal functor categories, and is the same for both the scope readings, as shown in (268).

(268) $((A \cdot teacher) \cdot ((persuaded \cdot (every \cdot student)) \cdot (to \cdot (study \cdot logic))))$.

(269) a. Surface scope:
$some'(teacher')(\lambda y.every'(student')(\lambda x.persuade'(x)(study'(logic'))(y)))$

b. Inverse scope:
$every'(student')(\lambda x.some'(teacher')(\lambda y.persuade'(x)(study'(logic'))(y)))$

[11] Thanks to Alex Lascarides for reminding me of this.

Next, we consider scope switch between a QNP as the main clause subject and a QNP in an object position of the embedded infinitival verb. Consider (270).

(270) A teacher persuaded John to study every/each subject.

$$a > every, ??every > a$$

The proof in (271) shows that we can treat *persuade Jack to review* in (270) as a complex predicate.

(271) a.

$$\dfrac{\dfrac{\dfrac{(((NP\backslash_j S)/_i(NP2\backslash_j S))/_j NP2 \circ_j NP2) \circ_i ((NP2\backslash_j S)/_j NP3 \circ_j NP3) \vdash NP\backslash_j S}{((((NP\backslash_j S)/_i(NP2\backslash_j S))/_j NP2 \circ_j NP2) \circ_i (NP2\backslash_j S)/_j NP3) \circ_j NP3 \vdash NP\backslash_j S}\, A_{i,j}}{(((NP\backslash_j S)/_i(NP2\backslash_j S))/_j NP2 \circ_j NP2) \circ_i (NP2\backslash_j S)/_j NP3 \vdash (NP\backslash_j S)/_j NP3}\, /_j R}{(persuaded \circ_j John) \circ_i to \cdot study \vdash (NP\backslash_j S)/_j NP3}\, PF$$

b. Derived LF sequent:
$(\lambda x.\lambda V.\lambda y.persuade'(x)(V)(y)) \circ_j John') \circ_i \lambda u.\lambda v.study'(u)(v)$
$\vdash \lambda u.\lambda y.persuade'(john')(study'(u))(y)$

c. Derived PF sequent:
$(persuade \circ_j John) \circ_i (to \cdot review) \vdash (persuade \cdot John) \cdot (to \cdot study)$

We can then apply argument slot raising to the complex functor *persuade John to review* of category $(NP\backslash_j S)/_j NP3$ in terms of the two NP argument slots in two orders and derive the two scope readings. On the face of it, this might seem to overgenerate the scope readings. The inverse scope reading is not easy to get with (270), and the reading is noticeably more difficult to get with (272c) as opposed to (272a) and (272b).

(272) a. Jack persuaded a student to review every paper. $a > every$; $every > a$

b. A negotiator persuaded every criminal to submit. $a > every$; $every > a$

c. An editor persuaded Jack to review every paper.

$$a > every;\ ? * every > a$$

I do not have a good explanation of this contrast, but I speculate that the inverse scope reading is formally available with (272c). We only find it difficult to get because of some on-line processing difficulty.

7.3 Raising constructions

Raising verbs as in (273) can be treated in more or less the same way as control verbs.

(273) a. A student [*seems to review*] every paper. $a > every$, $every > a$

b. A student [*is likely to review*] every paper. $a > every$, $every > a$

To the functors *seem* and *likely*, I assign categories analogous to subject control verbs.

(274) a. seem: $< seem;\ (NP\backslash_j S)/_i(NP\backslash_j S);\ seem'_2 >$
where, $seem'_2 \stackrel{\text{def}}{=} \lambda V.\lambda x.seem'_1(V(x))$

b. (be) likely: $< likely;\ (NP\backslash_j S)/_i(NP\backslash_j S);\ likely'_2 >$
where, $likely'_2 \stackrel{\text{def}}{=} \lambda V.\lambda x.likely'_1(V(x))$
Types, $seem'_2 : (et)(et)$; $seem'_1 : (tt)$; $likely'_2 : (et)(et)$
$likely'_1 : (tt)$; $V : (et)$; $x, y : e$

I abstract away from the contribution of *be* in *be likely to*. We can see it as an identity function both in the syntax and in the semantics. An important difference between raising verbs and control verbs is that raising verbs do not have an external argument slot in their basic logical expression. That is, the semantic type of $seem'_1$ in $\lambda V_{et}.\lambda x_e.seem'_1(V(x))$ is type (tt). Thus, in the normalized logical form, the type e argument for *Jack* in the sentence *Jack seems to run* will saturate the external argument slot of the logical expression run' for the infinitive verb *run*, which is selected by the raising verb *seem*. The type e argument for *Jack* does not saturate the (non-existing) external argument slot of $seem'_1$. This analysis of raising verbs allows us to use the same predicate $seem'_1$ and $likely'_1$ in infinitival sentences as in (273) on the one hand, and in expletive sentences such as *It seems that a student reviews every paper* and *It is likely that a student reviews every paper* on the other. The verb *seem* may have a dative argument slot, as in *Tom seems to me to be a good student*, but I ignore this additional argument slot for presentation reasons.

Other than the lack of the external argument slot for subject raising verbs, the use of structural association rule $A_{i,j}$ to produce a complex functor is exactly the same as in the subject control case with *try* in (255). I only show the final categorial sequent with its semantic sequent for (273a), ignoring the contribution of the infinitive *to*.

(275) *seem to review*

a. $(NP1\backslash_j S)/_i(NP1\backslash_j S) \circ_i (NP1\backslash_j S)/_j NP2 \vdash (NP\backslash_j S)/_j NP2$

b. $\lambda V.\lambda x.seem'_1(V(x)) \circ_i \lambda u.\lambda v.review'(u)(v)$
$\vdash \lambda u.\lambda y.seem'(review'(u)(y))$

Then, we can apply the argument slot raising to the output in (293), deriving the two scope readings. I omit the process.

Verbs such as *expect* might be treated as a "raising to the object" verb, in that it can either select an infinitival clause or a tensed CP, as shown in (276).

(276) a. Tom expected a student to review every paper. $a > every$; $every > a$

b. Tom expected that a student would review every paper.

A difference from object control verbs such as *persuade* will then be the lack of the type e internal argument slot in the semantic functor expression for *expect*. The lexical entry for *expect* in (276a) will be as in (277).

(277) expect:

$< expect; ((NP1\backslash_j S)/(NP2\backslash S))/NP2; \lambda x.\lambda V.\lambda y.expect'(V(x))(y) >$

Other than this difference in the semantics, the derivations of the two scope readings for (276a) are the same as in the object control construction. I omit the derivations.

Finally, auxiliary verbs can be dealt with in basically the same way as subject control verbs, with the lexical entry for auxiliary verbs as in (278b). As I have indicated above, auxiliary verbs are not normally taken to assign their own theta roles, but simply percolate the (external) argument slots of the VPs they select, but because of this, the suggested complex predicate formation is even more straightforward in the case of auxiliary verbs. We can simply treat each pair of an auxiliary verb and the selected verb (whether it is intransitive/transitive/ditransitive) as a verbal complex.

The derivations of the two scope readings are the same as with subject control verbs, and I omit them.

(278) a. A proper researcher must review every paper.

b. must:

$< must; (NP\backslash_j S)\backslash_j (NP\backslash_j S); must' >$

$must' = \lambda V_{et}.\lambda z.must'(V)(z)$

Types, $must' : (et)(et)$

In the next section, I explain how we can explain passivization in terms of a structural rule controlled by mode indices.

7.4 Passive

7.4.1 Basics

Passivization in the two kinds of ditransitive constructions in English is sensitive to the PF linear order between the two objects in the active sentences. In other words, only the left object NP can be fronted to the sentence initial position in the passive construction, both for the DO and for the PP construction, as is shown in (279). In (279), the coindexed traces are used to indicate the PF positions of the NP arguments in question in the active sentences.

(279) a. DO: Meg_1 was shown t_1 the room.

b. DO: *The room_1 was shown Meg t_1.

c. PP: The room_1 was shown t_1 to Meg.

d. DO: *Meg_1 was shown the room to t_1.

To deal with the control/raising/auxiliary constructions, I have modified structural positions where NP arguments are merged in terms of a structural rule controlled by merge-mode specification. Because the passive involves merging an internal NP argument in a different position from where it is merged in the corresponding active sentence, it is natural to use a similar way of introducing a structural rule.

7.4.2 Analysis of the passive in MMTLG

This thesis provides only a sketch of my treatment of the passive and the rest is left for future research. My analysis is similar to the analysis of the passive in Bach (1980) in that some passive translation process is applied to the categories/logical expressions associated with the verbal functors in question. It is similar in that the process is triggered by information encoded in the abstract passive participle morpheme (represented by EN in notation) as well.

However, it is different in several ways. The most important difference is that unlike Bach, I do not generate a 'transitive verb phrase' before I apply a passive translation procedure. Rather, I apply the passive morpheme operator EN directly to lexical verbs, either transitive or ditransitive. In this way, I can avoid using wrapping which Bach's analysis uses in an essential way. As I indicated in chapter 3, this is mostly because of my methodological preference in which we start with the basic logical connectives that successively merge PF adjacent expressions, and then explicitly define structural rules to explain discontinuity phenomena. On the

face of it, this may lead to some problems, given the claim that passive operates at the level of phrases, not at the level of lexical verbs.[12] However, because the application of *EN* to lexical verbs is only the first part of my analysis of the passive, and also because the use of • in ditransitive verb categories gives us more flexibility in terms of VP structures, I argue that my analysis can cope with the linguistic phenomena that cause problems for a lexical analysis of the passive. Also, decomposition of the passivization process into sub-parts helps put the passive into a more general picture in theoretical linguistics. As we see shortly, the second part of the passivization process which I explain in terms of a structural rule corresponds to the phenomena that Minimalism explains in terms of movement that is triggered by the Extended Projection Principle.

As the first step of the passivization process, the abstract passive morpheme *EN* suppresses the external NP argument slot of the verbal functor in the syntax, whereas it existentially closes the external argument slot in the semantics. The entry for the passive morpheme *EN* is as in (280). This process is basically the same as in Bach (1980), though the operation applies to lexical verbs, rather than transitive verb phrases.

(280) *EN*

a. Syntax: $((NP\backslash_j S)/_j X)\backslash_m (S/_j X)$

where X can be instantiated as zero to n NP argument slots concatenated as one complex argument slot, such as

$X = (NP_1 \bullet_c (NP_2 \bullet_c \ldots \bullet_c NP_n))$.

b. Semantics:

$en' := \lambda P^{n+1}.\lambda x_1.\ldots.\lambda x_n.\exists y.P^{n+1}(x_1)\ldots(x_n)(y)$

P^{n+1} of type $((e_1 \times (e_2 \times \ldots \times e_n)),(et))$, where n varies from 0 to n.

c. Phonology: *EN* is instantiated in an appropriate PF form and attached as a suffix to the verb.[13]

The passive suffix *EN* is phonologically attached to (the stem of) the lexical verb. Unlike Bach, I do not take the step of generating a transitive verb phrase first (to which the passive translation process is applied). Rather, I apply the *EN* functor directly to verbs with varying arities. In (280a), 'X' represents zero$\sim n$ NP-object argument slots for some natural number n[14] Theoretically, this means that the

[12]See Keenan (1980) for some arguments supporting this claim.

[13]Such as $-n$ in *shown* or $-ed$ in *played*, according to phonological principles whose identities are not relevant to our discussion.

[14]Probably, the maximal number n is 2 for ditransitive verbs in natural language.

passive morpheme can be applied to (certain) intransitive verbs, which would generate the impersonal passive structure in some languages. But for some reason that I cannot specify, this is not possible in English and thus, I concentrate on the application of EN to transitive and ditransitive verbs.[15] Whatever the identity of the internal argument slot 'X' is, the passive morpheme EN syntactically suppresses the external NP argument slot. In the semantics, EN existentially closes the external argument slot.[16]

In the category for EN in (280a), the mode j specifies the mode of merge of a functor and an NP argument, the mode m specifies the attachment of the suffix in question to a verbal functor, and the mode c specifies the merge of two or more NP arguments into a binary configuration, which requires the n NP arguments to be merged into a complex $NP_1 \circ_c (NP_2 \circ_c \ldots \circ_c NP_n)$ argument before it is merged with the verbal functor.[17] When the verb is a transitive verb, then there is only one internal NP argument, and thus, the category of EN is $((NP\backslash_j S)/_j NP)\backslash_m (S/_j NP)$ and en' is $\lambda P^2{}_{e(et)}.\lambda x.\exists y.P^2(x)(y)$. The application of EN to a verbal functor, say, *play* in *Jack played tennis* will produce $EN(play)$ of category S/NP. As it is, this might seem to wrongly generate a string, **played tennis*, but I assume that there is some universal constraint that requires the 'specifier' position of the highest head of each propositional element (which corresponds to category S) to be filled. In other words, generation of a functor category in the form of $S/_j X$ as the output of (280) triggers the application of the following structural rule, EAP.

(281) Extended Association/Permutation, or EAP (by analogy to EPP).

a.
$$\frac{A \circ_j (\mathbf{B}_1 \circ_j (dtr \circ_j (B_2 \circ_j \ldots \circ_j (dtr \circ_j B_n)))) \vdash C}{\mathbf{B}_1 \circ_j (A \circ_j (dtr \circ_j (B_2 \circ_j \ldots \circ_j (dtr \circ_j B_n)))) \vdash C}\ EAP$$

b.
$$\frac{V \circ_j (\mathbf{x}_1 \circ_j (dtr \circ_j (x_2 \circ_j \ldots \circ_j (dtr \circ_j x_n)))) \vdash \phi}{\mathbf{x}_1 \circ_j (V \circ_j (dtr \circ_j (x_2 \circ_j \ldots \circ_j (dtr \circ_j x_n)))) \vdash \phi}\ EAP$$

[15] I assume that EN can be applied to verbs with an arity greater than 3, if such verbs exist in some languages, but because natural language verbs normally have maximally three NP arguments, I only consider transitive verb and ditransitive verbs.

[16] (280b) would be $en' := \lambda P^{n+1}.\lambda x_1.\ldots.\lambda x_n.some'(\lambda y.individual'(y))(\lambda y.P^{n+1}(x_1)\ldots(x_n)(y))$ in the lambda notations that I use elsewhere in the thesis, but because the nominal restriction does not provide restriction here, I choose to use existential quantifier explicitly in notations. This is for convenience only.

[17] As we see later, the mode c does not play an essential role in the structural rule that we use. Given the general nature of the concatenation operation by '•,' we may underspecify the nature of this merge mode, but I keep the description of this merge mode as in the main text for the moment.

In (281a), A is for the category for $EN(verb)$, and $B_1, \cdots, B_n$ are n internal arguments for some natural number n. In (281), *dtr* represent the binary concatenation operator which concatenate NP arguments into successively more complex arguments when it is used repeatedly, as in $NP_1 \circ_j (dtr \circ_j (NP_3 \circ_j (dtr \circ_j NP_3)) \vdash NP_1 \bullet_c (NP2 \bullet_c NP3)$. In ditransitive constructions, the complex NP object is made out of two NPs, so *dtr* is inserted only once, as we have seen in chapter 4. In (281), *dtr* is used in the rule for presentational convenience. That is, it could be some other binary concatenation functor, as long as the functor is merged with the n arguments by using the mode j, concatenating the arguments into a complex argument as is required by the verbal functor category A. For example, in the PP ditransitive construction, the prepositional functor for *to* will be placed instead of *dtr* between the two NP objects. EAP in (281a) forces the merge of the left-most subcomponent (i.e. B_1) of the complex NP object as the single NP argument to the left of the verbal functor, as we will see in its application to the DO construction in (287). The restructuring in the antecedent side does not influence the semantics in the succedent side, and ϕ stays the same. In other words, though the left NP object is merged in the subject position, it still saturates the initial internal argument slot of the verb. This is a significant difference from the use of $A_{i,j}$ in the control/raising/auxiliary constructions, where the structural rule generates a complex predicate whose argument slots NP arguments saturate. In the passive, the structural rule does not influence which argument slot of which functor an NP object saturates. It only allows an object NP to be merged in a different PF position than the initial position.

EAP in (281) applies only with the merge mode j (which is for the merge of functors and their NP arguments) in certain structural configurations. This avoids destroying the binary syntactic structures that are generated by the base grammar system NL elsewhere. Some might be worried about the essential context sensitivity of this structural rule, but in that regard, even a simple association rule, which we would need for creating a complex predicate to deal with control/raising/auxiliary constructions, would not be context free either.

(282) Structural Association: $A \circ_i (B \circ_j C) \Rightarrow_{A_{i,j}} (A \circ_i B) \circ_j C$

This thesis does not investigate whether my solution for the passive necessarily exceeds the diagnostically crucial limit of Mildly Context Sensitive grammar, and leaves potential improvement in the instantiation of EAP for future research.[18]

[18]See Keenan and Stabler 2000 for some of the diagnostics for mild context sensitivity.

Some might wonder why I have not incorporated the effect of EAP in (281) into the morphological information with EN in (280), so that the morpheme EN would extract the left-most component of its internal argument to the left. One reason why I did not do so was that I am hoping to define EAP as a more general rule, as I have indicated before. That is, presumably, merge of any functor category in the form of $S/_jX$ (at least in English) is to trigger the application of EAP, forcing the 'specifier' position of that category to be filled. But I do not investigate whether this claim is substantiated in cases other than the passive, and thus, as far as this thesis is concerned, it does not make a difference if we incorporate the effect of EAP into the lexical entries with EN.

Finally, to avoid treating a string such as $(Tennis \cdot played)$ as of category S, the minimal propositional category S would ultimately be licensed by finite tense.[19] But I abstract away from the extra complexity in this thesis.

Before we apply the algorithm to ditransitive sentences in (279), we apply it to a simple transitive verb sentence, *Jack played tennis.* First, we apply EN to the transitive verb *play.* We first merge the passive morpheme EN with *play.*

(283) $EN(play)$.

a. Syntax: $EN((NP\backslash_jS)/_jNP) = S/_jNP$

b. Semantics: $EN(play') = \lambda x.\exists y.play'(x)(y)$

c. PF: $EN(play) = played$

Given the output in (283), the rest of the derivation is straightforward. We ignore the contribution of *be.* We can see it as an identity function in the syntax and in the semantics. Thus, we can assume that the syntactic category and the semantics of *be played* are the same as those for *played.*

(284) a. Syntax:

$$\frac{\dfrac{S/_jNP \circ_j NP \vdash S}{NP \circ_j S/_jNP \vdash S}\,EAP}{tennis \circ_j (is \cdot played) \vdash tennis \cdot (is \cdot played)}\,PF$$

[19]For example, we might merge an abstract lexical functor category such as PAST of category $TP/_fS$ with each propositional category (where the mode index f is mnemonic for 'finite'), turning the LF expression into a properly truth evaluable element. The merge of PAST might then trigger a further structural rule that has the effect of Extended Projection Principle as in Minimalism.

b. Semantics:

$$\frac{\dfrac{V \circ_j x \vdash V(x)}{x \circ_j V \vdash V(x)}\ EAP}{tennis' \circ_j (en'(play')) \vdash \exists y.play'(tennis')(y)}\ LF$$

Because the internal NP argument is not complex and is made out of only one NP component, the unique object NP is the 'left-most component.' Thus, EAP in (281) allows this object NP to be merged in the position for the subject NP.

Next, in (286) below, we merge the passive morpheme EN with a ditransitive verb *show* to deal with the sentence in (285).

(285) a. Jack showed Meg her room.

b. Meg was shown her room.

The mode of the merge in (286) is m, dictated by the morphemic functor category for EN in (280a).

(286) $EN(show)$

a. Syntax:
$(NP3\backslash_j S)/_j(NP1 \bullet_c NP2) \circ_m ((NP\backslash S)/_j X)\backslash_m(S/_j X)$
$\vdash S/_j(NP1 \bullet_c NP2)$

b. Semantics: $show_3' \circ_m EN \Rightarrow \lambda\alpha.\exists x.show'(\pi 1(\alpha))(\pi 2(\alpha))(x)$
Types: $show_3' : ((e \times e), (et))$; $show' : e(e(et))$; $\alpha : (e \times e)$;
$EN : ((e_1 \times ... \times e_{1-n}), (et)), ((e_1 \times ... \times e_{1-n}), t))$

c. PF: $show \circ_m EN \Rightarrow$ *(be) shown*

The merge of EN and the ditransitive verb in (286) suppresses the external NP argument slot in the syntax and existentially closes the external argument slot in the semantics, maintaining the complex object argument slot. Applied to the DO construction, this generates the category $S/_j(NP1 \bullet_c NP2))$ as in (286a), with the corresponding semantics in (286b). The generation of the functor category $S/_j(NP1 \bullet_c NP2))$ triggers the application of EAP in (281). The PF generates a passive form *(be) shown.*

(287) EAP applied to *Meg was shown the book.*

a. Syntax:

$$\frac{\dfrac{S/_j(NP1 \bullet_c NP2) \circ_j (NP1 \circ_j (dtr \circ_j NP2)) \vdash S}{NP1 \circ_j (S/_j(NP1 \bullet_c NP2) \circ_j (dtr \circ_j NP2)) \vdash S}\ EAP}{Meg \circ_j ((is \cdot shown) \circ_j (\epsilon \cdot the \cdot room)) \vdash S}\ PF$$

where $dtr = (NP1\backslash(NP1 \bullet NP2))/NP2$, in which numbering on NPs is used for presentation reasons.

b. Semantics:

$$\frac{V^3 \circ_j (x_1 \circ_j (dtr' \circ_j x_2)) \vdash V^3(x_1 \bullet x_2)}{x_1 \circ_j (V^3 \circ_j (dtr' \circ_j x_2) \vdash V^3(x_1 \bullet x_2)} \; EAP$$

LF instantiation (replacing meta-variables with constant terms, together with π_i conversion):

$x_1 \circ_j (V^3 \circ_j (dtr' \circ_j x_2) \vdash V^3(x_1 \bullet x_2) \quad \Rightarrow_{LF}$

$meg' \circ_j (\lambda\alpha.\exists y[show'(\pi 1(\alpha))(\pi 2(\alpha))(y)] \circ_j (dtr' \circ_j the \cdot room))$

$\vdash \exists y[show'(meg')(\iota(room'))(y)]$

where $dtr' = \lambda u.\lambda v.v \bullet u$; $\iota : (et)e$

Remember that *dtr* is a binary operator that generates a complex NP argument and that its PF is null. Because *EAP* only allows the left-most NP sub-component of the complex NP object (if it is complex) to be merged to the left of the verb, only the left NP object can be extracted to the subject position.

The treatment of the PP construction is basically the same.

(288) *EAP* applied to *Meg was shown the book.*

a. Syntax:

$$\frac{\dfrac{S/_j(NP1 \bullet_c NP2) \circ_j (NP1 \circ_j (to \circ_j NP2)) \vdash S}{NP1 \circ_j (S/_j(NP1 \bullet_c NP2) \circ_j (to \circ_j NP2)) \vdash S}\; EAP}{the \cdot room \circ_j ((is \cdot shown) \circ_j (to \circ_j the \cdot room)) \vdash S}\; PF$$

where $to = (NP1\backslash(NP2 \bullet NP1))/NP2$

b. Semantics:

$$\frac{V^3 \circ_j (x_1 \circ_j (to' \circ_j x_2)) \vdash V^3(x_1 \bullet x_2)}{x_1 \circ_j (V^3 \circ_j (to' \circ_j x_2) \vdash V^3(x_1 \bullet x_2)} \; EAPS$$

LF instantiation:

$x_1 \circ_j (V^3 \circ_j (to' \circ_j x_2) \vdash V^3(x_1 \bullet x_2) \quad \Rightarrow_{LF}$

$\iota(room') \circ_j (\lambda\alpha.\exists y[show'(\pi 1(\alpha))(\pi 2(\alpha))(y)] \circ_j (to' \circ_j meg'))$

$\vdash \exists y[show'(meg')(\iota(room'))(y)]$

where $to' = \lambda u.\lambda v.u \bullet v \; \iota : (et)e$

Note that unlike $A_{i,j}$ in (250), EAP in (281) does not allow us to treat *(be) shown* as a complex predicate of category $(NP\backslash S)/NP$. Consider the Gentzen sequent proof for the DO in (287) again, with the category for *dtr* being spelt out this time.

(289)

$$\dfrac{\dfrac{\dfrac{S/_j(NP_1 \bullet_c NP_2) \circ_j (NP_1 \circ_j ((NP_1\backslash(NP_1 \bullet_c NP_2))/NP_2 \circ_j NP_2)) \vdash S}{NP_1 \circ_j (S/_j(NP_1 \bullet_c NP_2) \circ_j ((NP_1\backslash(NP_1 \bullet_c NP_2))/NP_2 \circ_j NP_2)) \vdash S}\, EAP}{S/_j(NP_1 \bullet_c NP_2) \circ_j ((NP_1\backslash(NP_1 \bullet_c NP_2))/NP_2 \circ_j NP_2) \vdash NP\backslash_j S}\, \backslash R}{(be \cdot shown) \circ_j (\epsilon \circ_j the \cdot room) \vdash (be \cdot shown) \cdot (the \cdot room)}\, PF$$

Look at the sequent in the second lowest row in the proof (that is, the categorial sequent just above the PF string) in (289). In the antecedent of that sequent, $NP2$ is embedded in the deepest position, and we cannot abstract away from that argument by using $/R$. This means that we cannot generate the category $(NP1\backslash S)/NP2$ for the string '$(be \cdot shown)$.' Thus, the only scope reading that we can derive between the two QNPs in *A student was shown every room* is the reading that we can generate by applying argument slot raising to *dtr*. As we have seen in chapter 5, we generate only the surface scope reading between the two object QNPs in the DO construction. Thus, according to the proposed analysis, even though the left object NP is allowed to be merged in the subject position, the scope reading between a QNP in this position and a QNP in the right object position stays the same as in the active sentence. In the PP construction, we can switch scope between the two object QNPs by applying argument slot raising to the functor *to* in two different ways, and thus, the sentence stays ambiguous after being passivized. This prediction seems to be correct. The inverse scope reading is significantly more difficult in (290a) than in (290b).

(290) a. A student was shown every room. $a > every$; $? * every > a$

b. A room was shown to every student. $a > student$; $every > a$

Because the semantics stays the same between the active and the passive, the analysis will preserve the scope ambiguity when the external QNP argument is maintained with *by*, as in (291).

(291) a. A book was chosen by every student. $a > every$; $every > a$

b. I was sent a report by every institution. $a > every$; $every > a$

However, this process will be more complex, and thus, I leave the exposition of the exact passivization process for (291) for future occasions.

In my analysis, EN operates on lexical verbs, not on whatever expressions that have the categories $(NP\backslash_j S)/_j NP$, $(NP\backslash_j S)/_j(NP \bullet NP)$, etc. Thus, the mechanism does not generate the ungrammatical sentences in (292), even though the expressions *try to review* and *persuade John to review* are assigned the category $(NP\backslash_j S)/_j NP$ by way of structural association.

(292) a. *The paper was [tried to review].

b. *The paper was [persuaded John to review].

As I indicated above, my analysis applies the passive functor EN directly to the lexical verbal functor, in contrast to Bach (1980), which applies the passive translation process to transitive verb phrases (which may be derived in the syntactic derivation). This may be problematic, given the claim that the passive is not a lexical phenomenon, as Bach (1980) and Keenan (1980) claim. I leave the treatment of such potential problems for future research, but note that because the initial generation of the functor category $S/_j X$ triggers some structural rule application, the analysis has a certain degree of flexibility to deal with apparently phrasal nature of the passive, as we have already seen in its application to the PP ditransitive construction, *The book was shown* t_1 *to Meg.*

In this section, I have briefly shown how the passive can be accommodated in my analysis. Though the fundamental way of introducing the structural association/permutation is the same between the control/raising/auxiliary constructions and the passive, for the latter, the structural rule does not allow us to generate a new complex predicate to which argument slot raising may newly apply. Thus, passivization does not generate new scope ambiguity on top of the scope ambiguity that the corresponding active sentence has. In the next section, I informally show that the structural rules that have been introduced under the control of mode specification does not allow us to generate a complex predicate beyond the local S expression.

7.5 Tensed CP boundary

The structural association rule $A_{i,j}$ in (250) is defined in terms of the two specific modes of merge, so that we can derive a complex predicate only when we have a verb that selects a VP as an argument, as is the case with a control verb, a raising verb or an auxiliary verb. In such a case, we can merge the verb with the VP before or after we merge the head of the selected VP with its internal argument(s). However, when the higher verb selects a sentential complement of category S (or in

Minimalist syntax, when the higher verb selects a CP as one of its arguments), it is different because we would have to merge the complementizer with the minimal S expression that contains QNPs in question. Have a look at (293).

(293) $((A \cdot teacher) \circ_j (reported \circ_j (that \circ_s (Tom \circ_j (robbed \circ_j (every \cdot bank)))))).$

In (293), the complementizer *that* is merged with the minimal sentential expression *Tom robbed every bank* of category S in the merge mode '$\circ_s$' (mnemonic for 'sentences') which is a different mode of merge from '$\circ_i$.' Thus, the structural association $A_{i,j}$ in (250) cannot apply beyond this 'S' expression, making it impossible to derive a complex predicate beyond the local S expression.[20]

From a formal viewpoint, how we assign such merge modes is just a stipulation. However, the goal is to limit the number of modes of merge to linguistically well-motivated ones. From this viewpoint, it makes some linguistic sense to distinguish the mode of merge between a verbal functor and its NP arguments (i.e. '$\circ_j$'), the mode of merge between control verbs/auxiliary verbs and the selected V(P)s (i.e. $\circ_i$) and the mode of merge of the complementizer and the selected sentential expression (i.e. $\circ_s$ in (273)).

Finally, remember that all of these merge modes are specified in some lexical functor categories. Thus, we can still maintain lexical inclusiveness which states that the syntactic derivation can be read off the lexical information.[21]

7.6 Conclusions

This chapter discusses how we can generate complex functors such as *try to review*, *seem to review* or *can review* in subject control/raising/auxiliary constructions. We have done so by using a structural association rule that is controlled by a specific pair of merge modes, i, j. In this way, we can naturally limit the formation of complex predicates within each verbal projection line T-v-V. I have also shown how a similar structural rule together with a certain lexical entry for the passive morpheme *EN* can deal with the passive in English. However, unlike my treatment of control/raising/auxiliary constructions, passivization in my analysis does not generate a complex predicate and thus does not create more scope readings than

[20]Based on some initial observation that inverse scope between a QNP inside the embedded CP and a QNP in the main clause becomes easier if the CP is subjunctive (and thus, the verb in the CP is infinite), we might need to fine-tune the analysis so that $\circ_s$ will only be the mode for merging a complementizer head and a 'tensed' S, as opposed to any S, but I will leave such further complications for future research.

[21]Though as is the case with the application of $A_{i,j}$ to form complex predicates, certain structural operations only optionally apply.

in the active sentence. Finally, I have briefly explained differences between the way that we introduce structural rules to form complex predicates and the way we introduce structural rules to explain A-bar extraction phenomena. This provides a transition to the discussion in the next chapter, that is, how MMTLG can deal with Wh-movement.

Chapter 8

Wh-extraction and structural rules

8.1 Introduction

This chapter serves two purposes. Firstly, to deal with Wh-extraction, I adopt a different way of defining modally controlled structural rules from the one that we saw in chapter 7.[1] To deal with QNP scope in the control/raising/auxiliary constructions, I used a structural association rule that is restricted by some merge mode specification. The mixed association rule introduced there is suitable for instantiating the percolation of an argument slot of a verb within a verbal projection line, as is shown in (294).

(294) a. $John \circ_j (tried \circ_i (to \cdot find \circ_j Eva))$.

b. $John \circ_j ((tried \circ_i to \cdot find) \circ_j Eva)$.

Linguistically, the modification of the standard structure in (294a) as the structure in (294b), which allows us to switch scope, is the creation of a complex predicate made out of the lower verb *find* and the higher verb *tried* before we apply either functor to its NP argument(s).[2] The required combination of the modes of merge for introducing association (i.e., the mode 'i' for merging a control/raising/auxiliary verb with the selected VP and the mode 'j' for merging a verbal functor with its NP argument(s)) is a restrictive way of instantiating the percolation of NP argument

[1] In this thesis, I compare QNP scope only with Wh-extraction. Ideally, I should compare QNP scope with A-bar extraction in general which includes Wh-movement and overt topicalization, but I leave it for future research.

[2] As in chapter 7, I assume the contribution of *to* is trivial relative to our current concerns, and thus I treat it as an identity function.

slots when we form a complex predicate. Because of this mode specification, the percolation cannot go beyond the final output category S of the main head verb. This is the Type Logical instantiation of my linguistic argument which states that the scope of a QNP cannot go beyond the minimal propositional expression (which corresponds to the minimal S, as we have seen in Chapter 2) that contains it. Also, because the internal NP argument slot of *find* in (294) is percolated till a later stage of derivation without changing its semantic type, the NP *Eva* is interpreted just as a type e argument both for (294a) and (294b). This is desirable because we do not want to change the interpretation of the object NP depending on which structure we use. Note that this association rule is not triggered by the QNP category. It represents the flexibility of the VP structures until we derive the minimal S (or TP), whether the arguments are NPs or QNPs. In other words, the extension of QNP scope beyond the strict final output of the local functor in the infinitival constructions as we have discussed in chapter 7 is a side effect of a certain flexibility of the VP structure until we derive the local S.

In order to set up our Type Logical system in such a way that it respects the observed linguistic differences at the data level, we want to introduce structural rules for Wh-movement in a fundamentally different way from that in which we have introduced a structural rule for complex predicate formation as above.

First, Wh-movement can cross a tensed clause, though it is blocked by more complex structural configurations which are generally called 'islands' to Wh-movement (cf. Ross (1967), Manzini (1992), Postal (1998)). Consider (295)

(295) a. Who_1 did you find out that Meg bumped into t_1 at yesterday's party?

b. Some/a detective found out that Meg bumped into every celebrity at yesterday's party. *some* > *every*, **every* > *a*

c. *Who_1 will John go home if Meg talks to t_1?

d. Some/a girl will go home if Meg talks to every boy.
some > *every*; **every* > *some*

e. ??Who_1 did you wonder if Meg talks to t_1?

f. Some/a girl wondered if Meg talks to every boy.
some > *every*; **every* > *some*

(295a) shows that Wh-movement can cross a tensed clause boundary, whereas (295b) shows that the universal QNP in the embedded tensed clause cannot (easily) take scope over the main-clause existential/indefinite. (295c) and (295d) represent adjunct-clause islands. Wh-movement out of such an island is impossible in (295c).

According to my analysis of QNP scope, the universal QNP *every boy* cannot take scope over the indefinite *some/a girl* in (295d) because the indefinite is outside the minimal tensed clause which contains the universal QNP (or because the indefinite is outside the minimal S which contains the universal QNP), not because the universal QNP is in a Wh-island. Compare (295e) and (295f), where the *if*-clause is an argument of the matrix clause verb *wonder*, rather than an adjunct as we have seen in (295c)~(295d). Wh-movement acceptability improves with (295e), in comparison to (295c). The corresponding inverse scope is still very difficult to get in (295f).[3] Because the indefinite *some/a girl* is still outside the minimal tensed clause that contains the universal QNP in (295f), my analysis predicts that the scope switch is impossible. In contrast, the increased acceptability of (295e) in comparison to (295c) indicates the importance of the adjunct/argument distinction to Wh-movement. Though I do not aim to discuss all the islands to Wh-movement, the initial data suggest that we should explain Wh-islands and QNP scope in fundamentally different ways in TLG.

Wh-extraction is also different from the control/raising/auxiliary constructions in that Wh-expressions cannot saturate type e argument slots of verbal functors as NPs can. As I have explained when I discussed the formation of a complex predicate in the control construction in (294), NP arguments saturate the corresponding type e argument slots of their functors in the control/raising/auxiliary constructions, whether they are merged locally to the lexical functor, or they are merged with complex functors that have been made out of sub functor expressions. Thus, in these constructions, modification of the VP structure is not because of some requirement of NPs. NPs can potentially saturate any of the type e argument slots in the VP. Instead, the structural ambiguity of the complex VP represents the flexibility of the order of the merges when higher order functors such as control/raising/auxiliary verbs select VPs headed by transitive/ditransitive verbs. To represent this observation in formalism, I have used the lexical entries of the control/raising/auxiliary verbs to trigger the application of structural rules.

In contrast, in Wh-movement, the landing site of the Wh-expression is semantically different from the in-situ position. In the informal semantics in (296b), the Wh-expression as a semantic operator takes as its argument a propositional expression abstracted away from the type e argument position that is linked to

[3]Even with contrastive focus on *every boy* (which often extends the scope taking possibilities), the inverse scope is still marginal.

A) A girl wondered if Meg likes Harold.

B) That's not surprising. A girl wonders if Meg likes EVERY boy.

(Dialogue by Rob Truswell)

the Wh-expression (that is, abstracted away from the 'in-situ' argument position x from which the Wh-expression has moved in movement analyses).[4] Consider (296).

(296) a. What did you eat?

b. LF (informal): $\text{What}_{((e,t),q)}$ ($[\lambda x.$ (did) you eat $x]$)

c. PF (informal): $What \cdot [did \cdot (you \cdot eat)]$

d. Informal restructuring involved in the derivation for *What (did) you eat*:

$$\dfrac{\dfrac{Wh \vdash Wh \quad \dfrac{\dfrac{NP \circ ((NP\backslash S)/NP \circ [NP]) \vdash S}{(NP \circ (NP\backslash S)/NP) \circ [NP] \vdash S}\,Assoc}{NP \circ (NP\backslash S)/NP \vdash S/NP}\,/R}{Wh/(NP/S) \circ (NP \circ (NP\backslash S)/NP) \vdash Wh}\,/R}{what \circ (you \circ eat) \vdash Wh}\,PF$$

Note that the informal LF structure in (296b) matches well with the PF structure in (296c). In an informal categorial calculus as in (296d), derivation of the string *(did) you eat* with the category S/NP as in the third row from the top is a prerequisite for merging the Wh-expression which semantically selects type (et) arguments.[5] In other words, the overt extraction of the Wh-expression facilitates the interpretation of Wh-expressions as sentential operators. As this 'movement' seems to be triggered by the requirement of Wh-expressions in this regard, it is natural to have the lexical categories of Wh-expressions trigger the structural rules involved in the derivation in (296d).

Based on the assumption that Wh-movement is both syntactically and semantically different from complex predicate formation, we want to define the structural rules that instantiate Wh-extraction in TLG in a way that is fundamentally different from the way we have defined $A_{i,j}$ for infinitival constructions. For Wh-phenomena (together with A-bar movement phenomena which are subject to more or less the same locality constraints as Wh-movement), I adopt the use of a residuated pair of unary operators, $\Diamond, \Box^{\downarrow}$, as in Vermaat (2006), which decorate the 'hypothetical NP trace' category that can be 'moved' to a string peripheral position before we abstract away from the hypothetical category.

[4] q in the type $(et)q$ for Wh-expressions is mnemonic for 'question.'

[5] The contribution of *did* is ignored in (296d). Because of this, we modify the categorial assignments and the derivations later.

After showing the basic use of structural rules for Wh-extraction in sections 8.2 and 8.3, I briefly discuss the Wh-extraction data in the two kinds of ditransitive constructions in section 8.4.

(297) a. *?Who$_1$ did you show t_1 the room?

b. What$_1$ did you show Ally t_1?

c. Who$_1$ did you show the room to t_1?

d. What$_1$ did you show t_1 to Ally?

e. What$_1$ did you show Ally t_1 yesterday?

In the PP ditransitive construction, as in (297c) and (297d), Wh-movement is possible from either of the two object positions. In the DO construction, the left object is impossible to extract, as (297a) shows.[6] Wh-movement of the right object in the DO construction is grammatical, as in (297b).[7]

In some sense, the difficulty of the extraction of the left object in (297a) is understandable, because we would need to permute the 'trace' position with the right object, before we associate the structure so that we can abstract away from the trace position by applying the rule '$/R$' in Gentzen Sequent presentation (or '$/I$' in Natural Deduction presentation). The required structural process is more complicated compared with a case in which we abstract away from a string peripheral position as in (297b) or (297c).

However, controlled permutation is also necessary for explaining the extraction of the left object in the PP construction, which is grammatical, and also for explaining the Wh-movement out of the right object position as in (297e), where an adverbial expression appears to the right of the 'NP trace' position. Thus, we have to introduce controlled permutation in such a way that we can exclude (297a) without excluding (297d) or (297e).

In order to distinguish (297a) on the one hand, and (297d) and (297e) on the other, I assign different structures to them and use the structure that is assigned

[6]There is some dialectal variation in this judgment. Neil Smith and Glyn Morrill (p.c.) find *Who did you show the room* well-formed, for example. See Hudson (1992: 258) for discussion and analysis. I speculate that the underlying structure of the sentence in (297a) for such speakers is *Who$_1$ did you show the room (to t_1)*, where the preposition is somehow dropped out of the overt PF string though it is still inherently present in the syntactic structure, rather than the structure I postulate for the preposition-less DO construction. As suggested by Kempson and Morrill, some speakers might have the structural ambiguity of the PP ditransitive construction as in (314) for the preposition-less ditransitive construction as well.

[7]This pattern does not match up with the one in the passivized DO construction that we saw in chapter 7. Compare (297a) with *Ally$_1$ was shown t_1 the room* and (297b) with **The room$_1$ was shown Ally t_1*.

to (297d) and (297e) (but not assigned to (297a)) as part of the licensing condition for the permutation rule that we use.

Section 8.2 briefly reviews the pair of unary categorial connectives that we have seen in chapter 3. The unary connectives are used to introduce structural rules and add locality constraints for explaining Wh-extraction in Multi-Modal Type Logical Grammar (MMTLG). Section 8.3 deals with locality of Wh-movement, and section 8.4 is the application of the analysis introduced in section 8.3 to the ditransitive data. Section 8.5 provides concluding remarks.

8.2 MMTLG

In this section, I briefly repeat the definitions of categorial formulas and structures in $NL_\Diamond$, as well as the logical rules for the residuated pair of connectives as in Moortgat (1997). I also show the basic association and permutation rule used for Wh-movement analysis.

The law of residuation for the unary connectives as in (298) suggests that the accessibility relations are the opposite for the two unary connectives.

(298) $\Diamond A \vdash B \Leftrightarrow A \vdash \Box^{\downarrow} B$

Logical rules for the unary connectives in (299) are set up in such a way that they respect the residuation law in (298).

(299) a.

$$\frac{\Gamma \vdash A}{(\Gamma)^{\Diamond} \vdash \Diamond A}\,\Diamond R \qquad\qquad \frac{\Gamma[(A)^{\Diamond}] \vdash B}{\Gamma[\Diamond A] \vdash B}\,\Diamond L$$

b.

$$\frac{(\Gamma)^{\Diamond} \vdash A}{\Gamma \vdash \Box^{\downarrow} A}\,\Box^{\downarrow} R \qquad\qquad \frac{\Gamma[A] \vdash B}{\Gamma[(\Box^{\downarrow} A)^{\Diamond}] \vdash B}\,\Box^{\downarrow} L$$

Given (299), the derivability relation between the relevant categories goes only in one direction.

(300) $\Diamond\Box^{\downarrow} A \vdash A \vdash \Box^{\downarrow}\Diamond A$

As we see later in derivations, the derivability relation in (300) plays an essential role in explaining Wh-extraction. Informally, the sequent '$\Diamond\Box^{\downarrow} NP \vdash NP$' means that we can insert the hypothetical category '$\Diamond\Box^{\downarrow} NP$' instead of the category 'NP' wherever the NP can be placed in the antecedent of a sequent. The inserted

hypothetical category can then trigger application of structural rules which I show next.

Given the pair of unary operators, the structural rules that are used in Wh-movement (or A-bar movement in general) can be provided as right association (AR) and right permutation (PR), as shown in (301) (cf. Bernardi 2002: 47, Moortgat 2001: 9). As usual, the rules are applicable for all the structures X, Y and for all the formulas A, B.

(301) a. AR:

$$\frac{X \circ (Y \circ \Diamond A) \vdash B}{(X \circ Y) \circ \Diamond A \vdash B}\ AR$$

b. PR:

$$\frac{(X \circ \Diamond A) \circ Y \vdash B}{(X \circ Y) \circ \Diamond A \vdash B}\ PR$$

Note that no merge mode specification is involved in the rules in (301). Because AR and PR are defined irrespective of the modes of merge, the application of these structural rules are not influenced by such information. The rules may apply as long as we have a hypothetical category '$\Diamond A$' in the required structural positions. Remember from (300) that '$\Diamond\Box^{\downarrow}A$' can be placed in whichever position that 'A' can occupy in the antecedent of a sequent. As we see in application, this explains the different locality constraints applicable to QNP scope on the one hand and Wh-movement on the other.

In the next section, I briefly explain how this MMTLG analysis works for Wh-movement in a simple case, including how to set up a complex NP island to Wh-movement.

8.3 Wh-movement and islands

I first show that with the structural rules in (301) and an adequate Wh-category as in (303a), we can license Wh-movement across a tensed clause. Then I show how an analysis based on Vermaat (2006), which is based ultimately on Morrill (1994), can deal with a complex NP island to Wh-movement.

(302) a. What$_1$ did John like t_1?

b. What$_1$ did Meg think that John likes t_1?

c. *What$_1$ did you know the man who likes t_1?

The lexical entries to the crucial items are provided in (303).

(303) a. what (Wh): $Wh/(inv/\Diamond\Box^{\downarrow}NP)$

b. do: $inv/(NP \bullet inf)$

c. know, like: inf/NP

d. likes: $(NP1\backslash S)/NP2$

e. who (Rel): $(N\backslash\Box^{\downarrow}N)/(NP\backslash S)$

f. the: NP/N

(cf. Vermaat (2006): chapter 3)[8]

Given the entries in (303), (304a) derives the basic clause internal Wh-movement from the object position for the sentence in (302a). (304b) is for (302b), which derives Wh-movement across a tensed clause.[9]

(304) a. What did John like?

$$\dfrac{\dfrac{Wh \vdash Wh \quad \dfrac{\dfrac{q \vdash inv \quad \dfrac{NP1 \vdash NP1 \quad \dfrac{\dfrac{inf \vdash inf \quad \dfrac{\dfrac{NP2 \vdash NP2}{(\Box^{\downarrow}NP_2)^{\Diamond} \vdash NP_2}\,\Box^{\downarrow}L}{\Diamond\Box^{\downarrow}NP_2 \vdash NP_2}\,\Diamond L}{inf/NP2 \circ \Diamond\Box^{\downarrow}NP_2 \vdash inf}\,/L}{NP1 \circ (inf/NP2 \circ \Diamond\Box^{\downarrow}NP_2) \vdash NP_1 \bullet inf}\,\bullet R}{inv/(NP1 \bullet inf) \circ (NP1 \circ (inf/NP2 \circ \Diamond\Box^{\downarrow}NP_2)) \vdash inv}\,/L}{(inv/(NP1 \bullet inf) \circ (NP1 \circ inf/NP2)) \circ \Diamond\Box^{\downarrow}NP_2 \vdash inv}\,AR}{inv/(NP1 \bullet inf) \circ (NP1 \circ inf/NP2) \vdash inv/\Diamond\Box^{\downarrow}NP_2}\,/R}{Wh/(inv/\Diamond\Box^{\downarrow}NP_2) \circ (inv/(NP_1 \bullet inf) \circ (NP_1 \circ inf/NP_2)) \vdash Wh}\,/L}{What \circ (did \circ (John \circ like)) \vdash Wh}\,PF$$

[8]Vermaat (2006) assigns category $q/(NP \bullet inf)$, rather than $inv/(NP \bullet inf)$, to *do* in (303b). The reasoning here is that a sentence with a *do*-inversion but without a Wh-expression, such as *Did John do such a thing*, is already a (Yes-No) question. However, it is not only a (main-clause) Wh-phrase that selects the inverted expression, cf. *Rarely did John do such a thing*. So I use the category *inv* instead, which explicitly indicates that what is involved is just an auxiliary verb inversion.

[9]I omit the indices on logical and structural connectives unless they are essential.

b. What$_1$ did Meg think that John likes t_1?

$$\dfrac{\dfrac{Wh \vdash Wh \quad \dfrac{\dfrac{\dfrac{inv/(NP \bullet inf) \circ (NP \circ (inf/S \circ (S/S \circ (NP \circ (TV \circ \Diamond\Box^{\downarrow}NP))))) \vdash inv}{(inv/(NP \bullet inf) \circ (NP \circ (inf/S \circ (S/S \circ (NP \circ TV))))) \circ \Diamond\Box^{\downarrow}NP \vdash inv}\,AR}{inv/(NP \bullet inf) \circ (NP \circ (inf/S \circ (S/S \circ (NP \circ TV)))) \vdash inv/\Diamond\Box^{\downarrow}NP}\,/R}{Wh/(inv/\Diamond\Box^{\downarrow}NP) \circ (inv/(NP \bullet inf) \circ (NP \circ (inf/S \circ (S/S \circ (NP \circ TV))))) \vdash Wh}\,/L}{What \circ (did \circ (Meg \circ (think \circ (that \circ (John \circ like))))) \vdash Wh}\,PF$$

where $TV = (NP\backslash S)/NP$

In (304a), after merging '$inv/(NP \bullet inf)$' for *did* in the sixth line from the top, we apply AR twice, and then 'discharge' the hypothetical category '$\Diamond\Box^{\downarrow}NP_2$' (so that we can abstract away from the corresponding argument slot). Because AR in (301a) is not dependent on the mode of merge, this association can cross a tensed clause boundary. The derivation for (302b), as is shown in (304b), is basically the same as the one for (302a); the only difference is that we merge more items before we apply AR. In (304b), I have omitted the stage until I apply AR.

Next, I show how MMTLG in Vermaat (2006) can deal with islands. For an earlier analysis along this line of treatment of syntactic islands, see Morrill (1994: 218-224). I only show an instance of a complex NP island, but other islands can be dealt with in a similar way.

(305) *the man who likes*

$$\dfrac{\dfrac{NP \vdash NP \quad \dfrac{\dfrac{\dfrac{N \vdash N}{N \vdash N \quad (\Box^{\downarrow}N)^{\Diamond} \vdash N}\,\backslash L}{(N \circ N\backslash\Box^{\downarrow}N)^{\Diamond} \vdash N}\,\Box^{\downarrow}L \quad \dfrac{\dfrac{NP \vdash NP}{(\Box^{\downarrow}NP)^{\Diamond} \vdash NP}\,\Box^{\downarrow}L}{\Diamond\Box^{\downarrow}NP \vdash NP}\,\Diamond L}{(NP\backslash S)/NP \circ \Diamond\Box^{\downarrow}NP \vdash NP\backslash S}\,/L}{(N \circ ((N\backslash\Box^{\downarrow}N)/(NP_1\backslash S) \circ ((NP\backslash S)/NP \circ \Diamond\Box^{\downarrow}NP)))^{\Diamond} \vdash N}\,/L}{NP/N \circ (N \circ ((N\backslash\Box^{\downarrow}N)/(NP\backslash S) \circ ((NP\backslash S)/NP \circ \Diamond\Box^{\downarrow}NP)))^{\Diamond} \vdash NP}\,/L}{the \circ (man \circ (who \circ (likes \circ \epsilon)))^{\Diamond} \vdash NP}\,PF$$

In (305), the formula '$(N\backslash\Box^{\downarrow}N)/(NP\backslash S)$' which we assign to the relative pronoun *who* creates a boundary around the structure 'X' as in the form '$(X)^{\Diamond}$' in the antecedent structure. Given the inference rule $\Box^{\downarrow}L$ in (299b), we must add a boundary indicated by a pair of parentheses with '$\Diamond$' as a superscript (i.e. '$(\cdot)^{\Diamond}$') around the category '$\Box^{\downarrow}N$' when we introduce it in the antecedent of a sequent, as in '$(\Box^{\downarrow}N)^{\Diamond} \vdash N$' in the top-left side of the proof in (305). We can then create an internal structure inside this pair of parentheses by using $\backslash L$ or $/L$, generating sequents such as '$(B \circ \ B\backslash\Box^{\downarrow}N)^{\Diamond} \vdash N$' or '$(\Box^{\downarrow}N/A \circ \ A)^{\Diamond} \vdash N$,' where the

internal structure can become successively more complex, as we can see in (305). However, whatever additional category is introduced inside this special boundary, we cannot insert a hypothetical category in that position and abstract away from that position at a level larger than the boundary. In other words, even if we insert a hypothetical category in one of these argument positions in the antecedent sequent, as in '$C/N \circ (\Box^{\downarrow}N/A \circ \Diamond A)^{\Diamond} \vdash C$,' we cannot associate the structure beyond the boundary marked by '$(X)^{\Diamond}$.' That is, '$(C/N \circ \Box^{\downarrow}N/A)^{\Diamond} \circ \Diamond A \not\vdash C$.' And because association cannot cross this boundary, we cannot generate a category required by a Wh-expression, as in '$(C/N \circ \Box^{\downarrow}N/A)^{\Diamond} \not\vdash C/\Diamond A$.'

We can make the antecedent structure in (305) more complex by continuing the derivation until we merge the matrix auxiliary *did*, as shown in the top line of (306), but then we get stuck after a few applications of AR.

(306) *(What$_1$) did you know the man who likes t_1?

$$\dfrac{\dfrac{\dfrac{(inv/(NP \bullet inf) \circ (NP \circ (inf/NP \circ (NP/N \circ (N \circ ((N\backslash\Box^{\downarrow}N)/(NP_1\backslash S) \circ (TV \circ \Diamond\Box^{\downarrow}NP_2)))^{\Diamond})))) \vdash inv}{(inv/(NP \bullet inf) \circ (NP \circ (inf/NP \circ (NP/N \circ ((N \circ ((N\backslash\Box^{\downarrow}N)/(NP_1\backslash S) \circ TV)) \circ \Diamond\Box^{\downarrow}NP_2)^{\Diamond})))) \vdash inv}\ AR}{inv/(NP \bullet inf) \circ (NP \circ (inf/NP \circ (NP/N \circ (N \circ ((N\backslash\Box^{\downarrow}N)/(NP_1\backslash S) \circ TV))))) \not\vdash inv/(\Box^{\downarrow}\Diamond)NP}\ {}^{*}/R}{did \circ (you \circ (know \circ (the \circ (man \circ (who \circ likes))))) \not\vdash inv/\Diamond\Box^{\downarrow}NP}\ PF$$

where $TV = (NP\backslash S)/NP$

As we can see in the second line from the top, the hypothetical 'trace' category $\Diamond\Box^{\downarrow}NP$ can only reach the right periphery inside the parentheses marked by the Diamond operator, as in $((\cdots)\circ\Diamond\Box^{\downarrow}NP)^{\Diamond}$, and it cannot get out of the parentheses, for the reason that I have explained above. Thus we cannot abstract away from this hypothetical category to generate the argument category '$inv/\Diamond\Box^{\downarrow}NP$' for the string *did you know the man who likes*. Because the Wh-operator category in (303a) requires the category '$inv/\Diamond\Box^{\downarrow}NP$' as its argument, the derivation does not converge. As I have explained above, the boundary '$(X)^{\Diamond}$' has been introduced by the category of the relative pronoun *who*. We can set up other Wh-islands by assigning analogous categories to relevant lexical items. For further examples of other island creating items and proofs, see Chapters 7 and 8 of Morrill (1994) and Chapter 3 of Vermaat (2006).

Now, I briefly explain the use of a permutation rule so that we can ignore the presence of adjuncts when we discharge a hypothetical category. In (302a), the Wh-gap is in the right peripheral position, but this is not always the case.

(307) a. What$_1$ did you buy t_1 (yesterday)?

b. Who$_1$ did you meet t_1 (at the pub)?

To deal with cases as in (307), we need a controlled permutation rule as in (301b). Then the derivation converges as in (308).

(308) What$_1$ did you buy t_1 yesterday?

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{inv/(NP \bullet inf) \circ (NP \circ ((inf/NP \circ \Diamond\Box^{\downarrow}NP) \circ inf\backslash inf)) \vdash inv}{inv/(NP \bullet inf) \circ (NP \circ ((inf/NP \circ inf\backslash inf) \circ \Diamond\Box^{\downarrow}NP)) \vdash inv}\,PR}{(inv/(NP \bullet inf) \circ (NP \circ (inf/NP \circ inf\backslash inf))) \circ \Diamond\Box^{\downarrow}NP \vdash inv}\,AR}{Wh \vdash Wh \quad inv/(NP \bullet inf) \circ (NP \circ (inf/NP \circ inf\backslash inf)) \vdash inv/(\Diamond\Box^{\downarrow}NP)}\,/R}{Wh/(inv/\Diamond\Box^{\downarrow}NP) \circ (inv/(NP \bullet inf) \circ (NP \circ (inf/NP \circ inf\backslash inf))) \vdash Wh}\,/L}{What \circ (did \circ (you \circ (buy \circ yesterday))) \vdash Wh}\,PF$$

In this section, I have shown how we can set up the grammar system so that Wh-movement can cross a tensed clause. The association rule that is licensed by the presence of a hypothetical category '$\Diamond A$' in the in-situ position for the Wh-expression is not sensitive to the merge-mode specification, and thus, it can cross the minimal S category that contains the in-situ position, even though at that stage of derivation, a different mode of merge is introduced, as we have sketched in chapter 7.

Because the association rule introduced in this way is not sensitive to the merge mode specification, lack of locality is the default setting for this kind of extraction phenomena. However, we can set up islands by assigning categories in the form of '$(B\backslash\Box^{\downarrow}C)/A$' to 'island creating' items, such as relative pronouns (where B and A may be complex or empty). Presence of such a category in the antecedent structure creates a boundary around the structure 'X' that derives the category C, as in '$(X)^{\Diamond} \vdash C$.'

The next section deals with Wh-movement in the two kinds of ditransitive constructions.

8.4 Wh-movement in the distranstive constructions

Wh-movement in the two kinds of ditranstive constructions poses some problems for the proposed analysis. Look at the data in (309), with the suggested VP structures following my proposal in chapters 3, 4. Remember that *dtr* is the associate of a ditransitive verb which concatenates the NP objects into a complex NP object in the DO construction. It is not pronounced in the PF string. I omit the merge

mode indices.

(309) a. *Who(m)$_1$ did you $(give \circ (\Diamond t_1 \circ (dtr \circ the \cdot book)))$?

b. What$_1$ did you $(give \circ (Avril \circ (dtr \circ \Diamond t_1)))$?

c. What$_1$ did you $((give \circ (Avril \circ (dtr \circ \Diamond t_1))) \circ yesterday)$?

d. Who$_1$ did you $(give \circ ((the \cdot book) \circ (to \circ \Diamond t_1)))$?

e. What$_1$ did you $(give \circ (\Diamond t_1 \circ (to \circ Avril)))$?

If the trace category $\Diamond\Box^{\downarrow}NP$ is the right sister in its merge, as in (309b), (309c) and (309d), then we can either apply AR or PR in (301) and ultimately, we can place the category in the right periphery of the antecedent structure in a sequent. However, in my analysis, the left object NP is structurally the left sister in its merge for both the DO and the PP constructions, and, as we can see in (310), we cannot move $\Diamond\Box^{\downarrow}NP$ from that position to the right periphery position of the antecedent structure. Consider (310).

(310) a. $(give \circ (Avril \circ (dtr \circ \Diamond t_1))) \circ yesterday \vdash VP$

$\Rightarrow_{AR}$ $((give \circ (Avril \circ dtr)) \circ \Diamond t_1) \circ yesterday \vdash VP$

$\Rightarrow_{PR}$ $((give \circ (dtr \circ the \cdot book))) \circ yesterday) \circ \Diamond t_1 \vdash VP$

b. $give \circ (\Diamond t_1 \circ (dtr \circ the \cdot book)) \vdash VP$

$\not\Rightarrow$ $(give \circ \Diamond t_1) \circ (dtr \circ the \cdot book) \vdash VP$

c. $give \circ (\Diamond t_1 \circ (dtr \circ the \cdot book)) \vdash VP$

$\not\Rightarrow$ $give \circ ((dtr \circ the \cdot book) \circ \Diamond t_1) \vdash VP$

d. $give \circ (\Diamond t_1 \circ (to \circ Avril)) \vdash VP$

$\not\Rightarrow$ $(give \circ \Diamond t_1) \circ (to \circ Avril) \vdash VP$

e. $give \circ (\Diamond t_1 \circ (to \circ Avril)) \vdash VP$

$\not\Rightarrow$ $give \circ ((to \circ Avril) \circ \Diamond t_1) \vdash VP$

f. (Dead-end configuration) $A \circ (\Diamond C \circ B) \vdash D$

(310a) shows that if the hypothetical category marked by '$\Diamond$' is the right sister in the initial configuration, as in (309b), then we can apply AR and PR to place it in the right periphery position, before we abstract away from this category by '$/R$.' (310b) shows that we cannot use AR to rebracket and put the hypothetical category $\Diamond t$ as the right sister of *give*. If we could do this, we might then apply PR to permute $\Diamond t$ with $(dtr \circ the \cdot book)$, to place $\Diamond t$ in the right periphery position, but this is impossible because the first step does not converge. (310c) shows that we cannot put $\Diamond t$ to the right of $(dtr \circ the \cdot book)$ by applying PR to the initial

configuration, either, because PR cannot permute between the sisters. (310d) and (310e) show that for the same reasons, Wh-movement out of the left object position is impossible in the PP ditransitive construction as well, given my treatment of PP ditransitive sentences as in chapter 4 and 5.

To summarize the current situation, the system cannot license Wh-movement of the left object in the DO construction as in (309a), whereas, (309b), (309c) and (309d) are all predicted to be grammatical. These predictions are all empirically supported. However, the problem is that we should be able to license Wh-movement of the left object in the PP construction, as in (309e), whereas (310d) and (310e) suggest that we cannot do this in the PP construction structure that I have used. At the moment, the hypothetical category $\Diamond t$ (which abbreviates $\Diamond\Box^{\downarrow}NP$) in (309) is the left sister in its merge both in (309a) and in (309e) in my analysis and the challenge is to apply permutation (and association) only in the latter case, that is, only with regard to the left object in the PP construction, not in the DO construction.

In 8.4.1, I provide a provisional solution to this problem with some comments.

8.4.1 VP adjunct analysis

I explain why Wh-movement of the left object is possible in the PP construction by postulating a structural ambiguity for (309e), as shown in (311) and (312). I add numbers as subscripts to the verbs and the prepositions for presentation reasons.

(311) PP ditransitive construction.

What$_i$ did you ($give_1 \circ (\Diamond t_i \circ (to_1 \circ Avril))$)?

The VP structure in (311) is the same as the one for the PP ditransitive construction in chapters 4 and 5. That is, the verb $give_1$ in (311) is the same as what was used for the PP ditransitive construction and its category is $(NP\backslash S)/(NP \bullet NP)$. The same applies to the preposition to_1, whose category is $(NP2\backslash(NP1 \bullet NP2))/NP1$.

(312) VP adjunct construction.

What$_i$ did you ($(give_2 \circ \Diamond t_i) \circ (to_2 \circ Avril)$)?

The VP adjunct structure in (312) is the same as the one for the VP adjunct structure in chapter 6 (cf. (233)). The item $give_2$ in (312) is analyzed as having a transitive category $(NP\backslash S)/NP$. The preposition to_2 in (312) is a common preposition which heads a PP as a VP adjunct (i.e., its category is $(VP\backslash VP)/NP$,

where VP abbreviates $NP\backslash S$). See discussion surrounding (320) for some additional motivation for the use of this adjunct structure.

Though I do not provide the exact semantics of the adjunct structure in (312), the semantic expression for the PP to_2 *Avril* would be a VP modifier of type ((et)(et)). Thus, the semantics of this structure would be different from the semantics of the PP ditransitive construction as in (311), in which the preposition to_1 simply combines the two object NPs into one complex NP object.

In section 6.3.3 of chapter 6, the PP-as-adjunct structure was considered to be the basic structure and an aim of the analysis was to derive the PP-as-complement structure from this more basic structure (though we could not do it within NL). In contrast, the complement-adjunct ambiguity in this chapter arises because of the pure lexical ambiguity of the head verb, that is, between $give_1$ and $give_2$, and no attempt is made to derive one from the other (also, $give_1$, which has the ditransitive verb category, counts as more basic). Thus the relation between the PP-as-adjunct structure to the other structure (that is, the PP-as-complement structure in chapter 6 and the PP ditransitive construction in this chapter) is quite different in the two cases.

The structural association rule AR in (301a) which is triggered by the lexical category of Wh-expressions allows us to derive Wh-movement from the right peripheral expression. To deal with cases such as *What*$_1$ *did you buy* t_1 *yesterday* or *Who*$_1$ *did you meet* t_1 *at the pub* in (307), I have adopted the structural permutation rule PR which is also licensed by the unary operator $\Diamond$, as in (301b), repeated here as (313).

(313) PR:

$$\frac{(X \circ \Diamond A) \circ Y \vdash B}{(X \circ Y) \circ \Diamond A \vdash B}\, PR$$

Given PR, and with the above two structures in (311) and (312), the derivation converges only for the adjunct structure, as is shown in (314).

(314) *What*$_1$ *did you give* t_1 *to Avril?*

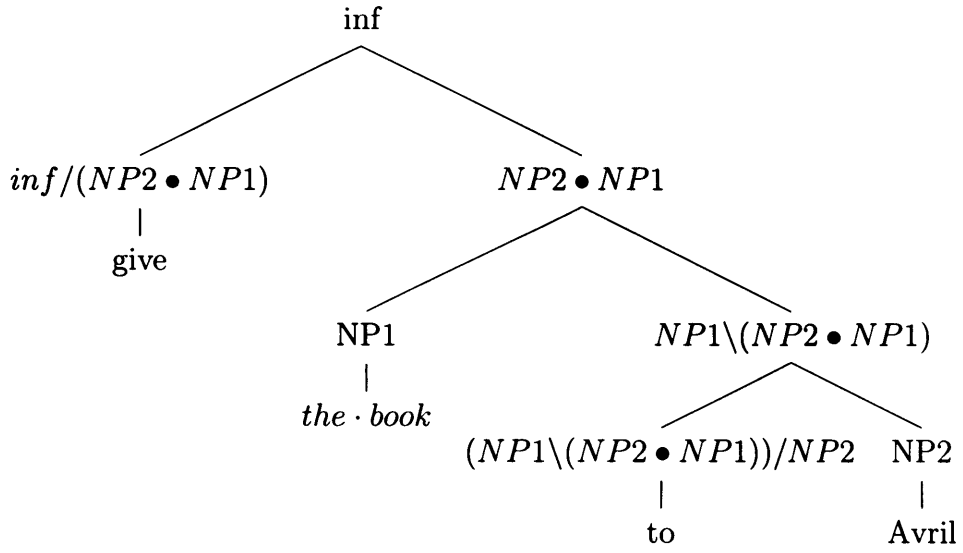
a. PP ditransitive structure: No conversion.
$give \circ (\Diamond t_1 \circ (to \circ Avril)) \not\vdash give \circ ((to \circ Avril) \circ \Diamond t_1)$.

b. Adjunct structure: Converge.
$(give \circ \Diamond t_1) \circ (to \circ Avril) \vdash_{PR} (give \circ (to \circ Avril)) \circ \Diamond t_1$

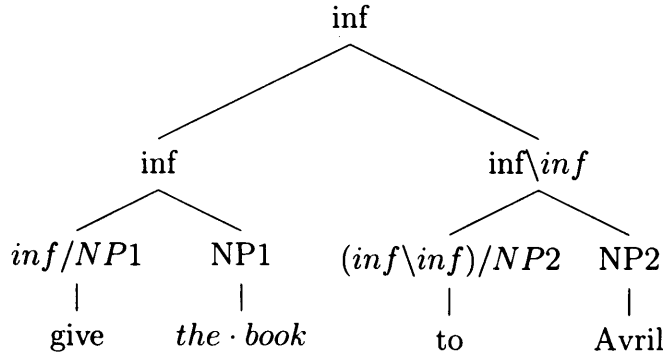
Note that the the derivation still does not converge for the PP ditransitive construction; we need to use the adjunct structure which places the hypothetical category in the correct location for PR to apply. Postulated structural ambiguity inside VP (or inf) is as in (316)~(317) for the sentence in (315):[10]

(315) Tom gave the book to Avril.

(316) VP structure for the PP ditransitive construction (which does not converge for (309e)).



(317) Alternative VP structure required for the application of PR for (309e).



A merit of this analysis is that we do not need anything more than the structural rules in (301) together with adequate Wh-categories. On the other hand, treating verbs such as *give* and *show* as lexically ambiguous between the ditransitive category $inf/(NP \bullet NP)$ (or $(NP\backslash S)/NP$ if finite) and the transitive category inf/NP (or $(NP\backslash S)/NP$ if finite) requires some more support.

[10]Both structures are before the application of AR or PR, that is, the same structures as the ones for the sentences which do not involve 'movement.'

As we have already seen, the reflexive binding asymmetry between the two objects at least does not require the VP adjunct structure in (317) above. Consider the reflexive binding sentences that we have seen before.

(318) a. John showed Avril to herself.

b. *John showed herself to Avril.

Only the PP ditransitive structure in (316) above licenses the correct binding relation in (318a). The question is whether the alternative adjunct structure in (317) together with the binding system that is proposed in chapters 3 and 4 licenses the unattested reflexive binding in (318b). It it did, it would provide a strong reason for not using the adjunct structure in my analysis.

Fortunately, the adjunct structure does not license the unattested binding in (318b), given my analysis of binding, and thus, (318b) does not prohibit us from postulating the adjunct structure in (317) above. Without going into details, a reflexive pronoun identifies itself with a co-argument of its functor with which the reflexive is merged sooner than the co-argument. In the alternative structure in (317), the two object NPs are no longer co-arguments of the same functor. That is, NP1 is the internal argument of the ditransitive verb *give*, and NP2 is the first argument of the prepositional functor *to*, where the PP functor *to Avril* takes the VP (i.e. *inf*) *give the book*, rather than the left object *the book*, as its second argument. Thus, the alternative structure does not generate the wrong reflexive binding relation as in (318b) when we put a reflexive in the left object position (that is, in the position of NP1 in the adjunct structure in (317) above).

To be confusing a little, there are some sentences which might support this 'wrong' binding relation.

(319) a. Who_1 did you introduce t_1 to [every student]$_2$?

b. His_2 teacher.

c. I returned his homework to each of the boys who came to collect it. (Cormack, pc)

Thus, the verdict is not clear for the adjunct structure in (317) with regard to binding data, but I leave this investigation for further research.

If some ditransitive verbs can be treated as syntactically transitive, then we expect that the adjunct to the VP in the VP adjunct construction can be omitted in the right context. This seems to be the case for *give*. On the other hand, the right object does not seem to be omittable in the DO construction, irrespective of the context. This contrast follows from the assumption that the adjunct structure

as in (317) is possible only if the verb is assigned the transitive verb category (which is only compatible with the PP as a VP adjunct). Consider (320).

(320) a. $[_{VP}[_{VP}Verb \circ Object] \circ PP]$

b. $[_{VP}[_{VP}Verb \circ [Object1 \circ [dtr/to \circ Object2]]$

c. John gave the book ?(to Avril).

d. John gave a speech.

e. John gave Meg *(the book).

The VP adjunct structure that I have postulated for sentences such as *Tom gave the book to Avril* is as in (320a). Because an adjunct is normally omissible without changing the grammaticality of the sentence, we expect the PP *to Avril* to be omissible in this structure. On the other hand, in my analysis of ditransitive verbs, we can only generate the structure as in (320b) with a ditransitive verb category, either for the DO or for the PP construction. Because the DO construction does not have the structure in (320a) as an option,[11] the analysis predicts that the right object is not omissible in the DO construction. Though (320c) without the 'adjunct' PP *to Avril* is not perfect as an English sentence, it is much better than the impossible sentence **John gave Meg*, as is shown in (320e). That is, it is easy to think of a context in which we can naturally use (320c), whereas that is not the case with (320d).[12] Also, there are some cases in which the PP is genuinely optional, as in (320d), though these cases might be idiomatic and therefore might better be considered separately. To summarize, the omission possibility of the right object (and the preposition) for a sentence such as *John gave the book (to Avril)*, as opposed to *John gave Avril *(the book)*, suggest that the adjunct structure is well-motivated as an alternative for the PP ditransitive structure.

As a diversion, I briefly discuss further restrictions that we might assign to PR. Permutation, as opposed to association, affects the left to right PF linear order. As far as the data that we have dealt with are concerned, PR has not overgenerated PF strings, but with interaction of other structural rules that we may add in the future, we might need to restrain it more. Given that all the cases we have seen so far involve an adjunct category to the right, we might want to use this as a

[11] If we assigned the category $((NP\backslash S)/NP)/NP$ to *give*, this would generate the VP structure $((give \circ Avril) \circ (the \cdot book))$, which would then license 'Wh-movement' from the position of *Avril*, but I do not use the curried ditransitive category in my analysis, and thus, this structure is not available for the DO construction via this assignment, either.

[12] For example, when we are trying to find out which specific object (such as the book, the clock, the doll) John gave to Meg, we can easily omit the PP *to Meg*, whereas even when we are trying to find out to which person John gave the book, (320e) without *the book* is not acceptable.

further restriction on PR. In MMTLG, there is a way of explicitly incorporating this 'adjunct structure' requirement for applying the permutation rule.

(321) a. PR_{ad}: $(A \circ \Diamond C) \circ_{ad} B \vdash D$
$\Rightarrow_{PR_{ad}} (A \circ_{ad} B) \circ \Diamond C \vdash D$

b. $(VP/NP \circ \Diamond\Box^{\downarrow}NP) \circ_{ad} Adverb \vdash VP$
$\Rightarrow_{PR_{ad}} (VP/NP \circ Adverb) \circ_{ad} \Diamond\Box^{\downarrow}NP \vdash VP$

c. $(VP/NP \circ \Diamond\Box^{\downarrow}NP) \circ_{ad} PP \vdash VP$
$\Rightarrow_{PR_{ad}} (VP/NP \circ PP) \circ_{ad} \Diamond\Box^{\downarrow}NP \vdash VP$

In (321a), permutation is controlled in two ways. First, as in PR in (301b), the permutable category C must be marked with $\Diamond$. Second, the B category has to be of an adjunct category, because of the adjunct merge mode *ad*. This extra constraint on the permutation rule is not required to deal with the basic Wh-movement cases in (309), but we might need this to avoid overgeneration that PR in (301b) may induce. I leave the evaluation of this additional possibility for further research.

I show some sequent proofs for Wh-extraction of the left object in the DO construction and the PP construction.

(322) What$_1$ did you give t_1 to Avril?

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{inv \vdash inv} \quad \dfrac{\dfrac{\dfrac{\dfrac{}{inf \vdash inf} \quad \dfrac{}{\Diamond\Box^{\downarrow}NP_1 \vdash NP_1}}{inf/NP1 \circ \Diamond\Box^{\downarrow}NP_1 \vdash inf}/L \quad \dfrac{}{inf \vdash inf}}{(inf/NP1 \circ \Diamond\Box^{\downarrow}NP_1) \circ inf\backslash inf \vdash inf}\backslash L \quad \dfrac{}{NP2 \vdash NP2}}{(inf/NP1 \circ \Diamond\Box^{\downarrow}NP_1) \circ ((inf\backslash inf)/NP_2 \circ NP_2) \vdash inf}/L}{NP \circ ((inf/NP1 \circ \Diamond\Box^{\downarrow}NP_1) \circ ((inf\backslash inf)/NP_2 \circ NP_2)) \vdash NP \bullet inf}\bullet R}{inv/(NP \bullet inf) \circ (NP \circ ((inf/NP1 \circ \Diamond\Box^{\downarrow}NP_1) \circ ((inf\backslash inf)/NP_2 \circ NP_2))) \vdash inv}/L}{inv/(NP \bullet inf) \circ (NP \circ ((inf/NP1 \circ ((inf\backslash inf)/NP2 \circ NP2)) \circ \Diamond\Box^{\downarrow}NP_1)) \vdash inv}PR}{(inv/(NP \bullet inf) \circ ((NP \circ ((inf/NP1 \circ ((inf\backslash inf)/NP2 \circ NP2)))) \circ \Diamond\Box^{\downarrow}NP_1 \vdash inv}AR}{inv/(NP \bullet inf) \circ ((NP \circ ((inf/NP1 \circ ((inf\backslash inf)/NP2 \circ NP2))) \vdash inv/\Diamond\Box^{\downarrow}NP_1}/R}{did \circ (you \circ (give \circ (to \circ Avril))) \vdash inv/\Diamond\Box^{\downarrow}NP_1}PF$$

We can derive the proper argument category $inv/\Diamond\Box^{\downarrow}NP_1$ which the Wh operator of category $Wh/(inv/\Diamond\Box^{\downarrow}NP)$ selects. Thus, Wh-movement of the (left) object is grammatical in the PP construction.

On the other hand, we cannot have the right configuration to apply PR with regard to the left object in the DO construction. I split the proof in (323a) into two parts.

(323) *Who$_1$ did you give t_1 the book?

a.

$$\dfrac{\dfrac{\dfrac{\Diamond\Box^{\downarrow}NP_1 \vdash NP_1 \quad (NP1 \bullet NP2) \vdash (NP1 \bullet NP2)}{\Diamond\Box^{\downarrow}NP_1 \circ NP_1\backslash(NP_1 \bullet NP_2) \vdash (NP_1 \bullet NP_2)}\backslash L \quad NP2 \vdash NP2}{\Diamond\Box^{\downarrow}NP_1 \circ ((NP_1\backslash(NP_1 \bullet NP_2))/NP_2 \circ NP_2) \vdash (NP_1 \bullet NP_2)}/L$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{inf \vdash inf \quad \Diamond\Box^{\downarrow}NP_1 \circ (DTR \circ NP_2) \vdash (NP_1 \bullet NP_2)}{inf/(NP1 \bullet NP2) \circ (\Diamond\Box^{\downarrow}NP_1 \circ (DTR \circ NP_2)) \vdash inf}/L \quad NP \vdash NP}{NP \circ (inf/(NP1 \bullet NP2) \circ (\Diamond\Box^{\downarrow}NP_1 \circ (DTR \circ NP_2))) \vdash NP \bullet inf}\bullet R \quad inv \vdash inv}{inv/(NP \bullet inf) \circ (NP \circ (inf/(NP1 \bullet NP2) \circ (\Diamond\Box^{\downarrow}NP_1 \circ (DTR \circ NP_2)))) \vdash inv}/L}{did \circ (you \circ (t_1 \circ ((dtr \circ the \cdot book)))) \vdash inv}PF$$

where $DTR = (NP1\backslash(NP1 \bullet NP2))/NP2$ and *dtr* is PF null

b.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{inf \vdash inf \quad \Diamond\Box^{\downarrow}NP_1 \vdash NP_1}{inf/NP1 \circ \Diamond\Box^{\downarrow}NP_1 \vdash inf}/L \quad NP2 \vdash NP2}{((inf/NP2)/NP1 \circ \Diamond\Box^{\downarrow}NP_1) \circ NP_2 \vdash inf}\bullet R \quad NP \vdash NP}{NP \circ (((inf/NP2)/NP1 \circ \Diamond\Box^{\downarrow}NP_1) \circ NP_2) \vdash NP \bullet inf}\bullet R \quad inv \vdash q}{inv/(NP \bullet inf) \circ (NP \circ (((inf/NP2)/NP1 \circ \Diamond\Box^{\downarrow}NP_1) \circ NP_2)) \vdash inv}PR}{inv/(NP \bullet inf) \circ (NP \circ (((inf/NP2)/NP1 \circ NP2) \circ \Diamond\Box^{\downarrow}NP_1)) \vdash inv}AR}{(inv/(NP \bullet inf) \circ ((NP \circ (inf/NP2)/NP1 \circ NP2)) \circ \Diamond\Box^{\downarrow}NP_1 \vdash inv}/R}{inv/(NP \bullet inf) \circ (NP \circ ((inf/NP2)/NP1 \circ NP2)) \vdash inv/\Diamond\Box^{\downarrow}NP_1}PF}{did \circ (you \circ (give \circ (the \cdot book))) \vdash inv/\Diamond\Box^{\downarrow}NP}$$

In the lower proof in (323a), the configuration in the Antecedent of the final sequent (i.e. the sequent just above the PF sequent) is $A \circ (B \circ (C \circ (\Diamond D \circ (DTR \circ E))))$ and $\Diamond\Box^{\downarrow}NP_1$ is the left sister in the second deepest position (that is, the position of $\Diamond D$ in the schematic configuration). Thus, we cannot apply the structural permutation PR. In (323b), we can apply PR after the fifth line from the top, but this proof is only possible with the assignment of the curried category, '$(inf/NP)/NP$' (or the category '$((NP\backslash S)/NP)/NP$' if we do not have the auxiliary *do*) to the verb *give*. My analysis only assigns the uncurried '$inf/(NP \bullet NP)$' (or '$(NP\backslash S)/(NP \bullet NP)$' if the verb is finite) to ditransitive verbs (unless they are treated as transitive verbs of category $(NP\backslash S)/NP$) and so this derivation is not available. Thus, the proposed analysis with the structural rules in (301b) and the availability of the alternative VP adjunct structure for the PP construction correctly licenses Wh-movement of the left object only in the PP construction, whereas the Wh-movement of the right

object is licensed for both the constructions (we either do not need permutation in this case, or we can apply permutation with regard to an adjunct such as *yesterday* in (308)).

8.5 Conclusion

In this chapter, I discuss Wh-movement from two viewpoints. First, based on the linguistic observation that Wh-movement can cross a tensed clause boundary, whereas QNP scope cannot, I pursued a different way of introducing a structural association rule (as well as a permutation rule) compared with how we introduced a structural rule for complex predicate formation. Roughly, the distinction in formalism is expressed in terms of the mode un-specific nature of structural rule application for Wh-expressions on the one hand, and mode-specific nature of the association rule for complex predicate formation on the other.

To explain the asymmetry of Wh-extraction of the left object in the prepositionless double object construction and in the ditransitive construction with a preposition, I have postulated multiple category assignments to normally ditransitive verbs such as *give* and *show*, assigning both the ditransitive verb category and the transitive verb category to these verbs. I leave the investigation of further justification for this postulated lexical/structural ambiguities for future research.

In chapter 9, I explain why extra scope taking behaviors of indefinites do not count as a counter example of the tensed clause locality of QNP scope.

Chapter 9

Indefinites: an extra argument slot analysis

9.1 Introduction

As we have seen, the scope of strong quantifiers such as universal quantifiers is roughly clause bound. However, the scope of indefinites is apparently not constrained in this way.

(324) a. Every teacher said that *a student* smoked at school. $\forall > \exists$; $\exists > \forall$

b. A teacher said that *every student* smoked at school. $\exists > \forall$; $*\forall > \exists$

(324a) has a reading which says that there is one student about whom all the teachers said that he smoked. This so-called 'wide scope reading' of indefinites is problematic because the universal quantifier in the embedded clause in (324b) cannot take scope over the main clause. (324b) does not have the reading that says that for each student, a possibly different teacher said that the student smoked. Based on covert quantifier raising (QR) in May (1977), Generative grammarians might simply postulate an exceptional long-distance movement that applies only to indefinites, but not to universal QNPs (cf. Beghelli and Stowell (1997)). But this would lose the uniformity of QR as syntactic movement. Alternatively, Reinhart (1997) uses the idea of Choice Functions, which can generate the reading that is roughly equivalent to the exceptional scope reading of indefinites without using an exceptional movement.[1]

However, Winter (2001) argues that neither the QR analysis nor the simple choice function analysis can explain a certain reading of the following sentence (cf.

[1]Reinhart maintains clause internal QR to explain the genuine QNP scope.

Winter 2001: 116).[2]

(325) Every boy_1 who hates [NP a (certain) woman he_1 knows] will develop a serious complex.

(325) has a reading in which each boy develops a complex if he hates a certain kind of woman that he knows, such as each boy's mother. This reading is problematic because it is different from either the narrow scope reading of the indefinite relative to the universal QNP, in which each boy develops a complex if he hates any woman that he knows, or the wide scope reading in which there is one woman such that all the boys can develop a complex if they hate the woman.[3] In order to deal with this reading, Winter (2001) 'Skolemizes' his choice function, which generates a reading in which a particular relation holds for all the *boy-woman* pairs (e.g. the mother-hood relation). Winter's analysis is empirically adequate, but because it is still a choice function analysis, it requires an existential closure operation on the choice/Skolem function variable. This operation is not constrained by standard syntactic islands, just like an exceptional QR for indefinites. If an alternative theory can explain the various readings of indefinites without assuming a syntactically unconstrained operation, that theory is preferable. Also, choice/Skolem function analyses make use of type ((et)e) function variables, to which we need to assign the choice/Skolem function property. This makes it difficult to follow Chomsky's principle of Inclusiveness, which expects us to derive all the information from the lexical items. Moreover, as I show in section 9.4, according to Winter's analysis, the binder/antecedent of a bound pronoun which appears in the nominal restriction of an indefinite cannot directly bind the argument slot that the pronoun introduces into the logical form. Given that the binders can directly bind the argument slots introduced by bound pronouns in simpler sentences, I argue that this goes against the spirit of semantic compositionality.

My QNP scope analysis keeps the scope of a QNP within the minimal S expression (or the minimal tensed clause) that contains the QNP, and thus, I need to explain the alleged exceptional scope of indefinites in some other way than by using my scope switch system. Using choice functions to explain such exceptional wide scope could be a possibility. However, choice functions are problematic as I have explained above. Also, introduction of choice functions and existential closure on

[2]Following Winter, I use the word *certain* to facilitate the 'exceptional wide scope' reading of indefinites. A few speakers do not get the exceptional scope reading at all when the indefinite is without *certain*.

[3]The narrow scope reading of the indefinite is possible only without the word *certain*. (325) does not have the indefinite wide scope reading because the pronoun inside the indefinite is bound by the universal quantifier.

choice function variables into our theory adds an extra complexity to the syntactic system. Because of these, I take an alternative approach. First, I adopt the basic idea of the domain restriction analysis of indefinites as in Schwarzschild (2002). According to the analysis, the exceptional wide scope of indefinites is not a matter of QNP scope. Instead, the reading in question arises when the set denoted by the nominal restriction of indefinites, such as the set of women for the indefinite *a woman*, is pragmatically restricted to a singleton set (i.e. a set that contains only one member).[4] In this analysis, we can talk about a particular individual by using an indefinite without accepting the exceptional quantificational scope of the indefinite. This analysis has an obvious merit because we can assume that the existential scope of indefinites is clause bound, like the scope of universal QNPs. On the other hand, there are some sentences that require the domain restriction of the indefinite to be dependent on some other element in the sentence. In order to explain this domain dependency of indefinites, I argue that the indefinite is lexically equipped with an extra argument slot which can be bound by another element that is merged later than the indefinite. Because the syntactic configurations that license such dependency relation have a certain similarity to the syntactic configurations licensing pronominal binding, I use the algorithm that I have explained for deriving pronominal binding in chapter 3 to derive the domain dependency of indefinites.[5]

Based on this analysis of indefinites, I show how we can compositionally derive the logical form of a sentence containing an indefinite in a Type Logical Grammar derivation. I reformulate Jacobson's analysis of pronoun binding (e.g. Jacobson (1999)) in Multi-Modal Type Logical Grammar as in Moortgat (1997) and show how we can percolate the extra argument slot of the indefinite into a later stage of the derivation and then have it bound by a quantifier.

In section 9.2, I explain the basic idea of choice functions. Section 9.3 introduces a problematic sentence in which an indefinite has a bound pronoun in its nominal restriction. As I have indicated above, Winter uses this sentence to motivate the use of his Skolemized choice functions. However, in section 9.4, I argue that Winter's analysis exhibits some compositionality problem to deal with this sentence. I also

[4]Strictly speaking, in the semantics that I have sketched in chapter 1, the nominal restriction *woman* denotes a function that maps individuals to truth values, rather than a set of individuals. But given the one-to-one correspondence between sets and their characteristic functions, I sometimes refer to the 'nominal restriction sets' of QNPs in my informal analysis.

[5]Kempson, Meyer-Viol, and Gabbay (2001) and Kempson and Cann (2006) analyze the whole phenomena of QNP scope switch in terms of the anaphoric dependency of indefinites to some other elements. In contrast, I preserve argument slot raising as an independent algorithm for explaining QNP scope. A comparison would be interesting.

explain more fundamental reasons why I do not adopt his choice/Skolem function analysis. In section 9.5, I explain the basic idea of domain restriction analyses, and argue that indefinites have an extra argument slot. Section 9.6 formalizes this idea in Type Logical Grammar, and applies it to a sentence which contains an indefinite with a bound pronoun in its nominal restriction, in order to show that we can compositionally derive an adequate logical form for Winter's sentence above. Section 9.7 discusses extensions and Section 9.8 provides concluding remarks.

9.2 Choice function

This section explains the basic idea of choice functions. A major motivation for the use of choice function is the exceptional scope taking of indefinites, as opposed to universal quantifiers which do not show exceptional scope. Consider the sentences in (324a) again, repeated here as (326a) and (327a). To represent quantifiers in the semantic terms, Winter uses (higher order) predicate logic notations. To make the comparison between Winter's analysis and my analysis clearer, I use the predicate logic notations, as in (326b) and (327b), in all the sections of this chapter except for section 9.6.

(326) a. Every teacher said that a student smoked at school.

b. $\exists x[student'(x) \ \& \ \forall y[teacher'(y) \rightarrow say'(smoke'(x))(y)]]$

(327) a. A teacher said that every student smoked at school.

b. $*\forall x[student'(x) \rightarrow \exists y[teacher'(y) \ \& \ say'(smoke'(x))(y)]]$

Again, in (326), the indefinite in the embedded tensed clause seems to have wide scope over the universal QNP in the main clause. In contrast, a universal QNP inside the embedded tensed clause in (327b) cannot easily take wide scope over the main clause indefinite.

Instead of extending the definition of QR to accommodate the exceptional behavior of indefinites, Reinhart (1997) and Winter (1997) explain the alleged exceptional scope reading of indefinites by using a fundamentally different mechanism, that is, 'choice functions.' The underlying idea is that the 'scope' of indefinites is different from the scope of universal QNPs because they use different mechanisms.

Informally, choice functions apply to sets of individuals denoted by nominal restrictions and choose a member from each set, if the set is not empty.[6] Win-

[6]I ignore the empty set problem of choice functions. Because of this, I do not apply another function of type $(et)e \rightarrow (et)t$ to the function variable f to generate a generalized quantifier as Winter does. The difference does not influence my arguments against Winter's analysis.

ter defines the property of being a choice function as 'CH' in Winter (2001: 89). Because I distinguish logical language expressions from their model theoretic denotations, I represent Winter's choice function property as the denotation of the functor expression 'CH'' of type (((et)e),t), as shown in (328). '$CH'(f)$' means that 'f is a choice function.'

(328) The denotation of $CH'_{((et)e)t}$

For all f of type (et)e, and for all A of type (et) where there exists at least one individual d in the model M such that $[\![A]\!]^M(d)$=1:

$[\![CH'(f)]\!]^M$ =1 iff $[\![A(f(A))]\!]^M$ =1 (cf. Winter 2001: 89)

In (328), the meta-variable 'A' is for some type (et) expression. For Winter, 'A' denotes a set of individuals, such as the set of students for *student'*.[7] Because each type (et) expression denotes a function from individuals to truth values in my semantics, in (328), I have expressed Winter's non-empty set requirement for his set-denotation of 'A' as the equivalent requirement on the function denoted by each 'A.'[8] However, for the reasons I have explained in footnote 4, I continue using set-based interpretations in informal explanations from now on as well. A function f of type (et)e maps non-empty sets of individuals to individuals. We need to assign the choice function property CH' as in (328) to the function f, so that f maps a set of individuals to a member of the set. Note that without this restriction, f might map a set of individuals to an individual that is not a member of the set. In (329b) and (329c) below, the function f with the choice function property CH' chooses a student from the set of students, and the chosen student acts as the type e argument of the functor expression *smoke'*.

(329) a. Every teacher said that some/a student smoked at school.

b. $\exists f_{(et)e}[CH'(f)\ \&\ \forall x[teacher'(x) \rightarrow say'([smoke'(f(student'))]_t)(x)]]$

c. $\forall x[teacher'(x) \rightarrow say'([\exists f_{(et)e}[CH'(f)\ \&\ smoke'(f(student'))]]_t)(x)]$

(329b) means that there is a function f such that f is a choice function and for each teacher x, x said that the student that f picks out smoked at school. Because the existential force associated with the function variable is outside the scope of the universal quantifier, f is the same function for all the teachers, and for all the teachers, it chooses the same individual out of the student set. This corresponds to the 'wide scope reading' of the indefinite, though in this choice function analysis,

[7]Again, though Winter does not distinguish typed lambda terms from their model theoretic denotations, I distinguish the two.

[8]In Winter's notation, the restriction on the arguments for the choice function 'f' is expressed as, '$A \neq \emptyset$' (where 'A' meta-represents a set of individuals).

the indefinite does not 'move' to a higher position to take wide scope. The idea is to leave the nominal expression *a student* and the function variable f in the base position of the indefinite noun phrase, whereas an existential closure can be introduced at various positions in the structure that have scope over the in-situ function variable.[9] In (329c), the existential closure on the function variable f is introduced within the scope of the universal quantifier. This means that for each teacher x, there is a possibly different choice function f involved. Because each f can choose a different student out of the student set, the identity of the student can co-vary with the teacher. This corresponds to the narrow scope reading of the indefinite.

In the next section, I introduce the main issue in this chapter together with Winter's solution.

9.3 Indefinites with a bound variable

Winter (1997) and Winter (2001) argue that the definition of choice functions in (328) cannot explain a particular reading of an indefinite noun phrase with a bound variable in the nominal restriction.

(330) a. Every boy_1 who hates [NP a (certain) woman he_1 knows] will develop a serious complex. (cf. Winter 2001: 116)

b. For each boy x, there is a (different) specific woman y among the women x knows such that if x hates y, x will develop a serious complex.

(330a) has the reading (330b). In this reading, the woman involved can co-vary with each boy, but this is not the ordinary narrow scope reading. Which woman we choose for each boy is relevant to the truth condition. That is, each boy will develop a complex only if he hates a woman who falls under a specific relation to him, for example, his mother, not if he hates some woman or other whom he knows.

We need to clarify the strange 'specificity' in (330b), in which the woman is not just some woman or other but can still co-vary with each boy. We could explain this specificity simply in terms of a different specific woman for each boy, such as Meg

[9]Winter (2001) does not specify at what level of representation an existential closure applies. In this chapter, I assume for convenience that it is introduced at LF. The function variable f is encoded with a phonologically null determiner head that selects the indefinite NP *a student* as its complement. The indefinite *a student* in this analysis denotes the set of students. Winter adopts this complex DP structure for explaining different behaviors of indefinites in different syntactic positions (see Winter 2005: 770 for details).

for Jack, Nancy for Sid, etc. But Cormack (personal communication) says that if she forces herself to get this reading, it has to be the case that a fixed relation holds for every pair of a boy and a woman, for example, the relation between a boy and his mother. Winter (2004) assumes that the reading (330b) is defined in terms of a function that maps the set of women each boy knows to another function that maps each boy to a member of that set. In some contexts, this second function can be understood as the fixed relation holding for every pair of a boy and the woman for him, such as the motherhood relation. In section 9.5, I adopt this assumption in a framework different from Winter's. In this section, however, I explain the strange specificity rather informally. The woman is specific in the sense that each boy is paired with only one truth-conditionally relevant woman. That is, it is not just an arbitrarily chosen woman. But the relevant woman can still co-vary with each boy. As Winter points out, neither the wide scope nor the narrow scope choice function logical form represent this reading with a type ((et)e) choice function.

(331) a. Choice function wide scope:
$\exists f_{(et)e}[CH'(f)\ \&\ \forall x[[boy'(x)\ \&\ hate'(f(\exists y.[woman'(y)\ \&\ know'(y)(x)])(x)]$
$\rightarrow develop\text{-}complex'_{et}(x)]]$

b. Choice function narrow scope:
$\forall x[[boy'(x)\&\exists f_{(et)e}[CH'(f)\ \&\ hate'(f(\exists y.[woman'(y)\ \&\ know'(y)(x)]))(x)]]$
$\rightarrow develop\text{-}complex'_{et}(x)]$

(332) a. $\exists y[woman'(y)\ \&\ \forall x[[boy'(x)\ \&\ know'(y)(x)\ \&\ hate'(y)(x)]$
$\rightarrow develop\text{-}complex'_{et}(x)]]$

b. $\forall x[[boy'(x)\ \&\ \exists y[woman'(y)\ \&\ know'(y)(x)\ \&\ hate'(y)(x)]]$
$\rightarrow develop\text{-}complex'_{et}(x)]$ (cf. Winter 2001: 116)

Neither the indefinite wide scope logical form in (331a) nor the corresponding classical indefinite wide scope logical form in (332a) represents the required interpretation. (331a) says that there is a choice function f such that for every boy x, if x hates the individual y that f chooses from the set of women x knows, x develops a complex. Consider a context in which all the boys happen to know exactly the same set of women. In this context, the logical form in (332a) means that the function f chooses one and the same woman for all boys, and that if each boy x hates that woman, x develops a complex. Note that there is only one function f involved for all the boys in (331a) because the existential quantifier binding f takes wide scope over the universal quantifier. If the function f is one and the same, and

the set from which f chooses an individual is one and the same, f chooses one and the same individual for all the boys. But the reading in (330b) implies that even if all the boys know exactly the same set of women, we should still be able to choose a different specific woman for each boy. For example, for each boy, we can choose his mother.

Note also that neither the indefinite narrow-scope logical form of the choice function analysis, given in (331b), nor its truth conditional equivalent in the classical notation, given in (332b), represents the reading in (330b). These narrow-scope logical forms say that for each boy x, if x hates a woman x knows, whichever woman it is, x develops a complex. I call this reading the **exhaustive reading** of the indefinite. But (330b) says that each boy x develops a complex only if x hates a specially chosen woman among the women x knows.

As the classical logical forms in (332) do not represent the reading (330b), the problem is not only for a choice function analysis. It is a problem for any analysis that explains the co-variation possibility of an indefinite solely in terms of the relative scope of the existential quantifier that is associated with the indefinite.

Winter solves this problem by re-defining choice functions as Skolem functions with flexible arities. For simplicity, I discuss a case in which the nominal restriction of the indefinite contains only one bound variable, as in (330a). Then the nominal restriction of the indefinite (i.e. *woman he knows* in (330a)) denotes a type (e(et)) functor and the Skolemized choice function takes this functor as its first argument. In (333), I use the constant expression 'SK^1' to refer to the property that Winter has assigned to a Skolem function 'f' whose first argument 'g' (which is for the nominal restriction of the indefinite that contains one bound pronoun) is of type (e(et)). '$SK^1(f)$' means that 'f is a Skolem function (whose argument has one pronoun inside).'

(333) The denotation of $SK^1_{((e(et))(ee))t}$:
For all f of type ((e(et))(ee)), for all g of type (e(et)) and for all x of type e where there exists at least one individual d in the model M such that $[\![g(x)]\!]^M(d) = 1$:
$[\![SK^1(f)]\!]^M = 1$ iff $[\![g(x)(f(g)(x))]\!]^M = 1$

The idea is that when a pronoun appears in the nominal restriction, the type of the logical expression for the nominal restriction is (e(et)), as we can see in (334) which is for the nominal restriction *woman he knows* in (330a).

(334) $\lambda m.\lambda n.[woman'(n)\ \&\ know'(n)(m)]$

For the sentence in (330a), the function g in (333) corresponds to the logical expression given in (334), which denotes a function that maps each individual m to the set of women that m knows. If the argument slot m of the nominal restriction functor in (334) is bound by another quantifier, say, *every boy*, then we get a possibly different set of women for each boy.

Given such nominal restriction functors with a bound pronoun inside, the Skolem function f defined in (333) denotes a function that maps each (e(et)) function g to another function $f(g)$, which in turn maps each individual x to a member of the set denoted by $g(x)$. If x is a boy, $g(x)$ denotes a set of women for the boy x. Now, what happens if $g(x)$ happens to denote the same woman-set for all the boys. $f(g)$ in (333) can still map each boy x to a different member of the woman-set denoted by $g(x)$. In other words, even if $g(x)$ denotes one and the same set of individuals, $f(g)$ can still map each individual x to a different member of that same set. Let me show this point in Winter's logical form in (335) for the sentence in (330a) with the reading (330b).

(335) $\exists f[SK^1(f)$ &
$\forall x[[boy'(x)$ & $hate'(f(\lambda m.\lambda n.[woman'(n)$ & $know'(n)(m)])(\mathbf{x}))(x)]$
$\rightarrow$ $develop\text{-}complex'(x)]]$
where f is of type (e(et))(ee) (Winter 2001: 118)

Again, the type (e(et)) function g in (333) corresponds to $\lambda m.\lambda n.[woman'(n)\&know'(n)(m)]$ in (335). In a context in which all the boys know exactly the same set of women, this nominal restriction function maps every boy x to the same set of women. However, even in this context, the function $f(\lambda m.\lambda n.[woman'(n)\&know'(n)(m)])$ in (335) can still map each boy x to a different member of that same set of women. This is possible because of the highlighted second argument $\mathbf{x}$ of the function f in (335). In (335), this argument $\mathbf{x}$ is bound by the universal quantifier in *every boy* and consequently, we can choose a different woman for each boy $\mathbf{x}$. Notice that (335) still implies that we pick out a specific kind of woman for each boy, rather than whichever woman it is in the set of women. That is to say, the existential quantifier ' $\exists f$' takes wide scope over the universal quantifier and because of that, there is only one function f involved for every boy x.[10] Thus, the logical form in (335) does not lead to the

[10]In other words, the fixed function f leads to the fixed relation holding for all the *boy-woman* pairs, as in Cormack's reading that I explained in the text following (330). In order to derive the narrow scope reading (in which each boy develops a complex if he hates any woman that he knows), Winter applies an existential closure in a position lower than the universal quantifier. To represent various scope readings, Winter needs to apply existential closure in different structural

exhaustive narrow scope reading as in (331b) or (332b).

In summary, the apparent specificity in (330b) is explained in terms of the widest scope of the existential quantifier binding the function variable, which leads to the use of the same Skolem function for all the boys, but we can still pick out a different woman for each boy, because this Skolem function applied to the same set of women can still choose a different woman for each boy from that same set, because of the function's second argument slot x, which can be bound by the universal quantifier in *every boy*. In the next section, I explain two problems with Winter's analysis.

9.4 Problems

9.4.1 Unconstrained existential closure

Winter uses a Skolemized function f as in (333) only for indefinite noun phrases with a pronoun inside the nominal restriction, to explain the strange reading as in (330b). The various 'scope readings' of indefinites are explained in terms of different positions at which existential closure is applied to the function variable f, whether the nominal restriction has a bound pronoun or not. For example, in (336), we can think of either the same girl for all the boys, or a possibly different girl for each boy, and Winter explains these readings by applying an existential closure in alternative positions, as in (336b) and (336c).

(336) a. Every boy said that Bob loves a girl.

b. $\exists f_{(et)e}[CH'(f)\&\forall x[boy'(x) \rightarrow say'_{(t(et))}([love'(f(girl'))(b')]_t)(x)]]$
$\exists > \forall$

c. $\forall x[boy'(x) \rightarrow \exists f_{(et)e}[CH'(f)\&say'_{(t(et))}([love'(f(girl'))(b')]_t)(x)]]$
$\forall > \exists$

However, the existential closure operation is no more structurally constrained than the exceptional scope movement for indefinites in QR based analyses. In (336b), closure is applied in a position outside the tensed clause where the indefinite is located, and in (337a) with the reading (337b), closure would have to be applied outside the complex NP in which the indefinite *a (certain) student* is placed.

(337) a. Every teacher over-heard [$_{NP}$ the rumour that a (certain) student smoked at school].

positions. See section 9.4.

b. There is one student such that every teacher over-heard the rumour that he smoked at school.

The choice-function logical form in (338a) is claimed to represent this reading in a better way than the LF representation in (338b), which covertly moves the indefinite out of the complex NP island at LF.

(338) a. $\exists f[CH'(f)$ & [every teacher over-heard [NP the rumour [that f(student) smoked at school]]]]

b. [some student$_1$ [every teacher over-heard [$_{NP}$ the rumour [that t_1 smoked at school]]]]

However, it is questionable whether introducing an unconstrained existential closure operation just to explain the exceptional scope taking of indefinites is any better than assuming an unconstrained covert movement just for that purpose. Also, if we adopt the idea of Inclusiveness as in Chomsky (1995) and assume that all information comes from the lexicon, we need to assume that the function variable f and the existential closure operator come from some lexical information as well. Remember that a mere existential closure over f is not enough; the function f has to have the choice function property denoted by CH'. In other words, we need an operation as in (339), where the existential closure is applied at the top of the logical form (the closure should alternatively be applicable somewhere within the scope of the universal quantifier as well, to derive the narrow scope reading of the indefinite).

(339) $ECC'(\lambda f.\forall x[boy'(x) \rightarrow say'([love'(f(girl'))(b')])(x)]) =$
$\exists f_{(et)e}[CH'(f)$ & $\forall x[boy'(x) \rightarrow say'([love'(f(girl'))(b')])(x)]]$,
where $ECC'_{(((et)e)t)t} \stackrel{\text{def}}{=} \lambda Q_{((et)e)t}.\exists f[CH'(f)$ & $Q(f)]$[11]

(cf. Winter 2001: 131)

The details in (339) are not essential here, but the existential closure operator 'ECC'' has to introduce not only an existential quantifier binding the variable f, but also the choice function property CH' of the function f. If an analysis that does not use choice functions can explain the exceptional scope taking of indefinites, that analysis is preferable in that we do not need the extra complexity of the theory.

[11] In Winter's notation, the definition of the existential closure operator on the choice function variable f is as in the following, $ECC_{(((et)e)t)t} \stackrel{\text{def}}{=} \lambda Q_{((et)e)t}.Q \cap CH' \neq \emptyset$ (Winter 2001: 131).

9.4.2 Compositionality problem

Another problem with Winter's analysis is that his Skolem function logical form cannot directly mark the binding relation between a quantifier and the pronoun bound by it. In a classical logical form, a bound pronoun is represented by a variable bound by the quantifier, as in (340b).

(340) a. Every boy_1 said that he_1 smokes.

b. $\forall x[boy'(x) \rightarrow say'_{t(et)}([smoke'(x)])(x)]$

In contrast to this, if we use Winter's Skolemized choice functions to deal with the problematic sentence *Every* boy_1 *who hates a woman* he_1 *knows develops a serious complex* in (330a), the external argument slot of the verbal functor *know'* cannot be directly bound by the universal quantifier in the logical form, even though the bound pronoun appears as the subject of *know* in the PF string, just as the bound pronoun *he* appears as the subject of the verb *smoke* in (340b). Look at (341).

(341) $*\exists f_{(e(et))(ee)}[SK^1(f)\&$
$\forall x[[boy'(x)\&hate'(f(\lambda n.[woman'(n)\&know'(n)(\mathbf{x})])(x))(x)]$
$\rightarrow develop\text{-}complex'(x)]]$

The logical form in (341) is illicit because the first argument of f does not have the required type (e(et)); its type is (et). This means that we cannot let the universal quantifier bind the highlighted external argument slot $\mathbf{x}$ of the verbal functor *know'*, even though this argument slot corresponds to the bound pronoun *he* which is the subject of the verb *know.*

Note that the following β reduction would be illicit in Winter's logical form in (335), repeated below as (343a), as it would collapse the two arguments of f into one (i.e. replacing x for m while deleting $\exists m$ would be illegal in (338)):

(342) $f(\lambda m.\lambda n.[woman'(n)\ \&\ know'(n)(m)])(x)$
$\not\Rightarrow_{\beta reduction}\ f(\lambda n.[woman'(n)\ \&\ know'(n)(x)])$

We cannot bind the argument slot m in this way either.

In Winter's logical form in (343a), the argument slot m for the bound pronoun *he* and the extra argument slot x of the Skolemized function f are set to denote the same individual only indirectly, through the definition of the Skolem function as in (333), repeated here as (343b).

(343) a. Every boy who hates a (certain) woman he knows develops a complex.
$\exists f_{(e(et))(ee)}[SK^1(f)$ &
$\forall x[[boy'(x)$ & $hate'(f(\lambda m.\lambda n.[woman'(n)$ & $know'(n)(m)])(x))(x)]$
$\rightarrow develop \cdot complex'(x)]]$

b. For all f of type ((e(et))(ee)), for all g of type (e(et)) and for all x of type e where there exists at least one individual d in the model M such that $[\![g(x)]\!]^M(d) = 1$:
$[\![SK^1(f)]\!]^M = 1$ iff $[\![g(x)(f(g)(x))]\!]^M = 1$

In (343b), the semantics of the Skolem function property SK' is defined in such a way that the first argument slot of the nominal restriction set 'g' is saturated by the same variable x as the one that saturates the second argument slot of the Skolemized choice function 'f.' In this way, the complex function '$f(g)$,' or '$f(\lambda m.\lambda n.[woman'(n)$ & $know'(n)(m)])$' in (343a) for the sentence in question, will denote a function that maps each boy x to a member of the woman-set for x, even though the external argument slot of the verbal functor *know* for the bound pronoun *he*, marked with m in (343a), is not directly bound by the universal quantifier in *every boy*. Technically, we could define the property SK^1 in a different way so that $f(g)$ maps each individual x to a member of the set denoted by $g(y)$, where $x \neq y$. If we applied this alternative definition to the sentence in (343a), then, m and x would denote different individuals, contrary to the interpretation required by the bound pronoun *he*. In this sense, the interpretation of the bound pronoun *he* necessitates the definition of SK^1 as in (343b). But in (339), it is the existential closure operator ECC' which introduces the choice function property CH'. Thus, the Skolem function property SK' in (343b) would also be introduced by the existential closure operator. The closure operator ECC' would presumably be associated with the indefinite NP via the choice function variable f, with or without the existence of a bound pronoun in the nominal restriction. It is not easy to modify the definition of ECC in such a way that the definition of the Skolem function property in (343b) is directly associated with the lexical information of the bound pronoun in the nominal restriction of the indefinite. Even if we could come up with a rule like that without violating Inclusiveness, the interpretational contribution of the bound pronoun *he* would still be different in the standard binding case as in (340) and in a case like (343a). Thus, Winter's logical form at least goes against the spirit of semantic compositionality, which predicts that the contribution of the bound pronoun to deriving the binding relation should be the same both for (340) and (343a).

Admittedly, the two points I have made are problematic only if we assume that the logical form is compositionally derived in a syntactic derivation following the Chomskian idea of Inclusiveness. If our primary concern is to explain the available readings of indefinites in an empirically adequate way, this might be less of a problem. But in this thesis, I assume that compositional derivation of interface logical forms is an essential thing.

In summary, Winter's analysis not only requires the introduction of the choice/Skolem function property during syntactic derivation but also an unconstrained existential closure operation over function variables in the derived logical form. It is not clear whether this additional complication of the theory is linguistically well-motivated. The other problem I have discussed is that Winter's logical form cannot directly represent the binding relation holding between the quantifier and the pronoun bound by it. This poses a problem for semantic compositionality.

9.5 Domain restriction

In this section, I informally motivate a domain restriction analysis with an inherent argument slot for the indefinite, and argue that it solves the problems I mentioned in the previous section. The formal analysis is provided in section 9.6.

First, I introduce the pragmatic domain restriction analysis proposed by Schwarzschild (2002) with one of its main motivations. Consider the example in (344).

(344) a. Every boy who hates a (certain) woman develops a complex.

b. $\exists x[woman'(x)\&\forall y[[boy'(y)\&hate'(x)(y)] \rightarrow develop\text{-}complex'_{et}(y)]]$

c. $\forall y[[boy'(y)\&\exists x[woman'(x)\&hate'(x)(y)]] \rightarrow develop\text{-}complex'_{et}(y)]$

In (344a), when the domain of the set of women is pragmatically restricted to a singleton set, the assertion is made only about the unique member of the set. Thus, we can derive the wide scope reading equivalent while assuming only the narrow scope reading, given in (344c), as the linguistically encoded meaning, where the pragmatic domain restriction enables us to talk about the unique woman.

Schwarzschild argues that the so-called exceptional wide scope reading of the indefinite is not a matter of the existential quantifier taking wide scope (2002: 298). Analyses that give exceptional quantificational scope-taking possibilities to indefinites assume that the indefinite *a (certain) woman* in (344a) can take scope over the universal QNP in the main clause, but it is not obvious whether the so-called wide scope reading some native speakers get with this string can be captured

by the wide scope logical form of the indefinite, given in (344b). (344b) is trivially true when there is an x such that x is a woman and no boy hates x, even if there is another woman y such that a boy who hates y does not develop a complex. This wide scope logical form does not correctly represent the specific reading of (344a). What we want to capture instead is the non-arbitrariness of the choice of a woman. Each boy develops a complex only if he hates a specific woman, say, Nancy; not when he hates some woman or other.

The domain restriction analysis can explain this neatly. If the domain is restricted to a singleton set, the other members of the original set that are excluded from the domain are irrelevant. On the other hand, if the sentence is understood as an assertion about women in general, the domain is not restricted to a singleton set and we do not get the specific reading. What happens if the domain is restricted to a singleton set that contains a woman that no boy hates? In that case, the sentence (344a) is simply true. Note that in this analysis, the woman no boy hates and the specific woman that is picked out by the indefinite *a (certain) woman* in this context have to be the same woman, because the domain-restricted set has only one member. So the above problem for the logical form (344b) does not arise. In an actual interpretation, it might be difficult to restrict the domain in this way. It is a pragmatic inference that determines to which member the domain is restricted and I assume that the pragmatic domain restriction is worked out on the basis of the linguistic meaning of the sentence and the relevant contextual information. This explains why in a normal context, it is difficult to restrict the domain in such a way that the truth value of the main clause *Every boy develops a complex* becomes irrelevant to the truth condition of the whole sentence.

Unlike the choice/Skolem function analysis, the domain restriction theory does not require an existential closure operation or a choice/Skolem function variable in the derivation of logical forms. This makes the derivation simpler and makes it easier to maintain the compositional derivation of logical forms based on the lexically encoded information. The existential quantifier is generated in-situ with the indefinite noun phrase, which does not take an exceptional wide scope. Also, because we interpret the indefinite quantificationally, it is easy to derive the exhaustive reading when the indefinite appears in a downward-entailing environment (and when the domain is not restricted to a singleton set), as in the narrow scope reading of *If I find a problem, I will let you know.*.

On the other hand, a challenge for the domain restriction analysis is the intermediate scope reading as in (345). Ruys (1992:101-102) and Abusch (1994: 84-88) argue that an analysis that predicts that the exceptional wide scope taking of an

indefinite always leads to the widest scope is wrong, based on sentences such as (345). Their criticism is aimed at the lexical ambiguity analysis of indefinites in Fodor and Sag (1982), in which referential indefinites, as opposed to quantificational indefinites, are always interpreted in the reading that corresponds to the widest scope reading of the indefinite. The criticism is not meant to be against the domain restriction analysis. However, if the domain restriction analysis always gives the widest scope when the domain is restricted to a singleton, it is subject to the same criticism.[12]

(345) Every student discussed every analysis that solved a (certain) problem in Chomsky 1995. (cf. Reinhart 1997: 346)

(345) has a reading that says that for each student x, there is a possibly different problem y in Chomsky 1995, and x discussed all the analyses that solved y. If the domain restriction to a singleton set is insensitive to other elements in the sentence, we predict incorrectly that whenever the domain is restricted to a singleton, the indefinite *a (certain) problem* has to denote one and the same problem for all the students.

One way to solve this problem is to assume that the indefinite has an inherent argument slot on which the domain restriction is dependent. When this inherent argument slot is bound by the universal quantifier *every student* in (345), the domain restriction can be done differently for each student. Thus, we can pick out a different problem for each student. The sentences in (346) (cf. Winter 2004: 331) will fall under the same sort of explanation.

(346) a. Every student$_1$ admired a (certain) teacher - his$_1$ homeroom teacher.

b. A woman that every man$_1$ loves is his$_1$ mother.

(346a) suggests that the specificity of the teacher can be relativized to each student; each student can admire a possibly different specific teacher, and if this specificity is the result of a domain restriction into a singleton set, the domain restriction has to be made in a possibly different way for each student.

The so-called functional reading provides another argument for this inherent argument slot of indefinites. The co-indexed pronouns in (346a) and (346b) pose a problem for a structural analysis of pronoun binding, because these pronouns are not within the surface c-command domain of the universal quantifiers. But if we assume that the indefinite has an inherent argument slot, which can be formally

[12]Cormack and Kempson (1991) also mentions the existence of the intermediate reading, though unlike Ruys and Abusch, they take a pragmatic approach to explain this reading.

linked to the universal quantifier, then we can claim that the equality of the functional relation holding between the universal quantifier and the indefinite on the one hand and the functional relation between the universal quantifier and the noun phrase containing the pronoun on the other justifies the use of the pronoun in this way.[13] In the relevant reading of (346a), the sentence is true if and only if the function mapping each student to a singleton teacher-set for him is the same as the function mapping each student to a singleton homeroom-teacher-set for him. In the same way, we can explain the relevant reading of (346b) by assuming that the function mapping each man to a singleton woman-set for him is the function mapping each man to the singleton set containing his mother as its unique member.

Motivated by these considerations, I propose that indefinites have an inherent argument-slot, which can be bound by another quantifier in the sentence, and which can make the domain restriction dependent on this quantifier.[14] Schwarzschild assumes that the dependency of the domain restriction is pragmatically derived without linguistic encoding, but I assume that indefinite noun phrases are lexically equipped with this extra argument slot, in order to compositionally derive the required dependency relations in the logical forms.

I start with the sentence in (347a). (347b) represents the reading in question.

(347) a. Every boy_1 respects a (certain) man (- his_1 father).

b. $\forall x[boy'(x) \rightarrow \exists y[sg'(man')(x)(y)\&respect'(y)(x)]]$

In (347b), the functor expression sg' is of type ((et)(e(et))) and it has three arguments: $man'; x; y$ in this order. In (347b), sg' denotes a function that maps the set of men to another function which maps each boy x to a singleton *man*-set.[15] That is, the function denoted by sg' maps the set of men to a possibly different singleton set for each boy x. In other words, sg' enables us to restrict the domain of the man-set to a singleton set differently for each x.

Next, I provide an informal analysis of an indefinite with a bound pronoun in the proposed domain restriction analysis. A more formal analysis in Type Logical Grammar is provided in section 9.6. The basic treatment of (bound) pronouns follows Jacobson (1999). Informally, if the nominal restriction of the indefinite noun phrase has a pronoun in it, the semantic type of the logical expression for the nominal restriction is (e(et)), rather than (et), which is for a nominal restric-

[13] Winter (2004) uses a similar argument to support his Skolem function analysis of indefinites.

[14] I do not discuss either a generic indefinite or an indefinite in a non-argument position, though later I will provide some preliminary suggestion about the latter in terms of my proposal.

[15] In the formal section, I slightly change the type of the extra argument slot of indefinites. See (351a)~(351b) and the following text.

tion without a pronoun. For Winter's sentence (330a), repeated here as (348a), the nominal restriction *woman he knows* is paired with the logical expression $\lambda x.\lambda y.[woman'(y)\&know'(y)(x)]$ of type (e(et)), where the pronoun *he* introduces an extra argument slot x. Because the functor expression sg' of type ((et)(e(et))) requires a type (et) argument, we cannot use a simple function application to merge the two expressions. As we do in the next section, we need to introduce some structural rule to percolate the extra argument slot introduced by the pronoun into a later stage of derivation separately from the inherent argument slot encoded with *a (certain)*. Given some extra set of rules, we can bind these two extra argument slots by the same quantifier, and derive the reading in question in (330b). The normalized interface logical form in (348b) represents this reading.

(348) a. Every boy_1 who hates [$_{NP}$ a (certain) woman he_1 knows] will develop a serious complex.

b. $\forall x[[boy'(x)\&\exists y[sg'(\exists z.[woman'(z)\&know'(z)(x)])(\mathbf{x})(y)\&hates'(y)(x)]] \rightarrow develop \cdot complex'(x)]$

The logical form in (348b) means that, given a set of women each boy x knows, we can map it to a different singleton set for each x, even if every boy happens to know exactly the same set of women. In (348b), the highlighted second argument $\mathbf{x}$ of the functor sg' marks the dependency of the domain restriction on x.

The external argument x in the formula $know'(z)(x)$ corresponds to the bound pronoun *he*. In (348b), this x is also bound by the same quantifier that binds the highlighted $\mathbf{x}$ (which, again, is the second argument of sg'), but this does not have to be the case. These two argument slots can be bound by different operators. See subsection 9.7.1 for one motivation for this formulation. Notice that, unlike in Winter's logical form in (335), the binding relation between the universal quantifier and the bound pronoun *he* is directly represented in (348b).

If the indefinite has the word *certain* overtly in it, such as *a certain woman*, then the set of women is obligatorily restricted to a singleton set. This forces a specific reading, but this specificity can be relativized because of the inherent argument slot of the indefinite. The indefinite *a woman* without the word *certain* still has this inherent argument slot, but there is no linguistic singleton set requirement. We can still optionally restrict the domain to a singleton set by using pragmatics. Then the identity of this singleton set can be dependent on the inherent argument slot. But normally, the domain restriction relativization is not noticeable with indefinite noun phrases without *certain* because the domain is usually not restricted to a singleton set with this type of indefinite. With this normal type of indefinite, the

domain is usually restricted to a set that still contains several members. This is why we tend to get the exhaustive reading when this type of indefinite appears in the nominal restriction of a universal noun phrase. In order to restrict the domain to a singleton and get an exceptional wide scope reading,[16] we need a special pragmatic context that justifies such extreme domain restriction. In fact, some speakers never get an exceptional scope reading with the normal indefinite such as *a boy*. I suppose this is because the existence of the more specific expression *a certain boy* blocks the application of the pragmatic domain restriction to a singleton set even in a suitable context.

How the domain is restricted in a particular use of such a sentence is a matter of pragmatics. I will not discuss the pragmatic process in detail. But roughly, when the indefinite *a certain woman* is used which linguistically requires the set of women to be restricted to a singleton set, then the hearer assumes that the speaker must have some evidence in mind which supports the domain restriction to a singleton set. If the speaker knows who the singleton member of the set is, it counts as good evidence, and this is why the hearer often has the impression that the speaker must know who the singleton member is. The supporting evidence does not have to be a specific individual; it can be a specific relation. The linguistic meaning may say that there is a certain relation holding between an element binding the inherent argument slot of the indefinite and the resultant singleton member of the woman set. A particular relation that the hearer takes the speaker to have in mind can then count as a ground supporting the singleton domain restriction. It might be the son-mother relation as in (346b). In contrast, in order to interpret a normal indefinite such as *a woman* as a singleton set of woman, the hearer needs some contextual information that indicates that the speaker must have some evidence as above in mind which supports a singleton set formation.

Before I show a syntactic derivation, I briefly mention two analyses of indefinites that have some similarities to mine. The first analysis is in Kratzer (1998). Kratzer's analysis is similar to mine in that she uses an extra argument slot for deriving the dependency of the indefinite to another operator. However, there are two crucial differences. The first difference is that like Winter, Kratzer uses Skolemized choice functions. Because Kratzer treats the functor term f as a strange kind of 'constant' expression (which is not a variable but whose identity may still vary depending on the context), her analysis does not need an existential closure operation as Winter's analysis does. On the other hand, unlike Winter's analysis, which

[16]For convenience, I keep on using the term 'exceptional wide scope,' though I do not see the phenomenon as a matter of scope.

treats f as a variable that ranges over certain types, it is not clear what kind of logical language expression Kratzer's functional term f really is (see section 9.7.3 for a related problem of my analysis and some modification of my analysis in that regard). Also, because Kratzer's analysis is basically the same as Winter's other than the constant status of the functional term f and the use of quantificational analysis of indefinites as well as the choice function analysis of indefinites, it is not clear whether Kratzer's analysis can deal with the compositionality problem that is associated with Winter's sentence I have discussed in section 9.4.2. As the second difference between Kratzer's analysis and my analysis, she is still in the spirit of the ambiguity analysis of Fodor and Sag (1982), in which indefinites are ambiguous between the referential meaning and the quantificational meaning. In contrast, my analysis is in the spirit of Schwarzschild (2002). In my analysis, all the indefinites appearing in argument positions are interpreted as existential quantifiers. Theoretically, the so-called specificity is a side effect of the domain restriction that applies to the nominal restriction set.

The other analysis is in Breheny (2003). Breheny treats indefinites with *certain* as existential. His analysis of indefinites is similar to mine in its essential use of pragmatics as well as his uniformly quantificational treatment of indefinites. On the other hand, the proposed theory itself is different. He explains the specificity with regard to *certain* in terms of some sort of 'specificity' property denoted by the constant $certain'_u$ of type (et).[17] Potential dependency of the specificity on another quantifier as in the intermediate scope reading of (345) is explained in terms of an extra argument slot of a related constant $certain'_{2u}$ of type (e(et)) which allows the 'specific' problem to co-vary with each student in (345). The so-called intermediate scope reading of the sentence in (345) without *in Chomsky 1995*) in Breheny's analysis will be as in (349).

(349) *Every student discussed every analysis that solved a (certain) problem.*

$\forall x[student'(x) \rightarrow$
$\forall y[[analysis'(y) \& \exists z[certain^2{}_u(x)(z) \& problem'(z) \& solve'(z)(y)]]$
$\rightarrow discuss'(y)(x)]]$ Cf. Breheny (2003: 46)

Though Breheny's main interest is the semantic/pragmatic interpretation, rather than the grammatical derivation of logical forms, incorporating his idea into Type Logical Grammar would make the grammatical composition more complex compared with my proposal because *certain* takes common nouns, rather than type

[17]The subscript u on the constant indicates that the property denoted by the constant is fixed relative to the utterance, where u is mnemonic for 'utterance.'

e denoting expressions, as arguments in the syntactic derivation. Because of this, his analysis is not suitable for the purpose of deriving the exceptional wide scope effect of indefinites by way of compositional pairing of PF strings with their interpretations based on lexical categorial information. In terms of the semantics, the domain restriction to a singleton set is only a derived effect in Breheny's analysis of indefinites with *certain*, where the essential encoded information is some sort of specificity property denoted by the functor $certain^2{}_u$. In contrast, in my analysis, the domain restriction is the essential information encoded with *certain*, and the property of specificity is a derivable side effect of restricting the domain into a singleton set.

In this section, I have argued that the indefinite is lexically equipped with an inherent argument slot on which the domain restriction can be dependent. Unlike Winter's analysis, this theory does not require an existential closure in the derivation of logical forms. And the logical form for a sentence that contains an indefinite with a bound pronoun directly represents the binding relation between the quantifier and the pronoun bound by it.

In the next section, I show a syntactic derivation in Type Logical Grammar for a sentence that has an indefinite with a bound pronoun in its nominal restriction.

9.6 Formal analysis

In this section, I show how an extra argument slot encoded with indefinites can be percolated into a later stage of syntactic derivation and get bound by another (Q)NP. The basic algorithm that I use is the same as in Jacobson's treatment of pronominal binding which we have seen in chapter 3. Translated into Multi-Modal Type Logical Grammar, this means that we introduce a hypothetical category marked with the unary operator '$\Diamond_p$' which triggers use of particular structural association/permutation rules. The hypothetical item will then be identified by an element that is merged later in the derivation. Because the structural rules licensed by $\Diamond_p$ are not sensitive to the modes of merge of the intervening PF items, this mechanism can derive a long distance dependency relation.

9.6.1 Deriving the dependency of indefinites in MMTLG

I show a type logical derivation for Winter's problematic sentence in (330a), which I repeat here with some modification in order to make the derivation less complex in unimportant parts.

(350) Every boy$_1$ who met [$_{NP}$ a (certain) woman that he$_1$ knew] left.

I provide the lexical entries of some of the items. I omit most of the indices for binary connectives, because I do not use structural rules defined in terms of different modes of merge in this section. All the structural rules in this section are controlled by way of unary modal operators. However, I specify the mode of binary merge 's' for the merge of relative pronouns with their complements, to indicate the clausal boundary.

(351) a. a: $< a; (N/\Diamond_p\Box^{\downarrow}NP)/N; \lambda A_{et}.\lambda u_e.\lambda v_e.a'(A)(u)(v) >$

b. a certain:
$< a \cdot certain; (N/\Diamond_p\Box^{\downarrow}NP)/N; \lambda A_{et}.\lambda u_e.\lambda v_e.sg'(A)(u)(v) >$

c. every (Nom):
$< every; (S/(NP\backslash S))/N; \lambda A_{et}.\lambda B_{et}.\forall x[A(x) \rightarrow B(x)] >$

d. some*(Acc): $< \epsilon; ((S/NP)\backslash_j S)/_n N; \lambda A_{et}.\lambda B_{et}.\exists x[A(x)\ \&\ B(x)] >$

e. he: $< he; NP/\Diamond_p\Box^{\downarrow}NP; \lambda x.x >$

f. who: $< who; (N\backslash N)/_s(NP\backslash S); rel' >$
where rel' of type (et)((et)(et)) $= \lambda A_{et}.\lambda B_{et}.\lambda x.[A(x)\ \&\ B(x)]$

g. that: $< that; (N\backslash N)/_s(S/\Diamond\Box^{\downarrow}NP); rel' >$
where $rel' = \lambda A_{et}.\lambda B_{et}.\lambda x.[A(x)\ \&\ B(x)]$

h. know: $< know; (NP\backslash S)/NP; \lambda x.\lambda y.know'(x)(y) >$

i. meet$_q$: $< meet;\ (NP\backslash S)/((S/NP)\backslash S);\ meet_q >$
where $meet_q \stackrel{\text{def}}{=} \lambda Q_{(et)t}.\lambda y_e.Q(\lambda x.meet'(x)(y))$

The entry for the pronoun is the same as in chapter 3, section 3.3, which triggers use of some structural rules, as we see shortly. The entries for relative pronouns in (351f) and (351g) are for subject relative pronouns and for object relative pronouns in English respectively. For the subject relative pronoun *who*, we could use the same entry that we use for the object relative pronoun *that*, if we added some permutation rule under the control of the operator $\Diamond$. This alternative might be preferable, considering that most of the relative pronouns in English do not morphologically distinguish their argument statuses relative to the verbal functor. However, the derivation would become unnecessarily complex with an extra structural rule, and thus, I stick to the entry for the subject relative pronoun *who* as in (351f) for presentation reasons. Now, I briefly explain the other entries.

The determiner *some** in (351d) has a null phonological entry ('ϵ' means PF-null). This item is inserted into the syntactic derivation as the left sister of the

indefinite *a girl*. The reason why I do not encode the existential quantifier in (351d) with the indefinite article *a* itself is that an indefinite noun phrase can be interpreted non-existentially, for example, as a predicate in the copula construction or as generic.

Having postulated the PF-null determiner *some** when indefinites appear in argument positions, we could have encoded the extra argument slot with this PF-null item, rather than with the indefinite article *a* as in (351a)~(351b). This choice is dependent partly on whether we can get the dependent 'specific reading' for an indefinite noun phrase in a non-argument position. The specific interpretation of the indefinite at the data judgment level seems to be obligatory with indefinites with *certain*. Thus, we can place an indefinite with *certain* in a predicate position to test whether the specificity in question can be dependent on another element in the sentence in a non-argument position. Consider (352).

(352) a. Every boy mistakenly believed that Mary was a certain woman.

b. Every boy mistakenly believed Mary to be a certain woman.

Can we pick out a different woman for each boy with the indefinites in these predicative positions? Though the judgment is subtle, I understand that the identity of the woman can co-vary with each boy in (352), which suggests that we should associate the inherent argument slot with *a (certain)*, rather than with the PF null functor *some** which is available only with an indefinite appearing in an argument position.

I have provided the entries for *certain* in (351b) to compare indefinites with *certain* with indefinites without *certain*. The entry in (351b) ignores the internal structure of the indefinite *[a [certain woman]]* for presentation reasons. We could respect the internal structure of the indefinite represented by the square brackets by treating *certain* as of type ((et)(et)) and the indefinite article *a* as of type (et)(e(et)). The extra argument slot would then be uniformly associated with the indefinite article *a* with or without *certain*. But the derivation would become longer with this alternative entry to *certain*, so I stick to the entry as in (351b). The essential point is that the singleton set requirement to the denotation of the nominal restriction expression saturating the A argument slot is linguistically encoded with *certain* in (351b), whereas if we use the indefinite article *a* without *certain*, the domain restriction to a singleton set is a pragmatic option, not a linguistic requirement. The functor expression a' may be encoded with some kind of singularity information, but I ignore such extra complication.

Both the logical expressions a' for simple indefinites without *certain* and sg' for indefinites with *certain* are of type ((et)(e(et))), where the informal interpretation of the functor expression sg' is as I have explained for (347) and (348) in section 9.5. The logical expression for *a certain woman* will then be $\lambda u.\lambda v.sg'(woman')(u)(v)$, where $sg'(woman')(u)$ denotes a singleton woman-set which can co-vary with each u, an element that saturates the extra argument slot. In contrast, the functor expression for *a boy* is $\lambda u.\lambda v.a'(boy')(u)(v)$. The functor expression $a'(boy')$ is also of type (e(et)), and has an extra argument slot u which may be bound by some other quantifier, such as the universal quantifier in *every boy*, as in *[Every boy]*$_1$ *loves [a woman]*$_1$. The expression '$a'(boy')(x)$' with 'x' being bound by the universal quantifier will then denote a possibly different set of boys for each x, but for each x, the set of boys does not have to be a singleton set, as far as the linguistic meaning is concerned. In other words, the dependency of the indefinite on the universal quantifier may still be linguistically marked, but without *certain*, the domain restriction to a singleton set is left for pragmatic inferences, and only if the pragmatics restricts the domain of the set of boys to a singleton set, does '$a'(boy')(x)$' denote the same set as '$sg'(boy')(x)$.'

The extra argument slot u of the indefinite is usually bound by a quantifier over individuals, such as *every boy*. Because of this, I have provisionally assigned type e to the extra argument slot. However, sometimes some other elements, such as the tense operator, might bind this argument slot, as we see in subsection 9.7.2. Thus, I later modify the type of the extra argument slot to an under-specified type τ, which covers the type e, the type for tense, and probably the type for event individuals, though the exact identity of this type is left for future research.

Finally, when I merge a QNP as an argument of a verb in this section, I skip the process of argument slot raising, and use the output of this process directly in the derivation. More specifically, I use the argument slot raised category for *meet* which has been shown in (351i). In the derivation, I also omit the reduction process that is based on the definition of $meet'_q$ in terms of the lexically encoded functor *meet'* of type (e(et)) as in (351i), which is as we have seen in chapter 2 and chapter 5.

9.6.2 Derivations

I show the important parts of the derivation for (350). I skip the derivational steps before the structural rules are applied; they are straightforward application of logical rules. The general strategy is the following. For each extra argument

slot, of either an indefinite or a bound pronoun, a hypothetical category marked with the operator $\Diamond_p$ is inserted in the position that can (hypothetically) saturate the extra argument slot. After merging some other items, we apply the association rule AR or/and the permutation rule PR to derive the configuration in which the 'binding' structural rule Z can apply. By way of Z, the hypothetical item that has provisionally saturated the extra argument slot is discharged and the antecedent binds the extra argument slot instead. We can bind more than one extra argument slot by one binder by applying Z in succession with regard to more than one hypothetical argument marked by '$\Diamond_p\Box^{\downarrow}NP$'. Use of relative pronouns requires insertion of their own hypothetical categories marked by $\Diamond$, but the hypothetical categories introduced for relative pronouns are distinguished from the hypothetical categories marked with $\Diamond_p$ for indefinites/pronouns because of the lack of the index p (mnemonic for 'pronouns'), as we see shortly.

For reference, the structural rules are repeated from chapter 3. The presentation is Gentzen Sequent.

(353) a. Association Rule AR (generally available for A-bar phenomena):

$$\frac{A \circ (X \circ \Diamond B) \vdash C}{(A \circ X) \circ \Diamond B \vdash C}\ AR$$

b. Mixed Permutation/Association PR (limited to pronouns/indefinites by the mode requirement):

$$\frac{(A \circ \Diamond_p B) \circ X \vdash C}{(A \circ X) \circ \Diamond_p B \vdash C}\ PR$$

$A, B, C \in F$ (i.e. the set of formulas) and $X \in S$ (i.e. the set of structures)

(354) Argument identification Z with regard to p (i.e. pronouns).

a. Syntax:

$$\frac{A \circ (X \circ \Diamond_p A) \vdash B}{A \circ X \vdash B}\ Z$$

b. Semantics:

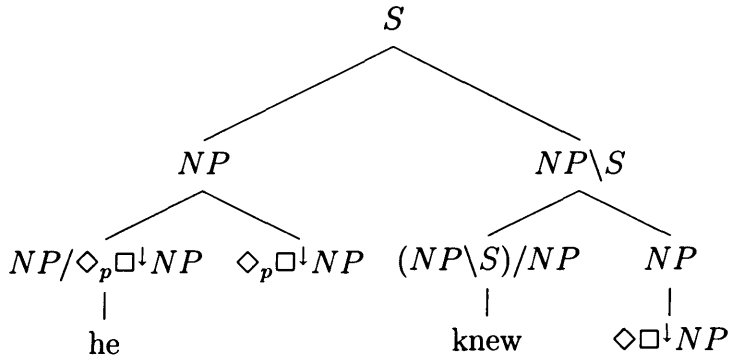
$$\frac{x \circ (R \circ y) \vdash \phi}{x \circ R \vdash \phi[y :\rightarrow x]}\ Z$$

As we have discussed in chapter 3, section 3, the rule Z in (354) might be too restrictive, because it requires the binder of the extra argument slot of an indefinite to appear in a position that c-commands the position of the indefinite if we think in terms of the tree structure that the categorial calculus would generate. But I ignore this extra complexity in this thesis.

More than one extra argument slot may be introduced (for example, by an indefinite and a pronoun) and then get bound by the same quantifier, as we see later, just as more than one pronoun can be bound by the same quantifier. As we saw in chapter 3 with regard to pronominal binding, we can achieve this by way of successive applications of Z in (354).

First, consider the derivation of the nominal restriction of the indefinite, that is, *woman that he knew* in (350). Before we apply structural rules, the derivation has got two 'hypothetical' categories, which must be discharged later, as is shown in (355).

(355) *he knew*



A hypothetical category $\Diamond\Box^{\downarrow}NP$ can saturate the internal NP argument slot of the verb *know*, because of the valid inference '$\Diamond\Box^{\downarrow}NP \vdash NP$.' Given the derivation in (355), we can apply the structural rule AR before we merge the result with the relative pronoun *that*.

(356) *he knew*

$$\dfrac{\dfrac{(NP/\Diamond_p\Box^{\downarrow}NP \circ \Diamond_p\Box^{\downarrow}NP) \circ ((NP\backslash S)/NP \circ \Diamond\Box^{\downarrow}NP) \vdash S}{((NP/\Diamond_p\Box^{\downarrow}NP \circ \Diamond_p\Box^{\downarrow}NP) \circ (NP\backslash S)/NP) \circ \Diamond\Box^{\downarrow}NP \vdash S}\,AR}{(NP/\Diamond_p\Box^{\downarrow}NP \circ \Diamond_p\Box^{\downarrow}NP) \circ (NP\backslash S)/NP \vdash S/\Diamond\Box^{\downarrow}NP}\,/R$$

(356) generates the required right-argument category for the functor *that* which has the category $(N\backslash N)/(S/\Diamond\Box^{\downarrow}NP)$. Note that Z in (354) is not applied with

regard to the normal hypothetical category '$\Diamond\Box^{\downarrow}NP$.' It is applied only with regard to '$\Diamond_p\Box^{\downarrow}NP$.' Another implication of this analysis is that the hypothetical category $\Diamond_p\Box^{\downarrow}NP$ for pronouns/indefinites cannot directly saturate a normal type NP argument slot, e.g. the internal argument slot of the verb *know* in (355). It must saturate the argument slot of a pronoun or an indefinite article which in turn may saturate an argument slot of a verbal functor.

Though the proposed analysis cannot 'bind' a 'trace' of a relative pronoun or a Wh-movement by a (Q)NP by way of Z, we can 'bind' a pronoun by a head of a relative or a Wh-operator. For example, the derivation for *I met a drunkard*$_1$ *who said that he*$_1$ *was sober* is fine. For this sentence, when we derive *said that he was sober*, we can identify the hypothetical category that is introduced by the pronoun *he* with the external argument slot for the verb *say* by way of Z. After this, we apply $\backslash R$ (that is, an abstraction rule), as is shown in (357), where VP represents $NP\backslash S$ and I treat the PF item '$(was \cdot sober)$' as of category VP $(= NP\backslash S)$ to simplify the derivation. The semantics combines PR and $AR \times 2$ into one step.

(357) *said that he was sober*

a. Syntax:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{NP \circ (VP/S \circ (S/_sS \circ_s ((NP/\Diamond_p\Box^{\downarrow}NP \circ \Diamond_p\Box^{\downarrow}NP) \circ VP))) \vdash S}{NP \circ (VP/S \circ (S/_sS \circ_s ((NP/\Diamond_p\Box^{\downarrow}NP \circ VP) \circ \Diamond_p\Box^{\downarrow}NP))) \vdash S}\,PR}{NP \circ ((VP/S \circ (S/_sS \circ_s (NP/\Diamond_p\Box^{\downarrow}NP \circ VP))) \circ \Diamond_p\Box^{\downarrow}NP) \vdash S}\,AR \times 2}{NP \circ (VP/S \circ (S/_sS \circ_s (NP/\Diamond_p\Box^{\downarrow}NP \circ VP))) \vdash S}\,Z}{VP/S \circ (S/_sS \circ_s (NP/\Diamond_p\Box^{\downarrow}NP \circ VP)) \vdash NP\backslash S}\,\backslash R}{said \circ (that \circ_s (he \circ was \cdot sober)) \vdash said \cdot (that \cdot (he \cdot (was \cdot sober)))}\,PF$$

b. Semantics:

$$\dfrac{\dfrac{\dfrac{\dfrac{z \circ (say' \circ (\lambda p_t.p \circ_s ((\lambda x.x \circ y) \circ sober'))) \vdash say'(sober'(y))(z)}{z \circ ((say' \circ (\lambda p_t.p \circ_s (\lambda x.x \circ sober'))) \circ y) \vdash say'(sober'(y))(z)}\,PR, AR}{z \circ (say' \circ (\lambda p_t.p \circ_s (\lambda x.x \circ sober'))) \vdash say'(sober'(y))(z)[y :\rightarrow z]}\,Z}{z \circ (say' \circ (\lambda p_t.p \circ_s (\lambda x.x \circ sober'))) \vdash say'(sober'(z))(z)}\,\backslash R}{say' \circ (\lambda p_t.p \circ_s (\lambda x.x \circ sober')) \vdash \lambda z.say'(sober'(z))(z)}\,\backslash R$$

Types, $say' : t(et)$; $sober' : et$; $p : t$,

where the LF item for *that* is an identity function $\lambda p.p$ of type (t, t).

In (357a), the external NP argument of the verb *say* is NP, rather than the hypothetical category $\Diamond\Box^{\downarrow}NP$. This is because we do not need to modify the binary

configuration of the antecedent before we apply $\backslash R$ to abstract away from this NP argument. Thus, we may identify the hypothetical category $\Diamond_p\Box^{\downarrow}NP$ which has been introduced by the pronoun with this NP by using Z before we abstract away from this NP argument. The semantics derives the correct binding relation.

We may need to modify the details of the analysis to explain more complicated data about the binding of pronouns by Wh-operators/relative pronouns but I do not investigate it any further here.

Getting back to the derivation of the sentence in question, given the result in (356), we merge three more items as shown in (359).

We contiune the derivation of (350). To reduce the size of the proof representations, I use the abbreviations in categorial formulas as in (358) from now.

(358) a. $Q1$ for $S/(NP\backslash S)$.
$Q2$ for $(S/NP)\backslash S$.

b. $R1$ for $(N\backslash N)/(NP\backslash S)$ (i.e. for *who*).
$R2$ for $(N\backslash N)/(S/\Diamond\Box^{\downarrow}NP)$ (i.e. for *that*).

c. TV for $(NP\backslash S)/NP$ (i.e. for *knew*).

d. TV_q for $(NP\backslash S)/((S/NP)\backslash S)$ (i.e. for *meet*).

e. $IndA_p/N$ for $(N/\Diamond_p\Box^{\downarrow}NP)/N$ (i.e. the category for the indefinite article, *a*).

f. Pro_p for $NP/\Diamond_p\Box^{\downarrow}NP$ (i.e. the category for the bound pronoun, *he*).

g. $\Diamond_p t_i$ for $\Diamond_p\Box^{\downarrow}NP$ (i.e. the hypothetical category for indefinites).

h. $\Diamond_p t_p$ for $\Diamond_p\Box^{\downarrow}NP$ (i.e. the hypothetical category for pronouns).

At the moment, the hypothetical category that the indefinite article introduces and the one that a (bound) pronoun introduces are the same (that is, $\Diamond_p\Box^{\downarrow}NP$), ignoring the anti-locality constraints that apply only to bound pronouns, as I have explained before. Thus, the different abbreviations that I use as in (358g) and (358h) are only for keeping track of which hypothetical category has been introduced for which expression in the presentation.

Now, given the result in (356), we merge three more items as shown in (359).

(359) *a woman that he knew*

$$\frac{(IndA_p/N \circ (N \circ (R2 \circ ((Pro_p \circ \Diamond_p t_p) \circ TV)))) \circ \Diamond_p t_i \vdash N}{((a \circ (woman \circ (that \circ ((he \circ \Diamond_p pro) \circ knew)))) \circ \Diamond_p ind \vdash a \cdot (woman \cdot (that \cdot (he \cdot knew)))}$$

I have inserted $\Diamond_p\Box^{\downarrow}pro$ and $\Diamond_p\Box^{\downarrow}ind$ in the antecedent of the PF sequent in the bottom row in the locations of the hypothetical categories for presentation reasons. Again, because we must discharge hypothetical categories by the end of each proof/derivation, there are no PF items corresponding to them when we actually derive the PF string.

Next, in (360) below, we merge the PF null existential determiner *some** with the output of (359). We also merge the transitive verb *meet* with the result.

In (360) below, we first merge the PF null quantificational determiner '*some**' with the output of (359) and then merge the verb *meet* with the result. After that, we apply *PR*, followed by *AR* seven times. '$\Diamond_p t_p$' and '$\Diamond_p t_p$' are not represented in the PF string.

(360) *met a woman that he knew*

a. Syntax:

$$\dfrac{\dfrac{\dfrac{\dfrac{TV_q \circ (Q2/N \circ ((IndA_p/N \circ (N \circ (R2 \circ ((Pro_p \circ \Diamond_p t_p) \circ TV)))) \circ \Diamond_p t_i)) \vdash NP\backslash S}{TV_q \circ (Q2/N \circ ((IndA_p/N \circ (N \circ (R2 \circ ((Pro_p \circ TV) \circ \Diamond_p t_p)))) \circ \Diamond_p t_i)) \vdash NP\backslash S}\,PR}{TV_q \circ (Q2/N \circ (((IndA_p/N \circ (N \circ (R2 \circ (Pro_p \circ TV)))) \circ \Diamond_p t_p) \circ \Diamond_p t_i)) \vdash NP\backslash S}\,AR}{((TV_q \circ (Q2/N \circ (IndA_p/N \circ (N \circ (R2 \circ (Pro_p \circ TV)))))) \circ \Diamond_p t_p) \circ \Diamond_p t_i \vdash NP\backslash S}\,AR}{met \circ (\epsilon \circ (a \circ (woman \circ (that \circ (he \circ knew))))) \vdash met \cdot (a \cdot (woman \cdot (that \cdot (he \cdot knew))))}$$

b. Semantics:

$$\dfrac{((TV_q \circ (Q2/N \circ (IndA_p/N \circ (N \circ (R2 \circ (Pro_p \circ TV)))))) \circ \Diamond_p t_p) \circ \Diamond_p t_i \vdash NP\backslash S}{((met_q \circ (some' \circ (a' \circ (woman' \circ (rel' \circ (\lambda u_e.u \circ know')))))) \circ x) \circ y \vdash \alpha}$$

where

$\alpha = \lambda z.some'(a'(rel'(\lambda v_e.like'(v)(x))(woman'))(y))(\lambda m_e.meet'(m)(z))$

$= \lambda z.some'(a'(\lambda v_e.[woman'(v)\&know'(v)(x)])(y))(\lambda m_e.meet'(m)(z))$.

Remember from (358g) and (358h) that $\Diamond_p t_i$ and $\Diamond_p t_p$ are the same hypothetical category, and the different indices on t are for distinguishing the two occurrences of the same category. In the semantics, we should not take x and y too literally. They are hypothetical variables to keep track of the locations of hypothetical arguments, and thus, they are meant to be discharged later. By the end of the proof, no term is left in the in-situ position where the hypothetical variable was introduced in the proof presentation.

For presentation reasons, I let α represent the final semantic output of (360b), as in the bottom line of (360b) and use it in the next step in (361b) below.

In the next step, we identify the hypothetical arguments y and x with the embedded subject argument by applying Z in succession with regard to these two hypothetical arguments. After that, we abstract away from the subject argument, which amounts to binding the three argument slots (that is, the extra argument slots introduced by the pronoun and the indefinite and the external argument slot z of the embedded functor, *meet'*) by one lambda operator.

(361) a. Syntax:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{NP \vdash NP \quad ((TV_q \circ (Q2/N \circ (IndA_p/N \circ (N \circ (R2 \circ (Pro_p \circ TV)))))) \circ \Diamond_p t_p) \circ \Diamond_p t_i \vdash NP\backslash S}{NP \circ (((TV_q \circ (Q2/N \circ (IndA_p/N \circ (N \circ (R2 \circ (Pro_p \circ TV)))))) \circ \Diamond_p t_p) \circ \Diamond_p t_i) \vdash S}\,\backslash L}{NP \circ ((TV_q \circ (Q2/N \circ (IndA_p/N \circ (N \circ (R2 \circ (Pro_p \circ TV)))))) \circ \Diamond_p t_p) \vdash S}\,Z}{NP \circ (TV_q \circ (Q2/N \circ (IndA_p/N \circ (N \circ (R2 \circ (Pro_p \circ TV)))))) \vdash S}\,Z}{TV_q \circ (Q2/N \circ (IndA_p/N \circ (N \circ (R2 \circ (Pro_p \circ TV))))) \vdash NP\backslash S}\,\backslash R}{met \circ (\epsilon \circ (a \circ (woman \circ (that \circ (he \circ knew))))) \vdash NP\backslash S}\,PF$$

b. Semantics (omitting $\backslash L$ in (361a), which would apply the functor α in (360b) to n as shown below the proof).

$$\dfrac{\dfrac{\dfrac{n \circ (((met_q \circ (some' \circ (a' \circ (woman' \circ (rel' \circ (\lambda u_e.u \circ know')))))) \circ x) \circ y) \vdash \beta}{n \circ ((meet'_q \circ (some' \circ (a' \circ (woman' \circ (rel' \circ (\lambda u.u \circ know')))))) \circ x) \vdash \beta[y :\rightarrow n]}\,Z}{met'_q \circ (some' \circ (a' \circ (woman' \circ (rel' \circ (\lambda u.u \circ know'))))) \vdash \beta[x :\rightarrow n][y :\rightarrow n]}\,Z}{met'_q \circ (some' \circ (a' \circ (woman' \circ (rel' \circ (\lambda u.u \circ know'))))) \vdash \lambda n.\beta[x :\rightarrow n][y :\rightarrow n]}\,\backslash R$$

where

$\beta = \alpha(n)$

$= (\lambda z.some'(a'(\lambda v.[woman'(v)\&know'(v)(x)])(y))(\lambda m_e.meet'(m)(z)))(n)$

$\Rightarrow some'(a'(\lambda v.[woman'(v)\&know'(v)(x)])(y))(\lambda m_e.meet'(m)(n))$

Thus,

$\beta[x :\rightarrow n][y :\rightarrow n]$

$= some'(a'(\lambda v.[woman'(v)\&know'(v)(n)])(n))(\lambda m_e.meet'(m)(n))$

Thus,

$\lambda n.\beta[x :\rightarrow n][y :\rightarrow n]$

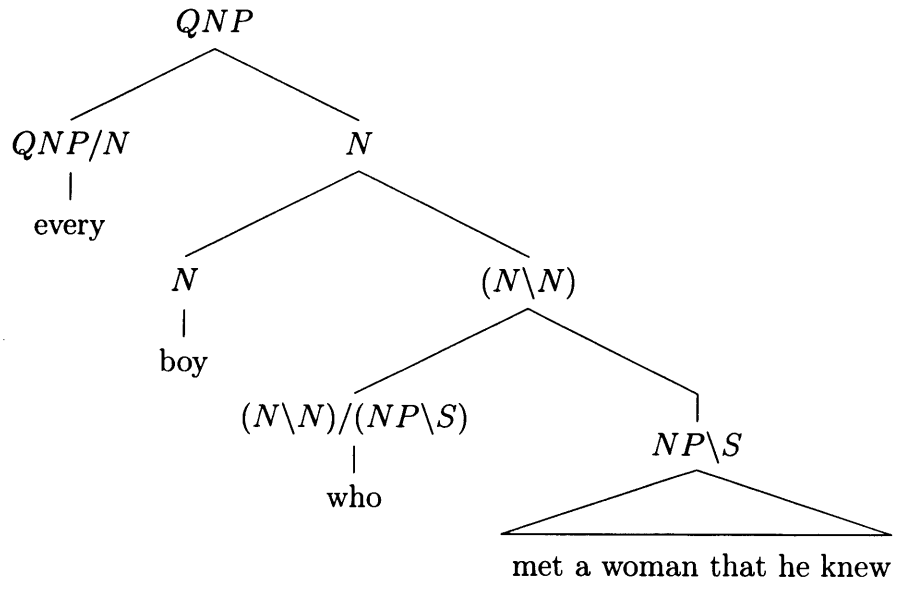
$= \lambda n.some'(a'(\lambda v.[woman'(v)\&know'(v)(n)])(n))(\lambda m_e.meet'(m)(n))$

Look at the final result in (361b) and notice that the lambda operator 'λn' now binds three argument slots. That is, from left to right in the lambda term, the

operator binds 1) the extra argument slot introduced by the bound pronoun (= the external argument slot of *know'*), 2) the extra argument slot introduced by the indefinite article (on which the domain of the indefinite will be dependent) and the external argument slot of the verb *meet*.

After we have derived the appropriate logical expression as in (361b) for the PF string, *met a woman that he knew*, the 'binding' of the three argument slots (filled out by n in the normalized lambda term in the bottom line in (361b)) by the universal quantifier is straightforward. It is just three steps of function application (that is, $\backslash E$ or $/E$). I show the syntax for this in the tree representation in (362). In the semantics in (363), I show the term reduction steps that lead to the binding of the two extra argument slots (for the pronoun and the indefinite) by the universal quantifier. Some of the obvious intermediate steps are omitted.

(362) Syntax:



(363) Semantics:

1) $boy \circ (who \circ (met \circ (woman \cdot that \cdot he \cdot knew))) :$
$rel'(\lambda n.some'(a'(\lambda v.[woman'(v)\&know'(v)(n)])(n))(\lambda m_e.meet'(m)(n)))(boy')$
$\Rightarrow \lambda x.[boy'(x)\&some'(a'(\lambda v.[woman'(v)\&know'(v)(x)])(x))(\lambda m_e.meet'(m)(x))]$

2) $every \circ (boy \circ (who \circ (met \cdot a \cdot woman \cdot that \cdot he \cdot knew))) :$
$\lambda A.\lambda B.every'(A)(B) \circ$
$\lambda x.[boy'(x)\&some'(a'(\lambda v.[woman'(v)\&know'(v)(x)])(x))(\lambda m_e.meet'(m)(x))]$
$\Rightarrow$
$\lambda B.every'(\lambda x.[boy'(x)\&$
$some'(a'(\lambda v.[woman'(v)\&know'(v)(x)])(x))(\lambda m_e.meet'(m)(x))])(B)$

In the step 1) in the semantics in (363), because of the logical expression for the relative pronoun *who* in (351f), all the n argument slots that have been bound by the lambda operator 'λn' in the bottom line in (361b) are now newly bound by 'λx,' which also binds the argument slot of *boy'*, the head of the relative. In the next step 2), the result expression of the step 1) saturates the first argument slot of the universal determiner functor *every'*, and thus, all of these x argument slots are bound by the universal quantifier, as desired for deriving the reading that we are interested in. If we merge the resultant universal QNP with the main clause verb *left*, we can derive the sentential logical form. My analysis merges QNPs as arguments of verbal functors, and thus, strictly speaking, we have to apply argument slot raising to the NP argument slot of *leave*, but because it leads to the same result as the derivation that merges the universal QNP as the functor in this case, I omit this process, and merge Q1 as the functor. This helps simplify the derivation of the normalized logical form.

(364) a. Syntax with PF:

$$\frac{(Q1/N \circ (N \circ (R1 \circ (TV_q \circ (Q2/N \circ (IndA_p/N \circ (N \circ (R2 \circ (Pro_p \circ TV))))))))) \circ (NP\backslash S) \vdash S}{(every \circ (boy \circ (who \circ (met \circ (\epsilon \circ (a \circ (woman \circ (that \circ (he \circ knew))))))))) \circ left \vdash S}$$

b. Semantics:

$(every \circ (boy \circ (who \circ (met \cdot a \cdot woman \cdot that \cdot he \cdot knew)))) \circ leave$:

$(\lambda B.every'(\lambda x.[boy'(x)\&some'(a'(\lambda v.[woman'(v)\&know'(v)(x)])(x))(\lambda m_e.meet'(m)(x))])(B))$
$(leave')$

$\Rightarrow every'(\lambda x.[boy'(x)\&some'(a'(\lambda v.[woman'(v)\&know'(v)(\mathbf{x})])(\mathbf{x}))(\lambda m_e.meet'(m)(x))])(leave')$

In the derivation in (364), the two extra argument slots (which are highlighted as **x** in the bottom line in (364)) are bound by the universal quantifier, as the reading in question requires. However, note that when we have two occurrences of the hypothetical category '$\Diamond_p \Box^{\downarrow} NP$' in the derivation, we may apply Z at different stages of the derivation with regard to these two hypothetical categories and with their (different) binders. Thus, we may apply Z at different stages As I briefly discuss in section 9.7.1, the two argument slots can be bound by different operators.

I have shown the derivation of a sentence that has a simple indefinite without *certain*, and though we may still restrict the set of women that each boy knew to a singleton set by using pragmatic inferences, the singleton set requirement is not linguistically provided. If we inserted the expression *certain* inside the indefinite,

then the singleton set requirement would become explicit in the logical form, as is shown in (365b).

(365) a. Syntax with PF:

$$\frac{(Q1/N \circ (N \circ (R1 \circ (TV_q \circ (Q2/N \circ (IndA_p/N \circ (N \circ (R2 \circ (Pro_p \circ TV))))))))) \circ (NP\backslash S) \vdash S}{(every \circ (boy \circ (who \circ (met \circ (\epsilon \circ (a \circ (certain \circ (woman \circ (that \circ (he \circ knew))))))))))) \circ left \vdash S}$$

b. Semantics:

$(every \circ (boy \circ (who \circ (met \cdot a \cdot woman \cdot that \cdot he \cdot knew)))) \cdot leave :$

$(\lambda B.every'(\lambda x.[boy'(x)\&some'(sg'(\lambda v.[woman'(v)\&know'(v)(x)])(x))(\lambda m_e.meet'(m)(x))])(B))$
$(leave')$

$\Rightarrow every'(\lambda x.[boy'(x)\&some'(sg'(\lambda v.[woman'(v)\&know'(v)(x)])(\mathbf{x}))(\lambda m_e.meet'(m)(x))])(leave')$

Looking at the derived logical form at the bottom in (364b) and in (365b), note that the extra argument slot that is introduced by the indefinite (that is, **x**, in bold face) and the extra argument slot that has been introduced by the pronoun (the external argument slot x of the functor *know'*) are separately bound by the universal quantifier. Thus, unlike in Winter's choice function analysis, we can represent the dependency of the indefinite and the dependency of the pronoun in the nominal restriction of the indefinite separately, even when these are dependent on the same antecedent. Again, I argue that this is desirable from the viewpoint of semantic compositionality.

In the next subsection, I provide a brief speculation about interaction of the suggested domain dependency mechanism with the QNP scope.

9.6.3 Interaction with QNP scope algorithm

We need to explain an asymmetry between subject position and object position in terms of the domain-restriction dependency. For example, in (366), the domain restriction of the indefinite in the subject position does not seem to be able to be dependent on the universal quantifier in the object position so easily.

(366) a. A certain woman loves every boy.

b. *? For each boy x, x is loved by x's mother (for example).

(366a) does not easily get the reading (366b). In the system that I have shown, the extra argument slot of the functor '*sg'*' associated with the indefinite *a certain woman* can only be bound by a quantifier that is merged later in the derivation. Because the scope switch mechanism that I have proposed in this thesis does not

change the order of the merges of the two QNPs in question, the combination of my QNP scope analysis using argument slot raising and the domain dependency analysis of indefinites does predict the difficulty of getting the inverse dependency reading with the sentence in (366a). On the other hand, the judgment in (366) is not very clear, and I leave further investigation of the interaction between the domain dependency of indefinites and QNP scope for future research.

In section 9.6, I have shown some Gentzen Sequent proofs for deriving the dependency of an indefinite to another quantifier in Moortgat's Multi-Modal Type Logical framework. In section 9.7.1, I discuss some of the linguistic loose ends, abstracted away from Type Logical Grammar.

9.7 Extensions (Speculation)

9.7.1 Multiple binding

The analysis in the previous section allows us to percolate more than one extra argument slot into later stages of derivation, to deal with multiple bound pronouns appearing in a sentence:

(367) Every father$_1$ [$_{VP}$ told [his$_1$ son]$_2$ [$_{CP}$ that he$_1$ would buy him$_2$ a$_1$ present]].

At the derivational stage of the embedded CP, the composed logical form should be of type (e(e(et))), because of the three extra argument slots that have been introduced by the two pronouns and the indefinite in the embedded clause. One of the pronouns inside the embedded CP (i.e. *him*$_2$) gets bound before the main clause VP is completed, but by the time we derive the main clause VP, we have got another bound pronoun (i.e. *his*$_1$). Thus, at that stage, there are again three extra argument slots in total. After that, both of them get bound by the subject QNP *every father*. Remember that the different extra argument slots introduced by different pronouns/indefinites do not have to be bound by the same antecedent in my analysis. They can be bound by the same operator by applying Z in succession with the same antecedent, as is the case with the argument slots introduced by *he*$_1$ and *his*$_1$ and the argument slot introduced by a_1 *(present)* in (367).

As I have suggested with my derivation for Winter's sentence in (330) in section 9.6.2, this means that even when a bound pronoun appears in the nominal restriction of an indefinite, we can separately bind the two extra argument slots that the indefinite and the pronoun introduce.

On the other hand, do the data really support a 'wide scope' specific reading in which the extra argument slot introduced by the pronoun and the extra argument slot with the indefinite *a (certain)* are bound by different operators in the sentence? Consider (368):

(368) Every psychiatrist says that every child$_1$ who hates a certain woman he$_1$ knows will develop a complex.

Can the relation between each child x and the woman for x co-vary with each psychiatrist? It is not very easy to get the reading which says that for each psychiatrist, there is a possibly different relation holding between each child and the woman concerned. On the other hand, this might be due to difficulty processing the complex sentence.

It is possible to formulate the theory in a way such that whenever some pronoun in the nominal restriction gets bound, the extra argument slot introduced with the indefinite also has to be bound. But I do not see a strong reason for adding that extra condition, so I just assume that a further percolation of the indefinite argument across the universal QNP *every child* in (368) is linguistically possible, but because it is pragmatics that actually restricts the domain within that linguistic information, relativizing the domain restriction both to a bound pronoun and to an inherent indefinite argument is quite difficult, as a matter of non-linguistic interpretation.

9.7.2 Wide scope indefinites: bound by the tense operator?

In order to explain the reading corresponding to the inverse scope reading of the indefinite, I need to have the extra argument slot of the indefinite bound by an element other than a quantifier in a QNP. One candidate might be the tense operator that can be higher than the subject QNP in the syntactic structure.[18]

(369) a. Every boy loves a certain girl.
(Inverse scope: $a \cdot certain > every$)

b. Indefinite dependent on the universal QNP:
$\exists t_\sigma[G'_{(\sigma,t)}(t)\&\{\forall x[boy'(x) \rightarrow \exists y[(sg'(woman'))(x)(y)\&love'(y)(x)]]\}_{(\sigma,t)}(t)]$

c. Indefinite dependent on the Tense operator:
$\exists t_\sigma[G'_{(\sigma,t)}(t)\&\{\forall x[boy'(x) \rightarrow \exists y[(sg'(woman'))(t)(y)\&love'(y)(x)]]\}_{(\sigma,t)}(t)]$

[18] $\{\phi\}(t)$ in (369) is a notational device that shows that the logical expression ϕ is a function from a time t to a proposition. The type σ is a type for an expression denoting a tense. G' is some tense, such as *Present'*, *Past'* whose exact identity does not matter here.

However, the tense operator does not always take wide scope over the subject quantifier.

(370) a. Every kid ran.

b. A friend often came to see Tom in London.

(370a) has a reading in which each kid ran at a possibly different time, and (370b) has a reading in which the same friend visited Tom many times. This does not necessarily stop us from using the tense operator to explain the wide scope of the indefinite over another QNP, as long as the tense operator can at least sometimes take the widest scope, but the issue requires further research in terms of the interaction of the scopes of QNPs and the tense or some other operators that can bind the extra argument slot of indefinites.

9.7.3 The functor sg′

Some might claim that the treatment of sg' as a constant expression is problematic. Given a set of individuals, say, the set of women denoted by $woman'$, and given some individual, say, Tom denoted by tom', for example, there are many candidate singleton sets from which we can choose the denotation of $sg'(woman')(tom')$, unless the set of women for Tom is already a singleton set in the given model before we apply the functor sg' to it.

From a more empirical viewpoint, in (371) below, the hearer is usually not expected to know the identity of the specific relationship that is supposed to hold between every pair of a boy and the man for him, even though the father-son relationship is a possible relation that the speaker can have in mind, as is shown in the parentheses to the right of the sentence.

(371) Every boy_1 respects a (certain) man (that is, his father_1).

To explain the data, the function denoted by $sg'(man')$ should be able to map an individual x to the same singleton set that the function denoted by $\lambda x.the\text{-}father\text{-}of'(x)$ does in some context, where the latter function maps an individual x to the singleton set that contains the father of x as its unique member. But this is not always the case. In a different context, $sg'(man')$ should also be able to denote a function that maps an individual x to the same singleton set that the function denoted by $\lambda x.the\text{-}maternal\text{-}grandfather\text{-}of'(x)$ does.

Does this mean that we have to treat sg' as a variable, which would lose one of the proposal's merits in comparison to Skolem/choice function analyses, that is, the compositional derivation of logical forms?

I argue that the definition of sg' as a constant functor is no more a problem than the definition of man' as a constant expression. With regard to a specific relation which, according to some speakers' data judgment, should hold for all the boy-man pairs and which should be able to vary with each context, remember that the specific relation between the individuals in question is not linguistically encoded in the proposed analysis. What is encoded with the use of *certain* is the function sg' which maps each boy to a singleton man-set for him, and it is only that the hearer may pragmatically speculate that the speaker must have a specific relation in mind which holds for all the pairs of individuals, given that the speaker has used an expression that requires a formation of a singleton set for each boy in the model. If the indefinite is without *certain*, even the singleton set interpretation is totally left for pragmatic inferences. In this partly pragmatic analysis, the hearer is actually not expected to know the identity of the specific relation that holds for all the pairs of individuals. Thus, even if the hearer takes it that the speaker has in mind a specific relation between each pair of individuals with his use of '$sg'(man')$,' and even if that relation happens to be the one denoted by '$\lambda x.the\text{-}father\text{-}of'(x)$' in the speaker's mind in some context, that does not mean that the two functor expressions have the same meaning, even in that given context.

The co-extension of the term $sg'(man')$ and $the\text{-}father\text{-}of'$ (which maps each individual x to the singleton set that contains x's father) in some context is not a problem for a similar reason, but in this case, the denotations of both the expressions become the same relative to the model. Because the denotations might not be the same in other models, this is not necessarily a problem, but some more must be said about the intended interpretation of sg' as a constant expression.

The potential problem of fixing the interpretation of $sg'(woman')(tom')$ in each model (even though there is potentially more than one singleton woman-set from which we can choose) will be a real problem if we have to use more than one singleton woman-set for each individual (i.e. Tom) in one model (or, informally, in one pragmatic context), but for lack of clear data which require such multiplication, the alleged problem seems to me to be more a matter of methodological preference. I do not see any essential problem about postulating that the function sg' maps each set of individuals to exactly one singleton set for each individual in the model in each context, though probably this becomes even less a problem with the alternative formulation of my analysis which I briefly discuss in section 9.7.4.

9.7.4 Extra argument slots with nominal restriction sets

I have encoded the extra argument slot in question with indefinites in argument positions, with or without *certain*, but the dependency of the domain of the nominal restriction seems to be more general than being limited to the indefinite, as we see in the text around (374) later, and thus, I show an alternative formulation in which the extra argument slot is encoded with the nominal restriction.

In this alternative formulation of the domain restriction analysis, the normalized logical form for the sentence in (371) would be as in (372a) which is equivalent to (372b), with the lexical entries of the crucial items as in (372c)~(372d).

(372) *Every boy*$_1$ *respects a certain man*$_1$.

a. $\forall x[boy'(x) \rightarrow \exists y[a_2(sg_2'(man_2'(x)))(y)\ \&\ respect'(y)(x))]]$
$\Leftrightarrow$ (372b)

b. $\forall x[boy'(x) \rightarrow \exists y[[man_2'(x)(y)\ \&$
$\forall z[man_2(x)(z) \rightarrow y = z]]\ \&\ respect'(y)(x))]]$

c. man'$_2$: $< man; N/\Diamond\Box^{\downarrow}U; \lambda x_\tau.\lambda y_e.man_2'(x)(y) >$
$man_2 \stackrel{\text{def}}{=} \lambda x.\lambda y.[man'_{et}(y)\ \&\ for'_{e(et)}(x)(y)]$

d. a certain: $< a \cdot certain\ ; N/N;\ sg_2' >$
where $sg'_2 \stackrel{\text{def}}{=} \lambda B_{et}.\lambda y_e.[B(z)\ \&\ \forall z_e[B(z) \rightarrow y = z]]$

e. a: $< a; N/N; \lambda B_{et}.\lambda x_e.a_2'_{(et)(et)}(B)(x) >$

f. some*: $< \epsilon;\ QNP/N;\ some' >$
where $some' \stackrel{\text{def}}{=} \lambda A_{et}.\lambda B_{et}\exists y[A(y)\ \&\ B(y)]$

The common noun *man* in (372c) is lexically given the category $N/\Diamond\Box^{\downarrow}U$ and the type $(\tau, (et))$. U is usually instantiated as NP and τ is normally instantiated as e. Then we get the category and the type that are usually given to a relational noun such as *mother* or *father*. The logical expression $man_2'(x)$ denotes a set of men that is possibly different for each individual x. We can define man'_2 of type (e(et)) from the standard expression of type (et), that is, man', as shown in (372c), where the constant for' expresses some adequate binary relation between individuals.

The logical expression for *certain* is now the functor sg_2' of type ((et)(et)), which denotes a function that maps this set of men to a singleton set of men, which is again possibly different for each individual x. I treat the indefinite article *a* as an identity function of the category/type that is provided in (372e), ignoring the contribution in terms of its singularity information. The PF null existential

determiner *some** is maintained as in section 9.6, though I provide its definition in the predicate calculus notation in (372f) which is equivalent to the original higher order lambda language notation.

The binding of the extra argument slot that is now associated with the nominal restriction by some operator that is merged later than the indefinite is as in section 9.6. In (372b), the extra argument slot ends up being bound by the universal quantifier in *every boy*. Then we get the desired reading that says that for each boy x, there is a possibly different singleton man-set and x respects the unique member of that set, e.g. x's father y in (371).

As in the section 9.7.3, note that this analysis does not imply that the intended interpretation of $a(sg'(man'(x))$ is the same as the interpretation of $\lambda y.the\text{-}father\text{-}of'(x)(y)$ for the same individual x, as should become clear from the logical form for *Every boy*$_1$ *respects his*$_1$ *father* as shown in (373) in which I interpret the definite description by using the Russellian quantificational interpretation.[19]

(373) *Every boy*$_1$ *respects his*$_1$ *father.*
$\forall x[boy'(x) \rightarrow \exists y[[father\text{-}of'(x)(y)\ \&$
$\forall z[father\text{-}of'(x)(z) \rightarrow y = z]]\ \&\ respect'(y)(x))]]$
where $father\text{-}of'$ is type (e(et))

Actually, the constant status of sg'_2 is even less problematic than the constant status of sg'_1. With regard to sg'_2, what we have postulated was that the set of men for each individual (say, for each boy) is fixed in each model, and I can not think of an essential reason to require that the set of men for a particular individual to be able to be variable within one semantic model.[20]

As I have indicated above, this formulation of my analysis implies that the domain restriction dependency is no longer limited to indefinites. If common nouns are generally equipped with this extra argument slot, then the domain restriction

[19]I assume that $father\text{-}of'(x)$ denotes the set of fathers for the individual x. The fact that the father is unique for each x comes from some world knowledge and irrelevant to the encoded semantic type of the functor $father\text{-}of'$. Thus, this uniqueness must be stated separately by attaching the definite determiner.

[20]Cooper (1996) argues that the domain of nominal restriction should be able to be restricted in a different way for each DP appearing in a sentence, arguing for a situation theoretic treatment of DP interpretations. Cooper's proposal's implication for my analysis is not clear because his sets of individuals are not relativized to other elements as with my proposal. Also, the choice of the domain for each nominal restriction set (as opposed to its dependency to other quantifiers) does not involve any obviously structural element with it, and the selection depends heavily on the hearer's inference with regard to the speaker's intentions. Thus, it is not clear if the model theory needs anything more than models (probably in possible world semantics, but I ignore intentionality).

applicable to other QNPs should also be able to be dependent on another quantificational element. This seems to be correct, as we can see in (374). The set of weak points can possibly be different for each player.

(374) Only those players who got rid of every weak point could play in the Major League.

Treating common nouns in general as if they were relational nouns might need more linguistic justification, but the compositional semantics seem to work in a better way with this analysis. Some explanation must be provided about why the definite description does not easily lead to the same kind of dependency in *Every* boy_1 *respects the* man_1, but I speculate that this is because the stronger uniqueness requirement encoded with the definite determiner blocks the co-variation of the man with each boy for some pragmatic reason which is outside the scope of the proposed analysis. Note that even with the relational noun *father*, we cannot get this dependent reading easily. That is, *Every child respects the father* does not mean *Every* $child_1$ *respects* his_1 *father.*

The logical form as in (372a) is postulated for explaining the attested dependency of the indefinite to the universal quantifier in the compositional derivation of the phonological and semantic representations by way of categorial calculus. Given the hypothesis that everything that is derived during the syntactic derivation should have its source in some lexical information (that is, the Chomskian idea of Inclusiveness), it is natural to attribute the source of the dependency relation to some lexical information. This lexicalist assumption is postulated in order to avoid some operation taking place during a syntactic derivation beyond the operation that can be read off the lexically encoded information, maintaining the compositionality of the semantics.

In contrast, whether we need the extra argument slot encoded with the indefinite article or the common noun for the purpose of explaining on-line interpretation data is a separate issue. For example, it is possible to modify the proposed analysis in such a way that the semantic information encoded with the indefinite *a certain man* only assigns the instruction to restrict the set of men into a singleton set. If the pragmatic inference can form the dependency relation of the nominal restriction set to the universal quantifier somehow, then we can derive the attested dependent specific reading without encoding the extra argument slot with the common noun *man.* This alternative analysis would still be able to explain the 'exceptional wide scope' of indefinites without accepting the exceptional existential scope of the indefinite, as well as explaining the domain dependency that we have observed, and

thus it would serve the most important purpose of the thesis (which argues for the tensed clause boundedness of QNP scope for all the QNPs including indefinites).

On the other hand, the domain dependency in the data that we have discussed seems to be sensitive to certain syntactic/structural elements. Given the similarity between the structural configuration that licenses the domain dependency and the structural configuration that licenses the binding of overt pronouns, a purely pragmatic analysis of inferring the domain dependency without manipulating any structured representations would be non-explanatory. In comparison, a partially pragmatic analysis which may construct certain sub-parts of structured Language of Thought representations (which will be expressible by typed lambda terms) beyond the semantic information encoded with overt language expressions might be able to form such dependency relations in an adequate way.[21] However, in that case, whether some sub-terms of the resultant logical form are pragmatically inferred, rather than being encoded with the overt language expressions used by the speaker, is not essential from the viewpoint of the type logical composition of the logical form in question. Even after accepting the possibility of pragmatic addition of information beyond the linguistically encoded information, we could still model the compositional derivation of the logical form as in (372a) based on its subparts, some of which might then have been provided as the result of pragmatic enrichment processes, rather than being the result of the linguistic decoding.

Because of such differences of the purposes of the theories, my proposal does not necessarily support the analyses which use hidden indexical/pronominal elements for explaining natural language interpretation data, such as Stanley (2000), which assumes that whenever the apparent meanings of the overt language expressions do not fully specify the truth condition, there are hidden variables in the encoded meaning representation so that assigning values to those variables will derive the truth condition of the utterance.[22] See also Hall (2006) for some arguments against

[21] I am not suggesting that pragmatics may compose and interpret Language of Thought representations bottom up as in syntactic tree developments. Such structure development and compositional interpretation processes should be explained separately by way of the syntax and the semantics of typed lambda expressions. But as long as pragmatic inferences manipulate such LoT representations, the process of pragmatic enrichment would still be constrained by structural elements in an indirect way.

[22] Also, all the variables that are used in my LF representations are bound, and thus, we can represent the LF terms without using variables if we want, whereas Stanley uses free variables which makes it hard to define compositional semantics of logical language representations, though Stanley might not be concerned about the compositionality of the meanings encoded with the language expressions abstracted away from the context. Use of free variables in the logical language requires use of variable assignment functions in the interpretation and thus, it makes it very difficult to sustain compositionality of the interpretation from the lexical level (see Jacobson (1999) for this point). Though Stanley could argue that his variables can never be free and must

abundant use of hidden indexicals for explaining interpretation data.[23]

Whether the extra argument slot in question should come from the encoded linguistic information or not for the purpose of explaining natural language interpretation data, it seems that the theoretical process of deriving the attested dependency relation in a compositional manner can be captured well by the suggested type logical analysis.

9.8 Summary

In this chapter, I argued that the exceptional wide scope of indefinites is not a matter of QNP scope. It is better explained by Schwarzschild's domain restriction analysis. When the domain of an indefinite's nominal restriction set is restricted into a singleton set, we get the impression that the utterance is about a specific individual. But the intermediate 'scope' reading and the functional reading of the indefinite suggests that this specific individual can co-vary with some other element in the sentence. In order to explain the variability of the specific individual with another element, such as a universal quantifier (say, *every boy*), I argued that the expression *a (certain)* has an extra argument slot of the under-specified type τ. If this slot is bound by a universal quantifier in *every boy*, the domain is restricted in a different way for *each boy*, which leads to a relativized specific reading.

By reformulating Jacobson's pronoun binding algorithm in Multi-Modal type logical grammar as in Moortgat (1997), I have shown how this extra argument slot of an indefinite is compositionally percolated in a syntactic derivation and then gets bound by another element in the sentence.

be bound by lambda operators (without specifying the type of the missing elements strictly), such modification would leave a lot of work for the pragmatics to do, and would make his analysis closer to contextualist views.

[23]Though the implication of such contextualist views for analyses that use extra argument slots (as opposed to additional indexical elements) in the encoded semantic representation is not very clear.

Chapter 10

Conclusions and prospects

10.1 General

This thesis has discussed scope of QNPs in Type Logical Grammar. The linguistic goal of the thesis has been to provide an analysis of QNP scope which can naturally explain why the scope of QNPs stays inside the minimal tensed clause which contains it.

The goal of the thesis in terms of Type Logical Grammar has been to set up the deductive grammar system in a way that respects the division line between QNP scope, A-movement (or more accurately, complex predicate formation) and A-bar movement phenomena, which I argue are all linguistically different.

The conclusion chapter includes the essential points of the proposal with some implications and loose ends. Section 10.2 reviews argument slot raising (ASR) which has been postulated for explaining QNP scope, together with complex predicate formation which is independent of ASR, but may still influence QNP scope. Section 10.3 discusses the basic differences between complex predicate formation and A-bar extraction in my analysis and section 10.4 deals with the implication of my analysis for Type Logical Grammar. Section 10.5 provides the final remarks.

10.2 QNP scope

The informal linguistic idea of QNP scope in my analysis is simple. QNPs are taken in as arguments of the local functors (normally, verbal functors). Therefore, the scope of a QNP cannot exceed the final output category S of the local functor that takes the QNP as an argument.

I have regarded the so-called 'exceptional scope' of indefinites in terms of the

domain restriction of the nominal restriction set to a singleton set. I have also postulated an extra argument slot with the indefinite so that we can explain the so-called intermediate scope reading and functional reading of the indefinite in terms of the binding of the extra argument slot by another quantifier. In this way, we can maintain the uniformly tensed-clause bound logicality contraints on all the quantificational NPs, while explaining various readings of indefinites by way of the compositional derivation of the logical forms in the categorial calculus.

Ignoring exceptional cases in which scope switch is partly dependent on value raising or independent structural ambiguity with regard to adjuncts, the structural configuration for scope switch has been as in (375).

(375) a. $QNP1 \circ_j (Functor \circ_j QNP2) \vdash S$

b. $S/_j(NP\backslash_j S) \circ_j ((NP\backslash_j S)/_j NP \circ_j (S/_j NP)\backslash_j S) \vdash S$

In words, if we create an antecedent structure in which a functor takes in two QNPs in succession (with the merge mode j), producing the clausal category S in the succedent, then argument slot raising is applied to this functor in two different ways, creating scope ambiguity. (376a) is the basic case where the functor is a lexical verb. I treat words connected by '$\cdot$' as one unit for convenience. (376b) requires the formation of a complex predicate before we apply ASR, whereas such complex predicate formation cannot cross the clause boundary marked by the merge mode c in (376c).

(376) a. $A \cdot student \circ_j (reviewed \circ_j every \cdot paper) \vdash S$

some > *every*; *every* > *some*

b. $A \cdot student \circ_j ((tried \circ_i (to \circ_i review)) \circ_j every \cdot paper) \vdash S$

some > *every*; *every* > *some*

c. $A \cdot student \circ_j (thought \circ_j (that \circ_c (Hiro \circ_j (would \circ_i (review \circ_j every \cdot paper))))) \vdash S$

some > *every*; $*$*every* > *some*

The complex predicate formation is controlled by the structural association $A_{i,j}$ which can only go as far as the merge mode i continues, which marks the verbal projection line in each TP (while the merge of NP arguments with sub-component functors via the merge mode j can be postponed until we derive the complex functor). Thus, complex predicate formation in my analysis cannot cross the clause boundary marked by the mode c which merges the complementizer with the local S category expression.

An essential linguistic claim is that the distinction between QNPs and normal NPs is fundamentally semantic and does not influence the syntactic structure in itself. In other words, at least at the observational level, QNPs occupy the same PF positions as normal NPs across languages. Thus, there is no strong motivation for postulating 'syntactic movement' of QNPs out of the argument positions that normal NPs occupy, where the 'syntactic movement' in Type Logical Grammar would be instantiated by using some structural rule(s). Because of this, I have postulated ASR as a semantically motivated operation which applies in order to consume semantic operator expressions of type (e(et)) as arguments of the (verbal) functor in the syntactic derivation, which has led to the formulation of ASR which does not involve any structure modification rules such as structural association. In the current formulation of the theory, the difference between QNPs and NPs is still represented in terms of different 'syntactic categories' that are assigned to them. However, given ASR as a special rule, we can merge the relevant QNP categories in the NP argument positions without modifying the bracketed syntactic structure, while still being able to derive scope ambiguity. With the current formulation, we cannot claim that the QNP scope is not a matter of the syntax, because ASR is incorporated into the categorial calculus as a non-logical axiom. However, when we use ASR to switch scope between two QNPs, the bracketed configuration of categorial formulas stays the same for the two scope readings.

As we have discussed in chapter 2, there are some data which may suggest that QNP scope interacts with syntactic structures in a more essential way. One such interaction is in terms of generalized conjunction, as in (377).

(377) [Neil liked, but every other linguist hated], some/an idea (that I have proposed for QNP scope).

**every > some*; *some > every*

The claim is that the generalized conjunction forces the structure as indicated by the square brackets, according to a prevalent Categorial Grammar analysis as in Steedman (2000b) and this forces the scope reading *some > every*.[1] However, the implication of the data as above for my analysis is not at all clear. The structure in (377) is triggered by the conjunction, not by the existence of QNPs, and the resultant structure may affect QNP scope independent of ASR. In other words, I have only claimed that the scope taking of QNP itself does not involve structure modification, but independent factors, either complex predicate formation or gen-

[1] A QR based analysis may explain this in terms of the illegal status of an extraction only from one side of a co-ordinated structure.

eralized coordination, may affect QNP scope in an indirect way. Also, though the sentence *Jack took, but every (other) student avoided, a logic lecture* is not very natural, it seems grammatical. And we can easily get the surface scope reading, *every* > *a* with it. Thus, the relative difficulty of the surface scope reading with (377) may be a matter of pragmatics. Remember that the inverse scope possibility with the string as in (378b) (where the pronoun *him* is free) does not save the illegal binding relation in (378a), as we have seen in chapter 2.[2]

(378) a. *A present that I have given him_1 pleased every $friend_1$.

b. An/at least one argument that I have used against him (i.e. Michael) convinced everybody.

a > *every*; *?every* > *a*

Further investigation of QNP scope and structure modification is left for future research.

At the descriptive level, my treatment of QNP scope switch in the two kinds of ditransitive constructions implies that we need the presence of an overt functor which takes the two QNPs in question as its co-arguments in the categorial derivation. (379c) and (379d) have such functors between the two QNPs (*to* and *give*, respectively, though for the latter, the right QNP is a complex one). In contrast, the concatenating functor *dtr* which merges two or more (Q)NP arguments into one complex argument is PF null (ϵ in (379b) means PF null), and it does not help switch scope between the two QNP objects.

(379) a. Scope switch configuration in PF:
$QNP1 \cdot (Functor \cdot QNP2)$.

b. DO: $Meg \cdot (gave \cdot ((a \cdot student) \cdot (\epsilon \cdot (every \cdot book))))$

a > *every*; **every* > *a*

c. PP: $Meg \cdot (gave \cdot ((a \cdot book) \cdot (\mathbf{to} \cdot (every \cdot book))))$

a > *every*; *every* > *a*

d. $(A \cdot teacher) \cdot (\mathbf{gave} \cdot (me \cdot (\epsilon \cdot (every \cdot book))))$.

My analysis of QNP scope in the preposition-less double object construction in English has some implication in languages such as Japanese, in which the surface order between the QNPs in front of the main verb unambiguously shows their scope

[2]Again, I have argued that excluding (378a) by postulating QR of the object across the co-indexed pronoun, which would allegedly lead to Weak Cross Over violation, is not explanatory without strong independent justification for QR in the first place, because the pronoun is not c-commanded by the object QNP in the surface position in the first place.

relation, unless scrambling is involved.[3] Also, it may have some implication in the treatment of Hungarian scope data, where if QNPs are placed in front of the main verb, then the left to right order between the QNPs is their scope relation, whereas the scope of a post-verbal QNP relative to the scope of another QNP in front of the verb is still ambiguous, depending on the phonological stress assignment to this post-verbal QNP, as we have seen in chapter 1. If we ignore the influence of scrambling in Japanese, the proposed analysis predicts the fixed scope relation between QNPs which are all placed in front of the main verb, because these QNPs are formed into one complex argument by way of the PF null concatenating operator *dtr* which cannot switch scope. With regard to the effect of scrambling in Japanese, the so-called A-scrambling is known to create scope ambiguity, whereas A-bar scrambling which may go long distance does not lead to scope ambiguity. If we apply the suggested analysis in a simplistic manner, A-scrambling would be instantiated in terms of structural rules controlled by binary merge-mode specification, whereas A-bar scrambling would be instantiated in terms of introduction and discharge of a hypothetical category marked by $\diamond$. This alone does not explain the scope ambiguity data with scrambling because the scrambled QNPs will still all appear in front of the main verb one after another for a clause internal A-scrambling. But we can formulate the grammar in such a way that application of permutation under the control of the i, j modes allows us to merge the resultant verbal functor with the two QNPs in front of the verbal functor one after another, rather than at once, whereas in the base positions, we have to use the lexically provided verbal functor (e.g. of category $(NP \bullet NP)\backslash S$ for transitive verbs in Japanese), which would force the surface scope reading between the two QNPs in front of the verb in the base positions.

My analysis will deal with A-bar scrambling in terms of the introduction and discharge of a hypothetical category marked with '$\diamond$.' Because this procedure does not create a complex predicate to which ASR is applied, it does not create scope ambiguity that is otherwise available for independent reasons.

The exact locality of QNP scope in my analysis hinges on which PF strings can count as a complex predicate in the syntactic calculus, and which strings do not. Thus, the next section includes a brief comparison of complex predicate formation and A-bar extraction in my analysis.

[3]See Hornstein (1995) and Hornstein (1999a) for the fixed scope reading between unscrambled QNPs in Japanese.

10.3 Complex predicates vs. A-bar extraction

I have formed complex predicates by way of structural rules which are sensitive to binary merge mode specification, whereas Wh-extraction is explained by way of the use of a hypothetical category marked with a unary operator $\Diamond$. Formally speaking, there is no essential reason to use two ways of introducing structural rules in this way. Also, though I have argued that the structural association rule introduced under the control of merge mode specification is inherently short distance, it is formally possible to set up the binary merge modes and the association rule in a different way from my analysis so that we could treat a string such as *say that Meg likes* as a complex predicate in *A girl said that Meg likes every boy*, which would wrongly predict that the sentence is scopally ambiguous. Thus, the claim that my analysis naturally explains the tensed-clause bound locality constraints hinges on the linguistic identification of what I have called 'complex predicates.'

Given that NP arguments are merged as NP arguments irrespective of whether the functor is a complex functor or a lexical functor, I have assumed that the information that licenses complex predicate formation is encoded with the sub-part functor expressions, especially with the higher functor that selects VPs as their complements, rather than with the argument expressions. These linguistic formulations are easier to instantiate with regard to merge-mode specification, rather than by using unary operators.

In contrast, I have postulated that Wh-extraction is induced because of the lexical information of the Wh-expression itself. The lexical categorial information that I have postulated requires the use of a hypothetical category marked with '$\Diamond\Box^{\downarrow}NP$' as a temporary filler in the corresponding argument position, and I have also postulated that the discharge of this hypothetical category can be done in arbitrarily many steps after it is provisionally inserted in the in-situ argument position. Thus, the default setting of this mechanism is lack of locality constraints, though we can add locality constraints in terms of the unary operator '$\Box^{\downarrow}$' which can be encoded with certain 'island creating' lexical items such as the head of the relative *who*, or certain kind of prepositional heads such as *without*. Note that such an analysis still conforms to the Chomskian principle of lexical Inclusiveness, where the introduction/discharge of the hypothetical category and the creation of an island are all introduced by some lexical categorial information.

Given that a possibly long-distance overt extraction is associated with a certain kind of operator status of the extracted item in my analysis, one interesting phenomenon will be overt topicalization. As Johnson (2000) has shown, an overt

topicalization is subject to more or less the same locality constraints that Wh-movement is subject to, and considering its potentially long-distance nature, I will have to formulate it in terms of the structural rules controlled by the unary operators in my analysis, which would lead to some 'operator' interpretation of NPs. On the other hand, unless we do something special, use of a type-raised NP argument would lead to the same normalized logical form that we would derive by merging an NP in the in-situ position, as we can see in the informal representation in (380).

(380) Broccoli, Meg said that Ad likes.

a. $Broccoli \circ (Meg \circ (said \circ (that \circ_c (Ad \circ likes)))) \vdash S$

b.
$$\frac{S/(S/\Diamond\Box^{\downarrow}NP) \circ S/\Diamond\Box^{\downarrow}NP \vdash S}{Broccoli \circ (Meg \cdot said \cdot that \cdot Ad \cdot likes) \vdash S}$$

c. $\lambda P_{et} P(broccoli') \circ (\lambda x.say'_{t(et)}(like'(x)(ad'))(meg'))$
$\Rightarrow_{\beta reduction} say'((broccoli')(ad'))(meg')$

Future research may include how to incorporate the syntactic 'operator' status of the extracted NP which is not reflected in the normalized logical form in the explanation of the extra layer of the semantics as has been investigated in various kinds of information structure theories, as in Steedman (2000a).

10.4 Looser relation between categorial calculus and typed lambda terms

As I have implied above, one important linguistic claim is that the distinction between QNPs and normal NPs is fundamentally semantic and that is why my analysis of QNP scope by way of either argument slot raising or value raising does not affect syntactic structures. On the other hand, my analysis comes at a cost of losing the tight correspondence between the categorial calculus and the intended semantics at the LF side.

More specifically, the system that I have used is $NL_{\Diamond_x}$ plus ASR as a special rule, where $NL_{\Diamond}$ is non-associative, non-commutative Lambek calculus NL enriched with controlled structural rules in terms of families of binary and unary connectives. If we ignore the formal properties that underlie the operation of ASR and represent the output of ASR applied to the functor *like'* of type (e(et)) for the sentence *A boy likes every girl*, then the two scope readings can be represented as in (381a) and (381b).

(381) A boy likes every girl.

a. Surface scope: $like'_{qs}(some'(boy'))(every'(girl'))$
$\Leftrightarrow$ $some'(boy')(\lambda y.every'(\lambda x.(like'(x)(y))))$
where $like'_{qs} \stackrel{\text{def}}{=} \lambda Q1.\lambda Q2.Q2.(\lambda y.Q1.(\lambda x.like'(x)(y)))$

b. Inverse scope: $like'_{qi}(some'(boy'))(every'(girl'))$
$\Leftrightarrow$ $every'(girl')(\lambda x.some'(girl'(\lambda y.like'(x)(y))))$
where $like'_{qi} \stackrel{\text{def}}{=} \lambda Q1.\lambda Q2.Q1(\lambda x.Q2(\lambda y.like'(x)(y)))$

Note that given the definitions of the two variant functor expressions $like'_{qs}$ and $like'_{qi}$ as in (381a) and (381b), the structures of the two logical forms in the top lines in (381a) and (381b) are exactly the same. The structural correspondence between the two logical forms corresponds to the same syntactic structure that my analysis has implied for both the scope readings. The remaining question is how much we should investigate the meta type logical system that would be able to derive ASR as a theorem. As we saw in chapter 2, the associative and non-commutative Lambek Calculus L would derive the effects of ASR, but L would do more than what ASR (with NL) can do and over-generate the scope readings. Also, if we adopted it as the base grammar system, it would over-generate the PF strings (see Moortgat (1997) for some discussions in this regard). If ASR is all that we need on top of my MMTLG based grammar system, then we can simply use ASR as an un-decomposable special axiom. However, I have used a few other special rules on top of ASR in the thesis, such as the algorithm that has merged two object QNPs and has produced a complex QNP object in chapter 5, and the algorithm used for re-analyzing PPs that are VP adjuncts as an additional complement of the head of the VP in chapter 6. For the latter, I have suggested that I will reformulate this process as part of the categorial calculus, by introducing a structural rule that is somehow related to some of the rules that we have already used. This is because the latter rule is not essentially related to QNP scope itself, and has obvious structural elements involved in it. For the former, however, it is a matter of QNP scope, so in future research I will have to consider some underlying algorithm that can generate ASR, part of which might be usable for deriving the rule that merges two QNPs into one complex QNP with some modification.

Finally, I have not provided the intended semantics of the extension of NL, either in the PF side or in the LF side. Some semantics using Kripke frames, as has been suggested in Kurtonina (1994), would provide the basic intended semantics with regard to which $NL_{\Diamond_x}$ is sound and complete, but details are left for future research.

10.5 Final remarks

QNP scope has been explained in terms of a non-logical axiom ASR, which is postulated for interpreting QNPs that are semantically operators in the in-situ argument positions in the syntactic derivation. Because the operation applies to the local functor of the QNP argument(s), the QNP scope cannot exceed the final output of this functor, which is the minimal S expression (or the minimal TP in Minimalism) which contains the QNP(s). Instantiating this idea in Type Logical Grammar requires a Multi-Modal set up, but with two ways of introducing structural rules for complex predicate formation and A-bar extraction, the proposed grammar system succeeds in regarding the minimal S category that contains the QNP(s) in question as the maximal scope of the QNPs. The status of ASR as a non-logical axiom is problematic from a theoretical viewpoint, and thus further investigation about a better way of achieving the effects of ASR is left for future research.

Appendix A

List of abbreviations

1. AR: Association Right.
2. ASR: Argument slot raising.
3. L: associative, non-commutative Lambek Calculus.
4. LoT: Language of Thought.
5. MMTLG: Multi-Modal Type Logical Grammar
6. NL: non-associative, non-commutative Lambek Calculus
7. PR: Permutation Right.
8. QNP: quantificational noun phrase.
9. QR: Quantifier Raising.
10. TLG: Type Logical Grammar.
11. TL/TR: Type Lifting/Type Raising.

Appendix B

Overview of rules

B.1 NL

Formulas:

(382) $F ::= At \mid (F/F) \mid (F\backslash F) \mid (F \bullet F)$

Structures:

(383) $S ::= F \mid (S, S)$

B.1.1 NL in Natural Deduction

(384) Premise/Hypothesis Introduction:

$$\underline{\begin{array}{c}A\\ \vdots\end{array}} \qquad\qquad \underline{\begin{array}{c}[A]_1\\ \vdots\end{array}}$$

(385) NL: Eliminations rules for $\backslash, /$.

a. Syntax: For all $A, B \in F$ (the set of formulas)

$$\frac{\begin{array}{cc}\vdots & \vdots\\ A & A\backslash B\end{array}}{B}\ \backslash E \qquad\qquad \frac{\begin{array}{cc}\vdots & \vdots\\ B/A & A\end{array}}{B}\ /E$$

b. Semantics:

$$\frac{\begin{array}{cc}\vdots & \vdots\\ \beta & \alpha\end{array}}{(\alpha\beta)}\ \backslash E \qquad\qquad \frac{\begin{array}{cc}\vdots & \vdots\\ \alpha & \beta\end{array}}{(\alpha\beta)}\ /E$$

c. Phonology:

$$\frac{\overset{\vdots}{a} \quad \overset{\vdots}{b}}{(a \cdot b)}\ \backslash E \qquad\qquad \frac{\overset{\vdots}{b} \quad \overset{\vdots}{a}}{(b \cdot a)}\ / E$$

(386) $\backslash, /$ Introduction

a. Syntax: For all $A, B, C \in F$:

$$\frac{\overset{\vdots}{[A]_1} \quad \overset{\vdots}{B}}{\dfrac{C}{A \backslash C}}\ \backslash I_1 \qquad\qquad \frac{\overset{\vdots}{A} \quad \overset{\vdots}{[B]_1}}{\dfrac{C}{C / B}}\ / I_1$$

[N.B] B is not empty for $\backslash I$ and A is not empty for $/ I$.

b. Semantics:

$$\frac{\dfrac{\overset{\vdots}{[\beta]_1} \quad \overset{\vdots}{\alpha}}{\phi}}{\lambda x.\phi[\beta :\rightarrow x]}\ \backslash I_1 \qquad\qquad \frac{\dfrac{\overset{\vdots}{\alpha} \quad \overset{\vdots}{[\beta]_1}}{\phi}}{\lambda x.\phi[\beta :\rightarrow x]}\ / I_1$$

c. Phonology:

$$\frac{\dfrac{\overset{\vdots}{[a]_1} \quad \overset{\vdots}{b}}{(a \cdot b)}}{b}\ \backslash I_1 \qquad\qquad \frac{\dfrac{\overset{\vdots}{a} \quad \overset{\vdots}{[b]_1}}{(a \cdot b)}}{a}\ / I_1$$

(387) $\bullet I$ and $\bullet E$

a. $\bullet I$ Syntax LF PF

$$\frac{\overset{\vdots}{A} \quad \overset{\vdots}{B}}{(A \bullet B)}\ \bullet I \qquad \frac{\overset{\vdots}{\alpha} \quad \overset{\vdots}{\beta}}{(\alpha \bullet \beta)}\ \bullet I \qquad \frac{\overset{\vdots}{a} \quad \overset{\vdots}{b}}{(a \cdot b)}\ \bullet I$$

b. $\bullet E$ Syntax LF PF

$$\frac{\overset{\vdots}{(A \bullet B)}}{A \qquad B}\ \bullet E \qquad \frac{\overset{\vdots}{\gamma}}{\pi_1\gamma \qquad \pi_2\gamma}\ \bullet E \qquad \frac{\overset{\vdots}{(a \cdot b)}}{a \qquad b}\ \bullet E$$

For $\gamma = (\alpha \bullet \beta)$, we have $\pi_1\gamma = \alpha$ and $\pi_2\gamma = \beta$. PF simplified.

B.1.2 NL in Gentzen Sequent

(388) For all $A, B \in F$ and for all $\Gamma, \Delta \in S$:

a. Identity Axiom: $A \vdash A$

b. Cut Axiom

$$\frac{\Gamma \vdash A \quad \Delta[A] \vdash B}{\Delta[\Gamma] \vdash B}\ Cut$$

(389) Logical rules. For all $A, B, C \in F$ and for all $\Gamma, \Delta \in S$:

a.
$$\frac{\Delta[B] \vdash C \quad \Gamma \vdash A}{\Delta[(B/A, \Gamma)] \vdash C}\ /L \qquad \frac{(\Gamma, A) \vdash B}{\Gamma \vdash B/A}\ /R$$

b.
$$\frac{\Gamma \vdash A \quad \Delta[B] \vdash C}{\Delta[(\Gamma, A\backslash B)] \vdash C}\ \backslash L \qquad \frac{(A, \Gamma) \vdash B}{\Gamma \vdash A\backslash B}\ \backslash R$$

c.
$$\frac{\Delta[(A, B)] \vdash C}{\Delta[(A \bullet B)] \vdash C}\ \bullet L \qquad \frac{\Gamma \vdash A \quad \Delta \vdash B}{(\Gamma, \Delta) \vdash (A \bullet B)}\ \bullet R$$

GS rules decorated with meta variables for lambda terms:

(390) a. Identity Axiom $A : u \vdash A : u$

b. Cut: (σ is free for u in ϕ).
$$\frac{\Gamma \vdash A : \sigma \quad \Delta[A : u] \vdash B : \phi}{\Delta[\Gamma] \vdash B : \phi[u :\rightarrow \sigma]}\ Cut$$

(391) Logical rules ($\alpha(\beta)$ is free for X in γ for $/L, \backslash L$. x is fresh for $/R, \backslash R$).

a.
$$\frac{\Delta[B : X] \vdash C : \gamma \quad \Gamma \vdash A : \beta}{\Delta[(B/A : \alpha, \Gamma)] \vdash C : \gamma[X :\rightarrow \alpha(\beta)]}\ /L \qquad \frac{(\Gamma, A : x) \vdash B : \phi}{\Gamma \vdash B/A : \lambda x.\phi}\ /R$$

b.
$$\frac{\Gamma \vdash A : \beta \quad \Delta[B : X] \vdash C : \gamma}{\Delta[\Gamma, A\backslash B : \alpha] \vdash C : \gamma[X :\rightarrow \alpha(\beta)]}\ \backslash L \qquad \frac{(A : x, \Gamma) \vdash B : \phi}{\Gamma \vdash A\backslash B : \lambda x.\phi}\ \backslash R$$

c.
$$\frac{\Gamma[(A : u, B : v)] \vdash C : \gamma}{\Delta[(A \bullet B) : (u \bullet v)] \vdash C : \gamma[(u, v) \rightarrow (u \bullet v)]}\ \bullet L \qquad \frac{\Gamma \vdash A : u \quad \Delta \vdash B : v}{(\Gamma, \Delta) \vdash (A \bullet B) : (u \bullet v)}\ \bullet R$$

B.2 MMTLG

Extended Grammar: $NL_{\diamond_y}$
The set of formulas:

(392) $F_{\diamond x,y} ::= At \mid (F/_x F) \mid (F\backslash_x F) \mid (F \bullet_x F) \mid \diamond_y F \mid \Box^{\downarrow}_y F$
where $x, y \in I$ (= the set of mode indices)

The set of structures:

(393) $S_{\diamond x,y} ::= F \mid (S \circ_x S) \mid (S)^{\diamond}_y$

(394) Logical rules:

a.
$$\frac{\Delta[B] \vdash C \quad \Gamma \vdash A}{\Delta[(B/_x A \circ_x \Gamma)] \vdash C} /L \qquad \frac{(\Gamma \circ_x A) \vdash B}{\Gamma \vdash B/_x A} /R$$

b.
$$\frac{\Gamma \vdash A \quad \Delta[B] \vdash C}{\Delta[(\Gamma \circ_x A\backslash_x B)] \vdash C} \backslash L \qquad \frac{(A \circ_x \Gamma) \vdash B}{\Gamma \vdash A\backslash_x B} \backslash R$$

c.
$$\frac{\Delta[(A \circ_x B)] \vdash C}{\Delta[(A \bullet_x B)] \vdash C} \bullet L \qquad \frac{\Gamma \vdash A \quad \Delta \vdash B}{(\Gamma \circ_x \Delta) \vdash (A \bullet_x B)} \bullet R$$

Meta-variables: $A, B, C \in F$ and $\Gamma, \Delta \in S$

The logical rules for the two unary operators.

(395) a.
$$\frac{\Delta[(A)^{\diamond_y}] \vdash B}{\Delta[\diamond_y A] \vdash B} \diamond_y L \qquad \frac{\Gamma \vdash A}{(\Gamma)^{\diamond_y} \vdash \diamond_y A} \diamond_y R$$

b.
$$\frac{\Delta[A] \vdash B}{\Delta[(\Box^{\downarrow}_y A)^{\diamond_y}] \vdash B} \Box^{\downarrow}_y L \qquad \frac{(\Gamma)^{\diamond_y} \vdash A}{\Gamma \vdash \Box^{\downarrow}_y A} \Box^{\downarrow}_y R$$

B.3 Structural rules in MMTLG

Structural rule for Complex Predicate formation.

(396) Mixed Association with the modes, i, j, for all $A, B, C \in F$ and $X \in S$:

$$\frac{A \circ_i (X \circ_j B) \vdash C}{(A \circ_i X) \circ_j B \vdash C} A_{i,j}$$

Association (AR) and Permutation (PR) for A-bar extraction:

(397) For all $A, B \in F$ and $X, Y \in S$.

a. AR:

$$\frac{X \circ (Y \circ \Diamond A) \vdash B}{(X \circ Y) \circ \Diamond A \vdash B}\ AR$$

b. PR:

$$\frac{(X \circ \Diamond A) \circ Y \vdash B}{(X \circ Y) \circ \Diamond A \vdash B}\ PR$$

Additional rules for Pronominal binding.

(398) Mixed Permutation/Association PR (limited to pronouns/indefinites by the mode requirement). For all $A, B, C \in F$ and $X \in S$:

$$\frac{(A \circ \Diamond_p B) \circ X \vdash C}{(A \circ X) \circ \Diamond_p B \vdash C}\ PR$$

(399) Argument identification Z with regard to p (i.e. pronouns).

a. Syntax:

$$\frac{A \circ (X \circ \Diamond_p A) \vdash B}{A \circ X \vdash B}\ Z$$

b. Semantics:

$$\frac{x \circ (R \circ y) \vdash \phi}{x \circ R \vdash \phi[y :\rightarrow x]}\ Z$$

B.4 Argument slot raising (ASR)

Standard case:

(400) Argument slot raising (ASR), syntax:

a. $(NP\backslash T)/NP$, where T is normally S.

b. $(NP\backslash T)/NP \Rightarrow (NP\backslash T)/((S/NP)\backslash S)$
$\Rightarrow ((S/(NP\backslash S))\backslash T)/((S/NP)\backslash S)$

c. $(NP\backslash T)/NP \Rightarrow ((S/(NP\backslash S))\backslash T)/NP$
$\Rightarrow ((S/(NP\backslash S))\backslash T)/((S/NP)\backslash S)$

(401) ASR, semantics, Surface Scope:

a. Type Shift: (e(et)) ⇒ (((et)t), (e, t)) ⇒ (((et)t), (((et)t), t))

b. Semantics: $P \Rightarrow \lambda Q1.\lambda y.Q1(\lambda x.P(x)(y)) \Rightarrow$
$\lambda Q1.\lambda Q2.Q2(\lambda y.Q1(\lambda x.(P(x)(y))))$
Variable types, $Q1, Q2$: (et)t; x, y: e

(402) ASR, semantics, Inverse Scope:

a. Type Shift: (e(et)) ⇒ (e, ((et)t), t) ⇒ (((et)t), (((et)t), t))

b. Semantics: $P \Rightarrow \lambda x.\lambda Q2.Q2(\lambda y.P(x)(y)) \Rightarrow$
$\lambda Q1.\lambda Q2.Q1(\lambda x.Q2(\lambda y.(P(x)(y))))$

Bibliography

Abusch, D. (1994). The scope of indefinites. *Natural Language Semantics 2*(2), 83–135.

Ajdukiewicz, K. (1935). Die syntactische Konnexität. *Studia Philosophica 1*, 1–27.

Areces, C. and R. Bernardi (2004). Analyzing the core of categorial grammar. *Journal of Logic, Language, and Information 13*, 121–137.

Bach, E. (1980). In defense of passive. *Linguistics and Philosophy 3*, 297–341.

Bach, E. (1981). Discontinuous constituencies in generalized categorial grammars. *Proceedings of the 11th Annual Meeting of the North Eastern Linguistics Society, New York 1*(11), 1–12.

Bach, E. and R. Cooper (1978). The NP-S analysis of relative clauses and compositional semantics. *Linguistics and Philosophy 2*(1), 145–150.

Bale, A. and R. May (2006). Inverse linking. In M. Everaert and H. van Riemsdijk (Eds.), *The Blackwell Companion to Syntax, Vol.2*, Chapter 36, pp. 639–667. Oxford: Blackwell.

Bar-Hillel, Y. (1950). On syntactical categories. *Journal of Symbolic Logic 15*, 1–16.

Bar-Hillel, Y. (1953). A quasi-arithmetical notation for syntactic description. *Language 29*, 47–58.

Barss, A. and H. Lasnik (1986). A note on anaphora and double objects. *Linguistic Inquiry 17*, 347–354.

Barwise, J. and R. Cooper (1981). Generalized quantifiers and natural language. *Linguistics and Philosophy 4*, 159–219.

Beghelli, F. and T. Stowell (1997). Distributivity and negation: the syntax of *each* and *every*. In A. Szabolcsi (Ed.), *Ways of Scope Taking*, pp. 71–107. Dordrecht: Kluwer.

Bernardi, R. (2002). *Reasoning with Polarity in Categorial Type Logic.* Ph. D. thesis, Utrecht University, Utrecht, The Netherlands.

Bernardi, R. and M. Moortgat (2007). Continuation semantics for symmetric categorial grammar. In D. Leivant and R. de Queiros (Eds.), *Proceedings 14th Workshop of Logic, Language, Information and Computation (WoLLIC'07). LNCS 4576.* Springer.

Breheny, R. (2003). Exceptional-scope indefinites and domain restriction. In *Proceedings of Sinn und Bedeutung*, Volume VII, pp. 38–52.

Brody, M. and A. Szabolcsi (2003). Overt scope in hungarian. *Syntax 6*(1), 19–51.

Bruening, B. (2001). QR obeys superiority: Frozen scope and ACD. *Linguistic Inquiry 32*, 233–273.

Büring, D. (1995). The great scope inversion conspiracy. In M. Simons and T. Galloway (Eds.), *Proceedings from Semantics and Linguistics Theory V*, pp. 37–53.

Cann, R., R. Kempson, and L. Marten (2005). *The Dynamics of Language.* Amsterdam: Elsevier.

Carpenter, B. (1997). *Type-Logical Semantics.* Cambridge, Mass: The MIT Press.

Carpenter, B. and G. Morrill (2006). Switch graphs and parsing type logical grammars. In *Proceedings of the International Workshop on Parsing Technology, IWPT05*, Vancouver.

Chomsky, N. (1981). *Lectures on Government and Binding.* Dordrecht: Foris.

Chomsky, N. (1995). *The Minimalist Program.* Cambridge, Mass: The MIT Press.

Chomsky, N. (2001). Derivation by phase. In M. Kenstowicz (Ed.), *A life in language.* Cambridge, Mass: The MIT Press.

Collins, C. (2005). A smuggling approach to the passive in english. *Syntax 8*(2), 81–120.

Cooper, R. (1996). The role of situation in generalized quantifiers. In S. Lappin (Ed.), *The Handbook of Contemporary Semantic Theory*, pp. 65–86. Blackwell.

Cormack, A. (1985). VP anaphora: Variables and scope. In F. Landman and F. Veltman (Eds.), *Variables of Formal Semantics*, pp. 81–102. Dordrecht: Reidel.

Cormack, A. and R. Kempson (1991). On specificity. In *Meaning and truth: essential readings in modern semantics*, pp. 546–581. New York: Paragon House.

Cormack, A. and N. Smith (1997). Checking features and split signs. *UCL Working Paper in Linguistics 9*, 223–252.

Cormack, A. and N. Smith (2001). Don't move. *UCL Working Paper in Linguistics 13*, 215–241.

Cormack, A. and N. Smith (2002). Compositionaliry, copy theory and control. *UCL Working Paper in Linguistics 14*, 355–373.

Cormack, A. and N. Smith (2005). What is coordination? *Lingua*, 395–418.

de Groote, P. (1999). The non-associative Lambek Calculus with product is polynomial time. In N. V. Murray (Ed.), *Automated reasoning with analytic tableaux and related methods*, pp. 128–139. Springer.

Dowty, D. (1988). Type raising, functional composition, and non-constituent conjunction. In R. T. Oehrle, E. Bach, and D. Wheeler (Eds.), *Categorial Grammar and Natural Language Structures*, pp. 153–198. Dordrecht: D. Reidel.

Dowty, D. (1992). 'Variable-free' syntax, variable-binding syntax, the natural language deduction lambek calculus, and the crossover constraint. In *Proceedings of 1992 West Coast Conference of Formal Linguistics.*

Dowty, D. (1996). Non-constituent coordination, wrapping, and multimodal categorial grammars. In M. L. D. Chiara, K. Doets, D. Mundici, and J. van Benthem (Eds.), *Structures and Norms in Science*, pp. 347–368. Dordrecht: Kluwer.

Dowty, D. (2003). The dual analysis of adjuncts and complements in categorial grammar. In E. Lang, C. Maienborn, and C. Fabricius-Hansen (Eds.), *Modifying Adjuncts*, pp. 33–66. Hague: Mouton.

Dŏsen, K. (1988). Sequent-systems and groupoid models I. *Studia Logica*, 353–389.

Dŏsen, K. (1989). Sequent-systems and groupoid models II. *Studia Logica*, 41–65.

Dŏsen, K. (1992). A brief survey of frames for the Lambek Calculus. *Zeitschrift fur Mathematische Logik und Grundlagen der Mathematik 38*, 179–187.

Fodor, J. D. and I. A. Sag (1982). Referential and quantificational indefinites. *Linguistics and Philosophy 54*, 355–398.

Fowler, T. (2007, July). LC graphs for the Lambek Calculus with product. Abstract for Mathematics of Language 10 conference.

Fox, D. (2000). *Economy and semantic interpretation.* Cambridge, Mass: The MIT Press.

Fox, D. (2002a). Antecedent-contained deletion and the copy theory of movement. *Linguistic Inquiry 33*(1), 63–96.

Fox, D. (2002b). Reconstruction, binding theory, and the interpretation of chains. *Linguistic Inquiry 30*(2), 157–196.

Gamut, L. T. F. (1991). *Logic, language, and meaning.* Chicago: University of Chicago Press.

Geurts, B. (2000). Indefinites and choice functions. *Linguistic Inquiry 31*(4), 731–738.

Girard, J. Y. (1987). *Proof Theory and Logical Complexity.* New York: Elsevier.

Girard, J. Y. (1995). Linear logic: its syntax and semantics. In J.-Y. Girard, Y. Lafont, and L. Regnier (Eds.), *Advances in Linear Logic*, pp. 1–42. Cambridge: Cambridge University Press.

Girard, J.-Y., P. Taylor, and Y. Lafont (1990). *Proofs and types.* Cambridge: Cambridge University Press.

Grimshaw, J. (1990). *Argument Structure.* Cambridge, Mass: The MIT Press.

Hall, A. (2006). Free enrichment or hidden indexical. *UCL Working Paper in Linguistics 18*, 71–102.

Heim, I. and A. Kratzer (1998). *Semantics in Gernerative Grammar.* Oxford: Blackwell.

Hendriks, H. (1987). Type change in semantics: the scope of quantification and coordination. In E. Klein and J. van Benthem (Eds.), *Categories, Polymorphism and Unification.*, Chapter 6, pp. 96–119. Centre for Cognitive Science, University of Edinburgh. Institute for Language, Logic and Information. University of Amsterdam.

Hepple, M. (1990). *The grammar and processing of order and dependency: a categorial approach.* Ph. D. thesis, University of Edinburgh, Edinburgh.

Hirschbühler, P. (1982). VP deletion and across the board quantifier scope. In J. Pustejovsky and P. Sells (Eds.), *Proceedings of NELS*, Amherst, Mass. GLSA.

Hornstein, N. (1995). *Logical Form: From GB to Minimalism.* Oxford: Blackwell.

Hornstein, N. (1998). Movement and chains. *Syntax 1*(2), 99–127.

Hornstein, N. (1999a). Minimalism and quantifier raising. In D. Epstein and N. Hornstein (Eds.), *Working Minimalism*, Chapter 3, pp. 45–75. Cambridge, Mass: The MIT Press.

Hornstein, N. (1999b). Movement and control. *Linguistic Inquiry 30*(1), 69–96.

Howard, W. A. (1980). The formulae-as-types notion of construction. In J. R. Hindley and J. P. Seldin (Eds.), *To H.B. Curry: Essays on combinatory logic, Lambda Calculus and Formalism.* Academic Press.

Hudson, R. (1992). So-called "double objects" and grammatical relations. *Language 68*(2), 251–276.

Jacobson, P. (1992a). Antecedent contained deletion in a variable free semantics. In *Proceedings of the Second Conference on Semantics and Linguistic Theory*, pp. 193–213.

Jacobson, P. (1992b). Flexible categorial grammars: Questions and prospects. In R. Levine (Ed.), *Formal grammar: theory and implementation*, Chapter ?, pp. 129–167. New York, Oxford: Oxford University Press.

Jacobson, P. (1999). Towards a variable-free semantics. *Linguistics and Philosophy 22*, 117–185.

Jäger, G. (2003). Resource sharing in type logical grammar. In G.-J. M. Kruijff and R. T. Oehrle (Eds.), *Resource-sensitivity, binding and anaphora*, pp. 97–121. Dordrecht: Kluwer.

Johnson, K. (2000). How far will quantifiers go? In R. Martin, D. Michaels, and J. Uriagereka (Eds.), *Step by Step*, pp. 187–210. Cambridge, Mass: The MIT Press.

Keenan, E. (1980). Passive is phrasal (not sentential or lexical). In T. Hoestra, H. van der Hulst, and M. Moortgat (Eds.), *Lexical Grammar*, pp. 181–213. Dordrecht: Foris Publications.

Kempson, R. and R. Cann (2006). Dynamic syntax and dialogue modelling: preliminaries for a dialogue-driven account of syntactic change.

Kempson, R. and A. Cormack (1981). Ambiguity and quantification. *Linguistics and Philosophy 4*, 259–309.

Kempson, R., W. Meyer-Viol, and D. Gabbay (2001). *Dynamic Syntax*. Oxford: Blackwell.

Kobele, G. M. (2006). *Generating copies: an investigation into structural identity in language and grammar*. Ph. D. thesis, UCLA, Los Angeles.

Kobele, G. M. (2007, July). A formal foundation for a and a-bar movement. Abstract for Mathematics of Language 10 conference.

Kracht, M. (2004). The emergence of syntactic structure. Presentation at the Foundations of Natural Language Grammar, ESSLLI 2005 Workshop, Edinburgh.

Kratzer, A. (1998). Scope or pseudoscope? are there wide scope indefinites? In S. Rothstein (Ed.), *Events in Grammar*, pp. 163–196. Dordrecht: Kluwer.

Kripke, S. (1963). Semantical analysis of modal logic I, normal propositional calculi. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik 9*, 67–96.

Kurtonina, N. (1994). *Frames and Labels; a Modal Analysis of Categorial Inference*. Ph. D. thesis, Utrecht University, Utrecht, The Netherlands.

Kurtonina, N. and M. Moortgat (1997). Structural control. In P. Blackburn and M. de Rijke (Eds.), *Specifying syntactic structures*, pp. 75–113. CSLI Publications.

Kurtonina, N. and M. Moortgat (2007). Relational semantics for the lambek-grishin calculus. Abstract for Mathematics of Language 10 conference.

Lambek, J. (1958). The mathematics of sentence structure. *American Mathematical Monthly 65*, 154–170.

Lambek, J. (1961). On the calculus of syntactic types. In R. Jakobson (Ed.), *Structure of Language and its mathematical aspects: Proceedings of Symposia in Applied Mathematics*, Providence, Rhode Island, pp. 166–178. American Mathematical Society.

Lambek, J. (2004). A computational algebraic approach to english grammar. *Syntax 7*(2), 128–147.

Larson, R. (1988). On the double object construction. *Linguistic Inquiry 19*, 335–391.

Lasnik, H. (1999). Chains of arguments. In D. Epstein and N. Hornstein (Eds.), *Working Minimalism*, Chapter 8, pp. 189–215. Cambridge, Mass: The MIT Press.

Manzini, M. R. (1983). On control and control theory. *Linguistic Inquiry 14*(3), 421–446.

Manzini, M. R. (1992). *Locality, a theory and some of its empirical consequences.* Cambridge, Mass: The MIT Press.

May, R. (1977). *The grammar of quantification.* Ph. D. thesis, MIT, Cambridge, Mass.

May, R. (1985). *Logical Form.* Cambridge, Mass: The MIT Press.

Montague, R. (1970). English as a formal language. In *Formal philosophy: selected papers of Richard Montague*, pp. 108–221. New Haven: Yale University Press.

Montague, R. (1973). The proper treatment of quantificaiton in ordinary english. In J. Hintikka, J. Moravcsik, and P. Suppes (Eds.), *Approaches to Natural Language: Proceedings of the 1970 Standford Workshop of Grammar and Semantics*, pp. 221–242. Dordrecht: Reidel.

Moortgat, M. (1988). *Categorial Investigations: Logical and Linguistic Aspects of the Lambek Calculus.* Dordrecht: Foris Publications.

Moortgat, M. (1991). Generalized quantification and discontinuous type constructors. Technical report, Institute for Language and Speech, University of Utrecht.

Moortgat, M. (1994). Residuation in mixed Lambek systems. In R. Kempson (Ed.), *IGPL Bulletin Special Issue Language and Deduction, 3(2-3).*

Moortgat, M. (1996). Multimodal linguistic inference. *Journal of Logic, Language, and Information 5*, 349–385.

Moortgat, M. (1997). Categorial type logics. In J. van Benthem and A. ter Meulen (Eds.), *Handbook of logic and language*, Chapter 2, pp. 93–178. Cambridge, Mass: The MIT Press.

Moortgat, M. (2001). Structural equation in language learning. In P. de Groote, G. Morrill, and C. Retore (Eds.), *Proceedings LACL 2001*, Chapter 1, pp. 1–16. Heidelberg: Springer.

Moortgat, M. (2007). Symmetries in natural language syntax and semantics: the lambek-grishin calculus. In D. Leivant and R. de Queiros (Eds.), *Proceedings 14th Workshop of Logic, Language, Information and Computation (WoLLIC'07). LNCS 4576*. Springer.

Moortgat, M. and R. T. Oehrle (1994). Adjacency, dependence and order. In *Proceedings of the Ninth Amsterdam Colloquium*, pp. 447–466.

Morrill, G. (1994). *Type Logical Grammar. Categorial Logic of Signs*. Dordrecht: Kluwer.

Morrill, G. (2000a). Incremental processing and accessibility. *Computational Linguistics 26*(3), 319–338.

Morrill, G. (2000b). Type-logical anaphora. Report de Recerca LSI-00-77-R, Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya.

Morrill, G. (2003). On bound anaphora in type logical grammar. In G.-J. M. Kruijff and R. T. Oehrle (Eds.), *Resource-sensitivity, binding and anaphora*, Chapter 6, pp. 159–177. Kluwer Academic Publishers.

Morrill, G. and M. Fadda (2005). Proof nets for basic discontinuity. In C. Fox and S. Lappin (Eds.), *Proceedings of Lambda Calculus, Type Theory and Natural Language, LCTTNL05*, pp. 1–16. Kings College London.

Morrill, G., M. Fadda, and O. Valentin (2007). Nondeterministic discontinuous lambek calculus. In *Proceedings of the seventh international workshop on computational semantics, IWCS7*, Tilburg.

Muller, G. (1995). *A-bar Syntax: a study in movement types*. Mouton de Gruyter.

Oehrle, R. T., E. Bach, and D. Wheeler (Eds.) (1988). *Categorial Grammar and Natural Language Structures*. Dordrecht: Kluwer.

Pentus, M. (2006). Lambek calculus is NP-complete. *Theoretical computer science 357*, 186–201.

Pesetsky, D. (1995). *Zero Syntax, experiencers and cascades*. Cambridge, Mass: The MIT Press.

Postal, P. M. (1998). *Three Investigations of Extraction*. Cambridge, Mass: The MIT Press.

Recanati, F. (2002). Unarticulated constituents. *Linguistics and Philosophy 25*, 299–345.

Reinhart, T. (1997). Quantifier scope: how labour is divided between qr and choice functions. *Linguistics and Philosophy 20*(4), 335–397.

Reuland, E. (2001). Primitives of binding. *Linguistic Inquiry 32*(3), 439–492.

Ross, J. R. (1967). *Constraints on variables in syntax.* Ph. D. thesis, MIT, Cambridge, Mass.

Ruys, E. G. (1992). *The scope of Indefinites.* Ph. D. thesis, Utrecht University, Utrecht.

Saito, M. (2003). A derivational approach to the interpretation of scrambling chains. *Lingua 113*, 481–518.

Schwarzschild, R. (2002). Singleton indefinites. *Journal of Semantics 19*, 289–314.

Stanley, J. (2000). Context and logical form. *Linguistics and Philosophy 23*, 391–434.

Steedman, M. (1996). *Surface Structure and Interpretation.* Cambridge, Mass: The MIT Press.

Steedman, M. (2000a). Information structure and the syntax-phonology interface. *Linguistic Inquiry 31*(4), 649–689.

Steedman, M. (2000b). *The Syntactic Process.* Cambridge, Mass: The MIT Press.

Steedman, M. (2007, September). Surface-compositional scope-alternation without existential quantifiers. Draft 5.2, School of Informatics, University of Edinburgh.

Szabolcsi, A. (1992). Combinatory grammar and projection from the lexicon. In I. A. Sag and A. Szabolcsi (Eds.), *Lexical matters*, Number 24 in CSLI Lecture Notes, Chapter 9, pp. 241–268. CSLI Publications.

Szabolcsi, A. (1997). Strategies for scope taking. In A. Szabolcsi (Ed.), *Ways of Scope Taking*, pp. 109–155. Dordrecht: Kluwer.

Szabolcsi, A. (2007, April). Scope and binding. Draft of article for Maienborn, von Heusinger and Portner (eds.), Semantics: an international handbook of natural language meaning.

Uchida, H. (2005a). Double-object in categorial grammar. In J. Gervain (Ed.), *Proceedings of the Tenth ESSLLI student session*, pp. 377–388.

Uchida, H. (2005b). Indefinites: an extra argument slot analysis. In S. Blaho, L. Vicente, and E. Shoorlemeer (Eds.), *Proceedings of Console XIII. ISSN: 1574-499X*, pp. 377–401.

Uchida, H. (2006). Frozen scope in categorial grammar. In C. Chang, E. Dugarova, I. Theodoropoulou, E. V. Beltrán, and E. Wilford (Eds.), *Proceedings of the fourth University of Cambridge Postgraduate Conference in Language Research*, Cambridge, pp. 237–245. Cambridge Institute of Language Research.

van Benthem, J. (1991). *Language in Action.* Amsterdam, New York: North Holland.

Venema, Y. (1995). Meeting strength in substructural logics. *Stuida Logica 54*, 3–22.

Venema, Y. (1996). Tree models and (labeled) categorial grammar. *Journal of Logic, Language, and Information 5*, 253–277.

Vermaat, W. (2006). *The Logic of Variation; a cross-linguistic account of Wh-question formation.* Ph. D. thesis, Utrecht University, Utrecht, The Netherlands.

Versmissen, K. (1996). *Grammatical Composition: Modes, Models, Modalities. Logical and linguistic aspects of multimodal categorial grammars.* OTS Dissertation series. Utrecht, The Netherlands: LEd.

Wansing, H. (1992). *The Logic of Information Structures.* Ph. D. thesis, Universitat Berlin, Berlin, Germany.

Wansing, H. (1998). *Displaying Modal Logic.* Dordrecht: Kluwer Academic Publishers.

Winter, Y. (1997). Choice functions and the scopal semantics of indefinites. *Linguistics and Philosophy 20*, 399–467.

Winter, Y. (2001). *Flexibility Principles in Boolean Semantics: coordination, plurality and scope in natural language.* Cambridge, Mass: The MIT Press.

Winter, Y. (2004). Functional quantification. *Research on Language and Computation 2*, 331–363.

Winter, Y. (2005). On some problems of (in)definiteness in flexible semantics. *Lingua 115*, 767–786.

Yngve, V. (1960). A model and an hypothesis for language structures. In *Proceedings of the American Philosophical Society*, Volume 104, pp. 444–466.