

NORTHWESTERN UNIVERSITY

Incremental Structure Building and Islands

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

Field of Linguistics

By

Peter Baumann

EVANSTON, ILLINOIS

September 2021

© Copyright by Peter Baumann 2021

All Rights Reserved

ABSTRACT

Human language processing is incremental. In this dissertation, I explore how an incremental perspective can help us clarify our understanding of transformational syntax, which typically proceeds bottom-up. As part of our exploration, I develop an incremental head-driven parsing algorithm for Minimalist Grammars. The two main innovations of this parsing algorithm are the formulation of a left-corner-like transformation for feature sequences of linguistic expressions, and the treatment of syntactic movement by inverting movement chains.

When viewed from the perspective of incremental parsing, some aspects of syntax become simpler: movement turns out to always be ‘covert’: the phonological form of a lexical item is processed when it is encountered in the input, but its meaning may get moved around in the derivation and the resulting logical form. In addition to the standard Minimalist Grammar operations MERGE and MOVE, I also discuss adjunction and coordinate structures. Adjunction is reduced to MERGE with the help of an ADJUNCTIVIZER head, which takes the phrase adjoined to as its complement and projects the same category that it selects. I also discuss the distinction between A-movement and \bar{A} -movement, arguing that \bar{A} -movement is in some sense optional and that the licensing features for \bar{A} -movement are not part of specific lexical items, but are instead introduced by functional heads.

I then sketch a compositional semantics that follows the incremental syntactic derivation: instead of being type driven, semantic composition will be driven by syntax, such that

every syntactic MERGE or MOVE operation corresponds to semantic FUNCTION APPLICATION with the syntactically selecting element uniformly acting as the function and the selected element as its semantic argument. Turning to the semantics of movement, I argue that an \bar{A} -moved expression makes separate meaning contributions at the top and at the base of a movement chain. The meaning contribution at the base of the chain is the core meaning of the moved item, while the meaning discharged at the top of the chain is provided by the head, which also introduces the licensing feature. So \bar{A} -movement always reconstructs, except when the moved element is a quantificational determiner phrase, which can take scope via Quantifier Raising licensed by a scope feature.

Finally, I use the proposed incremental syntax and semantics to argue that syntactic islands should not be viewed as a constraints within syntax. Instead different island effects may have different non-syntactic explanations. I first argue that specifiers are islands because of incremental structure building. Second, I offer an explanation of weak islands: I argue that weak island effects are identical to (focus) intervention effects, and that movement chains introduced by a Q head cannot span across (focus) interveners. But movement licensed by the scope feature of determiner phrases is not affected by intervention effects and can thus span across weak island interveners.

ACKNOWLEDGEMENTS

I would like to express my gratitude to everyone who has known me during my time at Northwestern and contributed in some form or other to the success of my graduate education and the development of this dissertation.

First and foremost I would like to thank two people who have been a constant source of encouragement from my first day at Northwestern, and without whose support I would have never come this far: Masaya Yoshida and Brady Clark. Masaya has been my advisor from day one, and he has always been incredibly supportive and patient with me, my ideas, and the many turns my graduate career has taken. I am honored to have him as the chair of my dissertation committee. Brady was the first person from Northwestern to reach out to me, and he has been a constant source of support and sound advice ever since. My regular conversations with him have kept me sane, and I have enjoyed every one of them.

I would like to thank my committee members Alexis Wellwood, Stefan Kaufmann, Greg Kobele, and Tim Hunter for their time and support, and for constructive questions and comments. Each one of them has helped shape this dissertation in more ways than they can imagine.

I would like to express my gratitude to Janet Pierrehumbert for her support while I was working with her on the IARPA Babel project, and to all members of team Swordfish. I would also like to acknowledge all other faculty and staff in the Linguistics department, especially Irene Sakk and Talant Abdykairov. And I would like to say a special thank you to Julia Moore, Erin Leddon, and the English Language Program.

I would like to acknowledge my amazing fellow grad students, colleagues, and friends. It is hard to overstate the impact they have had on me personally, and on the success and enjoyment of my time at Northwestern.

First, I would like to acknowledge my entire cohort of eight, and say a special thank you to Svetlin Dimov and Alex Schumacher, who have become dear friends. I would also like to thank all members of Masaya's lab, in particular David Potter, Mike Frazier, Kat Hall, Nayoun Kim, Ethan Myers, and Hyosik Kim, and all other graduate students in the Linguistics department, in particular Daniel Tucker, Tommy Denby, Clayton Bryson, Nicole Mirea, Dan Turner, and Anatha Latshaw.

Outside of Linguistics, I would like to say a special thank you to Ekke Beck, Dan Skibra, Jeremy Watt, Ahmet Colak, and Diego Soto. They each have made my time at Northwestern (and beyond) enjoyable and intellectually stimulating. And thank you to my wife Roz Kwan for being with me during these stressful times, und meinen Eltern, für alles. Vielen Dank.

To whom it may concern

TABLE OF CONTENTS

	Page
1 Introduction	14
1.1 Incremental Merge	17
1.2 Incremental Move	19
1.3 Outline	21
2 Incremental Syntax: An Incremental Parser for Minimalist Grammars	24
2.1 Minimalist Grammars and Parsing	26
2.1.1 Minimalist Grammars	26
2.1.2 Linearizing Complements and Specifiers	36
2.1.3 Parsing (Minimalist) Grammars	45
2.2 Incrementally Parsing Minimalist Grammars	58
2.2.1 Overview of the Incremental Parser	59
2.2.2 Incremental Structure Building: MERGE	61
2.2.3 Inverting the Chain: MOVE	77
2.2.4 Moving Incomplete Items: Remnant Movement	102
2.3 Adjunction	107
2.3.1 Adjuncts and Modifiers	108
2.3.2 Adjuncts in Minimalist Grammar	110
2.3.3 Incremental Adjuncts	118
2.4 Coordination	127

2.5	What Moves Where When How	131
2.5.1	A vs \bar{A} Movement	133
2.5.2	Introducing \bar{A} -Licensing Features via Adjunction	140
2.5.3	Covert Movement	150
2.5.4	Multiple <i>wh</i>	157
2.6	Taking Stock	158
2.6.1	Formal Definitions of Incremental MOVE and MERGE	160
2.6.2	Internal vs. External MERGE	163
2.6.3	Characteristics of the Parser and Previous Incremental MG Parsers .	164
2.6.4	Toward a Processing Model: Active Gap Filling and the Filled-Gap Effect	167
2.6.5	Parsing Head-final Languages	169
3	Incremental Morphology: Inverting Head Movement and the Lexicon	173
3.1	Head Movement and Inflectional Morphology	174
3.1.1	Inverse Head Movement	176
3.1.2	(Accusative) Case on DPs	180
3.1.3	A New Perspective on Alternations	183
3.2	Lexicon in Bottom-up and Left-to-right Syntax	185
3.2.1	Movement Paradoxes	186
4	Incremental Semantics: Composition and Scope	189
4.1	Compositional Semantics	191
4.1.1	Why Events?	193
4.1.2	Formal Assumptions and Notation	194
4.1.3	Compositional Event Semantics	196
4.1.4	Aligning Syntax and Semantics	199

	10
4.2	Incremental Compositional Event Semantics 202
4.3	Modification/Adjunction 208
4.3.1	Adjunctivizer + Function Application = Predicate Modification . . . 208
4.3.2	Thematic Roles as Semantic Modifiers 213
4.3.3	Thematic Roles and Scope 215
4.3.4	Full Thematic Separation and Exotransitives 217
4.3.5	Interim Summary 218
4.4	Semantics of Movement 219
4.4.1	Semantics of MOVE 221
4.4.2	Generalizing Scope Beyond Generalized Quantifiers 222
4.4.3	A-Movement: Scope and Case 226
4.4.4	Semantics in \bar{A} -Chains 229
4.4.5	Binding and Weak Cross-Over 235
4.4.6	Scope and c-Command 236
4.4.7	Inverse Movement Chains and λ -Abstraction 236
4.5	Incremental Compositional Event Semantics with Movement 238
4.6	Previous Work 247
5	Incremental Islands and Clausal Peninsulas 250
5.1	Specifier Islands 253
5.1.1	The Specifier-Island Constraint 253
5.1.2	Deriving the SPIC from Incremental Parsing 254
5.1.3	Subject Islands 256
5.1.4	A Different Specifier: Indirect Objects 263
5.1.5	Violations of Specifier Island Constraint 265
5.1.6	Interim Summary 267

5.2	Clauses and Peninsulas	267
5.2.1	Weak Island Effects as Intervention Effects	268
5.2.2	Definite and Open Clauses	272
5.2.3	Definite Clauses as Peninsulas	275
5.2.4	Open Clauses	280
5.2.5	Anti-Pronominal Contexts, Movement Types, and Weak Islands	285
5.3	Adjuncts and Coordinate Structures	287
5.3.1	Coordinate Structure Constraint	287
5.4	Summary and Other Factors Influencing Islands	292
6	Implications and Conclusions	295
6.1	Putting it all together in Phases	295
6.2	(Copy) Theory of Movement	297
6.3	Parsing and Grammar	300
6.4	Related Work	302
	References	308

LIST OF FIGURES AND ILLUSTRATIONS

		Page
1	Minimalist Grammar derivation tree for <i>Which book do the critics like?</i>	39
2	Derivation Tree for <i>Which book do the critics like</i> . Internal nodes contain the expression representing their sub-tree.	43
3	Structure building operations for Minimalist Grammars	45
4	Bottom-up parsing derivation of <i>The critics like the book</i>	55
5	Bottom-up parsing derivation with movement of <i>Which book do the critics like?</i>	57
6	Incremental parsing derivation of <i>The critics like the book</i>	75
7	Incremental parsing derivation of <i>The authors know that the critics like the book</i> .	76
8	A bottom-up movement chain	80
9	An inverse movement chain	80
10	Incremental derivation with movement of <i>Which book do the critics like?</i>	89
11	Incremental derivation with moved specifier: <i>Which critics like the book?</i>	97
12	Incremental derivation of <i>Which critics will seem to like the book</i>	101
13	Incremental derivation with remnant movement: <i>Admired, John was</i>	106
14	Incremental left adjunction: <i>John was quickly leaving London</i>	119
15	Incremental remnant movement with introduction of fronting feature via an empty head: <i>Admired, John was</i>	143
16	Incremental remnant movement with adjunct: <i>Deeply admired, John was</i>	147
17	Incremental structure building operations for Minimalist Grammars	161

18	Incremental structure building operations for Minimalist Grammars with full linguistic signs consisting of PF and LF components	162
19	Classification of incremental Minimalist Grammar rules into internal vs. external MERGE operations	164
20	Incremental derivation with Filled Gap Effect: <i>Which author does Anna like the brother of?</i>	168
21	Derivation with inverse head movement: <i>Do critics like books?</i>	180
22	Derivation tree for <i>rain heavily</i> , using an adjunctivizer head	211
23	(Impossible) incremental derivation with subject island violation: <i>Which actor do fans of think so?</i>	261
24	(Impossible) incremental derivation with subject island violation and head movement: <i>Which actor do fans of think so?</i>	262
25	(Impossible) incremental derivation with indirect object: <i>Who did Foucault give fans of beer?</i>	264
26	(Incremental derivation with extraction from a low subject: <i>Who are there pictures of (on the wall)?</i>	266

CHAPTER 1

INTRODUCTION

Human language processing is incremental. When reading or hearing a sentence like (1), human comprehenders do not wait until the end of the sentence to start building a syntactic and semantic representation.

- (1) Critics from several international newspapers attended the premiere of the musical Hamilton at the Public Theater in Manhattan.

Instead, human comprehenders can interpret (and react to) a sentence while reading or hearing it: for example, after reading the subject and the verb of (1), we know that there was an event of attending and that critics from several international newspapers were involved in said event as the party doing the attending (i.e. the agent or experiencer). We know this even if we don't hear or read the rest of the sentence, or only a part of it. Similarly, a human speaker may start uttering (1) while still trying to remember (or looking up) where the premiere was or where the theater is located.

Psycholinguistic evidence confirms the incremental nature of human language processing: sentence processing experiments show that human comprehender use contextual information and world knowledge to disambiguate a sentence while hearing it (e.g. TRUESWELL, TANENHAUS & GARNSEY 1994, TANENHAUS, SPIVEY-KNOWLTON, EBERHARD & SEDIVY 1995, ALTMANN & KAMIDE 1999, 2007), and form expectations about how the

rest of the sentence (most) plausibly continues and about upcoming words and syntactic structures in the input (e.g. KONIECZNY 2000, KONIECZNY & DÖRING 2003, VASISHTH & LEWIS 2006, YOSHIDA, DICKEY & STURT 2013, GRISSOM II, ORITA & BOYD-GRABER 2016). And when asked to do so, participants in experiments can detect syntactic and semantic anomalies while hearing or reading a sentence. Experiments and models of language production show that human speakers do not plan an entire sentence before they start speaking (e.g. PAUL 1880, LEVELT 1989, BOCK 1995, SCHRIEFERS, TERUEL & MEINSHAUSEN 1998, BROWN-SCHMIDT & TANENHAUS 2006, MOMMA, SLEVC & PHILLIPS 2016).

In contrast, modern theoretical syntax conceptualizes syntactic structure building as a bottom-up process: when analyzing a simple sentence like (2), we first combine the verb *chased* with its direct object *a big rat* to form a verb phrase (VP), and then combine the resulting VP with the subject *the cat*.

(2) The cat chased a big rat.

This bottom-up perspective of structure building has the desirable property that it preserves constituent structure (at least in the absence of syntactic movement): if an expression *X* becomes a constituent of an expression *Y* at any point in the derivation, *X* will be a constituent of *Y* in the final structure.

Obviously, bottom-up structure building is not compatible with incremental processing: applying the same logic to (1), a comprehender would have to wait until the end of the sentence, before they can start assembling the locative prepositional phrase (PP) *at the Public Theater in Manhattan* and the direct object *the premiere of the musical Hamilton* and combine them with the verb to form a VP.

Similarly, at least since MONTAGUE (1973), semantic composition is also conceptualized as a bottom-up process. Even worse, truth conditions and other inferences constituting

semantic meaning can only be calculated once the entire sentence has been processed. But the processing evidence cited above and everyday experience show that human beings are able to understand sentences incrementally and draw logical consequences from incomplete sentences: when reading the incomplete sentence in (3) we can infer that there is a critic and a state of regret whose experiencer is the critic, and there is an event of insulting whose agent is the critic, and the insulting event is the content of the state of regret.

(3) The critic regrets that he insulted ...

(4) The critic thinks that he insulted ...

But when reading the minimally different incomplete sentence in (4), we cannot infer that there is in fact an event of insulting, only that the critic thinks that there is one. So we draw different logical conclusions from (4) compared to (3), even though both sentences are incomplete. And if syntactic structure is to be interpreted directly or at least constrain semantic composition, the examples in (4) and (3) cannot be interpreted at all, because they do not form a syntactic constituent.

However, the fragments in (4) and (3) are almost constituents: all they are missing is a direct object. So we may call them incomplete constituents. But the missing direct objects in (4) and (3) may themselves be incomplete constituents as in (5).

(5) a. The critic regrets that he insulted the director of ...

b. The critic thinks that he insulted the ...

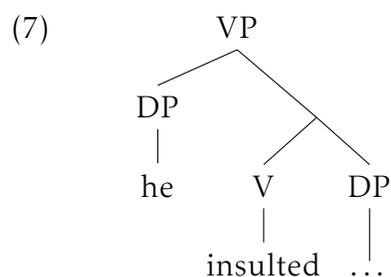
So syntax and semantics to be compatible with incremental processing we minimally require a way to assign syntactic structure to incomplete constituents and a way to

syntactically (and semantically) compose incomplete constituents. In this dissertation I explore one way of implementing this idea rather literally.

1.1 Incremental Merge

An incomplete constituent can be assigned a syntactic structure if it is missing only its complement: using standard notation the fragment in (6) can be represented as the syntactic structure in (7).

(6) He insulted



Formulated in terms of phrase structure rules, we say that the rule (8) can apply once the two elements in bold have been found in the input.

(8) $VP \rightarrow \mathbf{DP} \mathbf{V} DP$

Letting a phrase structure rule ‘fire’ before all its constituent parts are available is known as left-corner parsing, and the parsing procedure to be developed in the next chapter is a variant of left-corner parsing, in which the head (and everything to its left) act as the left corner. But instead of working with phrase structure rules like (8), we assume the framework of Minimalist Grammars (STABLER 1997), which is a formalization of the basic principles of syntactic Minimalism (CHOMSKY 1995).

Minimalist Grammars (MGs) are strongly lexicalized and encode syntactic selection information on lexical items in the form of features which trigger the general purpose structure building operation MERGE: the syntactic information contained in the lexicalized version (9) of the general phrase structure rule (8) is encoded in the feature sequence of the lexical entry (10) of the verb *insulted*, which selects two DPs to form a VP.

(9) VP \rightarrow NP insulted NP

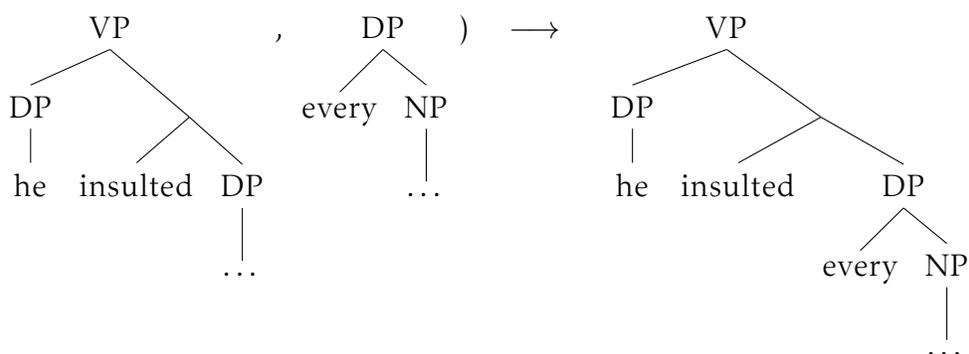
(10) =D =D V :: insulted

Selection features trigger an application of MERGE, and in bottom-up structure building, an expression like (10) has to satisfy (or check) all its selection features, before it can itself be selected. For our (rather conservative) variant of left-corner parsing, we allow MERGE to apply if the selected expression is incomplete and still has one active selection feature: after (10) has merged with *he*, the resulting expression in (11) can combine with a determiner like *every* to form the expression in (12).

(11) =D V :: he insulted

(12) =N V :: he insulted every

Written in terms of syntactic (or derived) trees, MERGE can apply as sketched in (13).

(13) MERGE(

However, the purpose of syntax is not to assign a hierarchical structure to a sentence, instead syntax is meant to regulate the formation of complex expressions from smaller expressions: a linguistic expression is (or contains) a linguistic sign in the sense of DE SAUSSURE (1916), consisting of a form (or pronunciation) and a meaning, and the structure building operations act on both pronunciation and meaning. The effect of MERGE on pronunciation is concatenation: the pronunciations of the two signs being merged are concatenated and the result is the pronunciation of the merged expression.

On the meaning side, MERGE corresponds to FUNCTION APPLICATION: every MERGE operation in the syntax triggers exactly one function application in the semantics, and the syntactic argument (selectee) of MERGE is always the semantic argument of function application (or function composition, in the case of an incomplete selectee).

The semantics is fully determined by syntax, or syntax-driven, which contrasts with the more traditional notion of type-driven semantics, where syntax determines which elements combine when, but then it is left to semantic type considerations to determine which element is the argument and which the function.

1.2 Incremental Move

Minimalist Grammars are transformational grammars and so in addition to MERGE they also define the structure building operation MOVE for syntactic movement. The basic idea to make syntactic movement compatible with incremental processing is to invert movement chains.

In traditional movement chains, an element is inserted in its base position and then its pronunciation moves to the top of the chains, while its meaning stays in situ. In inverted movement chains, an element is inserted into its moved position at the top of the chain and then its meaning moves to the base position, while the pronunciation stays in situ.

The core meaning of the moved expression (i.e. the meaning it has when not moved) is always discharged in its base position, otherwise there is no basis for assuming movement in the first place.

A moved expression may also make a meaning contribution in its moved position(s), but this meaning may be different from its core meaning: when forming a constituent question, the moved meaning of the *wh*-constituent has something to do with question semantics. So we claim that only the core meaning is contributed by the moved expression itself, while the moved meaning is introduced by a functional head, which also introduces the feature licensing movement. So when a verb like *admired* is fronted as in (14), it does not carry a fronting feature, which needs to be checked by the complementizer (spelled out as a comma) to license VP fronting.

(14) Admired, John was.

Instead the fronting feature is introduced by a functional head, which combines with the lexical item *admired*, before it combines with the complementizer via MOVE. Like this all licensing features for \bar{A} -movement are not a lexical property of lexical items (like *wh*-expressions), but are introduced by functional heads. For *wh*-expressions, the functional head introducing the $-\text{wh}$ feature is the Q-particle of CABLE (2007, 2010).

Combining the (moved) meaning of a moved expression with the meaning of the head, which licenses the movement step (e.g. a complementizer head), is done via function application: so the semantic effect of the operation MOVE is the same as that of MERGE: function application. This deviates from the traditional conception of the semantics of movement (HEIM & KRATZER 1998), where the licensing head makes no semantic contribution to movement.

After the licensing head and the moved expression have merged, the resulting expression takes the entire future derivation as its semantic argument: given an appropriate semantics of the licensing head, the moved expression then takes scope over the rest of the derivation. So syntactic movement is always potentially scope taking. Incidentally, this type of scope taking implies a c-command relation between the scope-taking element and its scope.

1.3 Outline

In Chapter 2, I sketch an incremental head-driven parsing algorithm for Minimalist Grammars (MGs, STABLER 1997), one recent formalization of CHOMSKY'S (1995) Minimalist Program. The two main innovations of this parsing algorithm are the formulation of a left-corner-like transformation for feature sequences, and the treatment of syntactic movement by quite literally inverting the movement chain.

When viewed from the perspective of incremental parsing, some aspects of syntax become simpler: movement turns out to always be 'covert': the phonological form (PF) of a lexical item is processed when it is encountered in the input, but its meaning may get moved around in the derivation and the resulting logical form (LF). In addition to the standard MG operations MERGE and MOVE, I also discuss adjunction and coordinate structures. Adjunction is reduced to MERGE with the help of an ADJUNCTIVIZER head, which takes the phrase adjoined to as its complement and projects the same category that it selects. The actual adjunct is the second element selected by the adjunctivizer. One critical distinction will be that between adjuncts that are linearized to the left of the modified phrase and those that occur to the right: left adjuncts will be shown to behave almost like functional projections along the clausal spine, while right adjuncts will be shown to pose a significant challenge for incremental structure building.

We also discuss the distinction between A-movement and \bar{A} -movement, arguing that \bar{A} -movement is in some sense optional and that licensing features for \bar{A} -movement like $-\bar{w}h$ are not part of specific lexical items like *wh*-words, but are instead introduced by a functional head. In the case of $-\bar{w}h$ features, the functional head is identified as the Q-head of CABLE (2007).

In Chapter 3, we will see that head movement can be a special case of (inverse) phrasal movement. Importantly, this simplified understanding does not require any changes to the grammar, the feature checking mechanism is standard MG. But our understanding of lexical items will need to be adjusted: lexical items can now be complex expressions and contain moving sub-expressions (or movers). This is not possible in bottom-up MGs, where only derived expressions can contain movers.

In Chapter 4, I sketch a compositional semantics that follows the incremental syntactic derivation: instead of being type driven, semantic composition will be driven by syntax, such that every syntactic MERGE or MOVE operation corresponds to semantic FUNCTION APPLICATION with the syntactically selecting element uniformly acting as the function and the selected element as its semantic argument. The semantics to be developed is a (neo-)Davidsonian event semantics, in which the external argument is severed from the verb, but the internal argument is projected by the verb. This differential treatment of internal and external argument is rooted in the SOV word order of English and the fact that all thematic roles are semantic modifiers.

Next we turn to the semantics of movement, and we argue that an \bar{A} -moved expression makes separate meaning contributions at the top and at the base of a movement chain. The meaning contribution at the base of the chain is the core meaning of the moved item, while the meaning discharged at the top of the chain is provided by the head, which also introduces the licensing feature. So \bar{A} -movement always reconstructs, except when the moved element is a quantificational DP: DPs can take scope via Quantifier Raising licensed

by a scope feature $-s$. And if a DP is fronted via another licensing feature (like $-wh$), it has two licensing features and can take scope at an intermediate position.

In Chapter 5 we finally reap the benefits of all this work: I will argue that syntactic islands (Ross 1967) should not be viewed as constraints within syntax. Instead different island effects may have different non-syntactic explanations. I first argue that specifiers are islands because of incremental structure building. Second, I offer an explanation of weak islands: I argue that weak island effects are identical to (focus) intervention effects, and that movement chains introduced by a Q head cannot span across (focus) interveners. But movement licensed by the scope feature of determiner phrases is not affected by intervention effects and can thus span across weak island interveners. I then propose that there are two types of embedded clauses: definite clauses and open clauses. Definite clauses are analyzed as having a definite determiner on top of their CP projection, and the definite determiner acts as an intervener for movement out complement. Definite clauses are thus weak islands, while open clauses shown no island effects.

Removing islands from the domain of syntax should be a welcome move, since islands are notoriously hard to formulate in a derivational theory of syntax, and there are more theories of islands than there are theories of verb meanings.

CHAPTER 2

INCREMENTAL SYNTAX: AN INCREMENTAL PARSER FOR MINIMALIST GRAMMARS

Human language processing is incremental. In this chapter we explore if and how an incremental perspective can help us clarify our understanding of transformational syntax, which typically proceeds bottom-up. As part of our exploration we develop an incremental parsing algorithm for Minimalist Grammars (MGs, STABLER 1997), one recent formalization of CHOMSKY'S (1995) Minimalist Program. The proposed parsing algorithm can be characterized as head-driven incremental parsing: once a head is seen, all dependents that preceded it can be merged together with the head, while all dependents that follow the head can be merged later. To achieve this we posit a left-corner-like transformation for feature sequences of MG expressions, and a mechanism to combine incomplete MG expressions via MERGE. Syntactic movement is made incremental by quite literally inverting the movement chain: the pronunciation of a lexical item is processed when it is encountered in the input (where else?), but its meaning may get moved around in the derivation and the resulting logical form (LF).

In addition to the standard MG operations, I also discuss adjunction and coordinate structures and how they can be made to fit into the incremental picture. Adjunction is reduced to MERGE with the help of an ADJUNCTIVIZER head, which takes the phrase adjoined to as its complement and projects the same category that it selects. The actual

adjunct is the second element selected by the adjunctivizer. One critical distinction will be that between adjuncts that are linearized to the left of the modified phrase and those that occur to the right: left adjuncts will be shown to behave almost like functional projections along the clausal spine, while right adjuncts will be shown to pose a significant challenge for incremental structure building.

Finally, we also offer a characterization of the distinction between A-movement and \bar{A} -movement, arguing that \bar{A} -movement is in some sense optional and that licensing features for \bar{A} -movement like $-\bar{w}h$ are not part of specific lexical items like *wh*-words, but are instead introduced by a functional head. In the case of $-\bar{w}h$ features, the functional head is identified as the Q-head of CABLE (2007).

Recently, there have been three related proposals of incremental left-corner parsers for Minimalist Grammars by HUNTER (2019), STANOJEVIĆ & STABLER (2018) and HUNTER, STANOJEVIĆ & STABLER (2019). In those parsers the left-corner transformation is applied to the basic operations MERGE and MOVE, making whichever item coming first in the string the left-corner of the rule to be applied. I am instead proposing to apply a left-to-right transformation to the feature sequences characterizing the lexical items in Minimalist Grammars, thus bringing the features into an order consistent with the left-to-right nature of processing. Incremental parsing can then proceed by feature checking, almost like in bottom-up structure building. Unlike some of this previous research, the focus here is linguistic. So we will not prove soundness or completeness of the incremental parser, instead we discuss the linguistic consequences of an incremental perspective. In this respect, the current work is similar to PHILLIPS (1996), but without committing to the claim that the parser is the grammar.

On a more conceptual level, the parsing procedure developed here does not follow the standard model of generative syntax, according to which a syntactic derivation is sent off to the interfaces only when it is completed. Instead we adopt the model of interpreted

languages developed in KRACHT 2011: we view any linguistic expression as a sign in the sense of DE SAUSSURE (1916), consisting of a form or pronunciation (the signifier) and a meaning (the signified). Not only simple words are signs in this sense, but also full sentences, And the principle of compositionality tells us how to assemble complex signs from simpler signs: the meaning of a complex expression is a function of the meanings of its parts and the mode of composition. So we assume that all expressions, simple and complex, have a form and a meaning. In keeping with standard terminology, we sometimes call the meaning of an expression its Logical Form (LF), and correspondingly the form or pronunciation of an expression its Phonological Form (PF).

2.1 Minimalist Grammars and Parsing

In this section, we introduce Minimalist Grammars, the grammar framework used throughout this dissertation, and discuss parsing, in particular bottom-up parsing, and how a Minimalist Grammar derivation can be written as a bottom-up parsing derivation.

2.1.1 Minimalist Grammars

Minimalist Grammars (MGs, STABLER 1997) are a precise and rather well-understood grammar formalism inspired by the central assumptions of transformational grammars in the tradition of the Minimalist Program (CHOMSKY 1995). MGs have been used as a framework to implement and evaluate specific minimalist syntactic analyses, and to gain a deeper understanding of the properties of syntactic theory in general. Like Combinatory Categorical Grammar (STEEDMAN 1985) and Tree-Adjoining Grammar (JOSHI, LEVY & TAKAHASHI 1975), Minimalist Grammars are mildly context-sensitive¹, making them an

¹ Though CCG may be strictly less expressive than MGs (STEEDMAN & BALDRIDGE 2011).

appropriately restrictive model for human language grammars, which are assumed to require more expressive power than context-free grammars, but not much more.

As a strongly lexicalized grammar formalism, MGs have a limited set of general structure building operations, with all construction and language specific information encoded in the expressions, which the structure-building operations act on to derive new expressions. The basic building blocks of MGs are lexical items, and any particular Minimalist Grammar is the set of all expressions, which can be derived from its lexical items through repeated application of the structure building operations.

The structure building operations used in most MGs² are MERGE and MOVE. They instantiate the two structure building operations of the same name commonly assumed in Minimalist analyses, but they are defined in a formally precise manner, see Section 2.1.2.2. As is common in the Minimalist tradition, all applications of MERGE and MOVE are licensed by syntactic features on lexical items (and derived expressions), which means that specifying a lexicon is all that is needed to define a particular grammar. And for syntactic structure building, the only relevant part of a lexical item is its syntactic features, which are structured as an ordered list of individual features, with only the first (i.e. leftmost) element accessible to license structure building operations.

A few examples will help to introduce the interaction of feature specifications and structure building.

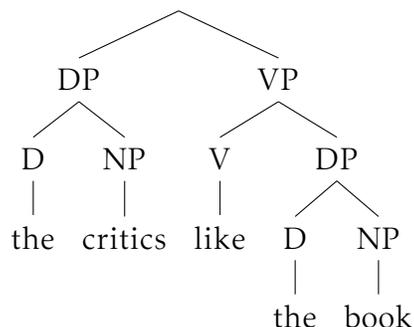
2.1.1.1 Feature-driven Structure Building by Example

In traditional transformational grammar (and many other grammar formalism) a simple declarative sentence like (15) is analyzed as having the constituent structure in (16).

(15) The critics like the book.

² For an overview of variations between different MGs proposed in the literature, see STABLER 2011.

(16)



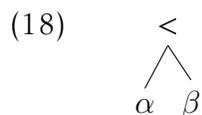
The tree structure in (16) can be derived in a bottom-up manner. First, we combine the determiner *the* and the noun (phrase, NP) *book* to form a determiner phrase (DP). Then we combine the verb *like* and the DP *the book* to form a verb phrase (VP).

How do we know that the result of combining *like* and *the book* is a VP and not, say, a DP? The resulting phrase is labeled VP and not DP, because the head of that phrase is the verb *like* and the head determines the relevant properties of the phrase it projects (MUYSKENS 1982). A second property of the head *like* is that it selects a DP as its complement. Similarly, it has become standard to assume that determiners project DPs (ABNEY 1987) and select NPs as their complements. But how do we know which words project? Or which words select what type of objects?

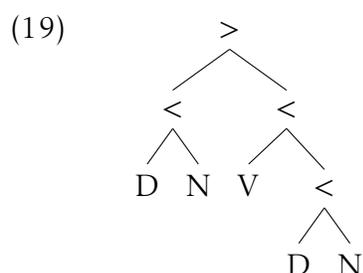
The properties projected by a head are sometimes called the ‘label’ of a phrase, and a simple way to model projection is to postulate that the projected phrase bear the same label as its head. This leads to the bare phrase structure representation in (17), where two items α and β combine, with α being the head. And so the resulting phrase is also labeled α . Instead of repeating the label of the head we can represent the same phrase as the tree in (18), with the symbol \langle (or \rangle) pointing towards the head.

(17)





The syntactic operation which combines two items in the way sketched in (18) is called MERGE³. The tree in (19) illustrates the bare phrase structure version of (15), with terminal nodes replaced by their labels.



As introduced above, the label of a phrase determines in what ways a phrase can combine with other elements, but it does not determine how the phrase itself is assembled. So the label of the phrase only encodes category information, while the way in which a phrase is built is determined by subcategorization (or selection) information, which must be encoded on the head of the phrase. This means labels are just one type of syntactic features.

In MGs, all syntactic behavior is encoded in features on syntactic expressions and explicitly spelling out those syntactic features and the corresponding lexical items fully determines a specific syntactic analysis and grammar. We can now specify the lexicon

³ In orthodox Minimalism, MERGE of two items α and β is claimed to yield the set $\{\alpha, \beta\}$, which is then combined with the label α to give the representation

- (i) $\{\alpha, \{\alpha, \beta\}\}$.

It is then argued that the representation in (i) is the most minimalist since it only involves the most basic mathematical operation, set formation. In my opinion, this insistence on formal simplicity is mostly meaningless, since (i) is just a formal notation for a concept. And that concept can be represented in many formally equivalent ways, one of them being a binary tree.

of a Minimalist Grammar to derive our original example in (15). The lexicon consists of four lexical items, corresponding to the four unique words in the sentence. The simplest lexical items are the nouns *critics* and *book*, whose only feature is their syntactic category feature **N** marking them as a nouns. Unlike the two nouns, the determiner *the* and the verb *like* both select other items. The determiner selects a noun, so in addition to its category feature **D**, it also carries a selector feature for a noun, which for the moment, we write **=N**, as is common in MGs. Its full feature specification is thus **=N D**. A transitive verb like *like* is traditionally analyzed as selecting two DPs as arguments. For the moment, we entertain this analysis and specify the feature sequence of *like* as **=D =D V**.

A lexical item consists of syntactic features and a linguistic sign in the sense of DE SAUSSURE 1916, representing the form and meaning of the lexical item (see also KRACHT 2011). In this chapter we follow established syntactic tradition and reduce the sign to its orthographic spelling. With this the lexical items discussed so far are listed in (20), there we separate syntactic features from the linguistic sign by ::.

- (20) **N** :: critics **N** :: book
 =N D :: the **=D =D V** :: like

Note how we use the usual labels for category features⁴, and how all the features are written in sequence. The order of the sequence is meaningful and represents something

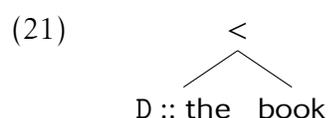
⁴ In the literature on MGs, it is common to denote category and selector features by lower-case letters (but cf. STABLER 2011) and to write the features after the orthographic representation of the word, as in

- (i) like :: =d =d v.

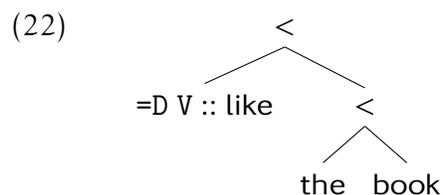
These are just notational variants of an otherwise fairly ‘standard’ introduction of MGs. My choice to invert the order of syntactic features and word form represents the order of importance of syntactic features and word form in syntactic structure building. Using upper-case category features is more in line with traditional syntactic labels (and it allows to distinguish **V** and **v** in the usual manner without making **V** stand out).

like a function signature: the lexical item $=N D :: the$ takes a noun as ‘input’ and ‘returns’ a DP, and the lexical item $=D =D V :: like$ takes a DP and returns a phrase of type $=D V$.

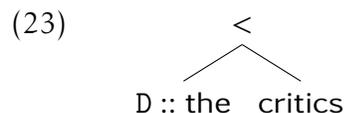
With the lexical items in (20), we can derive the tree in (19), by applying binary MERGE whenever a selector feature finds a matching category feature and deleting both features⁵ after they are ‘checked’. First we merge the lexical items $=N D :: the$ and $N :: book$ and delete the matching features N and $=N$ as in (21).



We then merge the lexical item $=D =D V :: like$ with the expression in (21) to obtain (22).

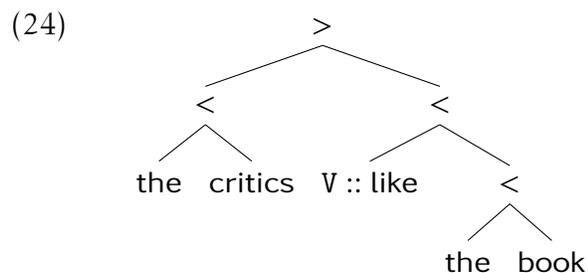


Next, in (23), we merge the lexical items $=N D :: the$ and $N :: critics$.



Notice how this step does not rely on the previous ones, one may say it is performed in a separate workspace. Finally, we merge (22) and (23) to obtain (24).

⁵ In Section 2.5.4 the feature calculus will be revised to allow for category features to be ‘persistent’ (STABLER 2006, 2011). For the moment, we are treating all syntactic features as ‘uninterpretable’ features.



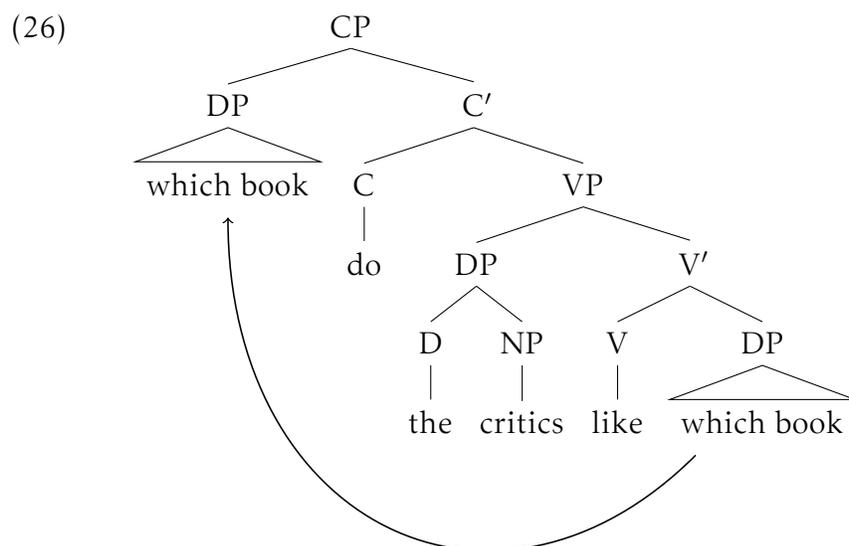
2.1.1.2 Adding Movement

So far, our MG is just a fancy way of writing a context-free phrase structure grammar. What makes MGs a transformational grammar formalism is the possibility of transformations, in the case of MGs (and Minimalism in general) the operation **MOVE**, which allows for a constituent to be pronounced in a different position from where it is most plausibly interpreted.

As we saw above in (24), a noun phrase (or DP) is interpreted as the (direct) object or theme of the action or state predicated by the verb, but in the constituent question in (25) the direct object *which book* appears at the front of the question.

(25) Which book do the critics like?

Within the transformational tradition, it is assumed that in the derivation of (25) the DP *which book* starts out as the complement of *like* and subsequently ‘moves’ to the front of the sentence, as sketched in (26), where *which book* is shown in both positions with an arrow to indicate the movement step.



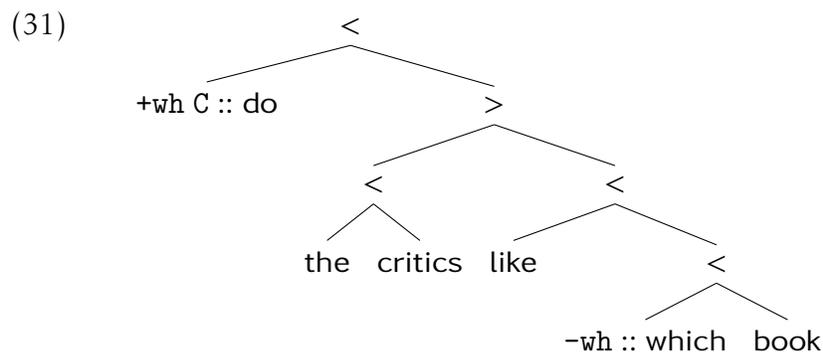
As a structure building operation, MOVE cannot occur freely, it must be licensed by features. So let's assume that the DP *which book* carries a licensee (or 'goal') feature, which is checked by a licensor (or 'probe') feature on the interrogative complementizer head, which in (26) is spelled out⁶ as *do*. We shall write licensee features as $-f$ and licensor features as $+f$, indicating that when they match, they both disappear.⁷ So we can add the following two items to our lexicon

- (27) =N D $-wh$:: which
 =V $+wh$ C :: do

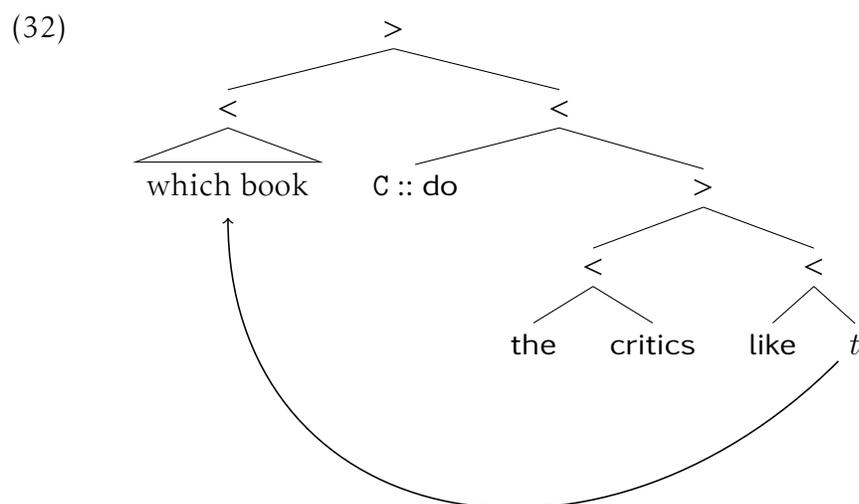
The operation MOVE shall be subject to the SHORTEST MOVE CONSTRAINT (SMC) which says that a mover must move as soon as it can, which is equivalent to requiring that it perform

⁶ The analyses in this section are simplified to avoid head movement. Head movement can easily be accommodated in MGs (KOBELE 2006), and in Chapter 3 I develop my own account of head movement from an incremental perspective. Unlike all other operations investigated, head movement seems to be significantly different when viewed incrementally compared to bottom-up.

⁷ So licensing features are never persistent and always uninterpretable.



The last feature to be checked in (31) is the licensing feature $+wh$, which triggers the operation *MOVE*, moving the phrase carrying the corresponding $-wh$ feature. This gives us (32), where the movement step is indicated by an arrow and the emptied base position is marked with the symbol t .



Taking stock, we note that features come in pairs consisting of an attractor feature (marked by a + or = prefix) and an attractee feature. Furthermore we distinguish two types of features: selection features which trigger *MERGE* and licensing features which trigger *MOVE*. The distinction between selection and licensing features is not a necessary one

(STABLER 2011), but we will maintain it throughout. The reason is that we will eventually make some features, namely selectee (or category) features, interpretable or persistent, i.e. they will not be deleted by MERGE. However, for the moment all features get deleted and feature checking is symmetric.

2.1.2 Linearizing Complements and Specifiers

In the example derivations above, we implicitly assumed that the leaf nodes of the derived syntax trees reflect the linear order of the corresponding expression. But how do we determine the order of the string resulting from MERGE? In the examples above, specifiers are merged to the left of the expressions selecting them, while complements are merged to the right. One common approach in MGs (e.g. STABLER 2011, STANOJEVIĆ & STABLER 2018, HUNTER ET AL. 2019) is to hard-code the linear order in the definition of MERGE, by specifying that elements merging with lexical items are linearized to the right, while elements merging with derived items are linearized to the left. Note how this definition explicitly encodes the notion of complements as ‘first-merged elements’.

An alternative is to formulate a directional MG (STABLER 2011, TORR & STABLER 2016), where the linearization order of MERGE is encoded in the feature diacritics: instead of writing a selector feature as =f, we now write it as either >f or <f, depending on whether the selected element is to be linearized to the right or to the left of the selecting element, respectively. As a concrete example, an element with a >D feature selects a determiner phrase to its right, while an element with a <D feature selects a determiner to its left, yielding the lexical entry in (33) for a transitive verb like *like*⁸.

⁸ In the directional MGs defined in STABLER 2011, TORR & STABLER 2016, the direction of selection is indicated by writing the feature diacritic = on the side in which the selected element is linearized, as in

(i) D= =D V :: like

(33) >D <D V :: like

If the directionality of selection is irrelevant, we may write a selection feature as =f, which is meant as a short-hand for the union of two feature specifications, one with selection feature <f, and one with selection feature >f.

Directional MGs are similar to Categorical Grammars (up to movement), and using them instead of a regular MG has the added benefit of not requiring the definition of MERGE to necessarily encode the notion of complements as first-merged objects⁹. This does not mean that there is no distinction between complements and specifiers, just that that distinction is not hard-coded in the structure building rules. Notice how we (and TORR & STABLER 2016) only specify the directionality for selector features, while licensing features always get checked in one direction, which hard-codes the assumption that movement is always to the left and up the tree. This also narrows the locus for parametric variations and offers a lexicalized version of the head (directionality) parameter.

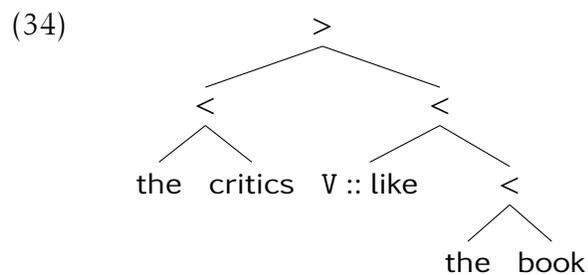
One may wonder whether a directional MG is more expressive or less restrictive than an MG without directionality specifications. The answer is no: both formalisms are equally expressive (STABLER 2011). More interestingly, in a case study of cross-linguistic word order variation in determiner phrases (STABLER 2011), both directional and non-directional MGs can derive the same set of word orders, which include all attested word orders according to GREENBERG's (1963) Universal 20, but less than the number of logically possible word orders (cf. CINQUE 2005, ABELS & NEELEMAN 2012).

Again, this is just a notational variant. In the variant chosen here, all feature diacritics are prefixes, and > and < have a clear pointing character that is already used to point to the head (or label) in derived trees.

⁹ TORR & STABLER 2016 still encode this distinction, since otherwise there is nothing prohibiting lexical items from containing 'movers', as discussed in the next section. We do not impose this restriction. Instead when modeling the inverse of head movement in Chapter 3, we will explicitly allow lexical items to contain movers.

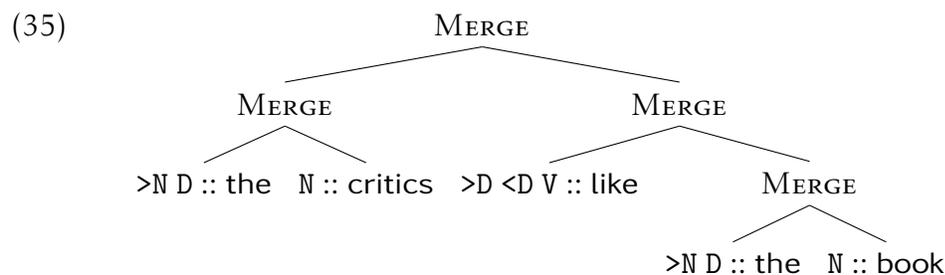
2.1.2.1 Derivation Trees, Collapsing Trees, and Notation

The syntactic structure of a linguistic expression is hierarchical, because it reflects the expression's constituency structure. And so a tree structure like the one in (34) is a common representation of the result of a syntactic analysis.



Another reason why the structure in (34) has the form of a tree, is that it is the result of a bottom-up derivation involving the structure building operations MERGE and MOVE. In a derived tree like (34), the specifics of the derivation, e.g. which operation applied when, are rather implicit.

Instead of focusing on the result of a syntactic analysis, we can view a syntactic derivation as a description of the derivational process. And this process description can be represented as a derivation tree. Example (35) shows a derivation tree corresponding to the derived tree in (34).



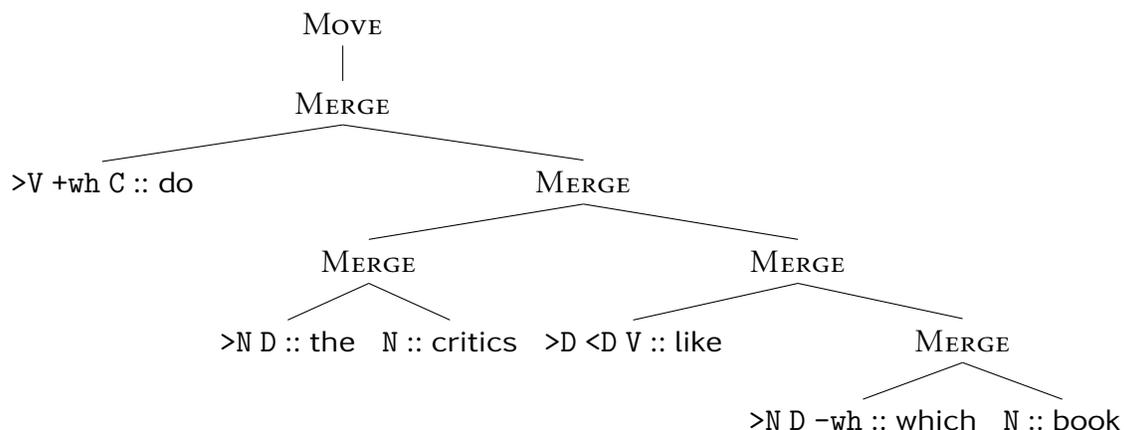
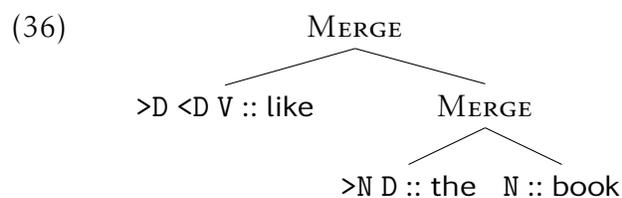


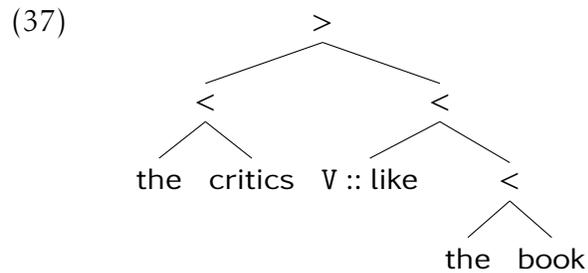
FIGURE 1: Minimalist Grammar derivation tree for *Which book do the critics like?*

Notice how in (35) each subtree corresponds to a state of the bottom-up derivation: e.g. subtree shown in (36) represents the derivation after the verb *like* merged with the direct object *the book*.



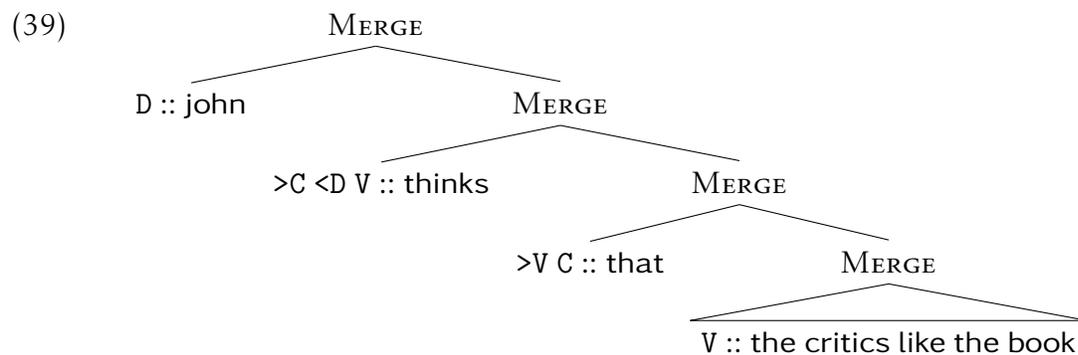
As long as the derivation consists only of **MERGE** steps, the derivation tree is isomorphic to the derived tree of the full derivation, except that the derivation tree (or its yield) need not reflect the surface word order of the represented string. Once a derivation also involves **MOVE** steps, the derivation tree is not a binary tree any more, since **MOVE** is a unary operation, and the derivation tree is not isomorphic to the derived tree any more. A derivation tree for the *wh*-question in (25) with its derived tree in (32) is shown in Figure 1; notice how the top node is a unary **MOVE** node, which checks the **-wh** feature of the phrase *which book*.

Looking once again at derived trees, we notice that most of the tree structure in a derived tree like (37) is functionally inert, it does not have any effect and cannot change once the derivation has moved on.



Since structure building is strictly feature driven, no structure building operation can alter the internal structure of a subtree that does not contain any features. So if (37) is part of a larger linguistic expression, as in (38), all future structure building operations treat it as a single unit, as indicated in (39) using the short-hand triangle notation.

(38) John thinks that the critics like the book.

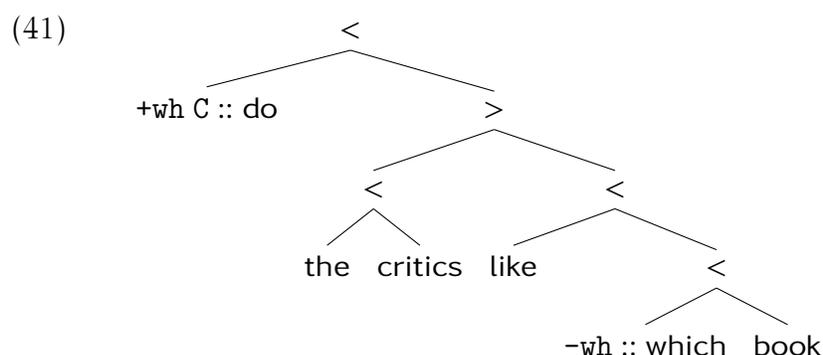


While not part of any formal definition, the triangle in (39) has a function: it marks its subtree as a derived expression whose internal structure is omitted. But as we just observed the internal tree structure of the triangled expression in (39) is functionally inert and

cannot change, so we may as well get rid of the triangle and write derived expressions in the same form as we write lexical items, e.g. write the triangled subtree in (39) as (40).

(40) V :: the critics like the book

In expressions with moving parts, like the one in (41), not all internal structure is inert: the phrase *which book* is a mover and so its position within the structure is going to change once its licensing feature is checked.



While the triangle notation is inadequate to represent (41) as a unit, we do not need its full hierarchical structure either. All we need is a representation of all the moving sub-expressions in addition to the non-moving part of the expressions. And since moving sub-expressions are also expressions, we can write an expression with moving sub-expressions as a (comma-separated) list of expressions, where the first expression is the main one, and any following expressions are movers. With this notational convention, the expression in (41) can be written as in (42).

(42) [+wh C :: do the critics like, -wh :: which book]

The brackets are optional and help to visually delineate complex expressions.

As a small adjustment of notation, we shall introduce a way to distinguish attractor features from attractee features, since the two have different functions, resembling the distinction between input and output types of functions in programming languages: a lexical item of a transitive verb like *like* takes two elements of type D as inputs and produces an output of type V. Looking ahead, this distinction will become relevant, and so to separate the two types of features visually (but not too much), I insert a dot¹⁰ between them. The features still form a single list, but the two parts can be referred to separately in rules. The full representation of the lexical item *like* is shown in (43), and (44) shows the full representation for the derived expression resulting from *like* merging with *what*.

(43) >D <D. V :: like

(44) [<D. V :: like, -wh :: what]

With all notation fixed, we can write the derivation tree of (25) in yet another way: instead of labeling non-terminal nodes with the structure-building operation as we did in (36), we can let the non-terminal nodes indicate the resulting expressions. This is shown in Figure 2. Since a derived expression is a collapsed version of a derived tree, the derived tree in Figure 2 is actually a tree of trees (each internal node represents a derived tree), and we can replace any subtree with its root and still validate the rest of the (bottom-up) derivation.

To conclude our discussion of Minimalist Grammars and to fix the notation for the rest of the chapter, we give a formal definition of Minimalist Grammars based on STABLER (2011: A.1), but with the modifications just discussed.

¹⁰ This is reminiscent of the way the programming language Lisp (McCARTHY 1960) displays pairs (or cons cells), which in turn are the building blocks of lists.

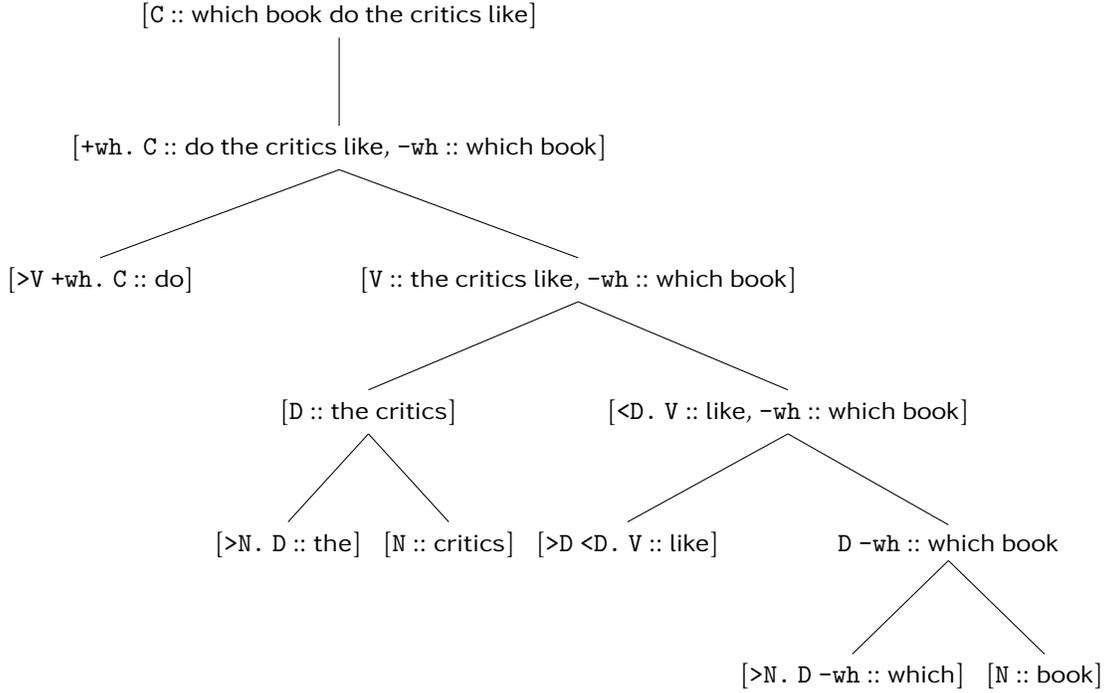


FIGURE 2: Derivation Tree for *Which book do the critics like*. Internal nodes contain the expression representing their sub-tree.

2.1.2.2 A Formal Definition

Let Σ be a vocabulary, including the empty string denoted by ϵ . Let B_S and B_L be two sets of basic features, the sets of selection and licensing features.. The features in B_S and B_L are concatenated with prefixes specifying their role to yield the sets

$$\begin{aligned}
 (45) \quad B_{L+} &= \{ +f \mid f \in B_L \} && \text{of licensors} \\
 B_{L-} &= \{ -f \mid f \in B_L \} && \text{of licensees} \\
 B_{S+} &= \{ <f \mid f \in B_S \} \cup \{ >f \mid f \in B_S \} \cup \{ =f \mid f \in B_S \} && \text{of selectors} \\
 B_{S-} &= \{ f \mid f \in B_S \} && \text{of selectees or categories.}
 \end{aligned}$$

Let F be the set of role-marked features, defined as the union of the sets in (45), i.e.

$$(46) \quad F = B_{L+} \cup B_{L-} \cup B_{S+} \cup B_{S-}.$$

A simple expression (or a chain) is a pair consisting of a feature sequence and a string (i.e. a sequence over Σ), an expression is a non-empty sequence of chains. Let the set of chains C and the set of expressions E be defined as

$$(47) \quad \begin{aligned} C &= F^* \times \Sigma^* \\ E &= C^+ \end{aligned}$$

Let the lexicon $\Lambda \subset C$ be a finite set. Then a Minimalist Grammar G is defined as a tuple

$$(48) \quad G = \langle \Sigma, F, \Lambda, c, \{\text{MERGE}, \text{MOVE}\} \rangle$$

where Σ is the vocabulary, F the set of role-marked features, Λ the lexicon, $c \in B_S$ a start symbol (or start category) and $\{\text{MERGE}, \text{MOVE}\}$ the structure building operations (or generating functions) as defined in Figure 3.

The set of structures $S(G)$, i.e. the set of expressions that can be generated by G is the closure of the lexicon Λ with respect to the structure building operations:

$$(51) \quad S(G) = \Lambda^{\{\text{MERGE}, \text{MOVE}\}}$$

A sentence structure is a structure whose only feature is the start category c , and the set of sentence structures $\mathcal{L}(G)$ generated by the grammar G is

$$(52) \quad \mathcal{L}(G) = \{ c :: s \mid c :: s \in S(G) \},$$

whereas the set of sentences $L(G)$ is defined as the set of sentence strings

For strings $s, t \in \Sigma^+$, for feature sequences $\gamma \in F^*$, $\delta \in F^+$ and $\mathbf{f} \in B_S$, $\mathbf{g} \in B_L$ an arbitrary selection, licensing feature, and for (possibly empty) sequences of chains $\vec{\phi}, \vec{\psi}$, let MERGE and MOVE be defined as follows:

- (49) MERGE is the union of the following three operations
- a. $\text{MERGE}_{>}([>\mathbf{f} \gamma :: s, \vec{\phi}], [\mathbf{f} :: t, \vec{\psi}]) \longrightarrow [\gamma :: st, \vec{\phi}, \vec{\psi}]$
selected item is linearized to the right
 - b. $\text{MERGE}_{<}([<\mathbf{f} \gamma :: s, \vec{\phi}], [\mathbf{f} :: t, \vec{\psi}]) \longrightarrow [\gamma :: ts, \vec{\phi}, \vec{\psi}]$
selected item is linearized to the left
 - c. $\text{MERGE}^M([=\mathbf{f} \gamma :: s, \vec{\phi}], [\mathbf{f} -\mathbf{g} \delta :: t]) \longrightarrow [\gamma :: s, -\mathbf{g} \delta :: t, \vec{\phi}]$
selected item becomes a Mover
- (50) MOVE is the union of the following two operations
- a. $\text{MOVE}^T([+\mathbf{g} \gamma :: s, \vec{\phi}, -\mathbf{g} :: t, \vec{\psi}]) \longrightarrow [\gamma :: ts, \vec{\phi}, \vec{\psi}]$
mover is linearized into Top of the chain
 - b. $\text{MOVE}^C([+\mathbf{g} \gamma :: s, \vec{\phi}, -\mathbf{g} \delta :: t, \vec{\psi}]) \longrightarrow [\gamma :: s, \vec{\phi}, \delta :: t, \vec{\psi}]$
mover Checks licensee feature (intermediate step)

FIGURE 3: Structure building operations for Minimalist Grammars

$$(53) \quad L(G) = \{ s \mid c :: s \in \mathcal{L}(G) \}$$

Note that $\mathcal{L}(G) \subset S(G) \subset E$.

2.1.3 Parsing (Minimalist) Grammars

Parsing is the process of syntactic analysis of sentences of a given language according to a grammar of said language, but with the implication that the analysis steps be formalized into an algorithm, which can (in principle) be run by a computer. A parsing algorithm

takes as input a sequence of lexical items¹¹ and returns some representation of a syntactic analysis. The most basic representation is just a yes/no verdict whether the input represents a valid sentence of the grammar or not. A more useful output of a parsing algorithm is a derivation tree like the one in Figure 2, which is also called a parse tree, and we will assume that the goal of parsing is to (at least implicitly) reconstruct the parse tree from an input sequence.

While there are many different parsing techniques (GRUNE & JACOBS 1990), all of them fall into one of two broad classes: top-down parsing and bottom-up parsing. In top-down parsing, the derivation or parse tree is constructed from the top (or the root) of the tree down along the branches. In bottom-up parsing, the derivation tree is constructed from the leaves at the bottom up along the branches.

The distinction between top-down and bottom-up parsing is particularly intuitive when considering a simple context-free phrase structure grammar like

- (54)
- | | | |
|-----|---|-----------|
| S | → | NP VP |
| NP | → | Det N |
| VP | → | V NP |
| N | → | cat rat |
| V | → | chased |
| Det | → | the |

The phrase structure rules in (54) can be read as production rules, which recursively rewrite all non-terminal symbols with their matching right-hand side until there are only terminal symbols left. Top-down parsing follows this production process, by expanding

¹¹ Determining the sequence of lexical items from a raw string is typically not done by the parser. When dealing with formal languages, this is the job of the lexer. When processing human language, this process is typically called syntactic category disambiguation, which may be a separate module in the architecture of human language processing (CROCKER & CORLEY 2002, BAUMANN 2013).

the start symbol (here S) and all non-terminals until the sentence in question is derived. The main problem with this approach is knowing which production rule to apply at which step to end up with the sentence in question and not any other derivable sentence.

Bottom-up parsing starts from the terminal nodes corresponding to the sentence in question, and tries to combine them into non-terminals by essentially inverting the production rules. This process of combination is then repeatedly applied to non-terminals as well, until only the start symbol is left.

So top-down parsing is most naturally viewed as a production process, while bottom-up parsing is more like a construction (or a recognition) process. These metaphors should not be equated with language production and recognition (or comprehension): both parsing strategies are in principle suitable for language comprehension and language production.

Since parsing is just syntactic analysis, a (pen and paper) syntactic derivation can also be performed either top-down or bottom-up. In the days when phrase structure rules like (54) were the main vehicle of syntactic theorizing, top-down derivations were the norm. But nowadays, syntactic structure building is typically conceptualized as proceeding bottom-up. One may argue that bottom-up structure building is more inductive, and that it is a better fit for the movement transformations employed in transformational grammar, but these reasons are more conventional¹² than conceptual.

2.1.3.1 Parser Internals

When implemented as a computer program, a parser consists of a set of data structures and a set of operations which manipulate those data structures.¹³ At the minimum, a parser needs two separate data structures: one to represent (a copy of) the input as it is being

¹² If the top-down perspective had prevailed, transformations would look different from today's general purpose MOVE but the concept of transformations is compatible with both bottom-up and top-down structure building.

¹³ In the language of object-oriented programming, a parser is an object.

parsed, and one to represent the internal state of the parser and (sometimes implicitly) the parse tree. The data structure representing the internal state of a parser is typically called the `PARSE STACK`, even if it is not technically a stack data structure. In the most general case, the parse stack is a mutable list-like data structure with random access to every element and no further restrictions. More commonly, the parse stack is either an actual stack or a (priority) queue data structure. A stack is a list-like data structure which implements the last-in, first-out (LIFO) principle: the last element added to (or pushed onto) a stack, is the first (and only) one that can be accessed (or popped), and it must be popped to access the elements pushed before it. So a stack behaves like a stack of plates at self-serve restaurant. A queue, on the other hand, implements the first-in, first-out (FIFO) principle: the first element added to the queue is the first (and only) one to be accessed or removed. Stacks and queues are restrictive data structure in that they do not offer random access, instead only two designated positions in the sequence are accessible, one for adding new elements and one for removing them.¹⁴ In parsing it is sometimes useful to have a stack with a designated top element while also allowing the occasional random access of elements lower on the stack.

The data structure representing the input is also a mutable list-like data structure with random access to every element, but eventually we want to parse an input sequence incrementally, meaning we want to impose the restrictions that elements can only be accessed and removed from the front of the input. For non-incremental parsing, like the bottom-up procedure illustrated next, we must allow random access. And if lexical items occur multiple times in an input sequence (think of the determiner *the*) we may add indices for disambiguation.

¹⁴ In the case of the queue these positions are the beginning and the end of the sequence, in the case of the stack they are the same position.

The operations manipulating the parsing data structures are dependent on the parsing strategy: for top-down parsing they are PREDICT and MATCH, for bottom-up parsing they are SHIFT and REDUCE. Here MATCH and SHIFT are largely independent of the grammar being parsed, while PREDICT and REDUCE correspond to applications of the rules of the individual grammar, possibly with some additional steps.

As sketched so far, the parser is mostly a bookkeeping device, it encodes what parsing steps are possible at what time in the derivation, but it does not know how to choose a specific step if more than one parsing action is possible (which may be the norm). What we are missing is a controller or a strategy to decide which parsing action to take at which time. And while the bookkeeping part of a parser is typically rather simple (it implements a non-deterministic automaton), the controller can be arbitrarily complex (GRUNE & JACOBS 1990: p. 69ff): it may have knowledge of the full grammar and it may even have been ‘trained’ on a (more or less) representative sample of possible input strings, as in the case of probabilistic parsers.

2.1.3.2 Parsing and Human Language Processing

In terms of the competence vs. performance distinction (CHOMSKY 1965), the controller is part of linguistic performance: in the case of human language comprehension, it encapsulates a comprehender’s experience with the language and may thus be the locus of a predictive component (cf. HALE 2001, 2006) in the human language processor. Resource limitations, such as a limited working memory, also affect the non-deterministic automaton by e.g. limiting the size of the stack. But if such resource limitations are abstracted away, the parsing automaton is most plausibly viewed as a part of linguistic competence, encoding the knowledge of what structures can be built and in what steps, under idealized circumstances.

In this sense, the present work is (almost) exclusively concerned with specifying the parsing automaton, and I will have very little to say about the control mechanism. I will generally assume that the correct parsing steps are chosen by an unspecified oracle, but occasionally I may hint at how certain conflicting parser states may provide a basis for a model of sentence processing difficulties.

While resource limitations typically affect the parsing automaton, they may also affect the controller, either directly by e.g. limiting the amount of search that the controller can do (for a search-based account of processing difficulty, see HALE 2014), or indirectly by leading to preferences for parses with shorter dependency lengths (e.g. GIBSON 1998, but cf. BAUMANN 2014 for a critical empirical assessment).

2.1.3.3 Bottom-Up Structure Building as Parsing

In Minimalism, syntactic derivations are typically built in a bottom-up manner, and bottom-up parsing can be viewed as an explicit algorithm for such bottom-up derivations. Like a syntactician armed with pen and paper, a bottom-up parser starts with the input sequence (or the numeration) and tries to reduce it to the start symbol. The two main operations of a bottom-up parser are `SHIFT` and `REDUCE` (hence also the name `SHIFT-REDUCE PARSER`). `SHIFT` moves or ‘shifts’ an element from the input onto the parse stack (and is typically not made explicit in manual syntactic derivations). `REDUCE` manipulates or ‘reduces’ the parse stack by applying a grammar rule to the top element(s) on the stack and pushing the result back onto the stack (in manual syntactic derivations there is typically no distinction between `REDUCE` and the grammar rule applied).

The parse stack of a bottom-up MG parser is an actual stack data structure, and elements on the stack are MG expressions as defined in (47), each being a non-empty sequence of pairs consisting of a feature sequences plus a linguistic sign, for the moment a string of words corresponding to its orthographic representation. Since the goal of bottom-

up parsing is to reduce the input to the start symbol, initially the parse stack contains only an item carrying the start symbol. Instead of choosing a specific syntactic category (say *C*) as the start symbol, we let the start symbol be an ‘extra-grammatical’ category feature *R*. Then the initial element on the stack is an item with this category feature *R* and one selector feature¹⁵, i.e. >X. R . When deriving full sentences, the start item should select a CP, i.e. have features >C. R , but we shall allow other selector features, both to avoid clutter (in the form of empty CP projections) and for more substantial reasons like a non-elliptical analysis of fragment answers, in particular non-clausal short answers to constituent questions.

From the initial stack, a bottom-up parsing derivation proceeds by shifting elements from the input onto the stack and reducing them whenever there is an applicable rule in the grammar, pushing the result back onto the stack. In general, a bottom-up derivation is not incremental and so we will allow random access to the input sequence, i.e. *SHIFT* can target any item in the input.

We can now give an example parsing derivation using all the notation and machinery introduced so far. The main purpose of this first parsing derivation is to introduce the parsing notation used throughout the rest of the text. And so we write the previous bottom-up MG derivation as a bottom-up parsing derivation. In the first derivations we will be using the following lexical items:

- (55)
- | | |
|----------------------------------|------------------------------------|
| $N :: \text{critics}$ | $N :: \text{book}$ |
| $\text{>N. D} :: \text{the}$ | $\text{>N. D -wh} :: \text{which}$ |
| $\text{>D <D. V} :: \text{like}$ | $\text{>V +wh. C} :: \text{do}$ |

¹⁵ *R* is a short hand for the symbol *ROOT* commonly used in parsing. In more linguistic terms, *R* is the category of an utterance, which has the selected phrase as its linguistic content. The feature *R* is part of the grammar, but not in any meaningful way, hence the label ‘extra-grammatical’.

Let's start with the simple declarative sentence (56) to illustrate the basic principles of bottom-up parsing of MGs. The sentence in (56) corresponds to the sequence of lexical items listed in (57), where we added indices to distinguish the two occurrences of the definite article, but left out the feature specifications to reduce clutter.

(56) The critics like the book.

(57) the₁ critics₂ like₃ the₄ book₅

Our oracle tells us that a bottom-up parse of (57) can be constructed by going through the input from end to start¹⁶, so we shall proceed in this order.

At the beginning, the parse stack contains just the start item >V. R. Starting from the end of the sentence, we shift the item book₅ onto the stack.

(58) SHIFT book₅

N :: book

>V. R

Since the two items on the stack have incompatible features, there is nothing to reduce, all we can do is shift the determiner the₄ onto the stack.

(59) SHIFT the₄

>N. D :: the

N :: book

>V. R

¹⁶ Being able to construct a bottom-up parse by going through the reversed input is not always guaranteed to work, but it is a decent heuristic in a mostly right-branching language like English.

- (63) SHIFT critics₂, the₁
- >N. D :: the
- N :: critics
- >D. V :: like the book
- >V. R

Now, we can finally reduce again: first we merge the top two items to form the subject DP (64), and then we can merge the resulting subject DP and the VP on the stack (65).

- (64) MERGE_> N
- D :: the critics
- >D. V :: like the book
- >V. R
- (65) MERGE_< D
- V :: the critics like the book
- >V. R

Finally, we can merge the remaining two items on the stack to obtain the full sentence with the start symbol R as its category:

- (66) MERGE_> V
- R :: the critics like the book

This is how most of our parsing derivations will be written. The full derivation without comments is shown in Figure 4. Of course, things become more complicated with movement, so in Figure 5 we give a bottom-up parsing derivation of (67) and the corresponding input sequence in (68). The derivation tree for this sentence was shown in Figure 2.

1. SHIFT book₅
N :: book
>V. R
2. SHIFT the₄
>N. D :: the
N :: book
>V. R
3. MERGE_> N
D :: the book
>V. R
4. SHIFT like₃
>D <D. V :: like
D :: the book
>V. R
5. MERGE_> D
>D. V :: like the book
>V. R
6. SHIFT critics₂
N :: critics
>D. V :: like the book
>V. R
7. SHIFT the₁
>N. D :: the
N :: critics
>D. V :: like the book
>V. R
8. MERGE_> N
D :: the critics
>D. V :: like the book
>V. R
9. MERGE_< D
V :: the critics like the book
>V. R
10. MERGE_> V
R :: the critics like the book

FIGURE 4: Bottom-up parsing derivation of *The critics like the book*.

(67) Which₁ book₂ do the critics like?

(68) which₁ book₂ do₃ the₄ critics₅ like₆

The parsing derivation in Figure 5 has some peculiar features: unlike the previous bottom-up parse, we do not consume the input from end to beginning, instead we start with the moved phrase *which book*, assemble it, and then merge it as the complement of *like* and turn it into a mover. But notice how the only occurrence of *which book* in the input string/sequence is in its fronted position, yet we merge it as if it occurred as the complement of *like*. This forces us to consume the input even more out of order than is typical for bottom-up parsing.¹⁷

Our goal is to parse sentences incrementally in the way humans naturally do (or are forced to do) when producing or comprehending spoken language. So in the next section I present an incremental parsing algorithm for Minimalist Grammars.

¹⁷ One may argue that the phrase *which book* is actually consumed in order, but then we have to jump to the end. Compared to this, a bottom-up parse without movement can often proceed from back to front, which is arguably more ‘in order’ than starting at the front and then jumping to the back.

1. SHIFT $wh_1, book_2$
 - >N. D - wh :: which
 - N :: book
 - >C. R
2. MERGE $_>$ N
 - D - wh :: which book
 - >C. R
3. SHIFT $like_6$
 - >D <D. V :: like
 - D - wh :: which book
 - >C. R
4. MERGE M D
 - <D. V :: like, - wh :: which book
 - >C. R
5. SHIFT $the_4, critics_5$
 - >N. D :: the
 - N :: critics
 - <D. V :: like, - wh :: which book
 - >C. R
6. MERGE $_>$ N
 - D :: the critics
 - <D. V :: like, - wh :: which book
 - >C. R
7. MERGE $_<$ D
 - V :: the critics like, - wh :: which book
 - >C. R
8. SHIFT do_3
 - >V + wh . C :: do
 - V :: the critics like, - wh :: which book
 - >C. R
9. MERGE $_>$ V
 - + wh . C :: do the critics like, - wh :: which book
 - >C. R
10. MOVE T wh
 - C :: which book do the critics like
 - >C. R
11. MERGE $_>$ C
 - R :: which book do the critics like

FIGURE 5: Bottom-up parsing derivation with movement of *Which book do the critics like?*

2.2 Incrementally Parsing Minimalist Grammars

Any incremental parsing or structure building procedure for a given grammar formalism should satisfy two requirements: it should be incremental, and it should use the grammar. Being incremental means that the input can only be processed from start to end, and only once, otherwise we can process elements in any order by picking them up in different rounds. In principle, incremental processing is compatible with the possibility of looking ahead at upcoming elements in the input, but unrestricted look-ahead goes against the spirit of incremental processing. The human language processor is not necessarily strictly incremental, syntactic parsing may lag behind lexical disambiguation, and semantic processing in turn may lag behind syntactic parsing, but psycholinguistic evidence clearly indicates that processing does not wait until a large unit of input (say, a whole clause, or phase, or proposition) is perceived, or production only starts after such a unit is built.

Using the grammar means that the parser operates over expressions as defined by the grammar and uses the structure building operations of the grammar. When building a parser (or developing a parsing algorithm) for a given grammar, these two requirements are strict requirements, and a number of incremental parsers adhering to these requirements have been proposed in the literature: for Minimalist Grammars there are the top-down and Early parsers in HARKEMA 2001, the top-down parsers in STABLER 2013 and KOBEL, GERTH & HALE 2013, and two left-corner parsers in STANOJEVIĆ & STABLER 2018 and HUNTER ET AL. 2019.

In contrast to these works, the goal of the present work is not so much to propose a (formally sound and complete) parsing algorithm for Minimalist Grammars (or to develop a sentence processing model), but to investigate Minimalist structure building from an incremental (as opposed to the prevailing bottom-up) perspective and discuss its linguistic consequences. So instead of reviewing and discussing previous incremental

parsers now, we first present our own parsing solution and then compare it to previous work in Section 2.6.3. In our own solution we will be a bit more lax about the two requirements: they will be our guiding principles, but they can be violated if it makes sense and we learn something from it.

As we will see, the parser we develop will not require any look-ahead and the necessary adjustments to the definitions in Section 2.1.2.2 will be fairly minimal: we will be adding a few additional rules to the definitions of MERGE and MOVE, which is rather unproblematic since the rules given in Figure 3 are just one of many possible ways of formalizing MERGE and MOVE (e.g. we chose a directional MG instead of the non-directional one we started with). While the additional rules can be classified as clear instances of either MERGE or MOVE, we will come to realize (and discuss) that the distinction between MERGE and MOVE is not as clear as bottom-up derivations make us believe.

The expressions the parser operates over will not change, the parser is a parser for Minimalist Grammars and so the expressions will be MG expressions as defined in Section 2.1.2.2, but some details will need to be made more precise in the process of developing the incremental parsing or structure building procedure.

2.2.1 Overview of the Incremental Parser

The existence of transformations in Minimalist Grammars (or any other transformational grammar frameworks) makes them somewhat challenging to parse incrementally: because of movement an expression may need to be parsed as though it was occurring in one position (typically its base position), but in the actual input string it occurs in a different position (typically its moved position). An instance of this situation was seen in Figure 5, where the phrase *which book* was merged with the verb *like* even though it appears at the other end of the input string. But what we did in the bottom-up derivation in Figure 5

was not quite right: we merged the moved phrase *which book* and the verb *like* according to MERGE_>, yet the moved phrase does not appear to the right of *like* in the input.

In the incremental parser proposed here, a moved phrase is ‘inserted’ into the position it is encountered in in the input, i.e. its moved position, and it then ‘moves’ to its base position by literally inverting the movement chain.

But the existence of MOVE is not the only challenge to building an incremental parser for Minimalist Grammars, the logic of bottom-up structure building is built into its lexical items and their feature specifications: the feature sequence of the lexical item in (69) specifies that the verb *know* first combines with a CP to its right and then with a DP to its left, but when we process a sentence like (70) incrementally, we would like to first merge *know* with the subject DP *the authors* and only afterwards with the complement CP.

(69) >C <D. V :: know

(70) The authors know that the critics like the book.

The solution I will be proposing is to transform lexical items from ‘bottom-up items’ to ‘incremental items’: for the item in (69) this means swapping the order of the two selector features:

(71) <D >C. V :: know

The so transformed item in (71) can now first combine with the subject DP, which linearly precedes it before combining with the complement CP.

Transforming lexical items is not quite enough though. Say we want to incrementally parse the embedded clause in (70), and we have the two expressions in (73).

(72) a. >C. V :: the authors know

b. >V. C :: that

Given the rules in Figure 3 we cannot combine the two items in (72), because the accessible features in the two items don't match. More intuitively, only completed items can be selected for MERGE. Since we are free to rewrite the rules, we will add a rule to allow MERGE to select incomplete items. But we will do so rather restrictively and only allow items with exactly one active selector feature to be themselves selected. This rule corresponds to the forward composition combinator in Combinatory Categorical Grammar.

In summary, the main characteristics of our incremental parser are

1. Transforming lexical items
2. Restrictively merging incomplete expressions
3. Inverting the movement chain

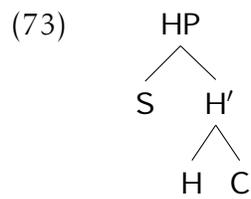
These will be discussed in more detail in the next sections. There I will be using terminology from both syntax and parsing, and I will not always draw a clear boundary between the two. To some extent this is just sloppiness for ease of exposition, but it also suggests that syntactic theorizing in the transformational tradition rests on certain mechanisms which are more compatible with parsing than with grammar.

2.2.2 Incremental Structure Building: MERGE

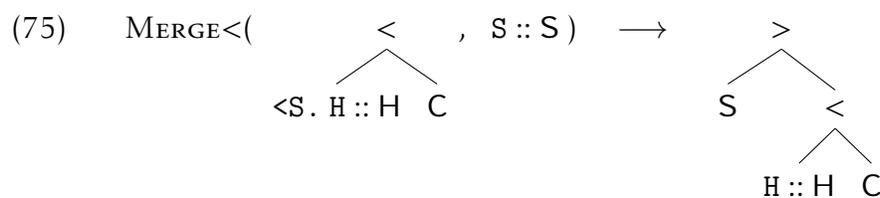
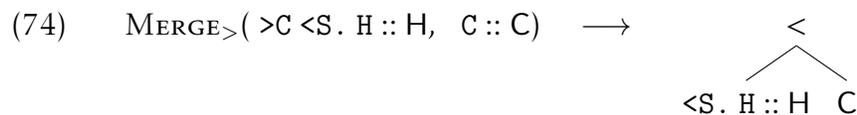
The structure building operation MERGE is the foundation of Minimalist syntax. And so we first show how syntactic structures without movement can be built incrementally, as opposed to bottom-up.

2.2.2.1 Transforming Lexical Items

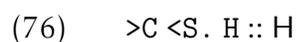
Modern transformational grammars often implement some form of X-bar structure (CHOMSKY 1970), according to which an phrase HP projected by a head H contains a complement C as the sister node of H and a specifier S as the sister of the H' formed by H and C:



In a bottom-up Minimalist derivation of (73), the head H first selects its complement C forming the expression on the right-hand side of (74), which in turn selects the specifier S as in (75).



So the complement C is merged before¹⁸ the specifier S is merged, and this order of merging is encoded in the feature sequence of the head:



¹⁸ It is sometimes claimed that structure building rules do not specify a temporal order and that temporal metaphors should thus be avoided when talking about structure building. Since grammars are characterizations of languages and thus at the computational level of MARR's (1982) hierarchy they do not come with an algorithmic or temporal specification for structure building. However, in the present work we attempt to formulate the algorithmic aspects of syntactic structure building, and later on we argue that some concepts in modern syntax, like phases or multiple spellout, are essentially temporal or parsing metaphors.

The feature sequence in (76) specifies that the head H first combines with complement C to its right and then with specifier S to its left. But in their linear order, the specifier S precedes the complement C and the head H , so a first step to make the feature sequence (76) more compatible with incremental structure building is to invert the order of its selector features and transform (76) into (77).

$$(77) \quad \langle S \rangle C. H :: H$$

Now the head H can first merge with the specifier S as in (78), and then the resulting expression merges with the complement C as in (79).

$$(78) \quad \text{MERGE}_{<}(\langle S \rangle C. H :: H, S :: S) \longrightarrow \begin{array}{c} > \\ / \quad \backslash \\ S \quad \rangle C. H :: H \end{array}$$

$$(79) \quad \text{MERGE}_{>}(\begin{array}{c} > \\ / \quad \backslash \\ S \quad \rangle C. H :: H \end{array}, C :: C) \longrightarrow \begin{array}{c} > \\ / \quad \backslash \\ S \quad < \\ \quad / \quad \backslash \\ \quad H :: H \quad C \end{array}$$

The derivations in (78) and (79) may look a bit strange, most notably the sister of S seems to change from (78) to (79), but this is just an artifact of the way trees are written: the node $\rangle C. H :: H$ is not in fact a terminal node, as it still has an active selector feature $\rangle C$. So the tree on the right-hand side of (78) is misleading in suggesting that H as a terminal node is the complement of S .¹⁹ It should be emphasized that despite their looks the derivations in (78) and (79) are possible according to the definitions of $\text{MERGE}_{<}$ and $\text{MERGE}_{>}$ in Figure 3. This means that all we need to incrementally build incomplete constituents like (80) is inverting the order of selector features of its constituent expressions.

¹⁹ A similar confusion can be observed in PHILLIPS 1996 and PHILLIPS 2003 which we will discuss in Section 6.4.

(80) The critics devoured . . .

The translation from ‘bottom-up’ to ‘incremental’ feature sequences is done by a transformation which operates over lexical items and inverts the order of all attractor features, not only selector features as shown above, but also licenser features as in (81), i.e. all features before the dot.

(81) $>V +wh. C :: do \longrightarrow +wh >V. C :: do$

Similarly, if an item contains licensee features in addition to a selectee (or category) feature as in (82), the licensee features need to be checked before the category feature.

(82) $D -wh :: who \longrightarrow -wh D :: who$

Intuitively, this serves the purpose of tracing the movement steps in their reverse order, starting at the final landing site and ending at the site where the element is first merged via its selectee feature. In Section 2.2.3 we further develop this view of ‘inverse’ movement, for the moment we just stipulate that attractor and attractee feature sequences both need to be inverted, but independently of each other²⁰, which is the reason we introduced the dot separating the two.

An example of the full transformation is given in (83) for a hypothetical lexical item with two selector features $>A <B$, one licenser feature $+x$, and two licensee features $-y -z$.

(83) $>A <B +x. C -y -z :: x \longrightarrow +x A. -z -y C :: x$

²⁰ Once again, a comparison with functions in programming languages (or the lambda calculus, CHURCH 1940) can be helpful, where attractor and attractee features correspond to input arguments and output values, respectively. Inverting the order of input arguments is occasionally useful, and so is inverting the order of outputs, but only independently of each other. There is no way inverting the feature sequence as a whole makes any sense.

Within the parsing procedure, we assume that this left-to-right transformation of lexical items happens as part of the SHIFT operation. So for the moment, the incremental parsing procedure is a parser for a standard bottom-up Minimalist Grammars: lexical items are as specified in bottom-up MGs, and are being transformed only when they enter the incremental derivation. With this, SHIFT can be defined formally as in (84), for a string $s \in \Sigma^+$ and for feature sequences $\gamma, \delta \in F^*$.

$$(84) \quad \text{SHIFT}(\gamma. \delta :: s) \longrightarrow \gamma^{-1}. \delta^{-1} :: s$$

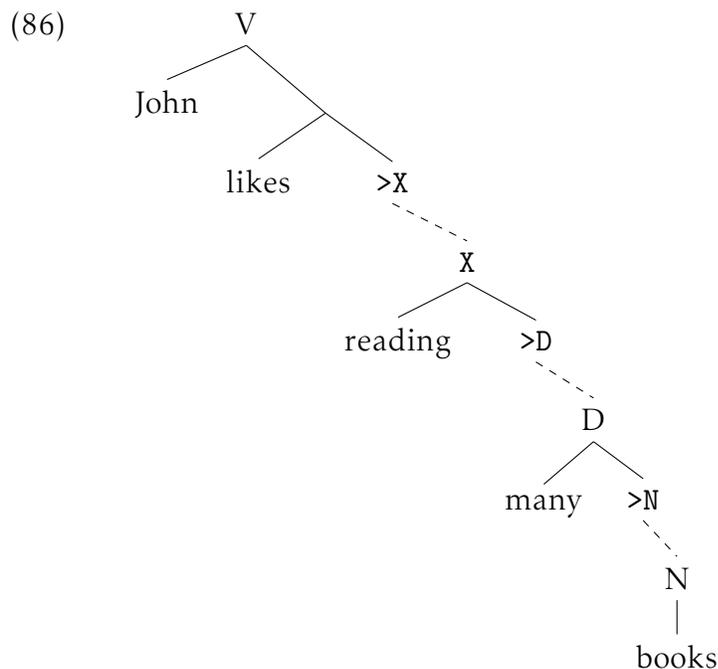
Here s^{-1} denotes the inverse of sequence s , e.g. $(a b c)^{-1} = c b a$.

2.2.2.2 Merging Incomplete Items

Incremental structure building proceeds through the complement position: a (derived) syntax tree grows by expanding its right edge, which is the complement position. As defined above, the operation MERGE_>, which fills the complement position, can only select complete items that have satisfied all their selection requirements, i.e. expressions that have checked all their selection features. But when building a tree incrementally we want to be able merge items as soon as possible. In particular, we want to be able to merge a head or derived tree T that has the selected category feature even if T 's own complement has not yet been merged. This is critical, because T 's complement may also select a complement, and so T would have to wait for its complement to be completed.

To see this consider the sentence in (85). Under a possible syntactic analysis, it consists of four phrases projected by the heads *likes*, *reading*, *many*, and *books*. These are shown in (86), where they are connected with dashed lines.

$$(85) \quad \text{John likes reading many books.}$$



For incremental structure building, we would like to merge the phrase projected by *likes* and the one projected by *reading* before the complement of *reading* is processed. In terms of our parsing notation, consider the parser state in (87) after *reading* was shifted.

(87) SHIFT *reading*

>D. X :: *reading*

>X. V :: *john likes*

>V. R :: ϵ

The two items on top of the stack have a matching X feature: the expression *john likes* selects an X and *reading* is of category X²¹. But *reading* still has an open selector feature >D, and because of that it cannot be selected by *john likes*: as defined in Figure 3, MERGE only allows expressions to be selected if they are *complete*, i.e. if they have no more open

²¹ Whatever X may be.

attractor features and their category feature is their active feature, i.e. the first feature in their feature sequence.

But we are free to rewrite the rules, and so we add a rule to allow MERGE to select an expression that has exactly one open selector feature. The new variant of MERGE corresponds to the forward composition combinator in Combinatory Categorical Grammar, and we write it as MERGE_{\gg} ('forward merge'). With it the derivation in (87) can continue as in (88).

$$(88) \quad \text{MERGE}_{\gg} X$$

$$\quad >D. V :: \text{john likes reading}$$

$$\quad >V. R :: \epsilon$$

A formal definition is given in (89) with the conventions from Figure 3, $\mathbf{x} \in B_S$ a category feature and $?$ the 'Kleene question mark', which marks zero or one occurrences of the feature thus modified ($?$ is whitespace-bounded).

$$(89) \quad \text{MERGE}_{\gg}([>\mathbf{f} \gamma :: s, \vec{\phi}], [>\mathbf{x}^? \mathbf{f} :: t, \vec{\psi}]) \longrightarrow [>\mathbf{x}^? \gamma :: st, \vec{\phi}, \vec{\psi}]$$

Note that there is no corresponding 'incomplete' or 'forward' variant for $\text{MERGE}_{<}$. This should be intuitively obvious: any item selected to the left will already need to have been seen before the selecting head is parsed. So skipping over a missing item to the left does not make much sense. The existence of movement will complicate this picture a little bit: we will be forced to introduce more operations over incomplete items to account for remnant movement in Section 2.2.4. The overall conclusion remains: only items expected to the right can be skipped over. This conclusion forms the basis of an explanation of subject islands (and specifier islands more generally) in Section 5.1.

Incomplete items can be merged only after they have merged with their specifier, as specifiers linearly precede their heads, and so if a head selects a specifier it must have been processed already and thus be on the parse stack. So our restriction on merging incomplete items is more a factual statement about the linear order of specifiers and heads than a substantial constraint on the application of MERGE: an element *S* must be processed before a head *H* if *S* is selected as the specifier of *H*. The substantive aspect of this constraint is that only incomplete expressions can be merged, but not predictions of expressions. Since all MG expressions are rooted in a head, we are designating the head as the left-corner in the sense of left-corner parsing (cf. the ‘head-corner parser’ developed in GIBSON 1991).

So the parser never tries to predict an unseen lexical item, its only ‘predictions’ are the category of an expected complement when merging an item missing that complement. So in a sense, our parser is strongly lexicalized, just like Minimalist Grammars: parsing operations can only be licensed by lexical items on the stack. This approach may be appropriate for languages like English, but it is probably not sufficient for head-final languages like Japanese (see Section 2.6.5 for a brief discussion of incremental parsing in head-final languages).

2.2.2.3 The Parser and an Example

The incremental parser is a shift-reduce parser, and thus essentially the same as the bottom-up parser described in Section 2.1.3.3; its parse stack is an actual stack, i.e. new elements can be pushed on top of the stack and popped from the top. Elements on the stack are MG expressions as defined in (47), each being a non-empty sequence of pairs consisting of a feature sequences plus a linguistic sign, for the moment a string of words corresponding to its orthographic representation. Initially the parse stack contains only a start element with the category feature *R* and one selector feature, e.g. the item in (90) when deriving a full CP.

(90) $\text{>C. R} :: \epsilon$

From the initial stack, a derivation proceeds by shifting elements from the input onto the stack and reducing them whenever there is an applicable rule in the grammar, pushing the result back onto the stack. Unlike the bottom-up parser in Section 2.1.3.3, the incremental parser has access to only one element of the input at a time, i.e. `SHIFT` can target only the front item of the input. In addition to just shifting elements onto the stack, `SHIFT` also performs the left-to-right transformation described in Section 2.2.2.1, thus acting as defined in (84).

For the moment, the possible operations of the parser are the MG structure building operations defined in Figure 3 plus the forward `MERGE \gg` as defined in (89) and `SHIFT` as defined in (84). In the language of a shift-reduce parser, the `MERGE` operations function as `REDUCE` and `SHIFT` is `SHIFT`. This set of operations will be expanded when discussing movement in Section 2.2.3, the full list of operations can be found in Figure 17. The start item in (90) does not ‘actively’ participate in incremental structure building, since it is meant to be an extra-syntactic item, it can only select a completed CP, not an incomplete item via `MERGE \gg` . It thus mostly acts as a way to allow multiple start symbols, but in Section 4.2 we argue that the start item may play the role of a assertion operator. As such it should also sit outside the syntactic derivation.

With the parser defined, we can illustrate the parsing algorithm with some example derivations. In the first derivations we will be using the following lexical items:

(91)

$\text{N} :: \text{critics}$	$\text{N} :: \text{book}$	$\text{N} :: \text{authors}$
$\text{>D} \text{<D. } \text{V} :: \text{like}$	$\text{>C} \text{<D. } \text{V} :: \text{know}$	$\text{>V} \text{+wh. } \text{C} :: \text{do}$
$\text{>N. } \text{D} :: \text{the}$	$\text{>N. } \text{D} \text{-wh} :: \text{which}$	$\text{>V. } \text{C} :: \text{that}$

Let's start with the simple declarative sentence (92) to illustrate the basic principles of our incremental parser. The sentence in (92) corresponds to the sequence of lexical items listed in (93), where we added indices to distinguish the two occurrences of the definite article, but left out the feature specifications to reduce clutter.

(92) The critics like the book.

(93) the₁ critics₂ like₃ the₄ book₅

For ease of comparison, the same example was used in the bottom-up parsing derivation in Figure 4, and the following parsing derivations are presented in the same format: we list the operation applied (e.g. SHIFT) and then print the stack as a vertical list with the top of the stack at the top.

At the beginning, the parse stack contains just the start item $>V. R :: \epsilon$. Starting with the first word, we shift the item the₁ onto the stack.

(94) SHIFT the₁
 $>N. D :: the$
 $>V. R :: \epsilon$

Since the two items on the stack have incompatible features, there is nothing to reduce, all we can do is shift the noun critics₂ onto the stack.

(95) SHIFT critics₂
 $N :: critics$
 $>N. D :: the$
 $>V. R :: \epsilon$

The top two items on the stack in (95) have a matching feature \mathbb{N} , so we can reduce them by applying $\text{MERGE}_{>}$ and pushing the result back onto the stack. Up to this point the derivation steps look very similar to those deriving the direct object DP *the book* in the bottom-up derivation in Figure 4.

(96) $\text{MERGE}_{>}$ \mathbb{N}

$\text{D} :: \text{the critics}$

$>\text{V}. \text{R} :: \epsilon$

Again there are no matching features and the only option is to shift the verb like_3 . Notice how SHIFT inverts the order of the two selector features of *like*.

(97) SHIFT like_3

$<\text{D} >\text{D}. \text{V} :: \text{like}$

$\text{D} :: \text{the critics}$

$>\text{V}. \text{R} :: \epsilon$

Now the top items on the stack in (97) have a matching feature D , so we can reduce by applying $\text{MERGE}_{<}$ to the verb *like* and the subject DP *the critics*. The result is pushed back onto the stack. Notice how the subject DP *the critics* did not trigger a prediction of an upcoming verb²², it instead got shifted onto the stack and only merged with the verb *like* after the verb was on the stack.

(98) $\text{MERGE}_{<}$ D

$>\text{D}. \text{V} :: \text{the critics like}$

$>\text{V}. \text{R} :: \epsilon$

²² This would be the case for the Minimalist Grammar left-corner parsers proposed in STANOJEVIĆ & STABLER 2018 and HUNTER ET AL. 2019, see Section 2.6.3.

At this point in the derivation subject and verb form a constituent in the form of the expression in (99).

(99) >D. V :: the critics like

It is generally considered a syntactic fact that subject and verb do not form a constituent to the exclusion of the direct object, and it should be emphasized that the expression in (99) is not ‘complete’, it still has an active selector feature. So subject and verb only form a constituent to the exclusion of the direct object until the direct object is parsed, and we notice that constituent structure can easily change during a left-to-right incremental derivation, unlike a bottom-up derivation, which is designed to build constituent structure in an additive and non-destructive manner. But non-standard constituents like the one in (99) can occur as actual constituents in right-node raising constructions like in (100), and the appearance of such non-standard constituents alongside more standard ones has led PHILLIPS (1996, 2003) to conclude that syntactic structure building proceeds strictly from left to right (see Section 6.4 for further discussions and comparisons).

(100) The critics like and the readers adore the book.

Returning to the example derivation, we may be tempted to apply MERGE_{>>} to the two items on the stack, as they have a matching feature V. But the selecting item is the start item and does not participate in incomplete merging. So we focus on the top item, which has an open >D selector feature and shift the next item the₄.

- (101) SHIFT the₄
- >N. D :: the
- >D. V :: the critics like
- >V. R :: ϵ

The shifted the₄ can be merged right away via MERGE_», since it is of category D. The single selector feature of the₄ gets ‘transmitted’ to the resulting item, which now selects a noun.

- (102) MERGE_» D
- >N. V :: the critics like the
- >V. R :: ϵ

Fortunately a noun is the last element in the input, and so book₅ is shifted onto the stack in (103) and then merged via an application of MERGE_> in (104).

- (103) SHIFT book₅
- N :: book
- >N. V :: the critics like the
- >V. R :: ϵ

- (104) MERGE_> N
- V :: the critics like the book
- >V. R :: ϵ

- (105) MERGE_> V
- R :: the critics like the book

Finally, the start item has a complete expression of category V to select and combines with it via MERGE_> in (105). This completes our first parsing derivation (a summary is shown

in Figure 6). As a second example, the full derivation of (106) is shown in Figure 7. There we leave out the start item during intermediate steps and only show it at the beginning and the end.

(106) The authors know that the critics like the book.

From a human language processing perspective a few aspects are worth remarking on: first, at most stages of the parsing derivation, the parse stack contains multiple unconnected items, which means that there is no fully connected parse tree. This may seem problematic, as human language comprehension as a whole is incremental, and not just parsing. So it must be possible to assign meaning to the parse of an incomplete sentence. But at least some elements on the stack can be interpreted independently, e.g. an item representing a DP can be interpreted by itself. So having multiple unconnected components is not necessarily an obstacle to incremental comprehension. But ultimately, we would like those components to be at least preliminarily connected. Such a connection could be achieved through an additional top-down or prediction layer, which may be more or less active in different types of languages, as comprehending a head final language may involve more predictions than comprehending an English-like language. This aspect of the proposal is mostly left for future research (a few more details are provided in Section 2.6.5).

Second, it should be emphasized that the derivation was successful because there were no more items in the input when there were no more items with active attractor features on the stack. If there were still items in the input and nothing on the stack selecting them, the derivation may have gone down a garden path. Since human comprehenders are at least sometimes able to recover from garden paths, a fully articulated model of human sentence processing would need to specify some mechanism for reanalysis. Since our focus

1. SHIFT the ₁	>N. D :: the >V. R :: ϵ
2. SHIFT critics ₂	N :: critics >N. D :: the >V. R :: ϵ
3. MERGE _{>} N	D :: the critics >V. R :: ϵ
4. SHIFT like ₃	<D >D. V :: like D :: the critics >V. R :: ϵ
5. MERGE _{<} D	>D. V :: the critics like >V. R :: ϵ
6. SHIFT the ₄	>N. D :: the >D. V :: the critics like >V. R :: ϵ
7. MERGE _» D	>N. V :: the critics like the >V. R :: ϵ
8. SHIFT book ₅	N :: book >N. V :: the critics like the >V. R :: ϵ
9. MERGE _{>} N	V :: the critics like the book >V. R :: ϵ
10. MERGE _{>} V	R :: the critics like the book

FIGURE 6: Incremental parsing derivation of *The critics like the book*.

is more on the competence aspect of parsing, proposing a reanalysis mechanism is outside the scope of this work.

1. SHIFT the, authors	N :: authors >N. D :: the >V. R :: ϵ
2. MERGE _{>} N	D :: the authors
3. SHIFT know	<D >C. V :: know D :: the authors
4. MERGE _{<} D	>C. V :: the authors know
5. MERGE _» V	>C. V :: the authors know
6. SHIFT that	>V. C :: that >C. V :: the authors know
7. MERGE _» C	>V. V :: the authors know that
8. SHIFT the, critics	N :: critics >N. D :: the >V. V :: the authors know that
9. MERGE _{>} N	D :: the critics >V. V :: the authors know that
10. SHIFT like	<D >D. V :: like D :: the critics >V. V :: the authors know that
11. MERGE _{<} D	>D. V :: the critics like >V. V :: the authors know that
12. MERGE _» V	>D. V :: the authors know that the critics like
13. SHIFT the	>N. D :: the >D. V :: the authors know that the critics like
14. MERGE _» D	>D. V :: the authors know that the critics like the
15. SHIFT book	N :: book >D. V :: the authors know that the critics like the
16. MERGE _{>} N	V :: the authors know that the critics like the book >V. R :: ϵ
17. MERGE _{>} V	R :: the authors know that the critics like the book

FIGURE 7: Incremental parsing derivation of *The authors know that the critics like the book.*

2.2.3 Inverting the Chain: MOVE

Syntactic movement is the defining characteristic of transformational grammars, and so any incremental parser for transformational grammars in general and Minimalist Grammars in particular must be able to handle movement. But why do transformational grammars use movement? The main reason is that in natural language there are expressions which are interpreted in a different position in the sentence from where they are pronounced: in (107), the phrase *which book* appears at the beginning of the question in the main clause, but it is interpreted as the direct object of the verb *like* in the embedded clause.²³

(107) Which book do the authors think that the critics like?

So one of the functions of syntactic movement is to decouple or desynchronize the derivations of the meaning and the pronounced form of an expression²⁴, which we refer to as Logical Form (LF) and Phonological Form (PF) derivations, respectively, using the terms LF and PF rather broadly to refer to the meaning and form of a linguistic expression or sign in the sense of DE SAUSSURE 1916, and not in their narrow syntactic sense.

When parsing a sentence with movement, a moved phrase needs to be parsed as if it was occurring in one position (its base position), although in the actual input string it occurs in a different position (its moved position). Since movement in Minimalism is always to the left, an incremental processor will first encounter a moved element in its moved position and later has to process the moved element again in its base position. But this is exactly the inverse of regular bottom-up movement, where a moved element is first merged in its base position and then has to be processed again in its moved position.

²³ Arguably, the moved phrase also contributes to the meaning of the question, but we leave this meaning contribution aside for the moment. See Section 4.4 for a more complete picture of the semantics of movement.

²⁴ Taken to its extreme this ‘functional’ explanation of syntactic movement implies that there should be no syntactic movement without some form of reconstruction.

So we can handle Minimalist Grammar movement incrementally by using its existing mechanism of movers, and literally inverting the movement chain.

As an illustration of the idea let's look at a bottom-up derivation of (108) after the verb *like* merged with the direct object *which book*.

(108) Which book do the critics like?

Since the direct object is a mover and will be pronounced somewhere higher in the tree, the expression is written as

(109) <D. V :: like ϵ , -wh :: which book

with the mover -wh :: which book. The derivation then continues and the first part of the expression in (109) 'grows', while the mover stays in place until the complementizer *do* is merged and we have

(110) +wh. C :: do the critics like ϵ , -wh :: which book

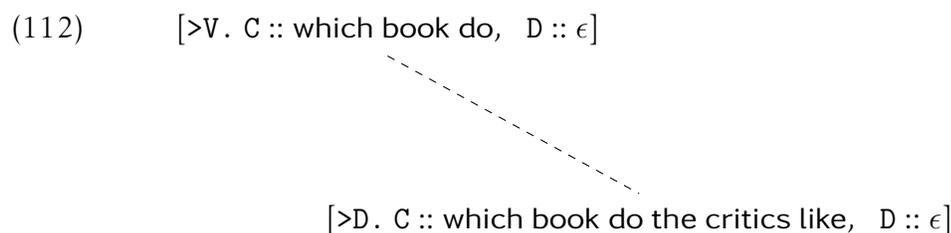
In the next step the mover -wh :: which book is discharged via MOVE, and so the expression in (110) marks the end of the movement chain.

Abstracting away all intermediate steps a movement chain can be reduced to the expressions at the top and the bottom of the chain linked together as shown in (111) for the bottom-up chain just discussed.

(111) [+wh. C :: do the critics like ϵ , -wh :: which book]

[<D. V :: like, -wh :: which book]

Now compare the movement chain in (111) to the slightly different, so far imaginary chain shown in (112): in both cases, the top and the bottom of the chain consist of an expression, which is a sequence of two basic expressions. The common interpretation is that the first element in such a sequence is the ‘main’ expression and the others are movers.



The mover in (111) is as expected, it carries a licensee feature $-wh$ which is to be checked by MOVE. But the ‘mover’ in (112) does not have a licensee feature, instead it only has a selectee (or category) feature D , which is checked by MERGE and not MOVE. Setting aside this issue (and remembering that the distinction between selection and licensing features is not a necessary one), we notice that in both (111) and (112) there is a feature match between the mover and the main expression: in (111) the match is at the top of the chain ($+wh$ matches $-wh$), while in (112) the match is at the bottom of the chain ($>D$ matches D).

So in a sense the movement chains in (111) and (112) are inverses of each other: the chain in (111) ‘ends’ at the top because that is where the mover and the main expression have a feature match. By this same logic, the chain in (112) ends at the bottom, because the mover feature matches the main expression in the bottom expression. The contrast between the two types of chains is illustrated a bit more pictorially in Figure 8 and Figure 9, where the continuation of the derivation is indicated by a solid arrow, leading to the same simple expression in both cases.

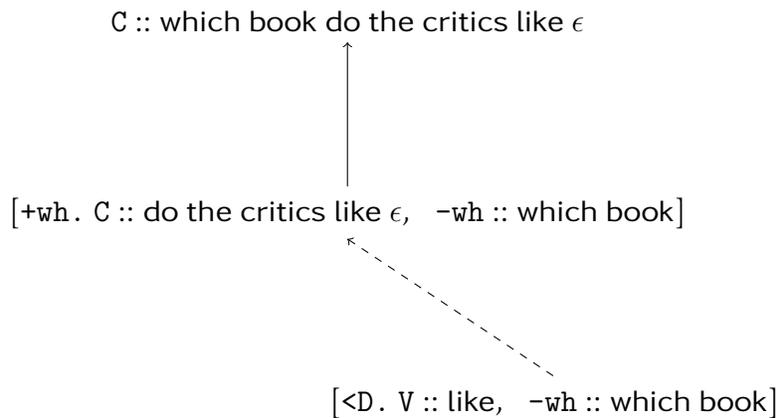


FIGURE 8: A bottom-up movement chain

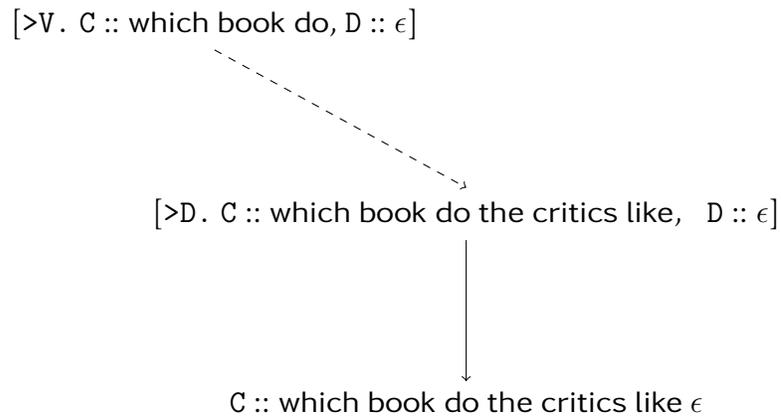


FIGURE 9: An inverse movement chain

How can the (so far imaginary) inverse movement chain in Figure 9 be made real? The basic idea is that during incremental structure building a moved phrase is ‘inserted’ into the top position of the chain (the position it is encountered in in the input), and then it ‘moves’ to its base position, where it gets merged. But neither of these operations, ‘inserting’ a mover into the top of the chain and merging a mover in its base position, are possible according to the structure building operations defined so far.

Merging a mover into its base position may seem rather straight-forward, the mover and the main expression have a matching selection feature, so the main expression selects

the mover and the two merge as in (113). But now we realize that the argument of MERGE in (113) is a single complex expression, making MERGE a unary operation, which is unusual since MERGE is conceptualized as a binary operation.

(113) $\text{MERGE}([\text{>D}. \text{C} :: \text{do the critics like}, \text{D} :: \epsilon]) \longrightarrow \text{C} :: \text{do the critics like}$

But MERGE in (113) is not really unary, it just operates internally of a single expression. So it is a form of ‘internal MERGE’. But internal MERGE is typically used as a synonym for MOVE.

The second challenge is how to ‘insert’ an element like the one in (114) into the top of a chain and turn it into a mover. The top of the chain for (114) is licensed by the complementizer do in (115).

(114) $-\text{wh D} :: \text{which book}$

(115) $+\text{wh >V}. \text{C} :: \text{do}$

So ‘inserting’ (114) into the top of the chain means combining (114) and (115) to get the expression in (116). But since the expressions in (114) and (115) have a matching licensing feature *wh* and licensing feature is checked when combining them into (116), the operation doing the combining must be an instance of MOVE as in (117).

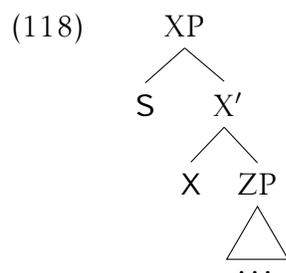
(116) $>\text{V}. \text{C} :: \text{which book do}, \text{D} :: \epsilon$

(117) $\text{MOVE}(+\text{wh >V}. \text{C} :: \text{do}, -\text{wh D} :: \text{which book}) \longrightarrow >\text{V}. \text{C} :: \text{which book do}, \text{D} :: \epsilon$

In (117) MOVE takes two independent arguments, which makes it a binary operation, while traditionally MOVE is conceptualized as a unary operation. But in some sense, MOVE always takes two arguments, the mover and the main expression, they are just parts of the

same complex expression, which is the reason why *MOVE* is also referred to as internal *MERGE*. So by analogy (and the exclusion principle), the instance of *MOVE* in (117) may be called external *MOVE*, since external *MERGE* is already taken.

From the perspective of incremental parsing, the Minimalist structure building operations *MERGE* and *MOVE* are not as clearly differentiated as they seem to be from a bottom-up perspective. But even when building a structure bottom-up, *MERGE* and *MOVE* can create isomorphic derived trees.



The specifier *S* of the *XP* in (118) can have come into its position in two different ways:

1. *S* is selected by *X* and enters into *Spec,XP* via *MERGE*
2. *S* is merged as a mover within *ZP*, then licensed by *X* and enters into *Spec,XP* via *MOVE*.

When building a tree bottom-up, these two options are clearly distinct, because in the second case *S* must have left a trace somewhere inside *ZP*, while in the first case there is no trace. But from an incremental perspective these two options are identical, at least as far as the PF exponent of *S* is concerned: *S* is pronounced in the specifier of *XP* whether it got there via *MERGE* or *MOVE*²⁵.

²⁵ But if *S* got inserted via *MOVE* some part of its LF may need to reconstruct into its base position via *MERGE*.

So the task of integrating an incoming lexical item into an existing derivation can be performed by either *MERGE* or *MOVE*, and when first integrating an incoming element, *MOVE*, like *MERGE*, is a binary (or external) operation: it takes an incoming moved element *S* and the tree containing the landing site of *S*, and in essence merges them together into one tree. Conversely, *MERGE* can be a unary (or internal) operation when a mover gets merged into the bottom of a chain, as seen in (113).

Formal definitions of external *MOVE_E* and internal *MERGE_I* are given in (119) and (120), respectively, assuming the variables and conventions from Figure 3.

$$(119) \quad \text{MOVE}_E([+\mathbf{g} \gamma :: s, \vec{\phi}], [-\mathbf{g} \delta :: t, \vec{\psi}]) \longrightarrow [\gamma :: ts, \delta :: \epsilon, \vec{\phi}, \vec{\psi}]$$

$$(120) \quad \begin{array}{l} \text{a. } \text{MERGE}_I^>([\mathbf{>f} \gamma :: s, \vec{\phi}, \mathbf{f} :: \epsilon, \vec{\psi}]) \longrightarrow [\gamma :: s, \vec{\phi}, \vec{\psi}] \\ \text{b. } \text{MERGE}_I^<([\mathbf{<f} \gamma :: s, \vec{\phi}, \mathbf{f} :: \epsilon, \vec{\psi}]) \longrightarrow [\gamma :: s, \vec{\phi}, \vec{\psi}] \end{array}$$

In (119), an item is initially inserted and becomes a mover (binary or External *MOVE_E*); in (120), a mover from within is selected and merged (unary or Internal *MERGE_I*). Together, the operations *MERGE_I* and *MOVE_E* perform the same functions as their bottom-up counterparts *MERGE^M* and *MOVE^T*: they both build movement chains. But *MERGE_I* and *MOVE_E* build inverted movement chains: the moved element enters a derivation and gets integrated into its final landing position position via *MOVE_E*. There it turns into a mover within the expression it integrated into and is then available for an eventual *MERGE_I* into its base position.

Since a moved element enters a derivation in the position where it is pronounced, it ‘loses’ its PF component once it turns into a mover. This encodes the principle of *PRO-NOUNCE HIGHEST*, which does not make much sense from an incremental perspective: an item from the input only becomes a mover once it has been integrated into the derivation, and an item can only appear in the input where it is pronounced. Therefore the rules

in (119) and (120) make explicit reference to the empty string ϵ as the PF exponent of movers. This may suggest that there is a significant difference between ‘incremental’ and ‘bottom-up’ movers, but notice that in a bottom-up derivation an element may ‘lose’ some of its meaning or LF component before turning into a mover via MERGE. This issue will be discussed further in Section 2.5 and Chapter 4.

Now it is time to illustrate the principles of incremental MOVE and MERGE with an example derivation, in which we use the lexical items specified in (91). The first sentence to be parsed is the *wh*-question in (121).

(121) Which book the critics like.

We will do the parsing derivation step by step, with comments interspersed. The full derivation without comments is shown in Figure 10. At the beginning, the parse stack contains just the start item $>C$. R. Starting at the beginning of (121), we shift the first two items *which* and *book* onto the stack.

(122) SHIFT *which*, *book*

N :: *book*
>N. -*wh* D :: *which*
>C. R :: ϵ

They have a matching feature N, so we can apply MERGE_> and push the result back onto the stack.

(123) MERGE_> N

-*wh* D :: *which book*
>C. R :: ϵ

Since there are no matching features, all we can do is shift the next item *do* in (124). So far the derivation should be familiar.

(124) *SHIFT do*

 +*wh* >*V*. *C* :: *do*

 -*wh* *D* :: *which book*

 >*C*. *R* :: ϵ

Now the top items on the stack have a matching licensing feature *wh*, so in (125) we can apply external *MOVE_E*.

(125) *MOVE_E wh*

 >*V*. *C* :: *which book do*, *D* :: ϵ

 >*C*. *R* :: ϵ

As an instance of *MOVE*, the incremental *MOVE_E* checks and deletes a licensing feature, in this case *wh*. It then inserts the element carrying the licensee feature (i.e. -*wh* *D* :: *which book*) into its landing position at the top of a new movement chain and turns it into a mover. Insertion into the landing position at the top of the chain means that the PF exponents of the licensee and the licensor are concatenated to form the PF exponent of the resulting expression >*V*. *C* :: *which book do*, and then the remaining part of the licensee (mostly its remaining *D* feature) becomes a mover within that expression. Turning the remaining part of the moved expression (or ‘filler’) *which book* into the mover *D* :: ϵ within the larger expression [>*V*. *C* :: *which book do*, *D* :: ϵ] bears some similarity to putting the filler into a ‘hold cell’ or similar storage, an idea repeatedly suggested in the psycholinguistic sentence processing literature (KAPLAN 1972, 1974, WANNER & MARATSOS 1978).

So both the bottom-up MOVE^T and the incremental MOVE_E concatenate two PF exponents, but incremental MOVE_E takes those from two separate expressions, like prototypical MERGE , while bottom-up MOVE^T gets them from within a single expression. Aside from the concatenated PF component of the resulting expression, incremental MOVE_E behaves mostly like bottom-up MERGE^M , which merges an expression with a future mover: like MERGE^M it is a binary operation and the resulting expression has one more mover than its arguments together. Since movement chains are inherently directed, a mover moves from start to end, we can say that both MERGE^M and MOVE_E create the ‘start’ of a movement chain, but in the incremental case the mover moves ‘downward’ in the tree, while in the bottom-up case the mover moves upward.

Next in the derivation, there is once again nothing to reduce, so we shift the next two items the and critics, which have a matching N feature and can be merged as in (126).

(126) SHIFT the, critics

	N :: critics
>N.	D :: the
>V.	C :: which book do, D :: ϵ
>C.	R :: ϵ
$\text{MERGE}_{>N}$	
	D :: the critics
>V.	C :: which book do, D :: ϵ
>C.	R :: ϵ

And so we shift again, this time the verb like, which selects the subject DP on top of the stack and can thus be merged with it.

(127) SHIFT like

<D >D. V :: like

D :: the critics

>V. C :: which book do, D :: ϵ >C. R :: ϵ MERGE_< D

>D. V :: the critics like

>V. C :: which book do, D :: ϵ >C. R :: ϵ

At the point in the derivation shown in (127), subject and verb again form a constituent to the exclusion of the direct object. And since the direct object is moved, this aspect of the constituent structure is not changing any more, at least not visibly. The final change to the parsed string occurs in the next step, where the ‘CP expression’ and the ‘VP expression’ on the stack merge by checking the V selection feature as in (128).

(128) MERGE_» V>D. C :: which book do the critics like, D :: ϵ >C. R :: ϵ

But after (128) the derivation is not yet completed, the expression on the stack still has an unchecked selector feature >D, and it contains a mover of category D. So we can apply internal MERGE_̂ to merge the mover into its base position.

(129) MERGE_I D

C :: which book do the critics like

>C. R :: ϵ

MERGE_> V

R :: which book do the critics like

It is worth comparing this last step in (129) with how the chain was started in (125), and together compare them to a corresponding bottom-up chain. The operation at the bottom of a chain is always MERGE, partly because of the bottom-up bias of Minimalist syntax, but also because MERGE is triggered by selection (or thematic) requirements, which in turn are associated with interpretation: a mover is primarily interpreted at the bottom of its chain. In the bottom-up case, a chain is always started by MERGE and ended by MOVE, while in the incremental case, a chain is always started by MOVE and ended by MERGE. But starting a chain is a binary or external operation, while ending a chain is a unary or internal operation, with the mover serving as the second argument.

1. SHIFT which, book
 - N :: book
 - >N. -wh D :: which
 - >C. R :: ϵ
2. MERGE_> N
 - wh D :: which book
 - >C. R :: ϵ
3. SHIFT do
 - +wh >V. C :: do
 - wh D :: which book
 - >C. R :: ϵ
4. MOVE_E wh
 - >V. C :: which book do, D :: ϵ
 - >C. R :: ϵ
5. SHIFT the, critics
 - N :: critics
 - >N. D :: the
 - >V. C :: which book do, D :: ϵ
 - >C. R :: ϵ
6. MERGE_> N
 - D :: the critics
 - >V. C :: which book do, D :: ϵ
 - >C. R :: ϵ
7. SHIFT like
 - <D >D. V :: like
 - D :: the critics
 - >V. C :: which book do, D :: ϵ
 - >C. R :: ϵ
8. MERGE_< D
 - >D. V :: the critics like
 - >V. C :: which book do, D :: ϵ
 - >C. R :: ϵ
9. MERGE_» V
 - >D. C :: which book do the critics like, D :: ϵ
 - >C. R :: ϵ
10. MERGE_I[>] D
 - C :: which book do the critics like
 - >C. R :: ϵ
11. MERGE_> C
 - R :: which book do the critics like

FIGURE 10: Incremental derivation with movement of *Which book do the critics like?*

2.2.3.1 Moved Specifiers

In our examples so far, moved elements started out from (or ended up in) complement positions, but it is also possible to move elements, which were base-generated in specifier positions: (130) shows constituent questions whose *wh*-phrases are the subject of the main clause and the embedded clause, respectively, and so are moved from a specifier position.

- (130) a. Which critics like the book?
 b. Which critics do the authors think like the book?

So far we also assumed that the specifier position where the subject *wh*-phrases in (130) move from is the specifier of VP, but that is an overly simplistic assumption. More realistically, an English clause can contain tense information expressed as an auxiliary like *will* in (131). There we see that the subject DP *the critics* linearly precedes the auxiliary.

- (131) The critics will like the book.

So if the auxiliary *will* is the head of a tense phrase (TP), then in (131) the subject DP must be in the specifier of TP. Subjects are still assumed to be merged within the VP (or vP) initially, but then they move from a VP-internal position to the specifier of TP (Spec,TP). The movement of the subject to Spec,TP is licensed by case: the subject DP moves to receive or check its nominative case marking. Inspired by subject-verb agreement like in (132), nominative case is assumed to be checked/assigned by the same projection which also checks/assigns verbal finiteness, typically the TP.

- (132) a. The critics like the book.
 b. The critic likes the book.

Non-finite clauses like the embedded infinitive in (133) cannot host a subject, but with a raising verb like *seem*, the embedded subject can raise to the matrix level.

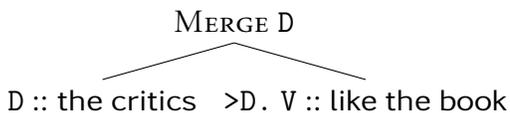
(133) The critics seem to like the book.

However, as it stands, our parsing procedure cannot actually handle moved subjects, or moved specifiers in general. The problem is a combination of at least two of our assumptions: first, a head must merge with any specifier it selects, before it can itself be selected and merged as an incomplete item. Second, once they have become movers, moved elements are only ‘visible’ for selection within their own expression (or tree), i.e. a mover can only be selected and merged by its main expression: an inverted chain is ended by internal MERGE.

Of these two assumptions, the second one is more fundamental and deserves some discussion: it says that elements on the parse stack are assembled independently (in different ‘workspaces’) and movers of one element are not visible from other elements. This assumption is built into our notation of MG expressions and also applies in bottom-up structure building. One particularly clear case is subjects: in the bottom-up derivation of (134) in Figure 4 we saw that the subject DP and the VP were assembled independently²⁶ and only merged when they were both completed, as shown in (135).

(134) The critics like the book.

(135)



MERGE D

D :: the critics >D. V :: like the book

²⁶ A standard bottom-up derivation is typically shown so that the VP is assembled first and then the subject DP, but this is not necessary, the derivation in Figure 4 could have started with building the subject.

This means that a mover within the VP cannot be licensed from within the subject. It is not easy to imagine a situation where this restriction could possibly apply, but (136) may be an example.

(136) *That the journalists asked [which critics]₁ the authors read annoyed *t*₁?

If movers within the VP were accessible from within the subject, in the derivation of (136), the mover *which critics* within the matrix VP could be licensed by the +wh feature on the embedded interrogative CP within the clausal subject and move there. But obviously (136) is ungrammatical.

To see how these two assumptions together do not allow us to parse moved specifiers, consider the following derivation of the question in (137). We assume that subject *wh*-questions are like other *wh*-questions and involve movement of the *wh*-phrase to the specifier of CP, despite the fact that there is no *do*-support, rendering CP a functional projection with an empty head. As in previous derivations, we will not include a TP projection, instead we assume that the subject *wh*-phrase moves from Spec,VP directly to Spec,CP. So we only need one additional lexical entry, the empty interrogative complementizer in (138); there and in the remainder of the chapter, the PF exponent of empty functional heads are replaced by their common category symbol in italics.

(137) Which critics like the book?

(138) +wh >V. C :: *C*_{wh}

Again we present the parsing derivation step by step (the full derivation is shown in Figure 11). At the beginning, the parse stack contains just the start item >C. R, so we shift the first two items *which* and *critics* onto the stack and merge them.

(139) SHIFT *which, critics*

N :: critics
 >N. -*wh* D :: which
 >C. R

MERGE_> N

-*wh* D :: which critics
 >C. R :: ϵ

Next we shift the empty interrogative complementizer C_{wh} and move the moved phrase *which critics* into its specifier as in (140), turning remaining parts of the moved phrase (i.e. its category feature and its LF) into a mover within the resulting CP expression.

(140) SHIFT C_{wh}

+*wh* >V. C :: C_{wh}
 -*wh* D :: which critics
 >C. R

MOVE_E *wh*

>V. C :: which critics C_{wh} , D :: ϵ
 >C. R :: ϵ

Now there is only one non-start item on the stack, and we have to shift again. This time the verb *like*. At this point, we have two expressions on the stack in (141), which cannot be combined without further assumptions (and shifting doesn't help either): the expression <D >D. V :: *like* needs to merge something of category D as its specifier before it can be selected as V and merged with the expression [>V. R :: which critics C_{wh} , D :: ϵ] via MERGE_{>>}. But the specifier (i.e. subject) of the verb *like* is a mover within the CP expression, the

remaining parts of the *wh*-phrase *which critics*. Since movers can only be merged within their own tree, the moved subject is not available for selection by the verb *like*.

(141) SHIFT like

<D >D. V :: like
 >V. C :: which critics C_{wh} , D :: ϵ
 >C. R :: ϵ

As discussed above, encapsulating movers within their own expression is a fundamental assumption of transformational grammar²⁷, so we should first try to loosen our restriction on merging incomplete items. The reason why we allow incomplete items to be merged only after they have merged with their specifier is that specifiers linearly precede their heads, and so if a head *H* selects a specifier *S*, the specifier *S* must have been processed already and thus be on the parse stack. So our restriction on merging incomplete items is more a factual statement about the linear order of specifiers and heads than a substantial constraint²⁸ on the application of MERGE: an element *S* must be processed before a head *H* if *S* is selected as the specifier of *H*. And indeed the specifier (or subject) selected by *like* has been processed in (141), it is just not an independent item on the stack any more, because it has been turned into a mover within another expression.

So we can allow an incomplete item *H* with a missing specifier to be merged if its selected specifier *S* has already been processed, i.e. if the selected specifier *S* is a mover within the expression *H* merges with. I will assume that the two operations—merging the mover *S* and merging the expression *H*—are somewhat interdependent and write them as one parsing step with two operations as in (142), notationally connected by a + sign.

²⁷ Even though this assumption is rarely made explicit.

²⁸ The substantial aspect of the constraint is that only incomplete expressions can be merged and not predictions of expressions. And since all MG expressions are rooted in a head, the constraint expresses the head-corner principle.

- (142) $\text{MERGE}_I^< D + \text{MERGE}_{II}^>> V$
- >D. C :: which critics C_{wh} like
 - >C. R :: ϵ

But nothing hinges on this assumption, all we need is to allow incomplete expressions with missing specifiers to be merged. If the selection requirement for the specifier cannot be met immediately after the incomplete item is merged, the derivation crashes because there is no way the specifier can be filled in later steps. After this successful double MERGE step the derivation can proceed without further surprises, as shown in (143).

- (143) SHIFT the
- >N. D :: the
 - >D. C :: which critics C_{wh} like
 - >C. R :: ϵ
- MERGE_> D
- >D. C :: which critics C_{wh} like the
 - >C. R :: ϵ
- SHIFT book
- N :: book
 - >N. C :: which critics C_{wh} like the
 - >C. R :: ϵ
- MERGE_> N
- C :: which critics C_{wh} like the book
 - >C. R :: ϵ
- MERGE_> C
- R :: which critics C_{wh} like the book

This treatment of moved specifiers may be generalized to expressions selecting more than one specifier: all moved specifiers will need to have been processed and thus be movers already, before they can be merged into their specifier base position, possibly together with additional unmoved specifiers which were shifted from the input onto the stack. Working out the details and verifying that there are no devils hiding in these details is left for future research.

1. SHIFT *which, critics*
 - N :: critics
 - >N. -*wh* D :: *which*
 - >C. R :: ϵ
2. MERGE_> N
 - wh* D :: *which critics*
 - >C. R :: ϵ
3. SHIFT C_{wh}
 - +*wh* >V. C :: C_{wh}
 - wh* D :: *which critics*
 - >C. R :: ϵ
4. MOVE_E *wh*
 - >V. C :: *which critics* C_{wh} , D :: ϵ
 - >C. R :: ϵ
5. SHIFT *like*
 - <D >D. V :: *like*
 - >V. C :: *which critics* C_{wh} , D :: ϵ
 - >C. R :: ϵ
6. MERGE_I[<] D + MERGE_E^{>>} V
 - >D. C :: *which critics* C_{wh} *like*
 - >C. R :: ϵ
7. SHIFT *the*
 - >N. D :: *the*
 - >D. C :: *which critics* C_{wh} *like*
 - >C. R :: ϵ
8. MERGE_> D
 - >D. C :: *which critics* C_{wh} *like the*
 - >C. R :: ϵ
9. SHIFT *book*
 - N :: *book*
 - >N. C :: *which critics* C_{wh} *like the*
 - >C. R :: ϵ
10. MERGE_> N
 - C :: *which critics* C_{wh} *like the book*
 - >C. R :: ϵ
11. MERGE_> C
 - R :: *which critics* C_{wh} *like the book*

FIGURE 11: Incremental derivation with moved specifier: *Which critics like the book?*

2.2.3.2 Subjects and Nominative

With a mechanism to move specifiers in place, we can expand our set of features and lexical items to make our grammar more empirically accurate. As discussed above, English subjects are pronounced in Spec,TP, but since subjects are still assumed to be merged within the VP (or vP), they move from a VP-internal position to Spec,TP for case reasons: the subject DP moves from VP to Spec,TP to receive or check its nominative case marking.

In Minimalism, movement is strictly feature-driven and so there must be a licensing feature for case, which in Minimalist Grammars is often written as **+k**. With this the lexical entry for the tense auxiliary *will* can be specified as

(144) >V **+k**. T :: will

But as the contrast in (145) shows, Spec,TP does not attract DPs with an arbitrary case as the generic **+k** might suggest, instead Spec,TP can only host nominative case.

- (145) a. They will like the book.
 b. *Them will like the book.

And since the pronoun *they* in (145) is inserted into the derivation with nominative case, it makes sense to give it a nominative licensee feature **-nom** (cf. GRAF 2013), and correspondingly restrict *will* to only license nominative case, as shown in (146) and (147).

(146) D **-nom** :: they

(147) >V **+nom**. T :: will

For the moment, we assume that there is no corresponding feature for accusative case, instead morphological accusative case is treated as the default case (SCHÜTZE 2001), in

which a DP appears if it carries no case feature²⁹. Since an adequate treatment of accusative checking requires a form of head movement, we postpone adding accusative case to the set of structurally checked cases until Section 3.1.2. There we assume that case is a necessary feature for being a DP.

With all the necessary features and lexical items in place we can derive (148) in a more empirically accurate manner. The derivation does not involve any new concepts or operations and is shown in Figure 12.

(148) They will like the book.

Finally, we can combine *wh*-movement and case-driven A-movement and make one feed the other as in

(149) Which critics will seem to like the book?

To derive this sentence we need the new lexical items listed in (150): we follow common practice and analyze infinitival *to* as a tense head.³⁰

- (150) a. >T +nom. V :: seem
 b. >V. T :: to

²⁹ Or if the DP has an unchecked case feature. This option is only available if we allow that licensee features can optionally remain unchecked. In bottom-up structure building, leaving licensee features unchecked is not really an option, as it would leave the corresponding mover hanging and not be integrated.

³⁰ Comparing the relative surface positions of the tense auxiliary *will* and infinitival *to* with respect to negation shows that they cannot be of the same category. We follow common practice and ignore this subtlety.

- (i) a. The critics will not have read the book.
 b. The critics seem to not have read the book.
- (ii) a. *The critics not will have read the book.
 b. The critics seem not to have read the book.

With this the derivation of (149) proceeds as shown in Figure 12. Notice how this derivation involves an intermediate movement step, which is an instance of the purely feature-checking operation Move_C .

1. SHIFT which, critics
 $N :: \text{critics}$
 $>N. -\text{wh} -\text{nom} D :: \text{which}$
2. MERGE $_>$ N
 $-\text{wh} -\text{nom} D :: \text{which critics}$
3. SHIFT C_{wh}
 $+\text{wh} >T. C :: C_{wh}$
 $-\text{wh} -\text{nom} D :: \text{which critics}$
4. MOVE $_E$ wh
 $>T. C :: \text{which critics } C_{wh}, -\text{nom} D :: \epsilon$
5. SHIFT T
 $+\text{nom} >V. T :: \text{will}$
 $>T. C :: \text{which critics } C_{wh}, D :: \epsilon$
 $>C. R :: \epsilon$
6. MOVE $_C$ nom + MERGE $_{\gg}$ T
 $>V. C :: \text{which critics } C_{wh} \text{ will}, D :: \epsilon$
7. SHIFT seem
 $>T. V :: \text{seem}$
 $>V. C :: \text{which critics } C_{wh} \text{ will}, D :: \epsilon$
8. MERGE $_{\gg}$ V
 $>T. C :: \text{which critics } C_{wh} \text{ will seem}, D :: \epsilon$
9. SHIFT to
 $>V. T :: \text{to}$
 $>T. C :: \text{which critics } C_{wh} \text{ will seem}, D :: \epsilon$
10. MERGE $_{\gg}$ T
 $>V. C :: \text{which critics } C_{wh} \text{ will seem to}, D :: \epsilon$
11. SHIFT like
 $<D >D. V :: \text{like}$
 $>V. C :: \text{which critics } C_{wh} \text{ will seem to}, D :: \epsilon$
12. MERGE $_I^<$ D + MERGE $_{\gg}$ V
 $>D. C :: \text{which critics } C_{wh} \text{ will seem to}$
13. SHIFT the
 $>N. D :: \text{the}$
 $>D. C :: \text{which critics } C_{wh} \text{ will seem to like}$
14. MERGE $_>$ D
 $>N. C :: \text{which critics } C_{wh} \text{ will seem to like the}$
15. SHIFT book
 $N :: \text{book}$
 $>N. C :: \text{which critics } C_{wh} \text{ will seem to like the}$
16. MERGE $_>$ N
 $C :: \text{which critics } C_{wh} \text{ will seem to like the book}$
 $>C. R :: \epsilon$
17. MERGE $_>$ C
 $R :: \text{which critics } C_{wh} \text{ will seem to like the book}$

FIGURE 12: Incremental derivation of *Which critics will seem to like the book*

2.2.4 Moving Incomplete Items: Remnant Movement

Without remnant movement, Minimalist Grammars are equivalent to context-free grammars (KOBELE 2010). And since human languages are most succinctly described by grammars that are not quite context free, it is important to see if the proposed parser can handle remnant movement, especially since remnant movement has been a problem for the first attempt at building a left-corner parser for Minimalist Grammars (HUNTER 2019).

Looking at remnant movement will also make us expand the number of operations which can work on incomplete items: since a remnant is by definition an incomplete item, we must allow incomplete items to be integrated via *MOVE* and to be passed along the inverse movement chain and finally merged.

A remnant is a phrase that another phrase has moved out of, and remnant movement is the movement of such a remnant phrase. Under a standard analysis of passive, the direct object of a transitive verb moves out of the VP into the canonical subject position. If after that the remnant VP itself moves, we get a sentence like (151), which is an instance of remnant movement.

(151) Admired, John was.

The critical insight for our analysis of remnant movement is that the fronted VP in (151) is an incomplete item, it is lacking a direct object, just like any instance of a transitive verb. And so we may account for examples like (151), by allowing incomplete items to be moved, just like we allow similar incomplete items to be merged.

In order to derive (151), we need to assume that subjects move out of their base position to check nominative case in *Spec,TP*, so *John* carries a nominative licensee feature *-nom* and *was* carries the corresponding licenser *+nom*. The VP-fronting is achieved by a

licensee feature $-f$ on the verb and a corresponding licenser $+f$ on the (almost³¹) empty complementizer. So we need the following lexical items

- (152) D $-nom$:: john $+nom$ $>V$. T :: was
 $>D$. $-f$ V :: admired $+f$ $>T$. C :: C_f

With these lexical items a derivation of (151) could proceed in the following way (the complete derivation is shown in Figure 13): first we shift the fronted verb *admired* and the empty complementizer. And since they have a matching f feature, we combine them via the forward variant of external $MOVE_E^{\gg}$ as shown in (153). This is the first critical step: *admired* still has an open selection feature $>D$, but like in the case of forward $MERGE_{\gg}$ we allow $MOVE_E^{\gg}$ to apply and ‘skip’ over the open selection feature. The result is an expression, which contains a mover with an open selection feature: $>D$. V :: ϵ , or a remnant mover.

- (153) SHIFT *admired*
 $>D$. $-f$ V :: *admired*
 $>C$. R :: ϵ
 SHIFT C_f
 $+f$ $>T$. C :: C_f
 $>D$. $-f$ V :: *admired*
 $>C$. R :: ϵ
 $MOVE_E^{\gg}$ f
 $>T$. C :: *admired* C_f , $>D$. V :: ϵ
 $>C$. R :: ϵ

³¹ Mind the comma!

Next, we shift the subject *john* and the tense head *was*, and combine them via MOVE_E based on their matching *nom* feature. The so formed TP can combine with the CP via MERGE_{\gg} checking their matching T feature. This part of the derivation is shown in (154).

(154) SHIFT *john*

-nom D :: *john*
 >T. C :: *admired C_f*, >D. V :: ϵ
 >C. R :: ϵ

SHIFT *was*

+nom >V. T :: *was*
 -nom D :: *john*
 >T. C :: *admired C_f*, >D. V :: ϵ
 >C. R :: ϵ

MOVE_E *nom*

>V. T :: *john was*, D :: ϵ
 >T. C :: *admired C_f*, >D. V :: ϵ
 >C. R :: ϵ

1. MERGE_{\gg} T

>V. C :: *admired C_f john was*, >D. V :: ϵ , D :: ϵ
 >C. R :: ϵ

The derivation concludes as shown in (155), which includes the second critical step: our remnant mover >D. V :: ϵ needs to be merged into its base position, but for this to be possible we need the forward variant of internal MERGE_I^{\gg} . After that we can satisfy the open selection feature of the moved remnant by merging the mover of the subject.

- (155) $\text{MERGE}_I^{\gg} V$
 >D. C :: admired C_f john was, D :: ϵ
 >C. R :: ϵ
- $\text{MERGE}_I^{\gg} D$
 C :: admired C_f john was
 >C. R :: ϵ
- $\text{MERGE}_> C$
 R :: admired C_f john was

For the derivation above to go through, we needed two more operations to work on incomplete items: external MOVE_E^{\gg} to integrate the fronted, but incomplete VP (it is missing its complement), and internal MERGE_I^{\gg} to merge the incomplete (remnant) mover into its base position, where its missing selected elements can be filled in. These operations are defined as in (156) and (157).

$$(156) \quad \text{MOVE}_E^{\gg}([+g \gamma :: s, \vec{\phi}], [>x^? -g \delta :: t, \vec{\psi}]) \longrightarrow [\gamma :: ts, >x^? \delta :: \epsilon, \vec{\phi}, \vec{\psi}]$$

$$(157) \quad \text{MERGE}_I^{\gg}([=f \gamma :: s, \vec{\phi}, >x^? f :: \epsilon, \vec{\psi}]) \rightarrow [>x^? \gamma :: s, \vec{\phi}, \vec{\psi}]$$

Incremental structure building is based on the idea of ‘skipping over’ one attractor feature. This idea makes immediate sense for $\text{MERGE}_>$. Incorporating remnant movement helps us make sense of the ‘skipping versions’ of MERGE_I and MOVE_E , and by extension also MOVE_C if the remnant mover goes through an intermediate landing site.

1. SHIFT admired
 - >D. -f V :: admired
 - >C. R :: ϵ
2. SHIFT C_f
 - +f >T. C :: C_f
 - >D. -f V :: admired
 - >C. R :: ϵ
3. MOVE \xrightarrow{E} f
 - >T. C :: admired C_f , >D. V :: ϵ
 - >C. R :: ϵ
4. SHIFT john
 - nom D :: john
 - >T. C :: admired C_f , >D. V :: ϵ
 - >C. R :: ϵ
5. SHIFT was
 - +nom >V. T :: was
 - nom D :: john
 - >T. C :: admired C_f , >D. V :: ϵ
 - >C. R :: ϵ
6. MOVE $_E$ nom
 - >V. T :: john was, D :: ϵ
 - >T. C :: admired C_f , >D. V :: ϵ
 - >C. R :: ϵ
7. MERGE \gg T
 - >V. C :: admired C_f john was, >D. V :: ϵ , D :: ϵ
 - >C. R :: ϵ
8. MERGE \xrightarrow{I} V
 - >D. C :: admired C_f john was, D :: ϵ
 - >C. R :: ϵ
9. MERGE \xrightarrow{I} D
 - C :: admired C_f john was
 - >C. R :: ϵ
10. MERGE $>$ C
 - R :: admired C_f john was

FIGURE 13: Incremental derivation with remnant movement: *Admired, John was*

2.3 Adjunction

Adjuncts are tricky, both from the perspective of Minimalist Grammar formalization and the perspective of incremental parsing. When building structure incrementally, adjuncts are problematic, because at least in the case of adjuncts that follow their adjunction site ('right adjuncts') the relevant category feature for an adjunct to adjoin to may not be visible any more, instead the intended adjunction may already be part of a larger phrase and its category feature may be deleted. All this could be fixed if we made two assumptions: first we need an adjunction operation that can alter an existing derived tree like the type of adjunction employed in Tree-Adjoining Grammar, and second category features are persistent. With these two assumptions, the incremental adjunct problem should be easy to solve (but obviously the devil is in the details).

Here we pursue a different approach, mostly to figure out how far we can get without a dedicated (or destructive) adjunction operation. Ultimately, such an operation may still be necessary, but maybe we learn something in the meantime. The main idea to be explored is that right adjunct attachment is mostly 'semantic' as suggested in HAIDER 2004 and ERNST 2020, but we apply a rather coarse notion of semantics. We assume that there are only two types of semantic objects, which are modified by adjuncts: verbal projections (or clauses) and nominal projections. In Chapter 4 we develop a semantics in which all clausal projections from VP up to (and including) CP are of the same semantic type, they are predicates of sets of events, while all nominal projections up to (but not including) DP continue to denote predicates of individuals. So an adjunct can either modify a clause (and semantically be a predicate of events) or modify a noun phrase (and be a predicate of individuals). In situations of attachment ambiguity, choosing between the two options may be achieved by delaying the integration of an item: VP attachment is always possible, but NP attachment must happen before the NP itself is integrated as the direct object.

2.3.1 Adjuncts and Modifiers

Adjunction or modification is a common phenomenon in natural language, in many languages there are parts of speech whose main usage is as modifiers: in English, adverbs (and maybe adjectives) are mostly used as modifiers. Adjuncts are often compared to arguments: in contrast to argument adjuncts are characterized as optional and freely iterable. In (158), the two adjunct PPs are optional: they can be left out without making the sentence ungrammatical. In contrast, the argument *the bread* is not optional, as leaving it out leads to ungrammaticality (159).

- (158) a. John cut the bread with a knife in the kitchen.
 b. John cut the bread.

(159) *John cut.

Adjuncts are also iterable: we can add a second locative PP to the sentence in (158), and we can also add multiple adjectives to modify the bread in (160). But we cannot add a second argument as shown in (161).

- (160) John cut the fresh fluffy French bread with a knife in the kitchen in his Manhattan apartment.

(161) *John cut the bread the steak.

However, the distinction between adjuncts and arguments is not always as clear-cut. First, arguments can be optional as well, as shown in (162). This is easily explained by stipulating that there are two different lexical entries for the verb *eat*: a transitive one that selects an internal argument, and an intransitive one. So instead of characterizing

arguments as obligatory and adjuncts as optional, we should better characterize arguments as selected and adjuncts as unselected.

- (162) a. John ate
 b. John ate the soup.

Second we note that in practice there are limits to how many adjuncts of a similar type can be added to a given sentence. Yet this may not so much be a syntactic limit on iterability, but instead an effect of plausibility and interpretation: too many adjuncts of the same type are bound to lead to implausible or even contradictory interpretations, as in the extreme example in (163).

- (163) *Mary lives in a large tiny red green blue yellow house in London in Paris in Berlin.

So adjuncts may not always be clearly identifiable, but a prototypical one is easy to spot.

A more clear-cut empirical distinction that will become important shortly is between left adjuncts and right adjuncts. We call an adjunct a left adjunct if it linearly precedes (is left of) the phrase it modifies (164), if it linearly follows (is to the right of) the phrase it modifies the adjunct is a right adjunct (165). This distinction will turn out to be critical when parsing adjuncts incrementally: there right adjuncts pose considerable difficulties. But even from a bottom-up perspective, the distinction between right adjuncts and left adjuncts matters: left adjuncts behave essentially like functional projections along the clausal spine, with the actual adjunct sitting in either the head or the specifier position. Assuming that the adjunct is in specifier position, right adjuncts (and conjuncts) would then imply specifiers to the right.

- (164) John quickly left the house.

(165) John left the house quickly.

2.3.2 Adjuncts in Minimalist Grammar

In transformational grammar and in Minimalism with its focus on MERGE and MOVE as the only structure building operations, there seems to be little consensus on how to treat adjunction. Consequently, there have been a fair number of proposals how to handle adjunction in Minimalist Grammars. Broadly speaking, these proposals can be divided into two classes (see GRAF 2014 for an overview): either adjunction requires a new, dedicated adjunction operation and a feature checking mechanism specific to adjuncts (e.g., FREY & GÄRTNER 2002, HUNTER 2011, TORR & STABLER 2016), or adjunction can be implemented using standard selection features and MERGE. In the spirit of minimalism, we only explore the latter option.

The simplest approach to reducing adjunction to selection and MERGE is to stipulate that adjuncts select their adjunction site, but do not project (cf. DOWTY 2003) by giving them the same category as the one they select, i.e. make adjuncts category-preserving. This is the most common treatment of adjuncts in Categorical and Type-Logical Grammars and it can be implemented in MG by giving adjuncts the feature specification $=X$. X for any category feature X , e.g. the adverb *quickly* as a VP modifier would have the lexical entry

(166) $\text{>V. V} :: \text{quickly}$

It selects an expression of category V and returns an expression of category V . With this we can derive

(167) John quickly left the house

One major drawback of this simple approach is that adjuncts do not have their own category or projection, even if they are internally complex. In (168) the VP adjuncts *with great diligence* and *diligently* would both be of type =V. V, even though one is a prepositional phrase (PP) and one is an adverb.

- (168) a. The president wrote her speech with great diligence.
 b. The president wrote her speech diligently.

Another consequence of not having their own category is that it makes it hard for adjuncts to be modified by other adjuncts, which is a common phenomenon, as in (169) where the adverb *ridiculously* modifies the adjective *long*, which itself modifies the noun *speech*.

- (169) The president started her tenure with a ridiculously long speech.

These problems can be solved by introducing an empty head, which selects both the adjunct and the category of the phrase the adjunct adjoins to and then projects that same category. Following TORR & STABLER (2016), we call such an empty head an ADJUNCTIVIZER. Abstractly, the feature specification of an adjunctivizer conforms to the following schema for category features X and A.

- (170) =X =A. X :: ∅

As a concrete instantiation consider the VP adjunctivizer for VPs and adverbs

- (171) >V <Adv. V :: ∅

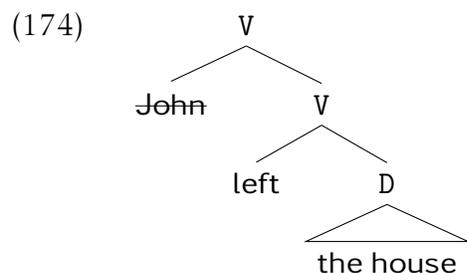
It selects an expression of category V as complement and an expression of category Adv as specifier, and returns an expression of category V.

With the adjunctivizer in (171) and the lexical entry for the adverb *quickly* in (172), we can derive the sentence in (173).

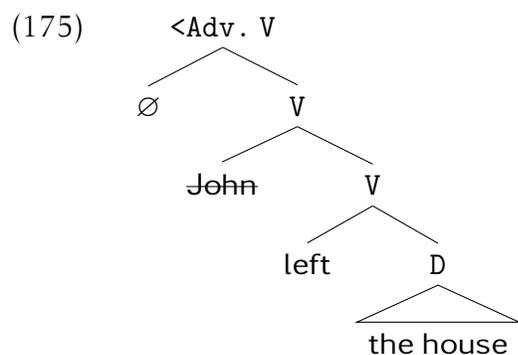
(172) Adv :: quickly

(173) John quickly left the house.

The sentence in (173) contains *quickly* as a left-adjunct, which looks almost like a functional projection along the clausal spine, except that it preserves the category of its complement. And the derivation proceeds exactly like that of a functional head with a specifier. We start with a fully formed VP, whose derived tree is shown in (174). Instead of using pointers to the head, we annotate phrases with their features.

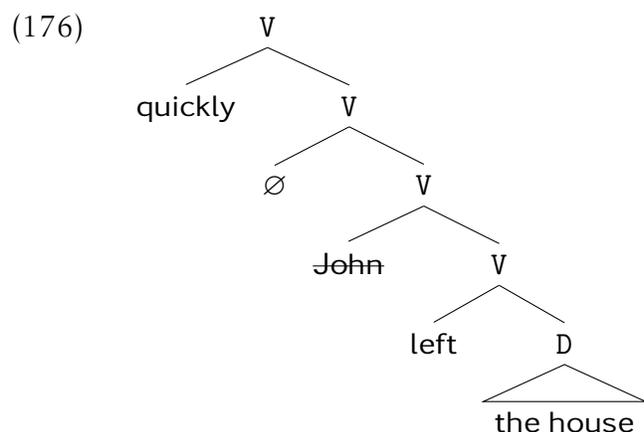


Next, we combine (174) with the adjunctivizer:



Using pointers would create a slightly confusing situation here, because the head pointer would have to point to the empty adjunctivizer head instead of the verb. This reflects the ‘dual’ nature of adjuncts (Dowty 2003): when an adjunct combines with a phrase, the adjunct (or the adjunctivizer) acts like the head, because it is the selecting one, just like a (selected) empty functional head along the clausal spine. But notice how in the chosen notation (and in the collapsed tree notation) this is not a problem.

Next we combine (175) with the adverb *quickly* to get a VP with an adverb adjoined



We can see that the adjunct itself has its own category, *Adv*. And since the adjunct has its own category, adjoining to the adjunct (i.e. modifying the adverb) is also possible with an adjunctivizer like

(177) >*Adv* <*Adv*. *Adv*

Another welcome consequence of adverbs and adjectives having their own category is that they can be selected by a lexical head other than an adjunctivizer. When an adjective is used as a predicate as in (178), we may analyze it as being selected by the copula *is*, just like the past participle is selected by an auxiliary to form the present perfect and the passive.

(178) Mary is tall.

And even adverbs can be selected: in (179) the adverb *poorly* is necessary for the intended meaning of *treated*. Without an adverb as in (180), *treated* strongly implies a medical procedure, which is not an association readily available for IRS officers.

(179) The IRS officer treated us poorly.

(180) #The IRS officer treated us.

The special meaning of *treat* in (179) can be captured if we assign it a lexical entry like (181).

(181) $\langle D \rangle \text{Adv. } V :: \text{treat}$

So we choose this approach and prefer reducing adjunction to selection and MERGE over introducing another structure building operation. Our approach may have the side effect of blurring the distinction between arguments and adjuncts (GRAF 2014): we take this as an overall welcome result, since we don't want to commit whether arguments and adjuncts should be treated differently, the same, or somewhere in between as BOUMA, MALOUF & SAG (2001) propose.

It is worth repeating that left adjuncts behave a lot like functional projections along the clausal spine. Consider the (active) voice head (182) proposed in KRATZER 1996: it is typically empty, selects the VP as its complement and the external argument as its specifier, and then projects a v(oice)P.

(182) $\langle D \rangle V. v :: \emptyset_v$

(183) $\langle \text{Adv} \rangle V. V :: \emptyset_A$

The main difference between a functional head like the active voice head in (182) and a VP adjunctivizer like (183) is that the phrase projected by the voice head is of a different category than the category it selects as its complement (v vs. V), while the adjunctivizer projects the same category as the one it selects as complement (V). And like functional projections, left adjuncts tend to be rather strictly ordered, while right adjuncts of the same type are more variable in their relative order. This led to the research project of Syntactic Cartography (CINQUE & RIZZI 2008) according to which many left adjuncts are heads of their own phrases (cf. ERNST 2004, 2020)

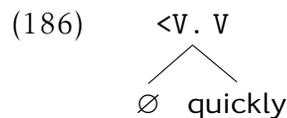
However, it is important to notice that as formulated above, the adjunctivizer as an empty head selecting the actual adjunct as its specifier only works for left adjuncts, but not for right adjuncts like the adverb *quickly* in (184): since specifiers are universally linearized to the left, right adjuncts cannot be specifiers.

(184) John left the house quickly.

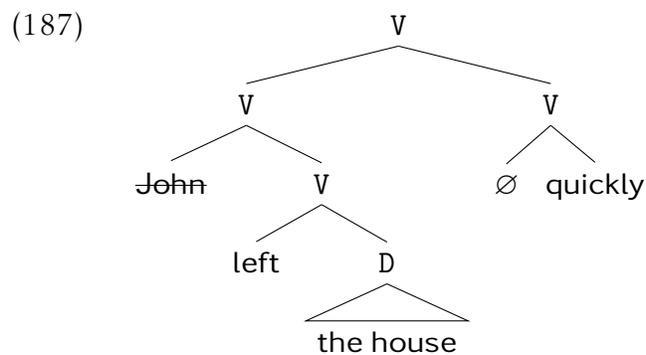
In traditional transformational grammar based on X-bar theory, the only options to make this approach work for right adjuncts are to either merge the host phrase (or adjunction site) in specifier position or to allow specifiers to the right. Neither of these options sound great. But in Minimalism (and in our version of directed MG in particular) the distinction between specifiers and complements is a bit less clear-cut and nothing bad happens if the actual adjunct is merged first and the host phrase is merged second. The main difference is that for right-adjuncts like the adverb *quickly* in (184) the order of selector features on the adjunctivizer needs to be inverted:

(185) $\text{>A <X. X :: } \emptyset$

With the adjunctivizer as defined in (185), we can now do the right-adjunct derivation of (184). We start by combining the adverb and the adjunctivizer. In the resulting tree the adverb is the complement of the empty adjunctivizer head, while the left-adjunct above was in specifier position.



Next, we combine (186) with the fully formed VP from (174) to form the expression in (187). Notice how in the collapsed tree notation, the two final expressions in (187) and (176) are equivalent except for their PF exponent, but their derived trees are rather different.



In (187), the VP host of the adverb *quickly* appears to be in specifier position, and so according to the Specifier Island Constraint, which we discuss in Chapter 5, the VP would be predicted to be an island for syntactic movement. But this prediction is obviously not borne out as examples like (188) show.

(188) Which car did you buy quickly?

In Chapter 5 we offer an explanation of certain Specifier Island effects based on the incremental parsing procedure developed in this chapter, and according to that explanation examples like (188) are not predicted to be bad. In essence, even if the host VP in (188) is a specifier, it only becomes a specifier after the adverb is merged, at which point the moved phrase will have already reconstructed into its base position. This is yet another indication that in Minimalism the distinction between complements and specifiers need not be as clear-cut as in traditional X-bar theory.

Looking at (187), we can also explain how right adjuncts of different syntactic categories can be conjoined like the adverb and prepositional phrase in (189): the adverb and the PP each combine with an adjunctivizer like the one in (185), which turns them into VP-adjuncts of category $\langle V, V \rangle$, which can then combine with the VP.

(189) The president wrote her speech slowly and with great diligence.

So adjuncts can have two types of syntactic category: their basic syntactic category like adverb (Adv) or prepositional phrase (P) as argued for based on the non-conjoined examples in (168), and their modifier category like $\langle V, V \rangle$ based on (189). Lifting an adverb from one category or (semantic) type to the other is the job of the adjunctivizer (and its semantic counterpart, which we discuss in Section 4.3)

Finally, it should be noted that adjunctivizer heads need not be empty. It may be reasonable (as TORR & STABLER 2016 do) to analyze *because* as an adjunctivizer, because *because*-clauses can host full CPs. But the clearest cases of a non-empty adjunctivizers are coordinating conjunctions like *and* and *or*, at least if we analyze coordination as a special case of adjunction, as we do in Section 2.4.

2.3.3 Incremental Adjuncts

From an incremental, or left-to-right perspective, the difference between left adjuncts and right adjuncts becomes even more pronounced. Left adjuncts can easily be integrated into the incremental picture, but right adjuncts pose a serious challenge. This challenge is not specific to our incremental MG parsing procedure, but also occurs in other grammar frameworks like Combinatory Categorical Grammar (STANOJEVIĆ, HALE & STEEDMAN 2020).

2.3.3.1 Left Adjuncts

As discussed above, left adjuncts behave a lot like functional projections along the clausal spine. Since in our previous derivations we were able to incrementally parse a clause with a TP or vP projection, we expect left adjuncts to be easy to parse incrementally. And indeed, a sentence like (190) with the VP adjunct *quickly* preceding the phrase it modifies can be parsed as shown in Figure 14.

(190) John was quickly leaving London.

2.3.3.2 Right Adjuncts

Right adjuncts are a challenge for incremental parsing. To see this consider a VP adjunct like *quickly* in a sentence like (191). The adverb *quickly* modifies a VP and so (after combining with an adjunctivizer of appropriate type) it has the feature sequence shown in (192): it selects an expression of category of V and returns an expression of category of V .

(191) John drank the beer quickly.

(192) =V. V :: quickly

1. SHIFT john
-nom D :: john
>T. R
2. SHIFT was
+nom >V. T :: was
-nom D :: john
>T. R
3. MOVE_E nom
>V. T :: john was, D :: ϵ
>T. R
4. SHIFT quickly
Adv :: quickly
>V. T :: john was, D :: ϵ
5. SHIFT \emptyset
<Adv >V. V :: \emptyset
Adv :: quickly
>V. T :: john was, D :: ϵ
6. MERGE_< Adv
>V. V :: quickly
>V. T :: john was, D :: ϵ
7. MERGE_{\gg} V
>V. T :: john was quickly, D :: ϵ
8. SHIFT leaving
<D >D. V :: leaving
>V. T :: john was quickly, D :: ϵ
9. MERGE_I D + MERGE_{\gg} V
>D. T :: john was quickly leaving
10. SHIFT london
D :: london
>D. T :: john was quickly leaving
11. MERGE_> D
T :: john was quickly leaving london
>T. R
12. MERGE_> T
R :: john was quickly leaving london

FIGURE 14: Incremental left adjunction: *John was quickly leaving London*

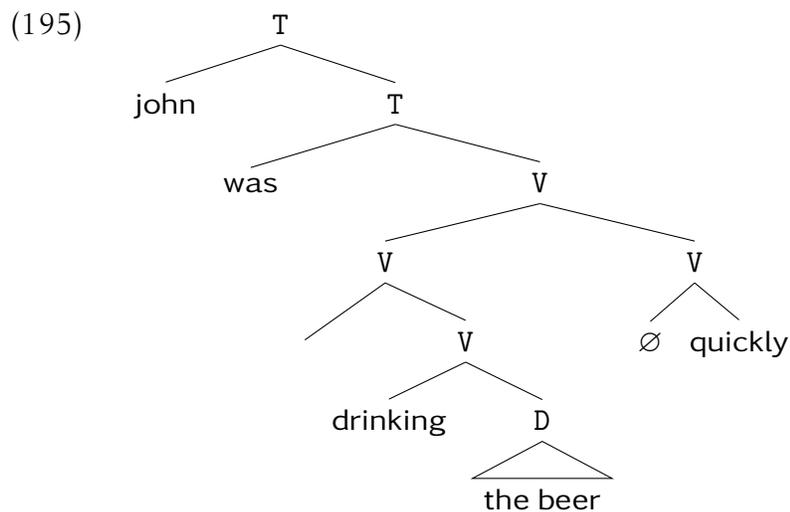
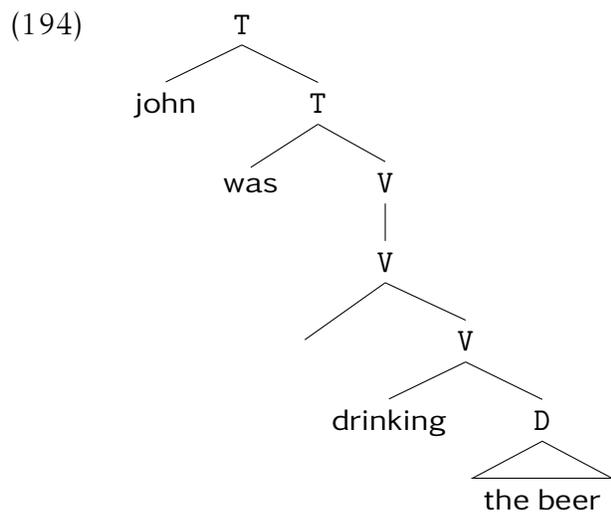
When building the structure bottom-up, this is straight-forward: *drank the beer* is of category *V* and *quickly* adjoins to that, resulting in the concatenation of the two items. But when building (191) from left to right, there is no *V* feature left once *quickly* is shifted: the string *john drank the beer* is at least a TP, corresponding to the expression in (193) of type *T*. The target feature *V* for adjunction is not available any more, because intermediate category features, like all features, get deleted when they are checked.

(193) T :: john drank the beer

Importantly, this issue is not specific to the incremental parsing algorithm proposed here, instead it seems to arise with any incremental parsing algorithm for standard Minimalist Grammars, provided that category features are deleted when checked. And the same issue with incremental parsing of right adjuncts (and conjuncts) arises in (Combinatory) Categorical Grammar (STANOJEVIĆ ET AL. 2020), because in CCG, like in MGs, category labels of selected items are deleted, once the item has combined with the selecting item.

However, deleting category features is not a defining property of Minimalist Grammars (unlike Categorical Grammars), and not deleting the category labels of maximal projections is probably in line with the intuitions of many syntacticians. This possibility is formally explored in GÄRTNER & MICHAELIS 2003 and STABLER 2006, where category features are treated as persistent properties, which are not deleted after being checked by MERGE. One has to make sure that persistent category features do not block further feature checking (e.g. of licensing features), this can be done by introducing a pointer to the active feature, or by stipulating that attractee features are not ordered, as argued in Section 2.5.4.

Having category features visible throughout the derivation is not quite enough to make right adjunction work though. Consider the tree shown in (194): adding a VP-modifying adverb like *quickly* would require us to transform (194) into a structure like (195).



Such a transformation requires some heavy machinery like the adjunction operation from Tree-Adjoining Grammars³², which takes apart an already built tree, inserts the adjunct and then re-assembles the tree. Such a powerful adjunction operation is most likely necessary to account for all possible cases of adjunction (and coordination), especially if we expect that the output of our parser be a fully connected derived tree representing the entire input.

³² For a discussion of the adjunction operation, its complexities and its impact on language development, see FRANK (1998), FRANK & VIJAY-SHANKER (2001)

However, if we give up this expectation and instead focus on the effects of right adjunction on the form and meaning component independently, a simpler picture emerges: on the PF side, right adjunction is just string concatenation. So as long as we are only concerned with the PF exponent, adjoining one MG expression onto another just means concatenating the two strings, since MG expressions are collapsed trees with no internal hierarchical structure. But having no internal hierarchical structure also means that there are no internal phrases to adjoin to.

Here we explore the idea that right adjunct attachment is mostly ‘semantic’ as suggested in HAIDER 2004 and ERNST 2020, but we apply a rather coarse notion of semantics. We assume that there are only two types of semantic objects, which are modified by adjuncts: verbal projections (or clauses) and nominal projections. Semantically an adjunct can either modify an event or an individual. If an adjunct modifies an event, like the adverb *quickly* in (195), it can adjoin to any projection that has ‘access’ to the event variable. Traditionally, these would only be VP and vP, but in Chapter 4 we develop a semantics in which all clausal projections from VP up to (and including) CP are of the same semantic type, they are predicates of sets of events, and so adjoining to TP or CP has the semantic effect of modifying the event variable within that TP or CP.

So adjoining *quickly* to the expression in (196), means that we need an adjunctivizer of appropriate type: it should select a TP and return a TP as shown in (197).

(196) T :: john was drinking the beer

(197) <T >Adv. T

Notice how having adjunctivizers offers us the flexibility to adjoin at any level, in particular there is nothing in the syntax or lexical entry of *quickly* that makes it a VP adjunct. All syntactic information about adjuncts is encoded in the lexical entries of adjunctivizers.

And since our selection features are directional, we can specify different adjunctivizers for right adjuncts and left adjuncts, allowing us to capture the empirical observation that left adjuncts are fairly rigidly ordered, while right adjuncts have greater positional flexibility (cf. HAIDER 2004, ERNST 2020).

Introducing a multitude of adjunctivizers into the lexicon may not seem like the most parsimonious analytical decision, but in Section 4.3 we show that the semantic function of an adjunctivizer is to lift or shift the semantic type of its argument into a modifier. Compared to the commonly held view that semantic type shifting can apply freely, lexicalizing some types of type shifter is a more restrictive position. It remains to be seen if all type shifting can be so lexicalized, but so far no counter-example has been found. An alternative to specifying adjunctivizers in the lexicon is the model of adjunction proposed in FOWLIE (2013), which handles left- and right-adjuncts and their different ordering restrictions in a unified manner, at the cost of introducing complex category features consisting of ordered pairs.

Denying the necessity of building a tree structure for right adjuncts offers a rather radical solution to another problem, namely deciding whether right adjuncts form a right-branching or left-branching structure (or both). PESETSKY (1995) observed that adjuncts to the right of the verb seem to form a left-branching structure when probed by constituency tests like VP fronting, but when probed by tests involving c-command the same adjuncts seem to form a right-branching structure. This has led PESETSKY (1995) to conclude that right adjuncts are simultaneously in both structures. But instead of having two syntactic structures, right adjuncts may as well have no syntactic structure at all, and be determined by semantic consideration as HAIDER (2004) and ERNST (2020) conclude.

Ultimately, the purely semantic approach to right adjuncts is probably not tenable, and we need to assume some form of nested or hierarchical structure. But critically, such structure is only needed on the meaning side. In Section 6.1 I will offer a rough sketch

how such a structured meaning representation may look like. One problem for a semantic approach to adjunction is to decide which variable an adjunct is supposed to modify. In syntax and sentence processing this problem goes under the name of adjunct attachment ambiguities.

2.3.3.3 Adjunct Attachment (and Extraposition)

One particularly clear case of an adjunct attachment ambiguity is the attachment of the PP *with binoculars* in a sentence like (198), which is ambiguous as to who has the binoculars, the man or the woman. In the latter case *with binoculars* modifies the woman (NP attachment), in the former case it modifies the event of seeing (VP attachment).

(198) The man saw the woman with binoculars.

In psycholinguistics and sentence processing attachment ambiguities like the one in (198) are typically viewed as parsing ambiguities. In our incremental parser the ambiguity in (198) can be modeled as resulting from a lexical ambiguity between the different possible adjunctivizers. To make this work we need to assume that there is limited look-ahead of the next item in the input, which can inform parsing decisions. This adds a shift-reduce conflict to the mix.

To see this consider the parse stack after the noun *woman* was shifted:

(199) SHIFT woman

N :: woman
>N. T :: the man saw the
>T. R

The obvious next step would be to reduce the two items on top of the stack via `MERGE`. And if the next item in the input is a verbal adjunctivizer like (200), reduce is the only option and the adjunctivizer then combines with the full clause as in (201), resulting in the VP attachment or event-modifying reading.

(200) `<T >P. T`

(201) `MERGE> N`

`T :: the man saw the woman`

`>T. R`

`SHIFT \emptyset_A`

`<T >P. T :: \emptyset_A`

`T :: the man saw the woman`

`>T. R`

`MERGE< T`

`>P. T :: the man saw the woman`

`>T. R`

`SHIFT with`

`>D. P :: with`

`>P. T :: the man saw the woman`

`>T. R`

`MERGE» P`

`>D. T :: the man saw the woman with`

`>T. R`

But if the next item in the input is instead a nominal adjunctivizer like (202), we should first shift and combine the shifted adjunctivizer with the noun on the stack, before

combining the modified noun with the TP. This derivation, which is shown in (203), gives us the NP attachment reading.

(202) <N >P. N

(203) SHIFT \emptyset_A

<N >P. N :: \emptyset_A

N :: woman

>N. T :: the man saw the

>T. R

MERGE_< N

>P. N :: woman

>N. T :: the man saw the

>T. R

MERGE_» N

>P. T :: the man saw the woman

>T. R

SHIFT with

>D. P :: with

>P. T :: the man saw the woman

>T. R

MERGE_» P

>D. T :: the man saw the woman with

>T. R

Notice how the derivation in (203) is in some sense not maximally incremental: merging the noun *woman* is delayed until it has combined with the following adjunct(izer).³³

Instead of attaching right adjuncts by making the derivation less incremental, we may also consider attaching right adjuncts via an anaphoric process along the lines suggested in GROVE & HANINK 2016. Such an anaphoric process may be independently needed for extraposed right adjuncts like in (204).

- (204) a. A book appeared about extraposed adjuncts.
 b. The president recited a poem yesterday by Shakespeare.

Alternatively, extraposed adjuncts and the height restriction on extraposed adjuncts (HUNTER & FRANK 2014) may require a mechanism resembling a true adjunction operation.

2.4 Coordination

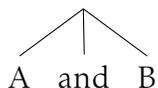
Like adjunction, coordination is tricky, both from the perspective of incremental parsing and the perspective of Minimalist Grammar formalization. And so we only provide a very rough sketch of how coordination could be made to work and what issues remain.

Coordination is often seen as a symmetrical relation between two or more conjuncts, for which a ternary structure like the one in (205) may be most appropriate.

³³ Making the derivation even less incremental, we may consider building the DP *the woman* without merging it with its clause and instead letting the PP adjoin to the entire DP instead of the NP. While rather unusual, such a derivation may give us the so-called inverse linking reading, where the modifying PP takes scope over its host. This is the most plausible reading for (i), stating that for every country there was a researcher from that country at the conference.

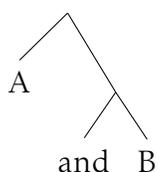
- (i) A researcher from every country attended the conference.

(205)



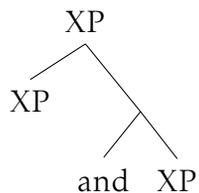
The structure in (205) is at odds with X-bar theory and the idea of a strictly binary branching syntactic structure. One way to reconcile coordination with binary branching is to assume that coordination is asymmetric (MUNN 1993: a.o.) as illustrated in (206).

(206)



A defining feature of coordination is missing from (206): coordination (typically) combines elements of the same syntactic category and yields an expression of that same category, so that an asymmetric coordinate structure looks like the one in (207).

(207)



One way to project a structure like (207) is for *and* to be head with a feature specification like the one in (208), which looks almost like an adjunctivizer head: it selects an XP to its left and to its right and then yields an expression of category X.

(208) $\langle X \rangle X$. $X :: \text{and}$

Treating coordinating heads like *and* like adjunctivizers also explains how a coordinate structure of two XP can be selected wherever a single XP can. If a coordinating head is an

adjunctivizer, then one of the conjuncts must be an adjunct, and in some sense they both are. In a sentence like (209) the second conjunct looks like a right adjunct, since the first part can stand alone.

- (209) The critic read the book
 a. and was disappointed.

And similarly, in (210) the first conjunct looks more like the adjunct, again because the second part of the sentence is complete by itself.

- (210) The authors and
 a. the critics attended a poetry reading.

The examples in (209) and (210) allow us to speculate that some conjuncts behave a bit like right adjuncts and other conjuncts a bit like left adjuncts, but ultimately the relationship between the two (or more) conjuncts is not as clearly specified as in the case of the adjunct and its host phrase. Maybe this is (at least part of) the reason why conjuncts are typically view as being on the same level (or coordinate), while an adjunct is typically subordinate to its head³⁴.

The main property that distinguishes coordination from adjunction is that it allows non-constituent coordination³⁵: in the right-node raising examples in (211) the first conjuncts would not be constituents of their respective sentences if the sentences ended before the word *and*.

³⁴ Although that is not as clear as it may seem: one may also view the adjunct as selecting its host, in which case one could call the adjunct the head (Dowry 2003).

³⁵ The term *non-constituent coordination* is a contradiction in terms: the “non-constituent conjunct” is a conjunct and so by definition a constituent of the coordinate structure.

- (211) a. Bill gives [money to banks on Monday] and [money to charities on Sunday]
 b. Bill gives money [to banks on Monday] and [to charities on Sunday.]

From an incremental perspective coordination poses the same challenges as adjunction (STANOJEVIĆ ET AL. 2020): for any kind of phrasal coordination, the left conjunct will have to be fully assembled (and most likely merged into the rest of the tree) before any evidence of a coordinate structure becomes available.³⁶ And right when the coordinator is processed, there is often an attachment ambiguity, as illustrated in (212).

- (212) Bill likes to talk to Mary and
 a. John likes to talk to Susan.
 b. to talk to Susan.

But the attachment ambiguity in (212) is different from the one we saw for adjunction: in the case of adjunction, the selected category of the adjunctivizer (i.e. the adjunction site) is typically rather constrained and determined by semantics, but in the case of coordination, the selected category of the ‘conjunctivizer head’ and is determined by the category of the second conjunct.

More importantly for our purposes, (212) shows that second conjuncts can ‘conjoin’ to a more diverse set of phrases than adjuncts. This means that our strategy of modeling attachment ambiguity via delayed incrementality to avoid introducing a destructive adjunction operation for right adjuncts is no longer feasible. The adjunction operation

³⁶ In the case of two-part coordinating conjunctions like *both ... and ...* or *either ... or*, this argument obviously does not apply. But it should be noted that in many languages such two-part constructions are somewhat marked, suggesting a pragmatic strengthening, like in the case of *either ... or*, which strengthens regular *or* to an exclusive or. From this perspective, it is interesting to note that negated coordinate structures are more likely to involve a two-part construction like the English *neither ... nor ...*

from Tree-Adjoining Grammars is still a good candidate, but figuring out the details is left for future research.

Here we just note that whatever the exact mechanism for adjunction and coordination, it should be able to combine incomplete items, just like MERGE. Then across-the-board (ATB) movement like in (213) can be handled in a straightforward manner: the two incomplete conjuncts of type >D. T in (213) combine and form a single expression of type >D. T as in (214) into which the moved phrase *who* can be merged.

(213) John knows who [Jack likes *t*] and [Mary hates *t*].

(214) John knows who [Jack likes and Mary hates] *t*.

2.5 What Moves Where When How

One of the main reasons for syntactic movement is the empirical observation that a constituent may be pronounced in a different position from where it is most plausibly interpreted. This split between pronunciation and interpretation can be handled in different ways. In a traditional Minimalist Grammar derivation, an expression with a licensee feature is merged at the site where it is interpreted, then becomes a mover, and is finally pronounced when its licensee feature is checked. So in a bottom-up derivation, a mover carries its pronunciation along the movement chain, and it makes sense to ask where a mover is pronounced, to which the answer is typically: PRONOUNCE HIGHEST: a mover gets pronounced at its final landing site, unless movement is covert, in which case it doesn't. No matter the exact answer, it remains a stipulation.

When parsing a sentence incrementally, words are processed as they appear in the input, so the pronunciation of a mover is fixed: words only become movers after they have been pronounced, and it makes no sense for a mover to carry along its pronunciation. One

may then conclude that movers consist of nothing but their syntactic features. And this is indeed exactly what our notation has suggested so far. But our notation has conveniently ignored the meaning side (or LF) of the linguistic sign. Once we take meaning into account, we need to conclude that a mover consist of its (remaining) syntactic features and (at least a part of) its meaning representation. So in an inverted movement chain, a mover carries its meaning along the chain.

Without any additional provisions, this means that a mover only makes a meaning contribution at its base position and nowhere else along the movement chain. We thus get full RECONSTRUCTION. But movers may also make a meaning contribution at other positions of their movement chain: in a long-distance constituent question like (215), the constituent in question *which books* contributes to the question meaning in the matrix clause, even though its base position is in the embedded clause.

(215) Which books do the authors think that the critics reviewed?

As another example, consider the *how many*-question in (216), which has two distinct readings: a wide scope reading and a narrow scope reading.

(216) How many books should the critics review this week?

The wide scope reading of (216) can be paraphrased as in (217), the assumption is that there is a fixed set of books that the critics should review, and the speaker is asking about the cardinality of that set. So under the wide scope reading, the *wh*-phrase *how many* takes scope above the modal auxiliary *should*.

(217) What is the cardinality of the set of books that the critics should review this week?

The narrow scope reading of (216) can be paraphrased as in (218). Critically, under the narrow scope reading it is not assumed that there is a specific set of books for the critics to review, instead there is a fixed number of books, and the speaker is asking about that number. So under the narrow scope reading, the *wh*-phrase *how many* takes scope under the modal auxiliary *should*.

(218) What is the number such that the critics should review that many books this week?

This shows that (at least in some cases) a moved expression can take scope and thus make a meaning contribution at more than one position of its movement chain. So we assume that the meaning of the linguistic sign within an MG expression can contain more than one meaning component. In particular, if an expression has multiple attractee features (e.g. a category and a licensing feature), we stipulate that the meaning of its sign (its LF) consist of as many components as their attractee features. This allows us to discharge one meaning contribution at each position in a movement chain.

2.5.1 A vs \bar{A} Movement

So far we have treated movement as a uniform phenomenon, but it is an established empirical generalization (but cf. VAN URK 2015) that there are (at least) two different types of phrasal movement: A-movement and \bar{A} -movement ('A-bar movement'). A-MOVEMENT (or argument movement) establishes a movement dependency between two argument positions (or A positions). \bar{A} -MOVEMENT (non-argument movement) moves an element to a non-argument position (\bar{A} position) from either an A-position or an \bar{A} -position. Examples of A-movement are the promotion to subject in passive transformations (219) and raising to subject (220).

(219) The critics were criticized.

(220) The critics seem to like the book.

Examples of \bar{A} movement are topicalization (221) and *wh*-movement (222), which is the prototypical case of \bar{A} movement (CHOMSKY 1977).

(221) The book, the critics like.

(222) Which books do the critics like?

Empirically, the two types of movement differ in how local they are and in how they interact with semantic scope, among other factors. In terms of analysis, \bar{A} -movement seems to be a genuinely syntactic process, while A-movement can be implemented as a lexical operation³⁷ like in Lexical-Functional Grammar (KAPLAN & BRESNAN 1982) and Head-driven Phrase Structure Grammar (HPSG, POLLARD & SAG 1994). Such a hybrid approach to movement can also be implemented within Minimalist Grammars, as shown in KOBELE 2010, where A-dependencies are established via hypothetical reasoning, while the MG operation MOVE is used only for \bar{A} -dependencies.

The principle explanandum for KOBELE (2010) is the BAN ON IMPROPER MOVEMENT, which states that A-movement cannot follow \bar{A} -movement ('once you go bar, you never go back'). The intuition to derive this ban is that an object entering an \bar{A} -dependency is already present in the derivation, while an object entering an A-dependency starts out as a hypothesis. We illustrate the idea of A-movement as hypothetical reasoning with a bottom-up derivation of the intransitive clause in (223) with the verb in (224).

(223) The critics will dance.

³⁷ \bar{A} -movement can also be implemented lexically, e.g. via slash features (GAZDAR, KLEIN, PULLUM & SAG 1985).

(224) <D. V :: dance

The derivation is shown in (225), where we insert parsing-like SHIFT-steps to indicate that the derivation starts with just the verb. The assumed subject is a DP, since the verb in (224) selects a D, but DPs always require case, so the assumption is of type D -nom. Following KOBELE (2010), in (225) we write the assumed subject D -nom like a mover, even though it is not a mover: in a bottom-up derivation a mover carries its pronunciation along with it, so as a mover would be D -nom :: the critics. Instead D -nom is just a hypothesis of a DP, stating that we expect to see an expression of type D -nom later during the derivation. In the remaining two steps in (225), the auxiliary will is shifted and merged.

(225) SHIFT dance

<D. V :: dance

>T. R

ASSUME D -nom

V :: dance, D -nom

>T. R

SHIFT will

>V +nom. T :: will

V :: dance, D -nom

>T. R

MERGE_> V

+nom. T :: will dance, D -nom

>T. R

The derivation then continues as in (226), where the subject DP is shifted and assembled, and the assumption of the subject DP can be discharged.

(226) SHIFT the, critics

```

                >N. D -nom :: the
                    N :: critics
                +nom. T :: will dance, D -nom
                    >T. R

MERGE> N

                D -nom :: the critics
                +nom. T :: will dance, D -nom
                    >T. R

DISCHARGE D -nom

                T :: the critics will dance, D -nom
                    >T. R

```

So the ‘A-moved’ subject DP is base-generated in its derived position, and only hypothesized (assumed) in its base position. Compare this to a derivation involving \bar{A} -movement, like the one of (227), whose initial derivation is as shown in (225).

(227) Which critics will dance?

After (226), the derivation of (227) then continues as in (228) by shifting and assembling the subject DP.

(228) SHIFT which, critics

>N. D -nom -wh :: which

N :: critics

+nom. T :: will dance, D -nom

>C. R

MERGE_> N

D -nom -wh :: which critics

+nom. T :: will dance, D -nom

>C. R

Now the assumed subject DP can be discharged against the *wh*-DP which critics just assembled, as shown in (229). But since after DISCHARGE the *wh*-DP which critics still has a -*wh* feature, it now becomes a regular mover.

(229) DISCHARGE D -nom

T :: will dance, -wh :: which critics

>T. C

The derivation then continues like a regular bottom-up derivation involving *wh*-movement, as shown in (230): we shift and merge the empty complementizer carrying the +*wh* licensing feature, and then apply MOVE.

- (230) SHIFT C_{wh}
- >T +wh. C :: C_{wh}
- T :: will dance, -wh :: which critics
- >C. R
- MERGE_> T
- +wh. C :: C_{wh} will dance, -wh :: which critics
- >C. R
- MOVE wh
- C :: which critics C_{wh} will dance
- >C. R

It should be emphasized that during its \bar{A} -movement step the subject looked like a regular mover, it consisted of a feature sequence and a pronunciation: -wh :: which critics. In contrast, during its 'A-movement step', the subject was just an (assumed) feature sequence D -nom without a pronunciation. However, since we wrote the assumption like a mover, let's assume that the assumption is indeed a mover, just in an inverted movement chain, and not in a bottom-up movement chain.

When going from bottom-up movement chains to inverted movement chains, we noticed that pronunciations are replaced with meanings: a bottom-up mover carries along its pronunciation, an inverted mover carries along its meaning. But according to KOBELÉ (2010) not all (bottom-up) 'movers' carry along their pronunciations, only \bar{A} -movers do. A-movers are just feature sequences representing a hypothesized expression. For inverted movement chains, the corresponding conclusion is that only \bar{A} -movers carry along their meanings, while A-movers are just feature sequences. As a first approximation, this matches nicely with the observation that A-movement does not reconstruct. But a more detailed discussion of the semantics of a A-movement must wait until Section 4.4.

Here we instead want to ask what (if anything) distinguishes the two types of movement at the feature level: what distinguishes an \bar{A} -movement licensing feature from an A-movement licensing feature? KOBELE (2010) does not discuss this question, the same type of licensing features are involved in both types of movement.

Although it may be useful to notationally distinguish \bar{A} -licensing features from A-licensing features, the main difference between the two types of features is how they are associated with the expression carrying them. I propose that A-movement features are inherent, they are a necessary feature of an MG expressions being a certain type of object. \bar{A} -movement features, on the other hand, are in some sense optional. They are not a necessary feature of the MG expressions they are typically associated with. And so I propose that \bar{A} -licensing features are introduced onto an expression by a (most often) empty head adjoining to that expression.

In the derivation above, the relevant licensing features were $-\text{nom}$ and $-\text{wh}$. It is obvious that $-\text{nom}$ is an inherent feature: there are no DPs without case, and so a case feature like $-\text{nom}$ or $-\text{k}$ is a necessary or constitutive feature of being a DP. It may very well be that all A-movement is case-driven and that A-positions are case positions, i.e. all landing sites of A-movement are positions that license a case feature. In this case A-licensing features are coextensive with case features. Alternatively, it is argued in VAN URK 2015 that A-movement is driven by ϕ -features, which enter into an AGREE relation with the licensing head. But like case, ϕ -features are also an inherent property of DPs.

But how is the $-\text{wh}$ feature optional? Doesn't it encode the inherent *wh*-iness of *wh*-words? A different type of \bar{A} -movement may illustrate the optionality of \bar{A} -licensing features more clearly.

2.5.2 Introducing \bar{A} -Licensing Features via Adjunction

When we discussed remnant movement in Section 2.2.4, we assumed that the fronted verb *admired* in (231) has a fronting feature $-f$, which is exactly how VP fronting is modeled in a bottom-up derivation: the element to be fronted has a feature licensing movement, which makes it become a mover after it has been merged into its base position.

(231) *Admired*, John was.

But how does the verb *admired* get its fronting feature $-f$? Since VP fronting is in theory possible for all verbs, so under the lexicalist approach of Minimalist Grammars we have to assume that all verbs (and adjectives, nouns, adverbs, ...) have two lexical entries: one with the fronting feature $-f$ and one without it. This is not very parsimonious.

An alternative to such a massive redundancy in the lexicon is to extend our conception of adjunction via empty heads, and to employ an empty head to introduce the fronting feature when needed. For the remnant movement derivation of (231), the empty head in (232) can introduce the fronting feature: it selects a V as its complement and projects (or returns) an expression of type $-f$ V, i.e. an expression of category V with a fronting feature $-f$.

(232) $\langle V, -f \rangle V :: \emptyset_f$

Notice how the head in (232) looks almost like an empty adjunct: if we focus only on category features, it is category preserving like any other adjunct. But \emptyset_f is not feature preserving, as it introduces a new licensing feature $-f$.

With the head in (232), we can introduce a fronting feature whenever needed. So our remnant movement derivation from Section 2.2.4 can proceed as follows (the full derivation is shown in Figure 15): first we shift the fronted verb *admired*, which now

has its standard feature sequence >D. V without any licensing features. Looking ahead to the fronting complementizer $[\text{+f >T. C} :: C_f]$ (spelled out as a comma) in the input, we notice that it has no matching features with the expressions on the stack, in particular no V feature. And so we shift the expression in (232) to allow the derivation to continue as in (233). This last-resort type introduction of the feature introducing head \emptyset_f is not necessary, it can instead be a regular (empty) item in the input.

(233) **SHIFT** *admired*

$\text{>D. V} :: \text{admired}$

>C. R

SHIFT \emptyset_f

$\text{>V. -f V} :: \emptyset_f$

$\text{>D. V} :: \text{admired}$

>C. R

The top two items on the stack in (233) have a matching V feature and so we apply **MERGE \gg** as in (234). Next we shift the fronting complementizer, and the derivation can continue as it did in Section 2.2.4.

(234) **MERGE \gg** *V*

$\text{>D. -f V} :: \text{admired}$

>C. R

SHIFT C_f

$\text{+f >T. C} :: C_f$

$\text{>D. -f V} :: \text{admired}$

>C. R

So we can see that the \bar{A} -licensing feature $-f$ can be introduced via an empty head, and that introducing the \bar{A} -feature $-f$ in this way may be more parsimonious than assuming that all verbs (and adjectives, nouns, adverbs, ...) have two lexical entries: one with the fronting feature $-f$ and one without it. Instead of quite literally doubling the lexicon, we only need to add a limited set of empty heads like (235) for all category features X .

(235) $\langle X, -f \rangle X :: \emptyset_f$

1. SHIFT admired
 >D. V :: admired
 >C. R
2. SHIFT \emptyset_f
 >V. -f V :: \emptyset_f
 >D. V :: admired
 >C. R
3. MERGE \gg V
 >D. -f V :: admired
 >C. R
4. SHIFT C_f
 +f >T. C :: C_f
 >D. -f V :: admired
 >C. R
5. MOVE \gg_E f
 >T. C :: admired C_f , >D. V :: ϵ
 >C. R
6. SHIFT john
 -nom D :: john
 >T. C :: admired C_f , >D. V :: ϵ
 >C. R
7. SHIFT was
 +nom >V. T :: was
 -nom D :: john
 >T. C :: admired C_f , >D. V :: ϵ
8. MOVE $_E$ nom
 >V. T :: john was, D :: ϵ
 >T. C :: admired C_f , >D. V :: ϵ
9. MERGE \gg T
 >V. C :: admired C_f john was, >D. V :: ϵ , D :: ϵ
10. MERGE \gg_I V
 >D. C :: admired C_f john was, D :: ϵ
11. MERGE $_I$ D
 C :: admired C_f john was
 >C. R
12. MERGE $_>$ C
 R :: admired C_f john was

FIGURE 15: Incremental remnant movement with introduction of fronting feature via an empty head: *Admired, John was*

2.5.2.1 (Remnant) Movement and Adjuncts

Introducing \bar{A} -licensing features via an empty head makes movement and adjunction play together nicely, especially in cases like (236), where the adverb *deeply* is adjoined to a moved phrase.

(236) Deeply admired, John was.

When building syntactic structure bottom-up, examples like (236) pose be a challenge for our reductionist approach to adjunction: if the verb *admired* carries the fronting feature $-f$ (after all it is VP fronting), how can the adverb *deeply* come along for the ride? One solution commonly suggested is LATE MERGER (e.g. LEBEAUX 1988, FOX & NISSENBAUM 1999), according to which adjuncts can be merged counter-cyclically after their host phrase has moved. However, it is not clear how late merger can be reconciled with the MG feature calculus: once (merged and) moved, the VP in (236) has lost its V feature.

In an incremental derivation, this problem does not occur, because the fronting feature can be added after the fronted phrase is assembled: so in (237), we first shift the adverb (for simplicity we assume it has combined with the appropriate adjunctivizer already) and the fronted verb, and merge the two.

- (237) SHIFT deeply
- >V. V :: deeply
- >C. R
- SHIFT admired
- >D. V :: admired
- >V. V :: deeply
- >C. R
- MERGE \gg V
- >D. V :: deeply admired
- >C. R

Then, in (238), we shift the head introducing the fronting feature $-f$, apply MERGE \gg because of the matching V feature. Next we shift the fronting complementizer, and derivation can continue as before.

- (238) SHIFT \emptyset_f
- >V. $-f$ V :: \emptyset_f
- >D. V :: deeply admired
- >C. R
- MERGE \gg V
- >D. $-f$ V :: deeply admired
- >C. R
- SHIFT C_f
- +f >T. C :: C_f
- >D. $-f$ V :: deeply admired
- >C. R

One may now wonder how we can rule out sentences like (239), which should be derivable if the adverb *deeply* combines with the verb before the selected direct object is shifted.

(239) *John admired deeply Mary.

One option is to stipulate that incremental right adjunction, whatever its precise mechanism, can only target maximal projections. This would rule out (239). But with the specifics of adjunction and merging of incomplete items, sentences like (239) are already ruled out: after combining with an appropriate adjunctivizer the adverb in (239) would need to be as in (240), to match the expression in (241) it wants to adjoin to.

(240) <T. T :: deeply

(241) >V. T :: john admired

But notice how (240) and (241) cannot be combined via MERGE, because there is no variant of MERGE for incomplete items linearized to the left, i.e. there is no MERGE \ll operation³⁸. Critically, the remnant movement case is unaffected, because there the adverb adjoined as a left adjunct to the fronted incomplete VP *admired*, and we know that MERGE \gg can skip over (missing) complements.

³⁸ An operation like MERGE \ll may be needed for certain cases of coordination, see Section 2.4. However, in Section 5.1 we argue that based on the mechanics of incremental processing MERGE \ll should not be available as a variant of general-purpose MERGE, and we use those arguments to derive the Specifier Island constraint as a consequence of incremental structure building. We leave resolving the tension inherent in this footnote for future research.

1. SHIFT deeply
Adv :: deeply
>C. R
2. SHIFT \emptyset
<Adv >V. V :: \emptyset
Adv :: deeply
>C. R
3. MERGE_< Adv
>V. V :: deeply
>C. R
4. SHIFT admired
>D. V :: admired
>V. V :: deeply
>C. R
5. MERGE_{>>} V
>D. V :: deeply admired
>C. R
6. SHIFT \emptyset_f
>V. -f V :: \emptyset_f
>D. V :: deeply admired
7. MERGE_{>>} V
>D. -f V :: deeply admired
8. SHIFT C_f
+f >T. C :: C_f
>D. -f V :: deeply admired
9. MOVE_E_{>>} f
>T. C :: deeply admired C_f , >D. V :: ϵ
10. SHIFT john
-nom D :: john
>T. C :: deeply admired C_f , >D. V :: ϵ
11. SHIFT was
+nom >V. T :: was
-nom D :: john
>T. C :: deeply admired C_f , >D. V :: ϵ
>C. R
12. MOVE_E nom
>V. T :: john was, D :: ϵ
>T. C :: deeply admired C_f , >D. V :: ϵ
>C. R
13. MERGE_{>>} T
>V. C :: deeply admired C_f john was, >D. V :: ϵ , D :: ϵ
14. MERGE_I_{>>} V
>D. C :: deeply admired C_f john was, D :: ϵ
>C. R
15. MERGE_I D
C :: deeply admired C_f john was
>C. R
16. MERGE_> C
R :: deeply admired C_f john was

FIGURE 16: Incremental remnant movement with adjunct: *Deeply admired, John was.*

2.5.2.2 \bar{A} -Features and the Q(uestion) Particle

Coming back to the distinction between A-features and \bar{A} -features: A-movement features are inherent and part of the lexical item carrying them. \bar{A} -movement features, on the other hand, are optional and get added to the respective moving expression via an empty head.

For the fronting feature $-f$ in the VP-fronting example above, this made sense. But what about standard *wh*-movement? Isn't the $-\bar{wh}$ feature an inherent property of *wh*-words like *who* or *which*, thus encode the inherent *wh*-iness of *wh*-words? It is both standard and suggestive to assume that *wh*-words are the carriers of a $-\bar{wh}$ feature. But this assumption is challenged by the possibility of pied-piping as in (242) and (243).³⁹

(242) To whom did the critics speak?

(243) Whose book did the critics review?

The problem with pied-piping is that the fronted phrase *to whom* in (242) is a PP, which contains the *wh*-word *whom* as the complement of the preposition: if the *wh*-word carries a $-\bar{wh}$ licensing feature, how can the whole PP be moved, and not just the phrase carrying the feature? One standard account of pied-piping is the Q-based analysis advanced in CABLE 2007, 2010. I propose that the head introducing the $-\bar{wh}$ feature is the same as CABLE's Q-head.

The standard picture regarding Q is that there is a Q-head, which has to move to CP,Spec to form a constituent question. Languages differ in how much stuff the Q-head

³⁹ In English, pied-piping of PPs is somewhat marked in constituent questions, instead the variant with preposition stranding in (i) is generally preferred. But in languages without preposition stranding, like German, pied-piping a PP is the only option and examples like (ii) are normal.

(i) Who did the critics speak with?

(ii) Mit wem haben die Kritiker gesprochen?
with whom have the critics spoken?

carries along: in *wh-in-situ*-languages like Japanese, Q itself can move, while in *wh*-fronting languages like English, Q pied-pipes its complement along. CABLE proposes that in *wh*-fronting languages, Q projects a Q-phrase and takes a *wh*-phrase as its complement, while in *wh-in-situ*-languages Q only adjoins to the relevant constituent in question, and then moves by itself to CP,Spec.

Incremental structure building offers a new perspective on pied-piping that combines the Q-based and $-\bar{w}h$ -feature based approach: a moved *wh*-phrase gets shifted onto the stack as a phrase without a $-\bar{w}h$ feature. And then the $-\bar{w}h$ feature, like the $-f$ feature above and all \bar{A} -features, gets added via (adjunction of) a head introducing the $-\bar{w}h$ feature. The head introducing the $-\bar{w}h$ feature is the Q-head as proposed by CABLE (2007, 2010). In many languages, including English, the Q-head is empty, but in some languages, e.g. Tlingit (Na Dene, spoken in the southeastern panhandle of Alaska), Q is spelled out as a particle.

Notice how from an incremental perspective, CABLE's distinction between Q-heads that project, and Q-heads that adjoin, does not seem to be the right distinction any more: when a Q-head adjoins to a fronted phrase, it may look a bit like it is forming a Q-phrase, but all it does is add a $-\bar{w}h$ feature. The category of fronted phrase is not changed. This also solves the problem of selection faced by CABLE's (2007) original proposal: if Q projects, then every verb must select both DPs and Q-phrases, potentially duplicating the lexicon. Treating Q uniformly as an adjoining head, which introduces the $-\bar{w}h$ feature, does not encounter this problem, even for *wh*-expressions that remain *in situ*: a DP carrying a $-\bar{w}h$ feature is still a DP and can be selected and merged as such.

Introducing \bar{A} -licensing features via a functional head opens the possibility that any phrase can in principle be fronted, but since fronting is rather constrained we have to come up with a way to constrain the addition of licensing features. The obvious approach is to stipulate that there is a fixed set of empty heads that can insert movement features, and

they select only the phrases that can front. So instead of duplicating every single lexical entry of a head projecting a potentially phrase, we specify one head for each frontable category.

2.5.3 Covert Movement

Syntactic movement allows for an expression to be interpreted in a different position from where it is pronounced. But as we saw above, a moved expression can be interpreted in both its base position and in its moved position. An example where this is necessarily the case is (244), where the moved expression *how many books* has to take scope in its moved position (because of the weak-island inducing embedding verb *realize*).

(244) How many books did the authors realize that the critics reviewed?

So the moved expression *how many books* in (244) is interpreted in both its base position and in its moved position: in its moved position, it takes scope; in its base position it fills a thematic role of the verb *review*, the theme of the critics' reviewing. Syntactically, a thematic requirement of an expression is realized as selection: the verb *review* selects the DP *how many books* and so the two combine via MERGE. Since at MERGE the DP still carries a $-\text{wh}$ licensing feature it later has to move and take scope in the moved position.⁴⁰ So more precisely, syntactic movement allows for an expression to be pronounced in a different position from where it was selected. This type of syntactic movement is called OVERT movement.

But movement can also be COVERT. In English, the empirical phenomenon most commonly analyzed as an instance of covert movement is the so-called inverse scope reading of quantificational DPs in object position. A sentence like (245) with a universally quantified

⁴⁰ Regular *wh*-movement does not necessarily shift scope, but it does in this case. See Section 4.4 for a discussion of movement and scope.

DP in object position has two truth-conditionally distinct readings, a surface scope reading and an inverse scope reading.

(245) Two critics read every book (on the list).

Under the surface scope (or subject wide scope) reading there are two highly motivated critics, and these two critics read every book on the (very long) reading list. Under the inverse scope (or object wide scope) reading none of the critics need to have worked long hours, as long as there are many critics to share the work. Because all it says is that every book on the list was read by two critics.

If we align the surface scope reading and the pronunciation of (245), there does not seem to be any need for movement⁴¹: all elements are interpreted (only) where they are pronounced. Under the inverse scope reading, the direct object DP takes scope higher than where it is pronounced: while it is pronounced next to the verb, by which it was selected, it takes scope above the subject. So there is no overt movement, yet the direct object (or at least its quantifier) is interpreted in a different position from where it is pronounced.

Since we already have movement in our syntactic toolbox, we can try to derive the inverse scope reading by expanding the scope of syntactic movement and allowing for it to be covert: all we need to move is the quantifier part of the quantificational DP *every book* and make it raise to its higher scope position, a process aptly called QUANTIFIER RAISING (QR, MAY 1977). Compared to the traditional notion of overt syntactic movement, where a mover carries with it its pronunciation, QR is unusual, in that a mover now has no pronunciation.

⁴¹ This does not mean that the derivation of a surface scope reading sentence does not involve any argument movement, a movement-based derivation may be desirable for various reasons: in Section 4.3.3 we argue that all arguments need to move for scope (and case) reasons if we want to assign thematic roles their most natural semantic denotations. But with a slightly more involved denotation for thematic roles, it is possible to derive a surface scope reading without argument movement, see Section 4.2 for an example derivation.

But this is exactly how movers are conceptualized from an incremental perspective in inverted movement chains: the linguistic sign of a mover has lost its pronunciation (or PF) and only consist of a meaning component. So covert movement can easily be integrated into the incremental picture developed here and it works (almost) exactly as in the bottom-up case: an expression which may take scope is merged in its base position via its selection feature and then becomes a covert mover.

Let's take quantifier raising to derive the inverse scope reading of (245) as an example. Like all movement, quantifier raising can only be licensed by a feature, so a quantificational determiner like *every* gets a licensing feature $-s$ to satisfy its scope taking requirement. Since SHIFT reverses the order of attractee features, the lexical entry of *every* needs to be as in (246) to allow for the category D to be checked first. This is the inverse of the attractee feature ordering expected for a lexical item with licensing features; in Section 2.5.4 we will see how the more familiar specification can be maintained.

(246) $\text{>N. } -s \text{ D} :: \textit{every}$

Under standard assumptions movement is to the left and up the tree,⁴² so when parsing incrementally the landing site of the covert mover has to be part of the tree already. For the moment we set aside the question of how the landing site gets into the tree and just make it happen by adding the licenser feature $+s$ to the expression selecting (246), thus giving us the incomplete expression in (247).

(247) $\text{>D } +s. \text{ C} :: \textit{two critics read}$

⁴² But see FOX & NISSENBAUM (1999) and CHESI (2013) for proposals that QR is to the right, a conclusion they reach to reconcile bottom-up structure building and incremental processing. Additionally, one may also argue that covert movement in head-final languages may occasionally need to be to the right. But once again, head-final languages will be ignored.

At this stage, we have only (247) on the parse stack and can now *SHIFT* the quantificational determiner *every* as in (248)..

(248) >D +s. C :: two critics read
 SHIFT every
 >N. -s D :: every
 >D +s. C :: two critics read

The two expressions on the stack in (248) have a matching selection feature *D* and so we can combine them via a (yet to be defined) variant of *MERGE* as in (249): since pronunciations cannot move, the lexical item *every* is pronounced where it appears in the input and its remaining *-s* feature turns into a mover. The mover does not have a PF component, but it does carry the quantificational part of the meaning of *every*, indicated here symbolically by a universal quantifier symbol (\forall). Some of the semantic aspects of scope taking via movement will be discussed in Section 4.4.

(249) *MERGE?* *D*
 >N +s. C :: two critics read every, -s :: $\langle \epsilon \mid \forall \rangle$

So far this is almost like regular movement in a bottom-up derivation (except that the mover carries no PF), but since the licensing feature *+s* is already active in the tree, the mover can now move immediately as in (250), assuming there is a suitable variant of *MOVE*.

(250) *MOVE?* *s*
 >N. C :: \langle two critics read every $\mid \forall \rangle$

Finally, the lexical item **book** can be shifted and merged to conclude the derivation as shown in (251).

(251) **SHIFT book**

N :: **book**

>**N**. **C** :: ⟨two critics read every | \forall ⟩

MERGE_{> N}

>**N**. **C** :: ⟨two critics read every **book book** | \forall ⟩

Adopting for a moment the copy theory of movement, the final result of the derivation can be represented as

(252) [every]₁ two critics read [[every]₁ book]

According to some (e.g. SAUERLAND 1998) this is an appropriate representation to derive the inverse scope reading, but in the case of universal quantification it seems necessary that the restriction move together with the quantifier to allow the semantic representation in (253) to be constructed.

(253) $\forall x[\mathbf{book}(x) \rightarrow \exists y[\mathbf{critic}(y) \wedge \mathbf{read}(y)(x)]]$

So it may be more appropriate for the syntactic derivation to generate a structure like the one in (254), which can be achieved by first assembling the entire quantificational DP on the stack and then merging it as a whole (see Section 2.3.3.3 for further discussion).

(254) [every book]₁ two critics read [[every book]₁

After having seen an example derivation we can formally define the new variants of **MERGE** and **MOVE** needed for covert movement: **MOVE**_? above is instantiated by **MOVE**_T

as defined in (255), which allows a covert mover to take scope at the Top of a movement chain. Similarly, $\text{MERGE}_?$ is instantiated by the two directional variants of MERGE_M as defined in (256), which allow an element to be selected and linearized, and then turned into a Mover.

$$(255) \quad \text{MOVE}_T([\mathbf{+g} \gamma :: s, \vec{\phi}, \mathbf{-g} :: \epsilon, \vec{\psi}]) \longrightarrow [\gamma :: s, \vec{\phi}, \vec{\psi}]$$

$$(256) \quad \text{a.} \quad \text{MERGE}_M^>([\mathbf{>f} \gamma :: s, \vec{\phi}], [\mathbf{f} \mathbf{-g} \delta :: t]) \longrightarrow [\gamma :: st, \mathbf{-g} \delta :: \epsilon, \vec{\phi}]$$

$$\text{b.} \quad \text{MERGE}_M^<([\mathbf{<f} \gamma :: s, \vec{\phi}], [\mathbf{f} \mathbf{-g} \delta :: t]) \longrightarrow [\gamma :: ts, \mathbf{-g} \delta :: \epsilon, \vec{\phi}]$$

MOVE_T and MERGE_M are rather similar to their bottom-up counterparts Move_T and MERGE_M^M as defined in Figure 3, the main difference is that in the incremental variants defined in (256) and (255) the movers have empty pronunciations, as is common for movers in inverted movement chains, while in the bottom-up variants movers carry their pronunciations along the chain.

Since $\text{MERGE}_M^>$ selects an element to the right, it should have a variant to select an incomplete complement, as in (257). The MERGE_M^{\gg} thus defined is the $\text{MERGE}_?$ used in (249).

$$(257) \quad \text{MERGE}_M^{\gg}([\mathbf{>f} \gamma :: s, \vec{\phi}], [\mathbf{>x}^? \mathbf{f} \mathbf{-g} \delta :: t]) \longrightarrow [\mathbf{>x}^? \gamma :: st, \mathbf{-g} \delta :: \epsilon, \vec{\phi}]$$

We still have to answer how the licensing feature or landing site for quantifier raising or covert movement in general gets into the tree: one option is that it merges like a normal projection along the spine. But this runs into at least two problems: first, how can a head with an open licensing feature and a selection feature for its spinal complement be merged? And, second, what happens if ultimately there is no scope taking element or it takes scope in-situ? When building the tree bottom-up, neither of these problems arise,

as we already know if we need any scope projections. But when building the structure incrementally, this has the feel of an oracle.

A better solution is to add the scope licensing feature at around the same time as the scope-taking DP is added to the derivation. The mechanism for adding the scope licensing feature is once again an empty head like (258), which adjoins to the expression over which the scope taker is supposed to take scope.

(258) $=T. +s T :: \emptyset_s$

The head in (258) takes the full expression to be scoped over as its syntactic and semantic argument, thus implementing the notion of scope advanced by BARKER (2015), according to which an expression X takes scope over an expression Y when Y serves as the semantic argument of X . Notice how in (258) we used the selection feature $=T$ without specifying directionality, because both the head \emptyset_s and the moving scope taker are phonologically empty. So there is no point to specify whether QR is to the left or to the right, as FOX & NISSENBAUM (1999) and CHESI (2013) argue. A concrete semantic implementation of scope will be given in Section 4.4.

Finally, the scope-taking feature $-s$ on the scope-taking quantifier can also be introduced by an empty head like (259), which has the same overall effect as a type lifter to achieve inverse scope. The fact that the scope feature $-s$ can be introduced by a head like (259) implies that Quantifier Raising is \bar{A} -movement, and the optionality of the $-s$ feature may be linked to the fact that inverse readings are often secondary and take longer to get.

(259) $=D. -s D :: \emptyset$

2.5.4 Multiple *wh*

From a bottom-up perspective, multiple *wh*-questions pose a challenge: if we are to assume that all *wh*-expressions have the same feature specifications and have to move, covertly or overtly, then a question like (260) violates the SHORTEST MOVE CONSTRAINT (SMC).

(260) Which critics like which book?

Once the complementizer is merged there are two movers with active $-\text{wh}$ features and so according to the SMC the derivation crashes. There are ways to derive (260) without abandoning the SMC (GÄRTNER & MICHAELIS 2010, STABLER 2011), but they are rather technical and require additional assumptions.

Interestingly, from an incremental perspective, multiple *wh*-questions do not need as much special treatment. When deriving (260) incrementally, only the moved *wh*-phrase *which critics* gets inserted into the derivation through MOVE_E , i.e. by checking its licensing feature $-\text{wh}$, the phrase *which book*, which remains *in situ*, gets inserted through MERGE_M : it is merged by checking its selection feature D, but since it also carries a licensing feature $-\text{wh}$, it becomes a covert mover. And like all covert movers, it can move immediately to its landing site up in the tree; provided that there is a landing site, as discussed above.

We are still left with a small problem though: what is the order of attractee features on the *wh*-elements in (260)? A standard lexical entry for the *wh*-determiner *which* is

(261) $\text{>N. D } -\text{wh} :: \text{which}$

After applying the left-to-right transformation during SHIFT , (261) becomes

(262) $\text{>N. } -\text{wh D} :: \text{which}$

and can be integrated into the derivation via MOVE_E by checking its licensing feature wh . But the same transformation prevents the *in-situ wh* phrase *which book* to be merged via D, because its active feature is now also the licensing feature $-\text{wh}$, and not the selection feature D required for MERGE .

Within the existing feature calculus, there is an easy technical fix for this problem: we can stipulate that attractee features are unordered and can be checked in any order. With the dot notation introduced above, we can continue to write the lexical entry of *which* as

(263) $\text{>N. D } -\text{wh} :: \text{which}$

with the additional understanding that features before the dot must be checked in order, while features after the dot can be checked in any order. This also means that the features after the dot need not be transformed to ‘switch’ between bottom-up and incremental structure building, another hint that the distinction between MOVE and MERGE may not be as clear cut as a strict bottom-up view of linguistic derivations may make us believe.

If attractee features are unordered, we can also stipulate that category features be persistent and not be deleted during feature checking (GÄRTNER & MICHAELIS 2003, STABLER 2006, 2011) while also allowing licensee features to be checked after an item was selected. Making category features persistent may also be necessary to handle some cases of adjunction (and coordination) as we discussed in Section 2.3.

2.6 Taking Stock

Human language processing is incremental, but transformational grammars like Minimalist Grammars conceptualize syntactic structure building as a bottom-up process. In this chapter, we developed an incremental parsing procedure for Minimalist Grammars, which resolves this conflict. For the structure building operation MERGE , the adjustments

necessary for incremental parsing were mostly technical in nature: when processing from left to right, phrases are necessarily incomplete, and so we allow MERGE to apply when the selected expression is incomplete and still has one active selector feature.

For the operation MOVE, adopting an incremental perspective is more consequential: starting from the intuitive idea of inverting movement chains, we come to realize that MOVE can be a binary (and external) operation, which combines two separate expressions: the moved element to be inserted into its moved position and the expression or tree for the moved element to be inserted in. And conversely, at the base of a movement chain, MERGE can be a unary (and internal) operation.

However, this confusion between MERGE and MOVE arises at least in part because of the way we defined the two operations: an operation is an instance of MOVE if it is triggered by matching licensing features, and it is an instance of MERGE if it is triggered by matching selection features. But the distinction between selection and licensing features is not a necessary one, and so we may want to find a more principled way to distinguish MOVE from MERGE. We will attempt to do so in Section 2.6.2.

In this chapter we nonetheless maintained the distinction between MOVE and MERGE as one between licensing and selection features, and in the next section we summarize the rules for incremental structure building by giving formal definitions of all the operations introduced so far.

We conclude this section and chapter by comparing the proposed parsing procedure to left-corner parsers for MGs proposed in STANOJEVIĆ & STABLER 2018 and HUNTER ET AL. 2019 and by briefly discussing two open questions: how does the proposed parsing procedure relate to actual human sentence processing phenomena, and how can head-final languages be made to fit into the picture.

2.6.1 Formal Definitions of Incremental MOVE and MERGE

As a summary of our discussions, in Figure 17 we show formal definitions for all (incremental) structure building operations introduced in this chapter. Operations that permit incomplete items as arguments are listed in their forward or ‘skipping’ version with a \gg subscript or superscript. All operations that allow incomplete items, also take completed items as arguments.

Looking ahead, in Figure 18 we extend the same formal definitions for all (incremental) structure building operations to show their effect the full linguistic sign within each MG expression, i.e. on the PF and the LF component within each expression. We assume that semantic composition is compositional and fully determined by syntactic composition: syntactic MERGE and MOVE always correspond to semantic function application, with the selecting expressions always being the function and the selected element always the argument. In Chapter 4 we show how this tight syntax-semantics interface can be implemented. Here we just list operations and their effect on PF and LF. Operations that permit incomplete items as arguments are written in their standard form in Figure 18,, for the ‘skipping’ versions see Figure 17.

For strings $s, t \in \Sigma^+$, for feature sequences $\gamma \in F^*, \delta \in F^+$ and $\mathbf{f}, \mathbf{x} \in B_S, \mathbf{g} \in B_L$ an arbitrary selection, licensing feature, and for (possibly empty) sequences of chains $\vec{\phi}, \vec{\psi}$, let MERGE and MOVE be defined as follows (with ? the ‘Kleene question mark’):

- (264) MERGE is the union of the following six operations
- a. $\text{MERGE}_{\gg}([>\mathbf{f} \gamma :: s, \vec{\phi}], [>\mathbf{x}^? \mathbf{f} :: t, \vec{\psi}]) \longrightarrow [>\mathbf{x}^? \gamma :: st, \vec{\phi}, \vec{\psi}]$
selected item is linearized to the right
 - b. $\text{MERGE}_{<}([<\mathbf{f} \gamma :: s, \vec{\phi}], [\mathbf{f} :: t, \vec{\psi}]) \longrightarrow [\gamma :: ts, \vec{\phi}, \vec{\psi}]$
selected item is linearized to the left (same as $\text{MERGE}_{<}$)
 - c. $\text{MERGE}_{\gg M}([>\mathbf{f} \gamma :: s, \vec{\phi}], [>\mathbf{x}^? \mathbf{f} -\mathbf{g} \delta :: t]) \longrightarrow [>\mathbf{x}^? \gamma :: st, -\mathbf{g} \delta :: \epsilon, \vec{\phi}]$
selected item is linearized to the right and becomes a Mover
 - d. $\text{MERGE}_{< M}([<\mathbf{f} \gamma :: s, \vec{\phi}], [\mathbf{f} -\mathbf{g} \delta :: t]) \longrightarrow [\gamma :: ts, -\mathbf{g} \delta :: \epsilon, \vec{\phi}]$
selected item is linearized to the left and becomes a Mover
 - e. $\text{MERGE}_{I}^{\gg}([>\mathbf{f} \gamma :: s, \vec{\phi}, >\mathbf{x}^? \mathbf{f} :: \epsilon, \vec{\psi}]) \longrightarrow [>\mathbf{x}^? \gamma :: s, \vec{\phi}, \vec{\psi}]$
selected item is a mover (**unary Internal MERGE**)
 - f. $\text{MERGE}_{I}^{<}([<\mathbf{f} \gamma :: s, \vec{\phi}, \mathbf{f} :: \epsilon, \vec{\psi}]) \longrightarrow [\gamma :: s, \vec{\phi}, \vec{\psi}]$
selected item is a mover (**unary Internal MERGE**)
- (265) MOVE is the union of the following three operations
- a. $\text{MOVE}_{E}^{\gg}([+\mathbf{g} \gamma :: s, \vec{\phi}], [>\mathbf{x}^? -\mathbf{g} \delta :: t, \vec{\psi}]) \longrightarrow [\gamma :: ts, >\mathbf{x}^? \delta :: \epsilon, \vec{\phi}, \vec{\psi}]$
item is inserted and becomes a mover (**binary External MOVE**)
 - b. $\text{MOVE}_{C}^{\gg}([+\mathbf{g} \gamma :: s, \vec{\phi}, >\mathbf{x}^? -\mathbf{g} \delta :: \epsilon, \vec{\psi}]) \longrightarrow [\gamma :: s, \vec{\phi}, >\mathbf{x}^? \delta :: \epsilon, \vec{\psi}]$
mover Checks licensee feature (intermediate step)
 - c. $\text{MOVE}_{T}([+\mathbf{g} \gamma :: s, \vec{\phi}, -\mathbf{g} :: \epsilon, \vec{\psi}]) \longrightarrow [\gamma :: s, \vec{\phi}, \vec{\psi}]$
mover is takes scope at Top of the chain

FIGURE 17: Incremental structure building operations for Minimalist Grammars

For strings $s, t \in \Sigma^+$, for feature sequences $\gamma \in F^*$, $\delta \in F^+$ and $\mathbf{f}, \mathbf{x} \in B_S$, $\mathbf{g} \in B_L$ an arbitrary selection, licensing feature, and for (possibly empty) sequences of chains $\vec{\phi}, \vec{\psi}$, let MERGE and MOVE be defined as follows (with ? the ‘Kleene question mark’):

(266) MERGE is the union of the following six operations

- a. $\text{MERGE}_{>}([\mathbf{>f} \gamma :: \langle s | F \rangle, \vec{\phi}], [\mathbf{f} :: \langle t | X \rangle, \vec{\psi}]) \longrightarrow [\gamma :: \langle st | F(X) \rangle, \vec{\phi}, \vec{\psi}]$
selected item is linearized to the right
- b. $\text{MERGE}_{<}([\mathbf{<f} \gamma :: \langle s | F \rangle, \vec{\phi}], [\mathbf{f} :: \langle t | X \rangle, \vec{\psi}]) \longrightarrow [\gamma :: \langle ts | F(X) \rangle, \vec{\phi}, \vec{\psi}]$
selected item is linearized to the left (same as $\text{MERGE}_{<}$)
- c. $\text{MERGE}_{\vec{M}}^{\vec{>}}([\mathbf{>f} \gamma :: \langle s | F \rangle, \vec{\phi}], [\mathbf{f} -\mathbf{g} \delta :: \langle t | X \rangle]) \longrightarrow [\gamma :: \langle st | F(X) \rangle, -\mathbf{g} \delta :: \langle \epsilon, X' \rangle, \vec{\phi}]$
selected item is linearized to the right and becomes a Mover
- d. $\text{MERGE}_{\vec{M}}^{\vec{<}}([\mathbf{<f} \gamma :: \langle s | F \rangle, \vec{\phi}], [\mathbf{f} -\mathbf{g} \delta :: \langle t | X \rangle]) \longrightarrow [\gamma :: \langle ts | F(X) \rangle, -\mathbf{g} \delta :: \langle \epsilon, X' \rangle, \vec{\phi}]$
selected item is linearized to the left and becomes a Mover
- e. $\text{MERGE}_{\vec{I}}^{\vec{>}}([\mathbf{>f} \gamma :: \langle s | F \rangle, \vec{\phi}, \mathbf{f} :: \langle \epsilon | X \rangle, \vec{\psi}]) \longrightarrow [\gamma :: \langle s | F(X) \rangle, \vec{\phi}, \vec{\psi}]$
selected item is a mover (**unary Internal MERGE**)
- f. $\text{MERGE}_{\vec{I}}^{\vec{<}}([\mathbf{<f} \gamma :: \langle s | F \rangle, \vec{\phi}, \mathbf{f} :: \langle \epsilon | X \rangle, \vec{\psi}]) \longrightarrow [\gamma :: \langle s | F(X) \rangle, \vec{\phi}, \vec{\psi}]$
selected item is a mover (**unary Internal MERGE**)

(267) MOVE is the union of the following three operations

- a. $\text{MOVE}_E([\mathbf{+g} \gamma :: \langle s | F \rangle, \vec{\phi}], [-\mathbf{g} \delta :: \langle t | X \rangle, \vec{\psi}]) \longrightarrow [\gamma :: \langle ts | F(X) \rangle, \delta :: \langle \epsilon | X' \rangle, \vec{\phi}, \vec{\psi}]$
item is inserted and becomes a mover (**binary External MOVE**)
- b. $\text{MOVE}_C([\mathbf{+g} \gamma :: \langle s | F \rangle, \vec{\phi}, -\mathbf{g} \delta :: \langle \epsilon | X \rangle, \vec{\psi}]) \longrightarrow [\gamma :: \langle s | F(X) \rangle, \vec{\phi}, \mathbf{x}^? \delta :: \langle \epsilon, X' \rangle, \vec{\psi}]$
mover Checks licensee feature (intermediate step)
- c. $\text{MOVE}_T([\mathbf{+g} \gamma :: \langle s | F \rangle, \vec{\phi}, -\mathbf{g} :: \langle \epsilon | X \rangle, \vec{\psi}]) \longrightarrow [\gamma :: \langle s | F(X) \rangle, \vec{\phi}, \vec{\psi}]$
mover is takes scope at Top of the chain

FIGURE 18: Incremental structure building operations for Minimalist Grammars with full linguistic signs consisting of PF and LF components

2.6.2 Internal vs. External MERGE

Over the course of this chapter we saw that from an incremental perspective, MERGE and MOVE are not as clearly distinct as in a bottom-up derivation: MOVE can be an external binary operation, which combines two separate expressions, and MERGE can be an internal unary operation which operates over a single expression and merges that single expression's main expression with one of its movers.

However, this confusion between MERGE and MOVE arises at least in part because of the way we defined the two operations: an operation is an instance of MOVE if it is triggered by matching licensing features, and it is an instance of MERGE if it is triggered by matching selection features. And moreover, it should be emphasized that MERGE and MOVE are always binary operation, they always combine two expressions, but questions whether both expressions are independent or whether one expression is a mover within the other.

So the critical distinction between MERGE and MOVE is one of external as opposed to internal: traditional MERGE is external MERGE, it combines two independent expressions, while MOVE is internal MERGE, it operates over a single expression and combines two of its sub-expressions. In this chapter, we instead distinguished MERGE from MOVE based on whether the triggering feature is a selection or a licensing feature, but this was not meant as a principled distinction, especially since the distinction between selection and licensing features is not a necessary one.

To get an idea of how the different variants of MERGE and MOVE discussed in this chapter (and defined in Figure 17) map onto the external vs. internal MERGE distinction, Section 2.6.2 shows a table classifying them as either internal or external. There we can see that the majority of operations already conform to the internal vs. external distinction, the only exceptions are the operations $MERGE_I$ and $MOVE_E$, which we introduced to handle inverse movement chains. So it may make sense to rename those two operations (as already

External MERGE	Internal MERGE
MERGE _{>} , MERGE _{<} MERGE _{M>} , MERGE _{M<}	MOVE _T MOVE _C MERGE _{I>} , MERGE _{I<}
MOVE _E	

FIGURE 19: Classification of incremental Minimalist Grammar rules into internal vs. external MERGE operations

suggested by their subscripts indicating ‘Internal’ and ‘External’), but in this text we stick with the names defined in Figure 17.

2.6.3 Characteristics of the Parser and Previous Incremental MG Parsers

The parser developed in this chapter is a variant of a left-corner parser, in which the head of a syntactic phrase serves as the left corner; so we may call our parser a ‘head-corner’ parser (GIBSON 1991) for (strongly lexicalized) Minimalist Grammars. Left-corner parsing is a variant of top-down parsing, but unlike top-down parsers where phrase structure rules can apply freely, in left-corner parsing a phrase structure rule can only apply if the left-most nodes(s) of the right-hand side of rule (the left corner of the rule) are found bottom-up. But our Minimalist Grammar does not have any obvious phrase structure rules. So how is the left-corner constraint to be applied?

We interpret MG expressions themselves and their feature specifications as phrase-structure rules, which can only apply if the head-corner criterion is met. To understand the basic principle, consider the MG expression for the transitive verb *like* in (268): in more traditional terminology, it selects two DPs and then projects a VP.

(268) <D >D. V :: like

Interpreted in terms of phrase-structure rules, which are typically written in a top-down manner, we can say that (268) can stand in for a VP if it is preceded and followed by a DP. So we can write (268) as the partially lexicalized phrase-structure rule in (269).

(269) $V \longrightarrow D \text{ like } D$

In the language of left-corner (or head-corner) parsing, the head *like* and the DP before it are the left-corner of the rule in (269), which can only apply if the head *like* and the DP before it have been processed. But in Minimalist Grammars an entire phrase structure rule like (269) is represented in the lexical item (268) of the head *like* of a phrase, so we can reformulate the head-corner constraint as the requirement that (268) can only be merged as *V* after it has checked its <D feature.

Compared to our approach of applying the left-corner constraint directly to MG expressions, the previous left-corner parsers for Minimalist Grammars in STANOJEVIĆ & STABLER 2018 and HUNTER ET AL. 2019⁴³ apply the left-corner constraint at the level of the rewrite rules used to define MERGE (and MOVE) in Figure 3, which are simple phrase structure rules. As an example consider the definitions of MERGE in Figure 3, more specifically, the definition of MERGE_> in (49a), which corresponds to the phrase structure rule in (270).

(270) $[\gamma :: st, \vec{\phi}, \vec{\psi}] \longrightarrow [>f \gamma :: s, \vec{\phi}] [f :: t, \vec{\psi}]$ MERGE_>

STANOJEVIĆ & STABLER (2018) and HUNTER ET AL. (2019) apply the left-corner constraint to the phrase-structure rules like the one in (270), specifying that the first element of the right-hand side serve as the left-corner. So MERGE can apply whenever the first element in

⁴³ The first sketches of a left-corner parsing algorithm for MGs appeared in a manuscript now published as HUNTER 2019. While providing a detailed psycholinguistic motivation for employing a left-corner parsing strategy that paper is mostly superseded by HUNTER ET AL. 2019.

(270) is encountered, in the case of $\text{MERGE}_{>}$ the first element is the selecting head, but in the case of $\text{MERGE}_{<}$, the first element is the selected specifier, and in the case of MOVE , the right-hand side consists of only one element.

Applying the left-corner transform in this way leads to a lot of prediction steps: the parser in STANOJEVIĆ & STABLER (2018) treats specifiers as left-corners for their respective MERGE steps, i.e. a specifier predicts its upcoming head, e.g. a subject predicts a verb. In comparison to that, our parser is more bottom-up and restricts prediction to complement positions. An unexpected consequence of this restraint is that there is no obvious need for unification as a mechanism to resolve feature variables. In fact, there are no feature variables: the only feature that is not checked before entering a derivation is the complement selector feature on a head, but the identity of that feature is known once the head is known (assuming we have an input oracle providing us with correct lexical items in the input).

Another interesting property of our parsing procedure is that it preserves the relative order of structure building operations compared to a bottom-up derivation: when parsing incrementally, the structure building operations must be applied in a fixed order, and if we invert the resulting sequence of operations, we get a valid sequence of operations for bottom-up parsing.⁴⁴ In the move-eager left-corner parser proposed in HUNTER ET AL. 2019 this property does not hold, instead a moved element is merged right after it was moved.

Like the previous left-corner parsers in STANOJEVIĆ & STABLER 2018 and HUNTER ET AL. 2019, our parser shares all the characteristics of left-corner parsers: in particular during the derivation the parser state is typically not represented by a single fully-connected parse tree, but instead by multiple unconnected components. Maintaining a fully connected

⁴⁴ In bottom-up parsing there is not just a single possible sequence of operations, since bottom-up parsing only specifies how the resulting tree is traversed along the vertical dimension, but not how it is traversed horizontally, i.e. in what order the yield or input string is consumed.

parse tree during the derivation is a characteristic of top-down parsers, but top-down parsers like the ones for Minimalist Grammars proposed by STABLER (2013) need to maintain a whole beam of possible parse trees, which are then matched against the input.

Not having a fully connected parse during incremental processing may seem problematic, but it may be possible to supplant our left-corner parser with a top-down component which links the individual items on the parse stack and predicts unseen one (see Section 6.1). Such a top-down component could also form the basis for a predictive component to the human sentence processor, e.g. as suggested in HALE 2006. This should be possible because of the general property of parsers that they can be conceptualized as two separate components (GRUNE & JACOBS 1990: p. 69ff): a non-deterministic automaton which determines the set of possible parsing actions, and a controller which decides which parsing action to take when there is more than one option. In this dissertation I am concerned with the non-deterministic automaton and have little to say about how parsing decisions are being made. One exception is the following section.

2.6.4 Toward a Processing Model: Active Gap Filling and the Filled-Gap Effect

With our incremental parsing procedure fully specify, we can try to investigate to what extent the parsing algorithm proposed in this chapter can serve as the basis for a processing or performance model. We are going to see that Active Gap Filling (FODOR 1978) and the Filled-Gap Effect (STOWE 1986) follow immediately from the proposed parsing algorithm.

From the sentence processing literature we know that human readers incur a processing penalty known as the Filled-Gap Effect (STOWE 1986) when they read the object DP in a sentence like (271). The most common explanation for this effect is that the parser is eager to discharge movers in a strategy known as Active Gap Filling (FODOR 1978, STOWE 1986). And so the parser is ‘surprised’ when it cannot discharge the mover (or filler) *which author*, because the gap position of the direct object is filled by *the brother*.

	>D. V :: anna like
	>V. C :: which author does, D :: ϵ
1. MERGE \gg V	
	>D. C :: which author does anna like, D :: ϵ
2. SHIFT the	
	>N. D :: the
	>D. C :: which author does anna like, D :: ϵ
3. MERGE \gg D	
	>N. C :: which author does anna like the, D :: ϵ
4. SHIFT brother	
	>P. N :: brother
	>N. C :: which author does anna like the, D :: ϵ
5. MERGE \gg N	
	>P. C :: which author does anna like the brother, D :: ϵ
6. SHIFT of	
	>D. P :: of
	>P. C :: which author does anna like the brother, D :: ϵ
7. MERGE \gg P	
	>D. C :: which author does anna like the brother of, D :: ϵ
8. MERGE $\overset{>}{I}$ D	
	C :: which author does anna like the brother of

FIGURE 20: Incremental derivation with Filled Gap Effect: *Which author does Anna like the brother of?*

(271) Which author does Anna like the brother of?

In the parser developed in this chapter the Filled-Gap Effect shows up as a shift-reduce conflict, which is an instance of parsing ambiguity. To see this consider the parse stack after *like* was shifted onto the stack and the subject *Anna* is merged and continue the derivation from there as shown in Figure 20.

This derivation succeeds and yields the desired result. But notice that in the second step above the stack was in a configuration that would have allowed the application of MERGE $_I$ and end the derivation there by treating the selected DP as the gap for the filler *which author*. Instead we chose to shift the next word from the input stream and fill the possible gap. So in the parser proposed here the Filled-Gap Effect is nothing but a

shift-reduce conflict, which means there is a parsing ambiguity. And with a reasonable linking hypothesis we may expect to see parsing ambiguity express itself as a processing cost.

Notice how this is actually the first example of parsing ambiguity we have seen, while more traditional left-corner parsers exhibit a lot of (sometimes spurious) parsing ambiguities. If it turned out that most of the parsing ambiguities exhibited by this parser can be measured as processing costs in human sentence processing, we would have a strong argument for the psycholinguistic plausibility of the proposed parser.

As a word of caution it should however be mentioned that the input given to the parser is generated by an oracle, i.e. the input is the correct sequence of lexical items with the correct feature sequences for the intended derivation. And it is known that a considerable part of the parsing complexity in strongly lexicalized grammar frameworks like Minimalist Grammars is to determine the correct lexical items from the input string (HALE & STABLER 2005). And in human sentence processing we can observe processing effects of syntactic category disambiguation (BAUMANN 2013).

2.6.5 Parsing Head-final Languages

In this dissertation I have set head-final languages aside and leave them for future research, but one may wonder how incremental structure building works in head-final languages, and if the incremental parsing procedure proposed in this chapter can be made to work for head-final languages. A main ingredient for structure building in head-final language is most likely going to be a highly predictive component, which uses heuristics to narrow down the feature composition of expected heads and of already processed but not yet integrated elements. Such a digging-in process is most likely also at play in English, but it features less prominently in the structure building process. In Japanese, digging-in is also at the heart of ambiguity resolution (ARAI & NAKAMURA 2016). One situation where such

narrowing down of expectations occurs in English, is in determining the (invisible) case of fronted *wh*-phrases: like all common nouns, the *wh*-phrase in (272) does not carry any case morphology and it is thus ambiguous between nominative and accusative case.

(272) Which house did your brother see the roof of?

But after seeing the auxiliary *did* human comprehenders know that the question is not a subject question and so the parser can disambiguate the case feature of the *wh*-phrase at that point at the latest, while a disambiguation through the input is only possible after having processed the subject *you* and possibly the head that licenses it. Critically, the predictive narrowing and ruling-out of options is not necessary for the parse to succeed in English, but it conforms with native speakers introspections.

A similar process of predictive narrowing is more common in head-final languages, where the narrowing does not only affect the features of already seen elements, but also of predicted elements: as a head-final language, the element to be predicted is the head and psycholinguistic evidence shows that human and machine comprehenders actively predict the type of the head (e.g. transitive vs. ditransitive verb) in head-final constructions in German (KONIECZNY 2000, KONIECZNY & DÖRING 2003), Hindi (VASISHTH & LEWIS 2006), and Japanese (GRISSOM II ET AL. 2016).

To get an initial idea of how head-final incremental parsing might work mechanically and how prediction matters, let's take a step back and consider when we can establish a maximal projection: a maximal projection (specifier, head, complement) can be established after we have seen a specifier and we have evidence for either the head or the complement. Then we can predict the other. So in English we can posit a maximal projection once we have seen a specifier and the head, and the remaining element to be predicted is the complement. This is the basis of our head-corner parsing strategy. In a head-final

language like Japanese, on the other hand, we first see a specifier and the complement, and the remaining element to be predicted is a head. So an incomplete maximal projection in Japanese is a projection missing its head, and we want to be able to combine such incomplete maximal projections.

In Minimalist Grammars, there is no notion of phrase or maximal projection independent of a head, so the only way to project a phrase is to predict a specific head in a top-down manner. Alternatively, we may assume a less strongly lexicalized top-down parsing procedure, where phrases can be projected independently of their heads or by predicting a generic head (say, a transitive verb, instead of a specific verb). Then pre-verbal arguments could be put into their corresponding phrases (or specifiers), which would help narrow the prediction of the type of verb to expect. A critical component in this process is going to be how case is licensed: if Japanese case is licensed by empty heads, case-bearing DPs (or their corresponding licensing heads) could be the backbones of the clausal spine. I will leave the interaction of case licensing and verb-final structure building for future research.

Parsing head-final languages may be inherently more top-down and involve prediction of the final heads, while parsing of head-initial languages is inherently more bottom-up in the way sketched in this chapter. But ultimately those two parsing strategies form the extremes of a continuum: left-corner is somewhere between them, though leaning more towards top-down, our head-driven parsing procedure is also in between, though leaning towards bottom-up. So human language processing may very well be flexible with respect to the parsing procedure employed. And ultimately, both strategies may be at work simultaneously: our head-driven parser is incremental, but the parse state does most often not represent a connected parse tree, instead there are several expressions on the stack. These expressions may be connected by a top-down parsing component, which also serves as a prediction component. In Section 6.1 I suggest that to allow for

right adjunction, expressions predicating over different semantic variables cannot be fully integrated (merged) and may thus need to remain on the stack as separate expressions, which are then linked by a top-down component. Working out this proposal in more detail is left for future research.

CHAPTER 3

INCREMENTAL MORPHOLOGY: INVERTING HEAD MOVEMENT AND THE LEXICON

In Chapter 2 we developed an incremental parsing procedure for Minimalist Grammars, trying to stay as close to possible to their standard bottom-up formulation. But in the process of developing the parsing algorithm, we were forced to extend standard Minimalist Grammars by adding additional structure building operations. In this chapter, we deviate even more from standard Minimalist Grammars, and develop the incremental parser into a broader incremental architecture.

We start with a treatment of inverse head movement as a model of inflectional morphology. Conceptually, our model is very similar to the one developed in PHILLIPS 1996, but it fits nicely into our general framework. Unlike the standard Minimalist Grammar treatment of head movement, we do not employ any special machinery, instead we use regular (phrasal) movement. The key insight (or trick) is to allow lexical items to be complex MG expressions, i.e. a lexical item may contain movers. Those movers are the lower heads of an inverse head movement ‘chain’. Given that movers cannot carry a PF, the only pronounced element of the head movement is the top head, which is exactly the behavior of regular head movement.

In the final section we offer some arguments that allowing lexical items to be complex expressions may result in a more accurate model of morphology and the lexicon: when

treated incrementally, input items are not lexical items in the sense of MG or Minimalism, rather input items are fully inflected expressions, and syntactic parsing also involves morphological parsing. But since inverse head movement is just a form of regular movement, we may be able to say that complex lexical items are the result of phrasal spell-out, as advocated in the framework of nanosyntax (STARKE 2009).

Our conception of inverse head movement deviates significantly from standard Minimalist Grammars, but we stay formally explicit and so we keep the same MG feature calculus. However we now dispense with the left-to-right transform and instead assume that selection features are ordered for incremental parsing.

3.1 Head Movement and Inflectional Morphology

In transformational grammars, head movement is a mechanism to model some aspects of inflectional morphology within a syntactic derivation, inspired by the empirical observation that in some cases a syntactic head may appear in a higher position than that of its own projection, and so it is natural to assume that the head moved to that higher position. An example of such head movement is T-to-C movement as illustrated in (273) and (274): in the declarative sentence (273), the auxiliary *will* appears after the subject, while in the interrogative sentence (274) *will* appears before the subject.

(273) The critics will review the book.

(274) Which book will the critics review?

Assuming that the subject is in the same position in both sentences in (274) and (273), and assuming that the surface position of the subject is Spec,TP, we are led to conclude that in the interrogative sentence (274), the auxiliary *will* moves from its base position in

T to the position of C. In the case of T-to-C movement, the form or pronunciation of the (potentially) moving head *will* is the same in both positions, but head movement can also be used to allow a moving head to ‘pick up’ inflectional affixes in the positions it moves through. Another peculiarity of T-to-C movement is that both head positions (T and C) can be realized overtly without head movement, the T position can be occupied by tense auxiliaries as in (273), and the C position can be occupied by an overt complementizer like *that* in embedded clauses like (275). However, head movement can also target positions which are never realized by an overt head, such as the v(oice) head position (or little v) assumed to introduce the external argument of a transitive verb, which also the position where the verb itself is actually pronounced.

(275) The authors knows that the critics will review the book.

In this section, we develop the incremental counterpart of head movement, which critically relies on the possibility of merging incomplete items as introduced in Chapter 2. The main idea behind my proposal is a decompositional approach to head movement as articulated in PHILLIPS 1996: rather than being built up through head movement, a lexical item is being deconstructed when head movement is inverted. A word enters a derivation in its final inflected form in the position where it is pronounced and it then loses its morphological features in the form moved heads through inverse head movement.

In the given framework, this intuition can be modeled in a way that uses (almost) the same movement apparatus for head movement as is already established for phrasal movement, in particular no special apparatus or notation is needed for head movement (unlike e.g. STABLER 2001, KOBELE 2006). For a collection of arguments why head movement should be treated (almost) like phrasal movement, see ROBERTS 2010.

3.1.1 Inverse Head Movement

The defining feature of our notion of inverse movement is that a moving element is pronounced in its final landing position, and that the only thing that actually moves are features and meaning components. Applied to head movement this implies a decompositional approach to inflectional morphology as proposed by PHILLIPS (1996), according to which a lexical item enters a derivation fully inflected and in the position it is pronounced. But if the item is to undergo (inverse) head movement, it is not a simple lexical item, but instead a complex expression: a head that needs to undergo head movement, carries its lower heads as movers, which it then discharges via selection.

As an illustrating example, consider a transitive verb like *like*, and let's assume that there is vP projection, which introduces the subject. Then the lexical item of the verb *like* could look like (276): it is of category v, and so its pronunciation *like* will be linearized in vP position. As far as its features go, it first selects a DP as the external argument and then a VP.

(276) $\langle D \rangle V. v :: \textit{like}, \quad \rangle D V :: \epsilon$

But critically, the selected VP is already contained within (276) as a mover, which is itself an incomplete item. This is very similar to what we have seen in the case of remnant movement in Section 2.2.4: the mover is a VP, which is lacking its complement DP. And like in the case of remnant movement, the mover gets merged into a complement position. If the moving head selects both a complement and a specifier, the moving head can only be merged after its selected specifier has been processed (either shifted onto the stack or turned into a mover of one of the items on the stack), so that the head can be merged with its specifier while itself being merged by its selecting main expression.

We illustrate the basic principles of inverse head movement with a derivation of the simple sentence in (277). In it the verb *like* requires having processed a moved specifier in order to be merged as the head of vP. The auxiliary *do* exemplifies a similar case, but here the relevant specifier *critics* is on the stack and needs to be combined with the T head, which is a mover within the higher C head, whose lexical entry is shown in (278).

(277) Do critics like books?

(278) >T. C :: do, +nom >v. T :: ϵ

We present the derivation of (277) step by step, the full derivation is shown in Figure 21. The derivation starts as in (279) by shifting the complex complementizer (278) and the subject DP *critics*.

(279) SHIFT do

>T. C :: do, +nom >v. T :: ϵ

>C. R

SHIFT critics

-nom D :: critics

>T. C :: do, +nom >v. T :: ϵ

>C. R

At the end of (279), we have two matching features: like in all instances of inverse head movement, the main component of *do* matches its mover via T, and the mover itself has a +nom licensing feature, which matches the corresponding nominative feature on the subject DP *critics*. So we can apply one instance of external MOVE, and one instance of MERGE, after which we shift the verb *like*, as shown in (280).

(280) $\text{MOVE}_E \text{ nom} + \text{MERGE}_{\gg} \text{ T}$

$>v. \text{ C} :: \text{do critics}, \text{ D} :: \epsilon$

SHIFT like

$<D >V. v :: \text{like}, >D V :: \epsilon$

$>v. \text{ C} :: \text{do critics}, \text{ D} :: \epsilon$

The verb *like* selects the D mover of the subject as its external argument, and is itself selected by the CP on the stack via a matching *v* feature. So we can apply two instances of MERGE, as shown in (281).

(281) $\text{MERGE}_I \text{ D} + \text{MERGE}_{\gg} \text{ T}$

$>V. \text{ C} :: \text{do critics like}, >D V :: \epsilon$

On the stack in (281), there is a V feature match between the main expression and its head mover, which is characteristic of inverse head movement. But the head mover still has an open selection feature $>D$ for the direct object, so in (282), we apply MERGE_{\gg} and let the derivation proceed as expected.

(282) MERGE_» V
 >D. C :: do critics like
 SHIFT books
 D :: books
 >D. C :: do critics like
 MERGE_> D
 C :: do critics like books
 >C. R
 MERGE_> C
 R :: do critics like books

As we alluded to during the derivation ending in (282), one of the defining characteristics of inverse head movement is that each head mover is selected by another mover or the main component within its own expression, as illustrated in (283) for a hypothetical expression with two head movers: the main expression X selects the first head mover via selection feature Y, which in turn selects the second head mover via Z.

(283) >Y. X :: X, >Z. Y :: ϵ , Z ϵ

The direct selection relation between all head movers as illustrated in (283) guarantees that inverse head movement is subject to the same strict locality requirements as regular head movement.

- (284) a. They liked the book.
 b. *Them liked the book.

In the head movement examples above we assumed that the input to our incremental parser consists of a sequence of fully inflected, potentially complex MG expressions, which are decomposed through inverse head movement. Critically, those fully inflected complex MG expressions are the lexical items of our incremental MG parser, and they are different from the lexical items of the bottom-up MG we started with. Following this logic, the nominative and accusative forms of the pronoun *they* can (and/or should) correspond to the two distinct lexical shown in (285), and this logic can be extended to all DPs, even though common nouns in English do not show any case distinctions.

- (285) a. D -nom :: they
 b. D -acc :: them

Another (more technical) motivation for adding a case feature to all DPs is that without a -acc feature, we may violate a variant of the SHORTEST MOVE CONSTRAINT when deriving a sentence like

- (286) Which book will the authors seem to like?

With no accusative feature, we would have two movers of category D after the subject *the authors* is inserted and has its -nom feature checked. So without an -acc feature on the fronted *wh*-phrase *which book* the two movers could not be distinguished. However, this argument should be taken with a grain of salt, since even without the additional -acc feature the two movers should be different: the fronted *wh*-phrase *which book* is an \bar{A} -mover, while the subject *the authors* is an A-mover.

Now that we have established that all DPs should carry a case feature (either *-nom* or *-acc*), we need to specify how and where the *-acc* is checked, since unlike subjects direct objects do not obviously move. Since checking a licensing feature implies movement, one solution is to make the movement step required to check *-acc* string vacuous, and so we assume that *-acc* that is checked by a dedicated Accusative phrase which sits between big VP and little vP: with this the lexical item of a simple transitive verb like *like* now contains two head movers as in (287), one of category *Acc* and one of category *v*.

(287) $\langle D \rangle \text{Acc. } v :: \text{like, } +\text{acc} \rangle V. \text{Acc, } \rangle D. V$

One remaining issue is how accusative case is checked for DPs headed by a preposition, as in (288).

(288) They stole from them.

Here we can only offer a slightly unsatisfactory technical trick: prepositions must have complex lexical entries, as exemplified in (289) for the preposition *from*, with an Accusative phrase intervening between the preposition and the DP complement.

(289) $\rangle \text{Acc. } P :: \text{from, } +\text{acc} \rangle D. \text{Acc}$

If we wanted to analyze English prepositional or pseudo passive as in (290) like regular passive, we would also need a standard lexical entry (291) for prepositions.

(290) The issue was talked about.

(291) $\rangle D. P :: \text{from}$

Alternatively, we may assume that there is an accusative checking empty head as in (292), which can be inserted when needed. But this option may overgenerate.

(292) $\langle D \rangle +acc. D$

Note that the solution via covert movement chosen in KOBELE 2015 is not necessarily available to us: so far we only allowed covert movement of the QR type, and assumed that the landing site for QR is inserted into the derivation after the moving DP is merged. Otherwise the DP could be inserted into its scope position via its licensing feature $-s$ (instead of into its base position via D), rendering the movement overt. And this is exactly what would happen here: with the $+acc$ feature in Spec,PP as in KOBELE 2015, the DP would need to be inserted there via $MOVE_E$ and thus pronounced before the preposition. In our solution above, the DP is also pronounced in a specifier position, but in Spec,AccP instead of Spec,PP, thus giving us the correct word order.

It should be noted that the head projecting the Accusative phrase can be the literal carrier of accusative case: it linearly follows the DP carrying accusative, and may thus be interpreted as an accusative suffix. For a much more detailed analysis of structural case suffixes represented as syntactic heads, see CAHA 2009.

3.1.3 A New Perspective on Alternations

Above we concluded that a transitive verb like *like* consists of three projections: VP – AccP – vP, which are bundled together into one complex lexical item. According to our logic of inverse head movement *like* needs to be pronounced in the highest head, so its lexical entry looks like (293).

(293) $\langle D \rangle Acc. v :: like, +acc \rangle V. Acc, \rangle D. V$

Based on this we can now specify lexical entries for intransitive verbs: an unergative verb like *arrive* has the same vP structure as a transitive verb, but its VP takes no arguments, as in (294). In contrast, the lexical entry of an unaccusative verb like *fall* in (295) consists of a simple VP, but for selection purposes it should be labeled vP, so we make it so.

(294) <D >V. v :: like, V

(295) >D. v :: fall

Finally, we can now specify lexical entries for verbs like *break*, which participate in the causative alternation, as shown in (297): the two senses/uses of *break* correspond to the two distinct lexical entries in (296).

(296) a. >D. v :: break

b. <D >V. v :: break, >D. V

(297) a. The plate will break.

b. Mary will break the plate.

Specifying complex lexical entries as in (296) may even provide a new perspective on why some verbs like *break* participate in the causative, while others like *fall* do not, as illustrated in (298).

(298) a. The painting will fall.

b. *Mary will fall the painting.

c. *Mary fell the painting

The contrast between (297) and (298) is a challenge for theories, which separate (sever) the external argument (or more precisely the agent role) from the verb meaning, and instead

introduce the agent role by means of an empty *v*(oice) or little-*v* head (e.g. KRATZER 1996). Assuming that both *break* and *fall* are unaccusative and thus assign a patient role to their respective internal argument (which in the intransitive use moves to Spec,TP), it is mysterious why the little-*v* head can combine with a VP projected by *break*, but not one projected by *fall*.

Under the incremental view on head movement, lexical items enter the derivation fully inflected and at their highest head movement position, and so the difference between the two verbs needs to be encoded in the lexicon: the lexical item *break* can be inserted as a little-*v* with a big *V* as its mover and projecting an agent argument, while *fall* does not project such an agent role. This is admittedly not much more than a restatement of the distributional facts in the lexicon, but such is the nature of the lexicon in lexicalist frameworks.

3.2 Lexicon in Bottom-up and Left-to-right Syntax

In the examples in this chapter we have assumed that the input to our incremental parser consists of a sequence of fully inflected, potentially complex Minimalist Grammar expressions, which are decomposed through inverse head movement, or regular phrasal movement in the case of case morphology and feature checking. Critically, those fully inflected complex MG expressions are the lexical items of our incremental MG parser, and they are different from the lexical items of the bottom-up Minimalist Grammars we started with.

So while in bottom-up Minimalist Grammars (and Minimalism more generally), lexical items always end up as terminal nodes in the derived tree, complex lexical items as used in our incremental derivations can correspond to entire subtrees (or treelets) of the derived tree. Put more colloquially, a word in the input of our incremental parser does not

necessarily correspond to a single terminal node in the derived tree, but may carry within itself a sequence of empty terminal nodes.¹

One may now wonder if and how the two types of lexical items, simple bottom-up and complex incremental items, are linked. If we want our syntactic derivation to also be a full morphological derivation, one rather radical proposal is the following: in bottom-up syntactic derivations, terminal nodes never correspond to ‘words’, instead terminal nodes in the syntax are as proposed in NANOSYNTAX (STARKE 2009): they are either functional heads encoding a single syntactic or morphological feature, or they are lexical roots. In incremental parsing, on the other hand, inputs are complex lexical items: each complex item is pronounced as a single word in the input, and it consists of a sequence of heads of the type used in the bottom-up derivation.

According to this proposal, the complex lexical items of incremental structure building can be viewed as the result of PHRASAL SPELL-OUT as proposed in nanosyntax. This means that inverse head movement is not in fact the inverse of bottom-up head movement, but instead the inverse of phrasal spell-out. This is suggested by the fact that the head movers in inverse head movement, are not just heads, they can also be viewed as incomplete phrases. So it may be possible to translate nanosyntactic analyses into the present incremental framework. One possible obstacle may be the frequent use of remnant movement in nanosyntax, navigating around this obstacle (or arguing it away) is left for future research.

3.2.1 Movement Paradoxes

The distinction between two types of lexical items, simple bottom-up and complex incremental items, may also be useful to help account for ‘movement paradoxes’ (BRESNAN, ASUDEH, TOIVONEN & WECHSLER 2016), where a morphological form of a word can appear

¹ In a bottom-up derivation, a lexical item may also be associated with multiple terminal nodes via head movement, but this is often not viewed as a property of the moving head, but instead as a property of the attracting heads.

in a moved position, but not in its supposed base position. Two prominent examples of movement paradoxes are morphological mismatches in VP fronting and subject-auxiliary inversion.

In the VP fronting examples in (299), only the first one is acceptable, which mirrors the acceptability pattern of the corresponding VPs in their base positions in (300).

(299) She said she would meet you, ...

- a. and meet you she will.
- b. *and met you she will.
- c. *and meeting you she will.

(300) a. She will meet you.

- b. *She will met you.
- c. *She will meeting you.

But with the perfect auxiliary *have*, it is possible to have a fronted VP in the infinitive (BRESNAN ET AL. 2016):

(301) He said he would meet me, ...

- a. and meet me he has.
- b. and met me he has.
- c. *and meeting me he has.

Yet an infinitive is not possible when the VP appears in base position:

(302) a. *He has meet me.

- b. He has met me.

- c. *He has meeting me.

If a fronted VP originates in its base position, as held in transformational grammars, how can the fronted VP in (301) appear in a morphological form which is impossible in its base position?

The same question arises when looking at subject-auxiliary inversion involving negation and the first person singular of the copula *be* (BRESNAN 2001): in the subject-auxiliary inversion case (303), i.e. when the auxiliary is fronted, using a contracted form of negation and the copula is possible in most dialects of English. Yet without subject-auxiliary inversion the same form is unacceptable in the standard dialects.

(303) Aren't I your friend.

(304) I aren't your friend.

Examples like these are key arguments against a transformational account of syntax, and in turn typically ignored by syntacticians working in the transformational tradition (BRESNAN ET AL. 2016). My goal here is not to give a transformational analysis of these movement paradoxes or even a sketch of such an analysis. I only observe that from an incremental perspective, the fronted elements in the above examples never appear in their base position as morphological entities. Instead all that 'reconstructs' into the base position is syntactic features (and a meaning component).

But these examples seem to point to a more fundamental problem with the traditional way of syntax, namely the assumption that the leaf nodes of our syntactic trees are 'words', while at the same time syntax is also the locus of (most of) morphology. Distinguishing bottom-up lexical items from complex incremental items, and noticing that only the latter, but not the former, correspond to words in a sentence, may help resolve this tension

CHAPTER 4

INCREMENTAL SEMANTICS: COMPOSITION AND SCOPE

Like syntactic structure building, semantic composition typically proceeds in a bottom-up manner, and it seems to be rather natural to express semantic derivations in this way: consider how aspect feeds tense (and not vice-versa), and how the entities involved grow in complexity when we climb from the bottom of the clausal spine to the top: (possible) worlds are arguably more complex entities than situations, which in turn are more complex than simple events. In addition, it is generally assumed that truth conditions and other inferences can only be calculated once a sentence is fully composed.

Obviously, this maximally non-incremental conception of traditional theoretical semantics is not compatible with the observation that human language processing is incremental and that human speakers and listeners can assign meanings to incomplete sentences while they process or produce them. And while incremental parsing for transformational grammars has received some attention in the literature, very few papers (BOTT & STERNFELD 2017, BECK & TIEMANN 2018) address the task of building semantic representations incrementally or assigning meanings to incomplete sentences or fragments.

In this chapter, I develop a compositional semantics that is fully compatible with the incremental structure building procedure developed in Chapter 2. More specifically, the compositional operations of the semantics will be driven by the derivational operations of the syntax: every MERGE operation in the syntax triggers exactly one FUNCTION APPLICATION

in the semantics, and the syntactic argument (selectee) is always also the semantic argument of function application. In many implementations of semantics for transformational (and Minimalist) grammars (e.g. HEIM & KRATZER 1998, KOBELE 2006), semantic composition is type-driven: syntax determines which elements combine when, but then it is left to semantic type compatibility to determine which element is the argument and which is the function. In a sense, the semantics developed here may be called syntax-driven, instead of type-driven.¹

The semantics to be developed is a (neo-)Davidsonian event semantics with FUNCTION APPLICATION as the only mode of semantic composition (CHAMPOLLION 2015, see also TOMITA 2016). In our semantics the external argument is severed from the verb, but the internal argument is projected by the verb. This differential treatment of internal and external argument is rooted in the SOV word order of English and the fact that all thematic roles are semantic modifiers.

As a semantics for a movement-based Minimalist Grammars, scope is handled via movement and not via type-shifting as in CHAMPOLLION 2015: first we argue that the semantic counter-part of the syntactic operation MOVE is also FUNCTION APPLICATION, just like for MERGE, and the moved (or licensed) element is always the semantic argument of the licensing head. Second, we discuss the semantics of movement chains and argue that an \bar{A} -moved expression makes separate meaning contributions at the top and the base of the movement chain. The meaning contribution at the base of the chain is the main meaning of the moved item, while the meaning discharged at the top of the chain is provided by the head, which also introduces the licensing feature (like the Q-head). So \bar{A} -movement always reconstructs, except when the moved element is a DP: DPs can take scope via Quantifier Raising licensed by a scope feature $-s$, and if a DP is fronted via

¹ In this sense, the semantics proposed by MONTAGUE (1970, 1973) is also more syntax-driven than type-driven, although a more accurate characterization may be to call MONTAGUE's syntax semantics-driven.

another licensing feature (like $-\text{wh}$), it has two licensing features and can take scope at an intermediate position.

We start with the Compositional Event Semantics developed in CHAMPOLLION 2015 and show how a neat syntax-semantics mapping can be reached with a few additional assumptions. This semantics can be made incremental without introducing any new assumptions or machinery.

Next, we take a step back and bring our semantics more in line with the spirit of transformational syntax by using syntactic movement for all instances of scope taking instead of the type-shifting approach to scope used in CHAMPOLLION 2015. As a first step, we deconstruct semantic modification and adjunction and show that a semantic modifier consists of a core predicate and a type shifter, which is the semantic counterpart of the adjunctivizer introduced in Section 2.3. We also observe that thematic roles are modifiers and thus justify some of the ad-hoc assumptions made to align syntax and semantics.

Finally, we give a semantics of syntactic movement. The core components there are the semantics of MOVE , which will turn out to be Function Application, and the semantics of movement chains. Movement chains will be shown to come in three types: A-movement chains, which are not really chains. \bar{A} -movement chains involving DPs, and \bar{A} -movement chains involving non-DPs.

4.1 Compositional Semantics

A linguistic sign consist of a form and meaning component, and two (or more) linguistic signs can be combined to form a new complex or derived linguistic sign. In Chapter 2, we discussed how the form or pronunciation of a derived sign is determined. In this chapter, we focus on the meaning component and assume that the meaning of a derived sign is determined by the meaning of the individual signs being combined and the mode

of composition. This is known as the PRINCIPLE OF COMPOSITIONALITY. One particularly successful framework for compositional semantics is the type-theoretic semantics based on the λ -calculus developed by MONTAGUE (1970, 1973) and popularized in HEIM & KRATZER 1998.

The basic building block of traditional type-theoretic semantics is the domain of individuals (or entities): a common noun like *cat* is a predicate of individuals as in (305), which is true of all individuals that are cats, and false otherwise.

(305) $\lambda x. \mathbf{cat}(x)$

Similarly, an intransitive verb like *run* is a predicate (306) which is true of all individuals that run and false otherwise.

(306) $\lambda x. \mathbf{run}(x)$

Treating nouns and intransitive verbs as formally the same may seem a bit curious: at least intuitively, nouns are more about individuals than verbs are, while verbs seem to be about events. So instead of denoting a set of individuals who run as in (306), the verb *run* may denote a set of running events. We follow the (neo-)Davidsonian line of event semantics (DAVIDSON 1967, PARSONS 1990, PIETROSKI 2005) and assume that verbs and sentences make reference to events.

Combining type-theoretic compositional semantics with (neo-)Davidsonian event semantics has not always been easy, but CHAMPOLLION (2015) developed a formal framework for Compositional Event Semantics, which we take as the starting point in this section.

4.1.1 Why Events?

Using events is not meant as an ontological or metaphysical commitment. The main reason is mostly formal: in the same way that we treat nouns as predicates of individual variables, verbs are predicates of events. Using events in natural language semantics is sometimes associated with philosophical questions like, are there negative events, are there impossible events, are events world particular or are they the same in all worlds. But notice that we are significantly less strict in the nominal domain, where we typically analyze *a unicorn* in a similar way as *a poet*, ignoring the fact that the former is fictional. And if we assume that common nouns denote individuals, we already include events in our domain of individuals, because what else does a noun like *riot* denote if not an event of rioting? Similarly deverbal nouns and event nominals like *observation* in (307) denote events, as evidenced by their ability to license aktionsart modifiers like *for several hours*.

(307) The observation of aerosol particles for several hours disproved our hypothesis.

So events are entities in our model theory, just like individuals, and we assume that the domain of entities consists of individuals and entities (and possibly other entities like propositional objects). Out of convenience, events are often given a semantic type different from individuals, but there is no semantic justification² to do so except convenience and comprehensibility.

Despite the fact that the introduction of events does not necessarily introduce any new ontological commitments and in some sense reduces the complexity of associated models (by reducing the arity of relations in the model), it is sometimes asked why we would want to introduce events if we can do without them. As appealing as this argument may sound, its razor sharpness only holds when looking at very simple semantic phenomena like the

² This is not to say that the distinction between individuals and events is semantically irrelevant, only that the type system and the mechanics of semantic composition need not make reference to this distinction.

compositionality of verbs and their arguments. Once we move on to realistic fragments, and try to cover phenomena like tense and aspect, this Occam argument actually turns into its exact opposite: any adequate treatment of tense in event-free semantics requires the introduction of some temporal primitives, like intervals, while with events this is not necessarily required (KAMP 1979). And arguably most treatments of aspect require events.

4.1.2 Formal Assumptions and Notation

With this we can lay out the basic assumptions of the framework in which we develop our semantics. We employ a type-theoretic semantics of the type developed by MONTAGUE (1970, 1973) and popularized in HEIM & KRATZER 1998. Following the Neo-Davidsonian line of event semantics (DAVIDSON 1967, PARSONS 1990, PIETROSKI 2005), we assume that verbs make reference to events (and/or states). So we include events in the domain of basic entities. Additionally, the domain of entities also includes propositional objects (MOULTON 2015, MOLTSMANN 2020), which denote the (propositional) content of speech acts and propositional attitude verbs. We thus assume that there are only two semantic types: a type t of truth values over the Boolean domain D_t , and a type e of non-Boolean basic entities whose domain D_e includes individuals, eventualities, propositional objects, etc. Semantic types and domains can now be defined as in (308) and (309), respectively. As explained above, we do not follow the common practice of distinguishing individuals and events in the type system, they are both part of the type and domain of entities, but for readability we sometimes add a subscript v and write the type of an event variable as e_v .

(308) SEMANTIC TYPES

- a. e, t are basic semantic types
(of entities and truth values respectively)
- b. If α and β are semantic types, then $\langle \alpha, \beta \rangle$ is a semantic type

c. Nothing else is a semantic type

(309) DOMAINS

a. $D_e = \{x \mid x \text{ is an individual}\} \cup \{e \mid e \text{ is an event}\}$

b. $D_t = \{\mathbf{true}, \mathbf{false}\}$

We assume only one mechanism of semantic composition: Function Application as defined in (310) takes one element of a complex type $\langle \alpha, \beta \rangle$ and one of (simpler) type α and combines them into one element of type β . There is no explicit operation to form complex types in the compositional semantics, i.e. λ -abstraction is not a semantic operation. Instead we assume that the meanings of some lexical items are of complex semantic type.

(310) FUNCTION APPLICATION (FA)

Let σ and τ be two expressions of types $\langle \alpha, \beta \rangle$ and α . Then the two expressions can be combined into an expression ρ of type β according to

$$\llbracket \rho \rrbracket = \llbracket \sigma \rrbracket (\llbracket \tau \rrbracket).$$

In (310), $\llbracket \cdot \rrbracket$ denotes the semantic value or meaning component of a linguistic sign or a Minimalist Grammar expression. Notice how the definition in (310) makes no reference to branching nodes or other syntactic constructs as e.g. in HEIM & KRATZER 1998. Instead Function Application is defined as an operation over meanings without regards to syntactic configuration, and we will argue that Function Application is the ‘semantic value’ of syntactic MERGE and MOVE.

We assume that one possible representation of linguistic meaning is as a formula in predicate logic, and the goal of compositional semantics as defined here is to construct such a formula for any grammatical sentence. This is not a claim that meanings are formulas in predicate logic, only that meanings can be represented as such.

4.1.3 Compositional Event Semantics

Type-theoretic compositional semantics (MONTAGUE 1970, 1973) and (neo-)Davidsonian event semantics (DAVIDSON 1967, PARSONS 1990, PIETROSKI 2005) have both led to a deeper understanding of the meaning of natural language. Yet combining these two approaches has occasionally been difficult. One of the problems has been the interaction of event semantics with quantificational arguments, and the question where and how the event variable is to be existentially closed: on the one hand, the event variable should be available for modification along the entire clausal spine (at least up to the tense level), but on the other hand, the event quantifier always takes narrowest scope with respect to other quantifiers in the clause. For example, direct objects seems to take scope over the event quantifier: the sentence in (311) does not require that there be a single reading event whose theme was every book on the shelf. Instead (311) is (also) true if there were many different reading events which had the individual books on the shelf as their theme.

(311) John read every book on the shelf.

Based on examples like (311), CHAMPOLLION (2015) proposes that lexical entries of verbs include the event quantifier (or the existential closure over their event argument). So verbs like *jump* do not denote a predicate of events like $\lambda e.\mathbf{jump}(e)$ (of type $\langle e_v, t \rangle$), but instead a predicate (of type $\langle \langle e_v, t \rangle, t \rangle$) of sets of events as in (312).

(312) $\llbracket \mathbf{jump} \rrbracket : \langle \langle e_v, t \rangle, t \rangle = \lambda f_{\langle e_v, t \rangle}.\exists e[\mathbf{jump}(e) \wedge f(e)]$

The expression in (312) is true of any set of events f that contains at least one event of which the predicate is true, i.e. at least one jumping event. Notice that (312) is the result of applying a standard generalized existential quantifier to a simple predicate of events, and that verbs (and verbal projections) now have the same type as DPs: $\langle \langle e, t \rangle, t \rangle$.

With the novel lexical entry for verbs in (312), CHAMPOLLION (2015) then develops a compositional (neo-)Davidsonian event semantics, which uses Function Application as its only mode of semantic composition, and notably does not rely on Predicate Modification as defined in HEIM & KRATZER 1998. Predicate Modification is typically the most important mode of composition in neo-Davidsonian semantics, because it allows two predicates of a given type (e.g. two predicates of events) to compose and form a combined predicate of the same variable. But the same effect can be achieved through function application and the use of higher-order types (and type shifters) for event denoting expressions: $\langle e_v t, t \rangle$ instead of $\langle e_v, t \rangle$.

Arguments of the verb, such as subject or direct object, are linked to the verb's event variable via thematic roles like agent (**ag**) or theme (**th**). Syntactically, thematic roles can be introduced by dedicated functional heads, or they can be projected by the verb. In CHAMPOLLION 2015, thematic roles are introduced by syntactic heads, which first combine with their DP argument and then take the verb (or its projection) as their argument. The functional heads introducing thematic roles also include a type-lifter, which allows arguments to take scope over the event they are thematically related to: as an example, (313) shows the semantic denotation of the head introducing the agent role.

$$(313) \quad \llbracket \emptyset_{\text{ag}} \rrbracket : \langle \langle et, t \rangle, \langle \langle e_v t, t \rangle, \langle e_v t, t \rangle \rangle \rangle \\ = \lambda Q_{\langle et, t \rangle}. \lambda V_{\langle e_v t, t \rangle}. \lambda f_{\langle e_v, t \rangle}. Q(\lambda x. V(\lambda e. [f(e) \wedge \mathbf{ag}(e, x)]))$$

The denotation in (313) implements a semantic treatment of quantifier scope, based on the type-shifting approach developed in HENDRIKS 1993. The alternative, a syntactic treatment of scope via movement, will be explored in Section 4.4 (see also LANDMAN 2000). For the moment, we stick with the semantic treatment of scope and the agent head in (313), which has (or selects?) two semantic arguments: a nominal projection ($\lambda Q_{\langle et, t \rangle}$) and a verbal

projection ($\lambda V_{\langle e_v t, t \rangle}$). These arguments can be filled by the DP *every horse* with its semantic denotation in (314), and the verb *jumped* in (312) according to the syntactic tree in (315).

$$(314) \quad \llbracket \text{every horse} \rrbracket : \langle et, t \rangle = \lambda P. \forall x [\mathbf{horse}(x) \rightarrow P(x)]$$

$$(315) \quad \begin{array}{c} \text{VP: } \langle e_v t, t \rangle \\ \swarrow \quad \searrow \\ \text{AgP: } \langle \langle e_v t, t \rangle, \langle e_v t, t \rangle \rangle \quad \text{jumped} \\ \swarrow \quad \searrow \\ \emptyset_{\text{ag}} \quad \text{every horse} \end{array}$$

First we combine the agent head with the DP *every horse*, yielding the agent-marked subject in (316), which then combines with the verb *jumped* to form the denotation in (317).

$$(316) \quad \begin{aligned} \llbracket \text{AgP} \rrbracket &= \llbracket \emptyset_{\text{ag}} \rrbracket (\llbracket \text{every horse} \rrbracket) \\ &= \lambda V. \lambda f. \forall x [\mathbf{horse}(x) \wedge V(\lambda e. [f(e) \wedge \mathbf{ag}(e, x)])] \end{aligned}$$

$$(317) \quad \begin{aligned} \llbracket \text{VP} \rrbracket &= \llbracket \text{AgP} \rrbracket (\llbracket \text{jumped} \rrbracket) \\ &= \lambda f. \forall x [\mathbf{horse}(x) \rightarrow \exists e [\mathbf{jump}(e) \wedge f(e) \wedge \mathbf{ag}(e, x)]] \end{aligned}$$

The main point to note is that the semantic type of the verbal projection does not change when the verb combines with the subject: both the verb itself and its projection including the subject are of type $\langle e_v t, t \rangle$. Also notice how in (317) the subject takes scope over the event variable; similarly, the head in (318) introducing the theme role allows a direct object to take scope over the event variable.

$$(318) \quad \begin{aligned} \llbracket \emptyset_{\text{th}} \rrbracket &: \langle \langle et, t \rangle, \langle \langle e_v t, t \rangle, \langle e_v t, t \rangle \rangle \rangle \\ &= \lambda Q_{\langle et, t \rangle}. \lambda V_{\langle e_v t, t \rangle}. \lambda f_{\langle e_v t, t \rangle}. Q(\lambda x. V(\lambda e. [f(e) \wedge \mathbf{th}(e, x)])) \end{aligned}$$

With the theme head in (318) we can form the denotation for the VP *read every book* in (319), which is true of any set of events containing a different reading event for each book.

$$(319) \quad \llbracket \text{read every book} \rrbracket = \lambda f. \forall x [\mathbf{book}(x) \rightarrow \exists e [\mathbf{read}(e) \wedge f(e) \wedge \mathbf{th}(e, x)]]$$

The formulas in (319) and (317) are not yet truth evaluable: they still contain the λ -bound variable f , which acts as a continuation variable in the sense of BARKER (2002), DE GROOTE (2006), and needs to be closed off. Closing f is achieved with the help of the sentence-level closure operator in (320), which is assumed to be located at the TP or CP level.

$$(320) \quad \lambda e. \mathbf{true}.$$

With (320) we can reduce (317) to a truth-evaluable formula of first-order logic, as shown in (321).

$$(321) \quad \llbracket \text{VP} \rrbracket (\lambda e. \mathbf{true}) = \forall x [\mathbf{horse}(x) \rightarrow \exists e [\mathbf{jump}(e) \wedge \mathbf{ag}(e, x)]]$$

4.1.4 Aligning Syntax and Semantics

In CHAMPOLLION 2015, all thematic roles are introduced by functional heads, but nothing really hinges on this choice, the thematic roles might as well be projected by the verb (see WILLIAMS 2015 for discussion, and e.g. TOMITA 2016 for an implementation). For reasons that will become clear shortly, we adopt the hybrid position advanced in KRATZER 1996 and assume that the external argument is introduced by its own projection, while the internal argument is introduced by the verb. One good empirical reason for not severing the internal argument from the verb is that the thematic role of the internal argument is

typically verb specific, while the thematic role of external argument is more prototypical (DOWTY 1991, WILLIAMS 2009, 2015).

In terms of the compositional semantics, this means that a transitive verb like *devour* has the meaning in (322), which includes the theme role of the internal argument, but not the agent role of the external argument. (The denotation in (322) is the result of applying the denotation of the theme head in (318) to the verb meaning assumed in (312)).

$$(322) \quad \llbracket \mathbf{devour} \rrbracket : \langle \langle et, t \rangle, \langle e_v t, t \rangle \rangle \\ = \lambda Q_{\langle et, t \rangle} . \lambda f_{\langle e_v t, t \rangle} . Q(\lambda x . [\exists e [\mathbf{devour}(e) \wedge f(e) \wedge \mathbf{th}(e, x)])])$$

The thematic role of the external argument (the agent role) is located in a syntactic head typically called voice head (KRATZER 1996) or (little) *v*, whose semantics is thus as in (323).

$$(323) \quad \llbracket v \rrbracket : \langle \langle e_v t, t \rangle, \langle \langle et, t \rangle, \langle e_v t, t \rangle \rangle \rangle \\ = \lambda V_{\langle e_v t, t \rangle} . \lambda Q_{\langle et, t \rangle} . \lambda f_{\langle e_v t, t \rangle} . Q(\lambda x . V(\lambda e . [f(e) \wedge \mathbf{ag}(e) = x]))$$

$$(324) \quad \text{>V <D. } v :: v$$

Notice how the semantic type of the *v*(oice) head in (323) is the same as its syntactic type in (324): it first combines with a VP (filling its semantic variable $V_{\langle e_v t, t \rangle}$) and then with a DP (filling in for $\lambda Q_{\langle et, t \rangle}$) to yield a *v*P, which denotes a predicate ($\lambda f_{\langle e_v t, t \rangle}$) over sets of events. The same can be said about transitive verbs like *devour*: its syntactic type is shown in (325), while its semantic denotation is as in (322), and the two are perfectly aligned: *devour* selects and combines with a DP to fill its semantic variable $\lambda Q_{\langle et, t \rangle}$ and yields a VP, which denotes a predicate ($\lambda f_{\langle e_v t, t \rangle}$) over sets of events.

$$(325) \quad \text{>D. V :: devour}$$

In the variant of Minimalist Grammars and its semantics developed in this dissertation, the following holds true for all for all linguistic expressions: there is a one-to-one correspondence between the syntactic and the semantic input and output types, respectively: whenever an expression X is selected as a syntactic argument by another expression F , the selected expression X also serves as the semantic argument of the selecting expression F . This means that the semantic counterpart of syntactic MERGE is always Function Application: as illustrated in (326), every application of MERGE with F selecting X corresponds to Function Application with the function F applied to the argument X .

$$(326) \quad \text{MERGE}([\text{>X. F} :: \langle \text{f} \mid F \rangle], [\text{X} :: \langle \text{x} \mid X \rangle]) \longrightarrow [\text{F} :: \langle \text{f x} \mid F(X) \rangle]$$

So the semantics developed so far has a much tighter syntax-semantics mapping than many others in the transformational tradition (but see TOMITA 2016, 2017 for proposals of an event-semantics for Minimalist Grammars with a similarly tight syntax-semantics mapping): semantic composition is fully determined by syntactic composition, and semantic types play no role independently of syntactic types. Instead of being type-driven, the semantics developed here is syntax-driven. But it should be emphasized that this tight syntax-semantics mapping is only possible because of some deliberate choices like incorporating the theme role into the verb, while severing the agent role into its own head. We will be justifying these choices (and investigate full thematic separation) when discussing adjunction and modification in Section 4.3. But first let's make this variant of Compositional Event Semantics incremental to see the basic principles of incremental semantic composition.

4.2 Incremental Compositional Event Semantics

Turning the syntax-driven variant of compositional event semantics just introduced into an incremental semantics is almost trivial: all we need is the ability to build syntactic structure incrementally. So we use the incremental parsing procedure from Chapter 2 to compose both the pronunciation and the meaning components of MG expressions: for the meaning components, syntactic MERGE always corresponds to semantic function application with the selectee being the argument and the selector the function. The operation MERGE \gg , i.e. MERGE involving an incomplete selected item, thus has to be function composition, defined³ as in (327).

$$(327) \quad \llbracket A \rrbracket \circ \llbracket B \rrbracket = \lambda A. \lambda B. \lambda C. \lambda D. A(B(C))(D)$$

The only substantial adjustment for incremental semantic composition is that we need to re-conceptualize how the event predicate $f(e)$ is closed off. Formally, closure still means applying (328) to a clause, but we do not locate this operator in a functional head of CP or TP.

$$(328) \quad \lambda e. \mathbf{true}$$

Instead, we assume that closure is an operation corresponding to truth evaluation and thus outside the semantics proper. In the case of declarative clauses, (328) may be viewed as an assertion operator. Connecting the semantics with our incremental parsing procedure, we locate the closure operator in the start item of category **R**, thus giving the start item an actual lexical entry, as in (329). In Chapter 2, the start item was conceptualized as outside of the syntax proper, and so it makes sense to also locate it outside of semantics proper and assign it the semantic function of an assertion operator.

³ In some derivations, the λ -terms for C and D may need to be swapped.

(329) $\text{>v. R} :: \langle \epsilon \mid \lambda V.V(\lambda e.\mathbf{true}) \rangle$

As a consequence, the CP is a verbal projection and of the same semantic type as a simple (intransitive) verb, which makes the entire clause an extended projection of the verb in the sense of GRIMSHAW (2000). It may seem somewhat strange to assign clauses the type $\langle \langle e_v, t \rangle, t \rangle$, but on closer inspection it is actually rather appealing: since clauses now have the same semantic type as DPs, this may explain, why many clause-embedding verbs (in particular factive and veridical verbs) also embed DPs, as noted in WHITE & RAWLINS 2018. In Chapter 5, we argue that clauses embedded under factive verbs are in fact definite DPs which quantify over propositional objects (MOULTON 2015, MOLTSMANN 2020). Conceptualizing propositions as propositional objects of semantic type e has the advantage that clauses become predicates, allowing us to semantically distinguish logically equivalent clauses.

Now we are already in a position to incrementally parse a simple declarative sentence like (330), and assign its meaning compositionally and incrementally.

(330) A cat ate every rat.

We first give an incremental derivation with the default scoping behavior of the thematic roles included in their lexical entry, and with the theme role projected by the verb. Assuming the framework of inverse head movement from Chapter 3, we assume that the verb *ate* is spelled out in the position of the *v(oice)* head, which carries an empty V head as a mover, as shown in (331). With their full semantic component specified, lexical entries can get rather unwieldy like the entry in (331), which is a single lexical entry consisting of two basic expressions. So we sometimes use variables for the LF components (here M_v and M_V) and specify their contents separately.

- (331) a. $\text{>V. } v :: \langle \text{ate} \mid M_v \rangle,$
 $\text{>D. } V :: \langle \epsilon \mid M_V \rangle$
- b. $M_v = \lambda Q. \lambda V. \lambda f. Q(\lambda x. V(\lambda e [f(e) \wedge \mathbf{ag}(e, x)]))$
 $M_V = \lambda Q. \lambda f. Q(\lambda x. [\exists e [\mathbf{eat}(e) \wedge f(e) \wedge \mathbf{th}(e, x)]]]$

Finally, we also list the full lexical entries for the constituents of *a cat* and *every rat* in (332). They are standard.

- (332) $\text{>N. } D :: \langle \mathbf{a} \mid \lambda N. \lambda P. \exists x [N(x) \wedge P(x)] \rangle$
 $\text{>N. } D :: \langle \mathbf{every} \mid \lambda N. \lambda P. \forall x [N(x) \rightarrow P(x)] \rangle$
 $\text{N} :: \langle \mathbf{cat} \mid \lambda x. \mathbf{cat}(x) \rangle$
 $\text{N} :: \langle \mathbf{rat} \mid \lambda x. \mathbf{rat}(x) \rangle$

With all requirements specified, we can present the derivation. We proceed step-by-step. At the beginning, the parse stack only contains the start entry so we shift the first two items as in (333).

- (333) **SHIFT a, cat**
- $\text{N} :: \langle \mathbf{cat} \mid \lambda x. \mathbf{cat}(x) \rangle$
 $\text{>N. } D :: \langle \mathbf{a} \mid \lambda N. \lambda P. \exists x [N(x) \wedge P(x)] \rangle$
 $\text{>v. } R :: \langle \epsilon \mid \lambda V. V(\lambda e. \mathbf{true}) \rangle$

Now we merge **a** and **cat** and combine the two meanings via Function Application as in (334). Since the determiner **a** selects the noun **cat**, the semantic value of **cat** serves as the semantic argument of **a**.

(334) MERGE_> N

$$\begin{aligned} D &:: \langle \text{a cat} \mid \lambda P. \exists x [\text{cat}(x) \wedge P(x)] \rangle \\ >v. R &:: \langle \epsilon \mid \lambda V. V(\lambda e. \text{true}) \rangle \end{aligned}$$

In (335) we shift the complex lexical item of the verb *ate*. The top two items on the stack have a matching D feature and so we can combine them via MERGE_< as in (336).

(335) SHIFT *ate*

$$\begin{aligned} <D >V. v &:: \langle \text{ate} \mid M_v \rangle, \\ >D. V &:: \langle \epsilon \mid M_V \rangle \\ D &:: \langle \text{a cat} \mid \lambda P. \exists x [\text{cat}(x) \wedge P(x)] \rangle \\ >v. R &:: \langle \epsilon \mid \lambda V. V(\lambda e. \text{true}) \rangle \end{aligned}$$

(336) MERGE_< D

$$\begin{aligned} >V. v &:: \langle \text{a cat ate} \mid M_1 \rangle, \\ >D. V &:: \langle \epsilon \mid M_V \rangle \\ >v. R &:: \langle \epsilon \mid \lambda V. V(\lambda e. \text{true}) \rangle \end{aligned}$$

Since the v(oice) head selects the DP *a cat*, the semantic value of *a cat* serves as the semantic argument of the denotation of the v(oice) head, as shown in (337):

$$\begin{aligned} (337) \quad M_1 &= M_v(\lambda P. \exists x. [\text{cat}(x) \wedge P(x)]) \\ &= \lambda V. \lambda f. \exists x. [\text{cat}(x) \wedge V(\lambda e. [f(e) \wedge \mathbf{ag}(e, x)])] \end{aligned}$$

Next we have a V feature match between the main expression and its mover, so we can apply internal MERGE_I. But since the mover is an incomplete item and has itself an open selector feature >D, we have to apply the forward combinator variant MERGE_I[>] as in (338). Notice that this is standard in our treatment of inverse head movement.

(338) a. $\text{MERGE}_I^{\gg} V$ >D. $v :: \langle \text{a cat ate} \mid M_2 \rangle$ >v. $R :: \langle \epsilon \mid \lambda V.V(\lambda e. \text{true}) \rangle$ b. $M_2 = M_1 \circ M_V$ $= \lambda Q.\lambda f.\exists x [\text{cat}(x) \wedge Q(\lambda y.\exists e [\text{eat}(e) \wedge f(e) \wedge \text{ag}(e) = x \wedge \text{th}(e, y)])]$

In the next step in (339) we shift the determiner *every* and merge it via MERGE_{\gg} . And finally, in (340), we shift the noun *rat* and merge it via $\text{MERGE}_{>}$.

(339) a. SHIFT every >N. $D :: \langle \text{every} \mid \lambda N.\lambda P.\forall x [N(x) \rightarrow P(x)] \rangle$ >D. $v :: \langle \text{a cat ate} \mid M_2 \rangle$ >v. $R :: \langle \epsilon \mid \lambda V.V(\lambda e. \text{true}) \rangle$ $\text{MERGE}_{\gg} D$ >N. $v :: \langle \text{a cat ate every} \mid M_3 \rangle$ >v. $R :: \langle \epsilon \mid \lambda V.V(\lambda e. \text{true}) \rangle$ b. $M_3 = M_2 \circ \lambda N.\lambda P.\forall x [N(x) \rightarrow P(x)]$ $= \lambda f.\lambda N.\exists x [\text{cat}(x) \wedge \forall y [N(y) \rightarrow \exists e [\text{eat}(e) \wedge f(e) \wedge \text{ag}(e) = x \wedge \text{th}(e, y)]]]$

(340) a. SHIFT rat

N :: ⟨rat | $\lambda x. \mathbf{rat}(x)$ ⟩

>N. v :: ⟨a cat ate every | M_3 ⟩

>v. R :: ⟨ ϵ | $\lambda V.V(\lambda e. \mathbf{true})$ ⟩

MERGE_> N

v :: ⟨a cat ate every rat | M_4 ⟩

>v. R :: ⟨ ϵ | $\lambda V.V(\lambda e. \mathbf{true})$ ⟩

b. $M_4 = M_3(\lambda x. \mathbf{rat}(x))$

$= \lambda f. \exists x. [\mathbf{cat}(x) \wedge \forall y. [\mathbf{rat}(y) \rightarrow \exists e [\mathbf{eat}(e) \wedge f(e) \wedge \mathbf{ag}(e, x) \wedge \mathbf{th}(e, y)]]]]$

The final step is to close the event continuation $f(e)$ by merging the assembled vP with the start item, as in (341).

(341) a. MERGE_> v

R :: ⟨a cat ate every rat | M_R ⟩

b. $M_R = \lambda V.V(\lambda e. \mathbf{true})(M_4)$

$= \exists x [\mathbf{cat}(x) \wedge \forall y [\mathbf{rat}(y) \rightarrow \exists e [\mathbf{eat}(e) \wedge \mathbf{ag}(e, x) \wedge \mathbf{th}(e, y)]]]]$

The meaning component of the final Minimalist Grammar expression in (341) is a formula of first-order logic, which represents the truth conditions of the surface scope reading of the sentence in (330), which we started with (ignoring tense). It is worth emphasizing that the above derivation is basically just a syntactic derivation over MG expressions, and the fact that MG expressions consist of a pronunciation and a meaning component means that it is also a semantic derivation.

4.3 Modification/Adjunction

Compositional semantics in the tradition of MONTAGUE (1970, 1973) focuses on predicate-argument relations as the backbone of linguistic meaning, and thus uses Function Application for semantic composition. Neo-Davidsonian event semantics (DAVIDSON 1967, PARSONS 1990, PIETROSKI 2005), on the other hand, is mostly based on modification relations, and thus often uses Predicate Modification as its main mechanism of semantic composition.

The compositional event semantics developed so far uses only Function Application for semantic composition, because of the direct correspondence between syntactic MERGE and semantic Function Application. Semantic modification can be achieved via Function Application if the semantic modifier takes the modified object as a semantic argument. This means that semantic modifiers must have higher-order semantic types, which conflicts with the intuitive idea that an adverb like *quickly* is a simple predicate of events.

The solution to this conflict is that simple predicates need to be type-lifted into semantic modifiers, which take the modified object as a semantic argument via Function Application. This type-lifter is the semantic equivalent of the ADJUNCTIVIZER head we introduced in Section 2.3.2: in the syntax, the Adjunctivizer allows us to reduce adjunction to MERGE. In the semantics, the Adjunctivizer (its semantic counterpart) allows us to reduce Predicate Modification to Function Application.

4.3.1 Adjunctivizer + Function Application = Predicate Modification

The main function of adverbs is to modify verbs, or more precisely the entities predicated over by verbs: eventualities. The sentence in (342) means that there is an event, which is a roaring event (by a lion), and that roaring (event) was loud, as shown in (343). So the adverb *loudly* modifies the event of roaring introduced by the verb *roar*.

(342) A lion roared loudly.

(343) $\exists x[\mathbf{lion}(x) \wedge \exists e[\mathbf{roar}(e) \wedge \mathbf{loud}(e) \wedge f(e) \wedge \mathbf{ag}(e, x)]]$

By itself an adverb like *loudly* is a predicate of events, as in (344). So how does it come to predicate over a specific event like the roaring event in (343)?

(344) $\llbracket \mathbf{loudly} \rrbracket : \langle e_v, t \rangle = \lambda e_{e_v}. \mathbf{loud}(e)$

As a simple predicate of events, the adverb in (344) is not a semantic modifier, it needs to be type lifted to become a modifier of adequate type. The desired modifier meaning of *loudly* would be as in (345): it takes a predicate over sets of events as an argument and returns a predicate over sets of events.

(345) $\llbracket \mathbf{loudly} \rrbracket_{\text{mod}} : \langle \langle e_v t, t \rangle, \langle e_v t, t \rangle \rangle$
 $= \lambda V_{\langle e_v t, t \rangle}. \lambda f_{\langle e_v t, t \rangle}. V(\lambda e. [\mathbf{loud}(e) \wedge f(e)])$

With the meaning in (345), the adverb *loudly* can take the verb *roar* (or an entire VP) as an argument to combine via Function Application as in (346).

(346) $\llbracket \mathbf{loudly} \rrbracket_{\text{mod}}(\llbracket \mathbf{roar} \rrbracket)$
 $= [\lambda V. \lambda f. V([\mathbf{loud}(e) \wedge f(e)])](\lambda f. \exists e [\mathbf{roar}(e) \wedge f(e)])$
 $= \lambda f. \exists e [\mathbf{roar}(e) \wedge \mathbf{loud}(e) \wedge f(e)]$

Lifting an adverb from a simple predicate of events to the desired higher-order modifier meaning can be achieved via the type shifting denotation in (347). Interestingly, the type shifter in (347) has exactly the same signature as the VP-adjunctivizer in (348), which we introduced in Section 2.3.2.

$$(347) \quad \lambda A_{\langle e_v, t \rangle} . \lambda V_{\langle e_v t, t \rangle} . \lambda f_{\langle e_v, t \rangle} . V(\lambda e . [A(e) \wedge f(e)])$$

$$(348) \quad \text{>Adv} \text{ <V. } V :: \emptyset_A$$

They both take an adverb (or simple predicate of events $\lambda A_{\langle e_v, t \rangle}$) as their first argument and a verbal projection ($\lambda V_{\langle e_v t, t \rangle}$) as their second argument, and return a verbal projection ($\lambda f_{\langle e_v, t \rangle}$). So the adjunctivizer in (348), which was introduced on purely syntactic and formal grounds to reduce adjunction to MERGE and make the Minimalist Grammar feature calculus work, can be identified with the type-shifter in (347), which is necessary to combine a simple predicate of events with a verb phrase it is supposed to modify using only Function Application. The full MG expression of a VP-adjunctivizer head can thus be specified as in (349).

$$(349) \quad \text{>Adv} \text{ <V. } V :: \langle \emptyset_A \mid \lambda A . \lambda V . \lambda f . V(\lambda e . [A(e) \wedge f(e)]) \rangle$$

To illustrate how form and meaning are assembled in parallel with the help of the VP-adjunctivizer head in (349), Figure 22 shows a derivation tree of the VP in (350) using MG expressions with both a pronunciation and a meaning component.

$$(350) \quad \text{rain heavily}$$

Our treatment of adjunction is not limited to the verbal domain, in the nominal domain the syntactic adjunctivizer is defined as in (351), while its semantic type-lifting counter-part has the denotation in (352).

$$(351) \quad \text{<Adj} \text{ >N. } N :: \emptyset_A$$

$$(352) \quad \lambda A_{\langle e, t \rangle} . \lambda N_{\langle e, t \rangle} . \lambda x_e [N(x) \wedge A(x)]$$

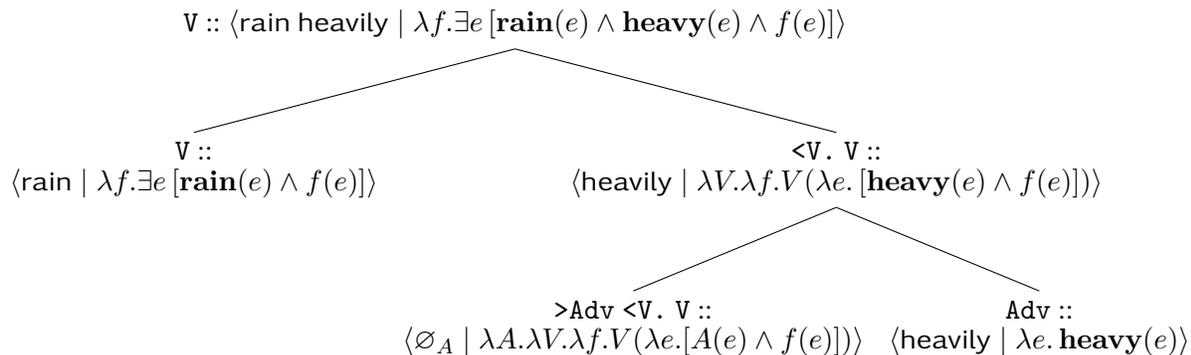


FIGURE 22: Derivation tree for *rain heavily*, using an adjunctivizer head

Putting these together we get the full MG expression (353) of the NP-adjunctivizer.

$$(353) \quad \langle \text{Adj} \rangle N. N :: \langle \emptyset_A \mid \lambda A. \lambda N. \lambda x [N(x) \wedge A(x)] \rangle$$

Since the semantic variable x , which the noun predicates over, is not existentially bound by the noun, but only later by the determiner, the nominal adjunctivizer does not need to ‘scope into’ the noun, as it does in the verbal case. With (353) and the lexical items for *grey* in (354), we can derive the DP in (355).

$$(354) \quad \text{Adj} :: \langle \text{grey} \mid \lambda x. \mathbf{grey}(x) \rangle$$

$$(355) \quad \text{a grey cat}$$

We show the derivation of (355) as an incremental parsing derivation, proceeding step-by-step. In (356), we shift the determiner *a* and the adjective *grey* onto the stack containing only the start item >D. R.

(356) SHIFT a, grey

$$\begin{aligned} \text{Adj} &:: \langle \text{grey} \mid \lambda x. \text{grey}(x) \rangle \\ >\text{N}. \text{D} &:: \langle \text{a} \mid \lambda N. \lambda P. \exists x [N(x) \wedge P(x)] \rangle \\ >\text{D}. \text{R} &:: \langle \epsilon \mid \lambda Q. Q(\lambda x. \text{true}) \rangle \end{aligned}$$

Since there are no matching features to reduce, we shift the NP adjunctivizer, and merge it with the adjective, as in (357).

(357) SHIFT \emptyset_A

$$\begin{aligned} <\text{Adj} >\text{N}. \text{N} &:: \langle \epsilon \mid \lambda A. \lambda N. \lambda x [N(x) \wedge A(x)] \rangle \\ \text{Adj} &:: \langle \text{grey} \mid \lambda x. \text{grey}(x) \rangle \\ >\text{N}. \text{D} &:: \langle \text{a} \mid \lambda N. \lambda P. \exists x [N(x) \wedge P(x)] \rangle \\ >\text{D}. \text{R} &:: \langle \epsilon \mid \lambda Q. Q(\lambda x. \text{true}) \rangle \end{aligned}$$
MERGE_> Adj
$$\begin{aligned} >\text{N}. \text{N} &:: \langle \text{grey} \mid \lambda N. \lambda x [N(x) \wedge \text{grey}(x)] \rangle \\ >\text{N}. \text{D} &:: \langle \text{a} \mid \lambda N. \lambda P. \exists x [N(x) \wedge P(x)] \rangle \\ >\text{D}. \text{R} &:: \langle \epsilon \mid \lambda Q. Q(\lambda x. \text{true}) \rangle \end{aligned}$$

Now we have a matching feature N, and so in (358) combine the two items on the stack via MERGE_{>>}.

(358) MERGE_{>>} Adj
$$\begin{aligned} >\text{N}. \text{D} &:: \langle \text{a grey} \mid \lambda N. \lambda P. \exists x [N(x) \wedge \text{grey}(x) \wedge P(x)] \rangle \\ >\text{D}. \text{R} &:: \langle \epsilon \mid \lambda Q. Q(\lambda x. \text{true}) \rangle \end{aligned}$$

Finally, in (359) we can shift the noun *cat*, merge it with the expression on the stack, and merge the result with the start item, which closes off the continuation term $P(x)$ of the generalized existential quantifier.

(359) **SHIFT** *cat*

$$\mathbf{N} :: \langle \mathbf{cat} \mid \lambda x. \mathbf{cat}(x) \rangle$$

$$>\mathbf{N}. \mathbf{D} :: \langle \mathbf{a\ grey} \mid \lambda N. \lambda P. \exists x [N(x) \wedge \mathbf{grey}(x) \wedge P(x)] \rangle$$

$$>\mathbf{D}. \mathbf{R} :: \langle \epsilon \mid \lambda Q. Q(\lambda x. \mathbf{true}) \rangle$$

MERGE_» **N**

$$\mathbf{D} :: \langle \mathbf{a\ grey\ cat} \mid \lambda P. \exists x [\mathbf{cat}(x) \wedge \mathbf{grey}(x) \wedge P(x)] \rangle$$

$$>\mathbf{D}. \mathbf{R} :: \langle \epsilon \mid \lambda Q. Q(\lambda x. \mathbf{true}) \rangle$$

MERGE_> **D**

$$\mathbf{R} :: \langle \mathbf{a\ grey\ cat} \mid \exists x [\mathbf{cat}(x) \wedge \mathbf{grey}(x)] \rangle$$

So adjuncts in both the verbal and the nominal domain can start out as predicates of events and individuals, respectively, and then be turned into semantic modifiers via a type-shifter. On the syntax side, this type shifter corresponds to an adjunctivizer head, which is necessary to establish the syntactic behavior of adjuncts: they select the phrase they adjoin to, but they do not project.

4.3.2 Thematic Roles as Semantic Modifiers

After appropriate type shifting (i.e. combination with an adjunctivizer head) semantic modifiers (like syntactic adjuncts) are characterized by their category or type preserving signature: as illustrated in (360), a semantic modifier takes an expression of semantic type $\langle \xi, t \rangle$ as an argument via $\lambda X_{\langle \xi, t \rangle}$ and returns an expression of semantic type $\langle \xi, t \rangle$ in the form of the λ -expression λx_{ξ} , which represents a predicate of type $\langle \xi, t \rangle$.

$$(360) \quad =X. X :: \lambda X_{\langle \xi, t \rangle}. \lambda x_{\xi}$$

According to this characterization, thematic roles are semantic modifiers: after it has combined with a DP, the agent role in (361) takes a VP as input ($\lambda V_{\langle e_v, t \rangle}$) and returns a VP ($\lambda f_{\langle e_v, t \rangle}$).

$$(361) \quad \llbracket \emptyset_{\text{ag}} \rrbracket (\llbracket \mathbf{a\ noun} \rrbracket) = \lambda V_{\langle e_v, t \rangle}. \lambda f_{\langle e_v, t \rangle}. \exists x [\mathbf{noun}(x) \wedge V(\lambda e. [f(e) \wedge \mathbf{ag}(e, x)])]$$

Given our tight syntax-semantics mapping, being semantic modifiers implies that thematic roles are syntactic adjuncts. For the agent role this may not seem obvious, because in an SVO language like English the agent (or the subject) precedes the verb, and so the agent modifier is a left adjunct with respect to its host, the verb (phrase). And since left adjuncts behave almost like functional projections, the agent role can be assigned to a functional head, such as the *v(oice)*-head in (362): it takes a VP as its complement and the agent DP as its specifier and projects a *v(oice)P*. Syntactically *vP* and VP are different categories and so the *v(oice)* head does not look like an adjunct, but semantically, VP and *vP* have the same type.

$$(362) \quad >V <D. v :: v$$

For the theme, this strategy is not available: in an SVO language, the theme follows the verb, which makes it a right adjunct of the verb phrase. And right adjuncts are a bit more mysterious. An (empty) syntactic head introducing the theme might look like (363): it first selects the theme-marked DP, then it selects the verb as its second argument, and returns a verbal projection.

$$(363) \quad >D <V. V :: \emptyset_{\text{th}}$$

But this makes the direct object a syntactic adjunct, because the theme head in (363) looks exactly like an adjunctivizer for its DP argument: the only difference between the theme head in (363) and the VP-adjunctivizer in (349) is the syntactic category of the actual adjunct, in (349) it is an adverb, in (363) it is a DP. Alternatively, the theme head could behave more like the agent head in (362) and not project the same verbal category V , which it selects. Instead the theme head could project a theme phrase (ThP) as in (364). But then what is commonly known as VP would need to be called ThP, because the verb is selected as an argument of the theme head and not vice versa.

(364) $\langle D \rangle \langle V \rangle . \text{th} :: \emptyset_{\text{th}}$

These brief observations may be added to the empirical and theoretical arguments discussed in KRATZER (1996) to sever the external argument from the verb into a $v(\text{oice})$ head containing the agent role, but let the theme role (or the internal argument more generally) be projected by the verb (but see WILLIAMS (2009) for arguments that all thematic roles are verb dependent).

Of course it is still possible to fully separate the theme role from the verb, but the head carrying the theme will need to look differently from the head carrying the agent role. In order to see this, we first need to disentangle thematic roles and argument scope.

4.3.3 Thematic Roles and Scope

The main empirical motivation behind the framework of Compositional Event Semantics (CHAMPOLLION 2015) is the fact that events always take scope lower than all arguments of the corresponding verb. To implement this lowest-scope behavior of the event quantifier, CHAMPOLLION introduces the event quantifier with the verb and adopts the type lifting approach of HENDRIKS (1993) to ensure that all its arguments take scope over the event

variable. So in the meaning of the agent role in (365), the scoping behavior of the external argument is hard-coded by nesting the VP argument V within the DP argument Q .

$$(365) \quad \llbracket \emptyset_{\text{ag}} \rrbracket = \lambda Q_{\langle et, t \rangle} . \lambda V_{\langle e_v t, t \rangle} . \lambda f_{\langle e_v, t \rangle} . Q(\lambda x . V(\lambda e . [f(e) \wedge \mathbf{ag}(e, x)]))$$

Adopting the type-lifting approach of HENDRIKS (1993) allows for a purely semantic treatment of argument scope. But argument scope can also be handled in the syntax via Quantifier Raising, and a syntactic treatment of scope is fully compatible with event semantics (cf. LANDMAN 2000; this option is also acknowledged in CHAMPOLLION 2015). In Section 4.4 we develop a syntactic approach to scope in general and argument scope in particular, and so we can remove any scoping provisions from the meaning of the thematic role heads.

If we ignore their scoping behavior, thematic roles are just plain semantic modifiers, and including no scope information in their semantic denotation makes them take scope below the event they modify. So a universally quantified DP like *every noun* marked as an agent would have the meaning in (366), where we use a universal quantifier to make the scoping behavior more clearly visible.

$$(366) \quad \llbracket \emptyset_{\text{ag}} \rrbracket(\llbracket \mathbf{every\ noun} \rrbracket) : \langle \langle e_v t, t \rangle, \langle e_v t, t \rangle \rangle \\ = \lambda V_{\langle e_v t, t \rangle} . \lambda f_{\langle e_v, t \rangle} . V(\lambda e . [f(e) \wedge \forall x [\mathbf{noun}(x) \rightarrow \mathbf{ag}(e, x)]])$$

As a thematic role, (366) has the type signature $\langle \langle e_v t, t \rangle, \langle e_v t, t \rangle \rangle$ of a semantic modifier, so it should be composed of a predicate of events and an adjunctivizer. The predicate corresponding to the agent marked DP in (366) is shown in (367).

$$(367) \quad \lambda e . \forall x [\mathbf{noun}(x) \rightarrow \mathbf{ag}(e, x)]$$

From (367) we can distill the core of the agent role as (368), or more generally for any thematic role, including prepositions, as (369).

$$(368) \quad \lambda Q_{\langle et, t \rangle}. \lambda e. Q(\lambda x. \mathbf{ag}(e, x))$$

$$(369) \quad \llbracket \emptyset_{\text{role}} \rrbracket : \langle \langle et, t \rangle, \langle e, t \rangle \rangle \\ = \lambda Q_{\langle et, t \rangle}. \lambda e. Q(\lambda x. \mathbf{role}(e, x))$$

$$(370) \quad >D. P$$

The meaning and semantic type in (369) is fully aligned with the syntactic type of prepositions in (370): it takes a DP as argument ($\lambda Q_{\langle et, t \rangle}$) and returns a new category P. Semantically, the new category P is a simple predicate of entities (type $\langle e, t \rangle$), either individuals or events, just like a noun of category N is a predicate of individuals (type $\langle e, t \rangle$). Taken together, (370) and (369) indicate that a full MG expression for a preposition like *in* must look like (371).

$$(371) \quad >D. P :: \langle \text{in} \mid \lambda Q. \lambda q. Q(\lambda x. \mathbf{in}(q, x)) \rangle$$

4.3.4 Full Thematic Separation and Exotransitives

In our semantics, we severed only the external argument from the verb and let the internal argument be projected by the verb (KRATZER 1996), but purists of a neo-Davidsonian approach may prefer to see full thematic separation in which the verb does not project any thematic role.

Continuing the arguments from Section 4.3.2, we can separate the thematic role of the internal argument (the theme) into its own head, but unlike the agent head, the theme head can only consist of a thematic core as in (369) without the associated type lifter or adjunctivizer. For a verb to select a thematic head like (369), the type-lifting normally

performed by the adjunctivizer needs to be included in the verb. So a verb selecting a theme head will need to have a denotation like the one in (372).

$$(372) \quad \llbracket \mathbf{verb} \rrbracket : \langle \langle e_v, t \rangle, \langle e_v t, t \rangle \rangle \\ = \lambda R_{\langle e_v, t \rangle} . \lambda f_{\langle e_v, t \rangle} . \exists e [\mathbf{verb}(e) \wedge R(e) \wedge f(e)]$$

This approach to (almost) full separation of the theme role from the verb can also be applied to verbs whose internal argument takes the form of an oblique, or a prepositional phrase. If such ‘exotransitive’ verbs (SCHUMACHER 2018) are to be analyzed in a way that locates the thematic role of the internal prepositional object in the preposition, then the verb itself needs to have a meaning like the one in (372). If we compare (372) to the verb meaning (322) with the theme role included, we notice that the transitive meaning is now distributed between the verb and the preposition.

4.3.5 Interim Summary

In the first half of this chapter, we developed a compositional semantics that is fully compatible with the incremental structure building procedure developed in Chapter 2. The semantics is a (neo-)Davidsonian event semantics with FUNCTION APPLICATION as the only mode of semantic composition, such that every MERGE operation in the syntax triggers exactly one FUNCTION APPLICATION in the semantics, and the syntactic argument (selectee) is always also the semantic argument of function application. In our semantics the external argument is severed from the verb into a v(oice) head, but the internal argument is projected by the verb, and we argued that this differential treatment of internal and external argument is rooted in the SOV word order of English and the fact that all thematic roles are semantic modifiers.

In the following, we will assume that the LF component of the v(oice) head is a composition of the (right) adjunctivizer and the simple agent role. Similarly, the LF component of a V head projecting the internal argument is the result of an adjunctivizer combining with the bare theme role and the verb.

4.4 Semantics of Movement

In Chapter 2, we argued that in order to make the syntax of movement compatible with incremental structure building, we need to invert the movement chains: a moved element gets inserted into its final landing site, where it discharges its pronunciation. It then becomes a mover consisting of its remaining syntactic features and its meaning (or LF), which can be merged into the base position of the movement chain. The basic idea of inverse chain formation is sketched in (374) for the sentence in (373).

(373) Who do you like?

(374)

$$\begin{array}{l}
 +\text{wh} >V. C :: \langle \text{do} \mid \llbracket \text{do} \rrbracket \rangle \\
 -\text{wh} D :: \langle \text{who} \mid \llbracket \text{who} \rrbracket \rangle \\
 \\
 \text{MOVE}_E \text{ wh} \\
 \\
 >V. C :: \langle \text{who do} \mid ? \rangle, D :: \langle \epsilon \mid \llbracket \text{who} \rrbracket \rangle \\
 \quad \vdots \\
 >D. C :: \langle \text{who do you like} \mid \llbracket \text{who do you like} \rrbracket \rangle, D :: \langle \epsilon \mid \llbracket \text{who} \rrbracket \rangle \\
 \\
 \text{MERGE}_I^> D \\
 \\
 C :: \langle \text{who do you like} \mid \llbracket \text{who do you like} \rrbracket \langle \llbracket \text{who} \rrbracket \rangle \rangle
 \end{array}$$

Forming a movement chain as illustrated in (374) gives us full reconstruction: the meaning $\llbracket \text{who} \rrbracket$ of the moved element is carried along the movement chain and ultimately merged

into its base position. This means that the only meaning contribution a moved element can make is in its base position. But we know that moved elements can also make meaning contributions in their moved positions: the topicalized direct object *every book* in (375) takes scope in its moved position, and the *wh*-element *who* in (373) and in (376) contributes to the question meaning in its moved position.

(375) Every book, two critics read.

So a moved expression makes separate meaning contributions at the top of the inverse movement chain and at its base. We argue that the meaning contribution at the base of the chain is always the core meaning of the moved item, i.e. the meaning the moved item would have if it was not moved. This aspect of the semantics of movement chains is accurately captured in the derivation in (374). What is not captured in (374) is what happens at the top of the movement chain, namely how the moved element *who* in (376) can contribute to the question meaning in the matrix clause, even though it reconstructs to the embedded clause. The question about the meaning contribution of moved elements at the top of the movement chain has at least two different aspects: first, what is the semantic function of movement chains, and what is the meaning contribution at the top of the chain? Second, what semantic operation corresponds to the syntactic operation MOVE, i.e. what is the meaning written as a ? in (374)?

(376) Who does Simone think that Simon likes?

The main semantic function of movement chains is scope taking, and the meaning discharged at the top of an \bar{A} -movement chain is a scope-taking meaning, which is introduced by the same functional head, which also introduces the licensing feature. Letting the moved meaning and the licensing feature be introduced by the same head, reflects the fact that

the two are obviously related. In the case of *wh*-questions, the head introducing the *-wh* feature and the question meaning contribution of the *wh*-expression is the Q head (or particle, CABLE 2007). So \bar{A} -movement always makes two meaning contributions, one at the base of the chain (where the moved element discharges its core meaning), and one at the top of the chain. This implies that \bar{A} -movement always reconstructs and always takes scope, at least according to the definition of scope provided below.

In order to make a meaning contribution at the top of the chain, the meaning of the moved element has to combine with the meaning of the licensing head. We argue that this is done via Function Application: the semantic equivalent of the structure building operation MOVE is the same as that of MERGE: Function Application.

4.4.1 Semantics of MOVE

We the first part of this chapter we developed a syntax-semantics mapping according to which all instances of the Minimalist structure building operation MERGE correspond to the semantic operation of Function Application, with the selecting expression the function and the selected expression the argument. Focusing now on MOVE, we may wonder which semantic operation corresponds to the structure building operation MOVE. Curiously, this question is rarely asked in standard treatments of movement (but see KLECHA & MARTINOVIĆ 2015). The operation MOVE is triggered by a matching licensing feature on two expressions and in a sense it combines those two expressions just like MERGE combines two expressions. This MERGE-like nature of MOVE is most obvious in the case of the binary external MOVE_E, but all variants of MOVE take two arguments.

We assume that a mover discharges a meaning component at each of its landing sites, and so that meaning component must combine with the meaning of the licensing expression. And since our only mode of semantic composition is Function Application, MOVE also has to correspond to Function Application. This brings us to one of our main

results: Function Application is the semantic equivalent of Minimalist MERGE and MOVE, and the selected or licensed element is always the semantic argument of the selecting or licensing expression or head.

4.4.2 Generalizing Scope Beyond Generalized Quantifiers

We assume that the main function of movement chains is scope taking, but what do we mean by scope taking? For quantificational elements like DPs this seems obvious: a DP like (377) denotes the generalized quantifier expression in (378) and takes scope over whatever gets inserted for the predicate variable P .

(377) every book

(378) $\lambda P_{\langle e,t \rangle}.\forall x[\mathbf{book}(x) \rightarrow P(x)]$

A different characterization is that the generalized quantifier in (378) takes scope over its semantic argument P , and according to BARKER (2015) an expression X takes scope over an expression Y if Y serves as the semantic argument of X . This definition of scope taking can easily be extended beyond generalized quantifiers. The negative adverb/particle *not* used to express linguistic negation may be assigned the denotation in (379): negation takes scope over whatever gets inserted for the variable P , or whatever the expression in (379) takes as its semantic argument.

(379) $\lambda P.\neg P$

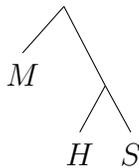
This notion of scope can even be extended to elements that do not take scope in the traditional sense, thus allowing expressions like the fronted VP in (380) to ‘take scope’ in its moved position.

(380) Admired, John was.

All the fronted VP in (381) has to do to ‘take scope’ is take the rest of the sentence as its semantic argument. In the case of the fronted VP in (380) this type of ‘scope taking’ may not have any semantic effect, but that is of no immediate concern.

What is of concern is how exactly a fronted element can take the rest of the sentence as a semantic argument. Here we assume that it is not the moved element *per se* that takes its scope as a semantic argument, but instead it is the head, which licenses the movement and hosts the landing site. The licensing head H combines with the moved phrase M via MOVE, which corresponds to semantic Function Application, and the resulting expression $H(M)$ then takes the scope of the moved expression as its semantic argument S .

(381)



(382) $H(M)(S)$

So a tree structure like the one in (381) corresponds to the semantic composition in (382): from this we see that the semantic denotation of the head H is critical in determining how a mover M takes scope. The semantic composition only guarantees that the scope term S is the semantic argument of the ‘scope-taking’ mover M combined with its licensing head H ; for M to take scope in the traditional sense, the head H needs to have an appropriate scope taking denotation like the one in (383), where M could be filled by a DP taking scope over the predicates of individuals in S .

$$(383) \quad \llbracket H \rrbracket : \langle \langle et, t \rangle, \langle \langle e, t \rangle, \langle et, t \rangle \rangle \rangle \\ = \lambda M_{\langle et, t \rangle} . \lambda S_{\langle e, t \rangle} . \lambda f_{\langle e, t \rangle} M(\lambda x [S(x) \wedge f(x)])$$

Incidentally, this notion of scope-taking implies that all c-command relations are scope-taking relations, see Section 4.4.6 for further discussion.

Let's illustrate the basic principle with a quantifier raising example. In Section 2.5.3 we argued that the landing site for covert quantifier raising (i.e. the scope taking position) can be introduced by an empty head like (384), to which we already applied the left-to-right transform.

$$(384) \quad +s = T. \quad T :: \emptyset_s$$

$$(385) \quad \llbracket \emptyset_s \rrbracket : \langle \langle et, t \rangle, \langle \langle e_v t, t \rangle, \langle e_v t, t \rangle \rangle \rangle \\ = \lambda Q_{\langle et, t \rangle} . \lambda V_{\langle vt, t \rangle} . \lambda f_{\langle e_v, t \rangle} . Q(\lambda x . V(\lambda e . f(e)))$$

The head in (384) expects a quantificational DP to satisfy its licensing feature, and then selects a TP. Assuming that both MERGE and MOVE correspond to Function Application, this determines the semantic type of (384) as $\langle \langle et, t \rangle, \langle \langle vt, t \rangle, \langle e_v t, t \rangle \rangle \rangle$. If we want the quantificational DP to take scope over the TP, the head in (384) must be can assigned the semantic value in (385): it takes two arguments, a quantifier over individuals ($\lambda Q_{\langle et, t \rangle}$) and an expression denoting a verbal projection ($\lambda V_{\langle e_v t, t \rangle}$), and returns an expression denoting a verbal projection ($\lambda f_{\langle e_v, t \rangle}$). Similarly, a T head or tense auxiliary like *will*, which licenses the external argument (or subject) may be defined as in (386).

$$(386) \quad \text{a. } +\text{nom} > \text{v}. \quad T :: \langle \text{will} \mid M_T \rangle \\ \text{b. } M_T : \langle \langle et, t \rangle, \langle \langle e_v t, t \rangle, \langle e_v t, t \rangle \rangle \rangle \\ = \lambda Q_{\langle et, t \rangle} . \lambda V_{\langle e_v t, t \rangle} . \lambda f_{\langle e_v, t \rangle} . Q(\lambda x . \exists t . V(\lambda e [f(e) \wedge \tau(e) > t]))$$

Just letting a quantifier take an expression as a semantic argument is not the whole story of scope taking though, the variable bound by the quantifier must also occur within the scope expression, otherwise there is no scope effect, as illustrated in (387) where the individual variable x bound by the universal quantifier does not occur within its scope.

(387) $\forall x[\mathbf{noun}(x) \rightarrow \exists e. \mathbf{verb}(e)]$

But in (386), there is nothing to ensure that the variable bound by the generalized quantifier expression inserted for Q appears in the scope expression. In our movement-based approach to scope, ensuring this is the function of the movement chain.

In our model of inverse movement, we assume that an expression X with a licensing feature $-\mathbf{x}$ has (at least) two meaning components: one to be discharged where the licensing feature $-\mathbf{x}$ is checked, i.e. in the moved position, and one to be discharged in the base position where the \mathbf{x} -based movement chain ends. In order to make all (phrasal) movement scope-taking, we have to stipulate that the meaning component discharged when the licensing feature is checked is always the scope-taking component. If we want movement to be scope-taking in the traditional sense, we have to prevent vacuous scope taking and ensure that the second component contains at least one instance of the variable quantified over in the scope-taking component.

Depending on how strictly the above requirements are ensured, we get different types of movement and scoping behavior: an \bar{A} -movement expression always has two meaning components, one for the top of the chain and one for the bottom of the chain, but depending on the specific type of chain (or head introducing the licensing) the meaning at the top of the chain may or may not be linked to the meaning at bottom of the chain. If the two meaning components are linked (and the licensing head has the appropriate denotation), the corresponding \bar{A} -movement chain represents a traditional scope relation.

If the two meaning components are not linked by a shared variable or if the top meaning is trivial, the \bar{A} -movement chain reconstructs with respect to scope. But critically, with respect to binding \bar{A} -movement always reconstructs, because there is always a non-trivial meaning component discharged at the base of the chain.

In contrast to this, an A-movement expression has effectively only one meaning component, which is discharged at the top of the chain, while the second meaning component at the base of the chain is just a bound variable. And so an A-moved expression always takes scope in its moved position and does not reconstruct.

4.4.3 A-Movement: Scope and Case

An expression with an A-licensing feature discharges its full meaning component at the position where it is inserted and its A-movement feature is checked (i.e. at its moved position), and the meaning component of the mover consists of nothing but the variable bound by the quantifier introduced in the moved position. And since movement chains are discharged from top to bottom, the variable carried along the A-movement ‘chain’ is always properly bound. We thus assume that A-movement is handled via the Minimalist Grammar operation Move_E , and not via hypothetical reasoning as presented in Section 2.5.1. But since A-movement does not really create a chain, our treatment of A-movement is essentially a notational variant of hypothetical reasoning.

In our notation, an A-movement expression can be represented as in (388): since the A-licensing feature $-\text{nom}$ is inherent, the two meaning ‘components’ needed for the movement chain must also be inherent of a DP. This contrasts with the meaning components of \bar{A} -movement expressions like (387): like the licensing feature, the non-primary meaning component can be introduced by an empty head like Q.

(388) $-\text{nom D} :: \langle \mathbf{a\ cat} \mid \lambda P. \exists x[\mathbf{cat}(x) \wedge P(x)], x \rangle$

This means that all A-movement is scope taking, and there is no (scope) reconstruction for A-movement: an A-moved expression makes its core meaning contribution at its moved position and fills its base position with nothing but a bound variable. This is a welcome result: CHOMSKY (1995) discusses the established observation that in (389) *everyone* cannot take scope under negation.

(389) Everyone seems not to be there yet. $\forall \gg \neg$

Since in (389) *everyone* raised from a position under negation via A-movement, CHOMSKY concludes that A-movement cannot reconstruct. Similarly, LASNIK (1999) observes that in an ECM-context like (390) with subject-to-object A-movement, the universal quantifier *every defendant* cannot take scope under negation.

(390) Mary proved every defendant not to be guilty during his trial. $\forall \gg \neg$

So it seems that A-movement does not reconstruct for scope-taking purposes. And with respect to binding, A-movement does not seem to reconstruct either:

(391) Every cat_i seems to its_i owner to be the cutest.

(392) #Which cat_i did its_i owner like the most?

In the A-movement case (391), the noun *cat* can serve as the antecedent of the pronoun *its* from its moved position, suggesting that it does not reconstruct for binding purposes. In contrast, the \bar{A} -movement example in (392) is infelicitous, suggesting that from an \bar{A} -position the noun *cat* cannot serve as the antecedent of *its*.

There is a substantial literature debating the reconstruction behavior of A-movement, and four examples cannot possibly justify the conclusion that A-movement never re-

constructs. So here we just assume that it is true, and discuss the consequences of the assumption that all A-movement is scope taking, and the connection between A-movement and case.

If all DP arguments have to move for case reasons, both the subject and the direct object will end up taking scope above the VP and so above the event variable. This removes their implausible lowest scope reading, which we introduced when deconstructing the thematic roles in Section 4.3.3 (see CHAMPOLLION 2015 for arguments why the lowest scope reading is implausible). It also simplifies the denotation of the agent-role-introducing v(oice) head to the one given in (393), where the external argument fills the slot of a simple individual-type variable (λx_e).

$$(393) \quad \llbracket v \rrbracket : \langle e, \langle \langle e_v t, t \rangle, \langle e_v t, t \rangle \rangle \rangle \\ = \lambda x_e . \lambda V_{\langle e_v t, t \rangle} . \lambda f_{\langle e_v, t \rangle} . V(\lambda e [f(e) \wedge \mathbf{ag}(e) = x])$$

So in summary case-driven A-movement solves a number of issues, depending on the theoretical perspective. From the perspective of CHAMPOLLION's (2015) compositional event semantics, A-movement can be used to implement a syntactic theory of argument scope: all verbal arguments have to take scope above their respective event variable, and all argument DPs have to move for case reasons to a scope taking position above the verb, so if all verbal arguments take scope via A-movement, a head introducing a thematic role like the v(oice) head can have the rather simple denotation in (393).

From a classical Montagovian perspective, notice that the argument variable in (393) is of simple type e . This is the famous mismatch between the type e of the direct object selected by the verb and the type $\langle \langle e, t \rangle, t \rangle$ of generalized quantifiers assigned uniformly to all DPs. Like in HEIM & KRATZER 1998 this type mismatch is resolved via movement, but in our case the movement is not forced by type considerations. Instead the syntactic

obligatoriness of A-movement for case reasons allowed us to assign the classical types to the arguments of the verb. With this the V head corresponding to a transitive verb like *devour* now has the almost classical denotation in (394).

$$(394) \quad \llbracket V_{\text{devour}} \rrbracket : \langle e, \langle e_v, t \rangle \rangle = \lambda x_e. \lambda f_{\langle e_v, t \rangle}. \exists e [\mathbf{devour}(e) \wedge \mathbf{th}(e, x) \wedge f(e)]$$

Finally, we can also assign a meaning to the Accusative head: it projects a scope position as its specifier, so its lexical entry is as in (395).

$$(395) \quad +\text{acc} > \mathbf{V}. \text{Acc} :: \langle \epsilon \mid \lambda Q \lambda V. \lambda f. Q(\lambda x. V(\lambda e. f(e))) \rangle$$

4.4.4 Semantics in \bar{A} -Chains

For \bar{A} -movement I assume that the moved expression always has two meaning components: one for the licensing position, i.e. the top of the chain, and one for the bottom of the corresponding movement chain. The bottom meaning component is the core meaning of the moved item, i.e. the meaning it would have in a non-moved position. The top meaning component is provided by the head introducing the licensing feature. So it is to be expected that \bar{A} -movement always reconstructs: the core meaning of the mover must be discharged in the base position, because the meaning component discharged at the top of the chain was added by the head that also added the \bar{A} -licensing feature.

But since the licensing head takes the rest of the derivation as its semantic argument, we can maintain that \bar{A} movement is always scope-taking, at least under an appropriate definition of scope. So \bar{A} -movement reconstructs **and** takes scope. This apparent contradiction can be resolved by observing that while \bar{A} -movers make a meaning contribution at the top and bottom of their respective chain, those meaning contributions need not be the same (or exact copies) as is assumed under the copy theory of movement. In addition, our

notion of scope taking is weaker than the traditional one and just means ‘taking the scope terms as a semantic argument’.

Employing a more traditional notion of scope, the scoping behavior of \bar{A} -movement is more complicated, and it may look like there are at least two types of \bar{A} -movement: scope-taking movement and movement that reconstructs (JOHNSON 2012, POOLE 2017). A prime example of \bar{A} -movement that allegedly allows the moved phrase to either take scope in its moved position or to reconstruct and take scope in its base position is standard *wh*-movement for question formation. A particularly illustrating case are *how-many*-questions like (396), which have two distinct readings: one in which *how many books* takes scope above the modal *should*, and one where it takes scope below (see Section 2.5).

(396) How many books should the critics review this week?

But the existence of the low scope reading does not imply that under that reading the *wh*-phrase *how many books* does not take scope in Spec,CP. After all the *wh*-phrase makes a contribution to the meaning of the question, and does so from the Spec,CP position under both readings.

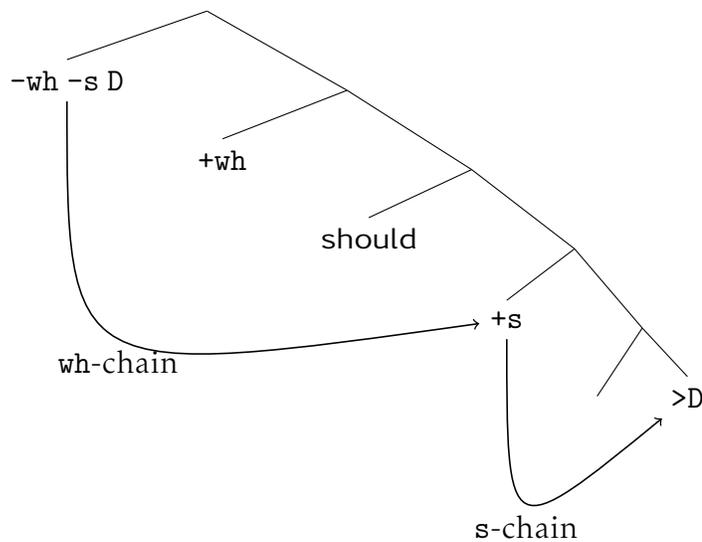
What enables the two readings of (396) is the fact that the phrase *how many books* contains two scope taking elements: one *wh*-element and one quantifier over individuals as part of the meaning of *many*. So the MG expression of *how many books* reads as in (397).

(397) $-\bar{w}h$ $-\bar{s}$ D :: how many books

Based on our characterization of \bar{A} -features as optional (see Section 2.5.2) we assume that the two licensing features in (397) can be introduced by adjoining heads, in particular the $-\bar{w}h$ feature is introduced by a Q-head (CABLE 2007). What is more important is that the two licensing features can be discharged in different positions as illustrated in (398): the

$-wh$ -feature is always checked by the question complementizer, but the scope feature $-s$ can be checked in any possible scope position. At that position the chain started by the feature $-wh$ (the wh -chain) ends, and a new s -chain starts by checking the $-s$ feature. If $-s$ is checked above the modal *should* the *many*-quantifier takes the entire future derivation as its semantic argument and so scopes over the modal. If $-s$ is checked below the modal *should* as in (398), the *many*-quantifier only takes scope over the lower part of the derivation.

(398)



Importantly, in both cases the meaning component corresponding to the $-wh$ -feature takes scope in the same high position. Only the scope taking position of the meaning component associated with the scope feature $-s$ can vary. So there are two connected movement chains: the wh -chain starts at Spec,CP (where the *wh* is inserted and $-wh$ is checked) and ends at the position, where the feature $-s$ is checked and *many* takes scope. That position is also the start of the s -chain, which in turn ends at the complement position of *review*. This example also illustrates that feature-checking (or internal) $MOVE_C$, like all variants of $MOVE$, has Function Application as its semantic operation, otherwise the *many*-quantifier would not be able to take scope.

4.4.4.1 Reconstruction Effects

The determiner *many* is not the only element introducing its own quantifier and scope feature $-s$. All determiners are quantifiers, and so we expect that all DP-*wh* be able to either reconstruct or take scope, in the traditional sense, since all DP-*wh* can have both a $-wh$ feature and a $-s$ feature. All other types of moved phrases, in particular all non-DP *wh*-phrases, cannot take scope in their moved positions and thus reconstruct, like the *wh*-phrase *how* in (399) carries, which a single $-wh$ licensing feature, but no additional $-s$ scope licensing feature.

(399) How should Mary paint the house?

The $-wh$ licensing feature is checked at Spec,CP and the corresponding meaning component contributing to the question meaning is discharged. The remaining selection feature and the non-question meaning of *how* form a mover, whose meaning is discharged when the selection feature is checked, i.e. in the base position (wherever that may be).

So only movement of a DP can be scope-taking in the traditional sense, because only DPs (denote generalized quantifiers and) can receive an additional scope licensing feature $-s$. Expressions with only a $-wh$ -licensing feature reconstruct. This is a reformulation of a basic generalization stated by POOLE (2017), who distinguishes scope-taking DP-movement (our s -chains) from movement of Q(uestion particle) phrases (or QP), our wh -chains, which always reconstructs.

POOLE mostly discusses examples like (400), which is interesting, because aside from the moved phrase it is identical to (399), but unlike (399) the moved phrase is a DP.

(400) Which color should Mary paint the house?

Yet the DP in (400) is different from ordinary DPs, since it denotes a property (type $\langle e, t \rangle$), and not a simple entity (type e), and so the example in (400) can only be QP-movement, and not scope-taking DP-movement. Trace positions that are not of simple entity type correspond to the ANTI-PRONOMINAL CONTEXTS in POSTAL 1998: they only allow reconstructing QP-movement ('A-TYPE extractions'), but not scope-taking DP-movement ('B-TYPE extractions'). For a detailed analysis of POSTAL's (1998) distinction in these terms, see POOLE 2017. For an explanation of why A-Type extractions are blocked by weak (or selective) islands, but B-Type extractions are not, see Section 5.2.5.

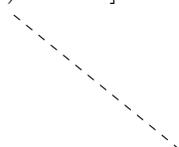
In terms of inverse movement chains, we can say that the meaning component corresponding to the $-\text{wh}$ feature (introduced by the Q-head) is in some sense rather independent from the core meaning component to be discharged in the base position. This is the norm for \bar{A} -movement chains: their meaning components are mostly independent of each other. The exception to this stipulation are the two meaning components associated with s-chains: to allow an actual scope taking relation between the two meaning components, the top and base meaning must share (i.e. predicate over) a common semantic variable, and so they are rather dependent on each other.

4.4.4.2 Taking Scope in \bar{A} -Chains

The prototypical case of movement for scope taking is Quantifier Raising, where a generalized quantifier takes scope from a position higher than its base position. In Section 2.5.3 we assumed that (covert) Quantifier Raising is triggered by a $-s$ licensing feature, which is introduced onto a DP by a functional head \emptyset_s . The functional head \emptyset_s also adjusts the meaning component of the DP it combines with. But unlike the Q head, which introduces the $-\text{wh}$ feature and a meaning component that can potentially be disconnected from the

base meaning, after the adjustment by \emptyset_s the two meaning components of the DP must be connected.

One possibility to connect the top and the base meaning is to take inspiration from Fox (1999, 2002) and assume that the base meaning (the meaning discharged in the position of the movement trace) is subject to Trace Conversion, while the top meaning contains the actual scope taking quantifier. So like all \bar{A} -movement chains, a scope-taking s -chain makes two meaning contributions, and both of them are generalized quantifiers. The two quantifiers are as proposed in Fox 1999: at the top of the chain is the actual generalized quantifier corresponding to the lexical entry of the determiner heading the DP, while at the bottom of the chain—the trace position—we insert a definite anaphoric expression, as sketched in (401) using an ι operator to represent a definite quantifier:

$$(401) \quad \forall x[A(x) \rightarrow \dots]$$


$$\iota y[A(y) \wedge y = x \wedge \dots]$$

With a single intervening existential event quantifier, this would give us the logical expression in (402), which is equivalent to (403), the intended wide-scope reading of the universal quantifier.

$$(402) \quad \forall x[A(x) \rightarrow \exists e[V(e) \wedge \iota y[A(y) \wedge y = x \wedge R(e) = y]]]$$

$$(403) \quad \forall x[A(x) \rightarrow \exists e[V(e) \wedge R(e) = x]]$$

It should be noted that s -chains are special in that because of trace conversion the core meaning of the moved expression is more faithfully represented at the top of the chain than in the base position. However, trace conversion ensures that the base position still

contains the full lexical content of the moved DP, which allows us to account for binding reconstruction effects as illustrated in (404), where the pronoun *his* can only be bound by the quantified DP if its base position (or trace) is c-commanded by the DP. The surface position of the pronoun seems to be irrelevant.

- (404) a. Which of his_1 students did every professor₁ talk to *t*?
 b. *Which of his_1 students *t* talked to every professor₁?

4.4.5 Binding and Weak Cross-Over

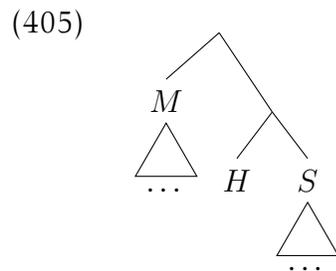
Inspired by (404), one may wonder how binding and Weak Cross-Over fit into the picture developed so far: following CHIERCHIA 2020 we assume that all binding is dynamic binding, and to the extent that CHIERCHIA's (2020) account of weak cross-over is adequate, we also have an account of that phenomenon

However, there may be no need for CHIERCHIA's stipulation that discourse referents can only be introduced by predicates (assuming that thematic roles are predicates, but nouns are not), we may instead explore the possibility that discourse referents are introduced when a noun predicating over that referent is integrated into the main representational structure via MERGE.

Even more interestingly, the dynamics needed for binding may be modeled by extending our use of continuations for structure building with the dynamic continuations proposed by DE GROOTE (2006) and further developed by LEBEDEVA (2012). The details are to be worked out in future research (BAUMANN in prep).

4.4.6 Scope and c-Command

It is worth emphasizing that the notion of scope advanced above necessarily implies a c-command relation between the scope taker and its scope. To see this consider the configuration in (405), where M c-commands S .



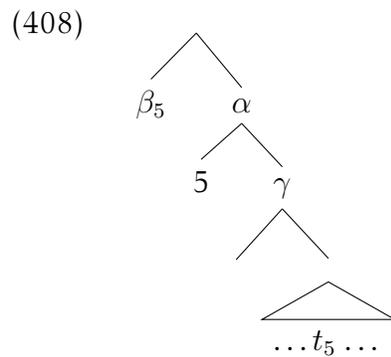
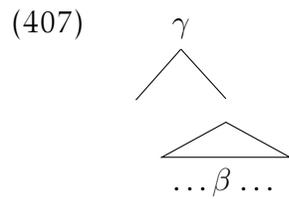
If M is a mover licensed by the head H , then M acts as the semantic argument of H , and the result then takes S as its semantic argument, as shown in (406).

$$(406) \quad [H(M)](S)$$

So M can take scope over S , because the expression $H(M)$, resulting from M combining with its licensing head H , takes S as its semantic argument. But notice how this notion of scope-taking implies c-command, but not in any meaningful way. In fact, c-command is just a label for the structural relation, which allows the semantic combination in (406).

4.4.7 Inverse Movement Chains and λ -Abstraction

One common way to model the semantics of bottom-up movement is through λ -abstraction (HEIM & KRATZER 1998), which proceeds as follows: applying the syntactic operation Move to a structure like (407) with β the expression to be moved results in an LF shown in (408), where 5 is a binding index.



The LF in (408) can be interpreted via PREDICATE ABSTRACTION, according to which for a branching node α with a binding index and γ as its daughters, and an assignment function g , its meaning is computed as in (409).

$$(409) \quad \llbracket \alpha \rrbracket^g = \lambda x. \llbracket \gamma \rrbracket^{g[x/5]}$$

Compared to our method of inverse movement chains, λ -abstraction as sketched in (408) and (409), seems to have at least two drawbacks: first, in (409) there is no role for the licensing head (cf. KLECHA & MARTINOVIĆ 2015), in fact the moved expression is not even a semantic argument of the licensing head. Second, it requires indexing and assignment functions to allow the λ -binder to bind the trace position. In our model, such machinery is not necessary, and incremental structure building in general obviates the need to introduce variables as free variables and only later bind them, because the binding operator is introduced before any instance of the variable it binds, and so there are no free variables at any point in the derivation. Instead of relying on assignment functions to handle

free variables, we build the semantic representation syntactically by combining semantic formulas as literal objects and keeping track of semantic variables in the syntax.

4.5 Incremental Compositional Event Semantics with Movement

In lieu of a summary we illustrate all basic principles discussed in this chapter by deriving the inverse scope reading of the sentence in (410).

(410) A cat will eat every rat.

Once again we assume that the semantic contribution of the start item >T . \mathbf{R} is the assertion operator in (411) which closes the continuation term $f(e)$ of the VP. Since this operator is meant to sit outside of the structure building system, it can only combine with the final TP. Semantic type considerations suggest a similar conclusion.

(411) $\text{>T}. \mathbf{R} :: \langle \epsilon \mid \lambda V.V(\lambda e. \mathbf{true}) \rangle$

The lexical items for the tense auxiliary *will* and the verb *eat* are listed in (412) and (413), respectively. Since tense and temporal semantics is not the focus of this work, the ‘tense part’ of lexical entry in (412) is only meant as a placeholder. The lexical entry of the verb *eat* in (413) contains three projections, which are linked via inverse movement: a v(oice) projection, an Accusative projection and a VP projection. The verb form *eat* itself is pronounced in the highest vP projection, and the other two are (head) movers within that expression.

(412) a. $+\text{nom} \text{>v}. \mathbf{T} :: \langle \text{will} \mid M_T \rangle$
 b. $M_T = \lambda Q \lambda V. \lambda f. Q(\lambda x. \exists t. V(\lambda e. [f(e) \wedge \tau(e) > t]))$

- (413) a. $\langle D \rangle_{\text{Acc}}. v :: \langle \text{eat} \mid M_v \rangle,$
 $\text{+acc} \rangle V. \text{Acc} :: \langle \epsilon \mid M_{\text{Acc}} \rangle,$
 $\rangle D. V :: \langle \epsilon \mid M_V \rangle$
- b. $M_v = \lambda x. \lambda V. \lambda f. V(\lambda e. [f(e) \wedge \mathbf{ag}(e, x)])$
- c. $M_{\text{Acc}} = \lambda Q \lambda V. \lambda f. Q(\lambda x. V(\lambda e. f(e)))$
- d. $M_V = \lambda x. \lambda f. \exists e. [\mathbf{eat}(e) \wedge \mathbf{th}(e, x) \wedge f(e)]$

The remaining lexical items are less complex than the one in (413) and are thus introduced when needed during the derivation. We present the derivation step by step, the first part of it should be familiar by now: in (414), we shift and merge the subject DP, then we shift the tense auxiliary will.

- (414) SHIFT a, cat
- $N :: \langle \text{cat} \mid \lambda x. \mathbf{cat}(x) \rangle$
- $\rangle N. D \text{-nom} :: \langle \mathbf{a} \mid \lambda N. \lambda P. \exists x. [N(x) \wedge P(x)] \ x \rangle$
- $\rangle T. R ::$
- MERGE \rangle N
- $D \text{-nom} :: \langle \mathbf{a} \ \text{cat} \mid \lambda P. \exists x. [\mathbf{cat}(x) \wedge P(x)], \ x \rangle$
- $\rangle T. R ::$
- SHIFT will
- $\text{+nom} \rangle v. T :: \langle \text{will} \mid M_T \rangle$
- $D \text{-nom} :: \langle \mathbf{a} \ \text{cat} \mid \lambda P. \exists x. [\mathbf{cat}(x) \wedge P(x)], \ x \rangle$
- $\rangle T. R ::$

Now we can integrate the subject via external MOVE $_E$, as shown in (415). Notice how the generalized quantifier representing the subject *a cat* gets inserted at the top of the

chain and combines with the T head via Function Application. Since this is a case of A-movement, the meaning component of the resulting mover is just the variable x bound by the subject quantifier.

(415) a. $\text{MOVE}_E \text{ nom}$

$\text{>v. T} :: \langle \text{a cat will} \mid M_1 \rangle,$

$\text{D} :: \langle \epsilon \mid x \rangle$

$\text{>T. R} ::$

b. $M_1 = M_T(\lambda P. \exists x [\mathbf{cat}(x) \wedge P(x)])$

$= \lambda V. \lambda f. \exists x [\mathbf{cat}(x) \wedge \exists t. V(\lambda e [f(e) \wedge \tau(e) > t])]$

Since there are no other matching features, the derivation continues in (416) by shifting the verb **eat**. And now we have an item selecting a moved specifier, so we perform two MERGE operations: we merge the $v(\text{oice})$ head with the TP containing the subject via v , and the mover of the subject with the $v(\text{oice})$ head via D.

(416) a. SHIFT eat

$$\begin{aligned}
\langle D \rangle \text{Acc. } v &:: \langle \text{eat} \mid M_v \rangle, \\
+\text{acc} \rangle V. \text{Acc} &:: \langle \epsilon \mid M_{\text{Acc}} \rangle \\
\rangle D. V &:: \langle \epsilon \mid M_V \rangle \\
\rangle v. T &:: \langle \text{a cat will} \mid M_1 \rangle, \\
D &:: \langle \epsilon \mid x \rangle \\
\rangle T. R &::
\end{aligned}$$

$$MR_I^{\leq} D + MR_{\gg} v$$

$$\begin{aligned}
\rangle \text{Acc. } T &:: \langle \text{a cat will eat} \mid M_2 \rangle, \\
+\text{acc} \rangle V. \text{Acc} &:: \langle \epsilon \mid M_{\text{Acc}} \rangle \\
\rangle D. V &:: \langle \epsilon \mid M_V \rangle \\
\rangle T. R &::
\end{aligned}$$

$$\begin{aligned}
\text{b. } M_2 &= M_1 \circ M_v(x) \\
&= M_1 \circ \lambda V. \lambda f. V(\lambda e. [f(e) \wedge \mathbf{ag}(e, x)]) \\
&= \lambda V. \lambda f. \exists x [\mathbf{cat}(x) \wedge \exists t. V(\lambda e [f(e) \wedge \mathbf{ag}(e, x) \wedge \tau(e) > t])]
\end{aligned}$$

We continue in (417) by shifting the quantificational determiner *every*. Since we want the direct object to take wide scope, we also shift the noun *rat* selected by the determiner, and merge the two.

(417) SHIFT every, rat

N :: $\langle \text{rat} \mid \lambda x. \text{rat}(x) \rangle$ >N. D -acc :: $\langle \text{every} \mid \lambda N. \lambda P. \forall y. [N(y) \rightarrow P(y)], y \rangle$ >Acc. T :: $\langle \text{a cat will eat} \mid M_2 \rangle,$ +acc >V. Acc :: $\langle \epsilon \mid M_{Acc} \rangle$ >D. V :: $\langle \epsilon \mid M_V \rangle$

>T. R ::

MERGE_> ND -acc :: $\langle \text{every rat} \mid \lambda P. \forall y. [\text{rat}(y) \rightarrow P(y)], y \rangle$ >Acc. T :: $\langle \text{a cat will eat} \mid M_2 \rangle,$ +acc >V. Acc :: $\langle \epsilon \mid M_{Acc} \rangle$ >D. V :: $\langle \epsilon \mid M_V \rangle$

>T. R ::

Then in (418) we shift the empty head \emptyset_s and merge it with the direct object. This adds the scope feature $-s$ onto the direct object.

(418) SHIFT \emptyset_s =D. D -s :: \emptyset_s D -acc :: $\langle \text{every rat} \mid \lambda P.\forall y [\mathbf{rat}(y) \rightarrow P(y)], y \rangle$ >Acc. T :: $\langle \text{a cat will eat} \mid M_1 \rangle$,+acc >V. Acc :: $\langle \epsilon \mid M_{Acc} \rangle$ >D. V :: $\langle \epsilon \mid M_V \rangle$

>T. R ::

MERGE_> DD -acc -s :: $\langle \text{every rat} \mid \lambda P.\iota y [y = z \wedge \mathbf{rat}(y) \wedge P(y)], y, \lambda P.\forall z [\mathbf{rat}(z) \rightarrow P(z)] \rangle$ >Acc. T :: $\langle \text{a cat will eat} \mid M_1 \rangle$,+acc >V. Acc :: $\langle \epsilon \mid M_{Acc} \rangle$ >D. V :: $\langle \epsilon \mid M_V \rangle$ >T. R :: $\langle \epsilon \mid \lambda V.V(\lambda e. \mathbf{true}) \rangle$

We assume that \emptyset_s also adds an additional meaning component and performs trace conversion (Fox 1999). We leave the details of this mechanism somewhat unspecified: schematically, \emptyset_s copies the core meaning component (but not the variable) and converts the one associated with the category feature into an anaphoric definite descriptions as shown in (419):

(419) $\emptyset_s([\lambda P.\forall y. [\mathbf{rat}(y) \rightarrow P(y)], y])$ $\implies [\lambda P.\iota y. [y = z \wedge \mathbf{rat}(y) \wedge P(y)], y, \lambda P.\forall z. [\mathbf{rat}(z) \rightarrow P(z)]]$

The derivation then continues as in (420): we merge the head mover of the Accusative phrase via *Acc*, which licenses external *MOVE_E* of the direct object into its specifier via *acc*.

- (420) a. $M_{V_E} \text{acc} + M_{R \gg} \text{Acc}$
- >V. T :: ⟨a cat will eat every rat | M_3 ⟩,
 >D. V :: ⟨ ϵ | M_V ⟩,
 D -s :: ⟨ ϵ | y , $\lambda P. \forall z [\mathbf{rat}(z) \rightarrow P(z)]$ ⟩
 >T. R ::
- b. $M_3 = M_2 \circ M_{acc} \left(\lambda P. \iota y [y = z \wedge \mathbf{rat}(y) \wedge P(y)] \right)$
 $= \lambda V. \lambda f. \exists x [\mathbf{cat}(x) \wedge \exists t. \iota y [y = z \wedge \mathbf{rat}(y)$
 $\wedge V(\lambda e. [f(e) \wedge \mathbf{ag}(e, x) \wedge \tau(e) > t])]]$

In the next step in (421), we have a matching V feature and so we merge the head mover of the VP via *MERGE_{\gg}*. As result there is a matching D feature, and so we merge the mover of the direct object into its base position. Notice how on the meaning side, the base position of the direct object is filled by the variable *y*, which is bound by the (trace converted) definite quantifier we merged into *Spec,AccP* in (420).

- (421) a. $\text{MERGE}_{\gg} V$
- >D. T :: ⟨a cat will eat every rat | M_4 ⟩,
 D -s :: ⟨ ϵ | y , $\lambda P.\forall z [\mathbf{rat}(z) \rightarrow P(z)]$ ⟩
 >T. R ::
- $\text{MERGE}_I D$
- T :: ⟨a cat will eat every rat | M_5 ⟩,
 -s :: ⟨ ϵ | $\lambda P.\forall z [\mathbf{rat}(z) \rightarrow P(z)]$ ⟩
 >T. R ::
- b. $M_4 = M_3 \circ M_V$
- c. $M_5 = M_4(y)$
- $= \lambda f.\exists x. [\mathbf{cat}(x) \wedge \exists t.\iota y [y = z \wedge \mathbf{rat}(y)$
 $\wedge \exists e [\mathbf{eat}(e) \wedge \mathbf{th}(e, y) \wedge f(e) \wedge \mathbf{ag}(e, x) \wedge \tau(e) > t]]]$

The resulting expression in (421) is a TP without any selection or licensing features, but it still contains a scope taking mover with the licensing feature -s. So we need to add a landing site for that mover, i.e. a scope position for Quantifier Raising. This can be achieved via the empty head in (422), which adjoins to a TP and licenses a scope taker via +s.

- (422) a. $+s >T$. T :: ⟨ ϵ | M_s ⟩
- b. $M_s = \lambda Q\lambda V.\lambda f.Q(\lambda x.V(\lambda e.f(e)))$

The empty head in (422) then merges with the TP via T and licenses a final movement step via s. Finally, we insert a head to attract the scope taking mover via +s.

(423) a. SHIFT \emptyset $+s >T. T :: \langle \epsilon \mid M_s \rangle$ $T :: \langle \text{a cat will eat every rat} \mid M_5 \rangle,$ $-s :: \langle \epsilon \mid \lambda P. \forall z [\mathbf{rat}(z) \rightarrow P(z)] \rangle$ $>T. R ::$ $M_{V_E} s + M_{R_{>}} T$ $T :: \langle \text{a cat will eat every rat} \mid M_6 \rangle,$ $>T. R :: \langle \epsilon \mid \lambda V. V(\lambda e. \mathbf{true}) \rangle$ b. $M_6 = M_s(\lambda P. \forall z [\mathbf{rat}(z) \rightarrow P(z)])(M_5)$ $= \lambda f. \forall z [\mathbf{rat}(z) \rightarrow \exists x [\mathbf{cat}(x) \wedge \exists t. \iota y. [y = z \wedge \mathbf{rat}(y)$ $\wedge \exists e. [\mathbf{eat}(e) \wedge \mathbf{th}(e, y) \wedge \mathbf{ag}(e, x) \wedge \tau(e) > t]]]]]$

As the last step, we merge the complete TP and the start item, and close off the continuation $f(e)$ of the event variable.

(424) a. MERGE $_{>}$ T $R :: \langle \text{a cat will eat every rat} \mid M_7 \rangle$ $>T. R :: \langle \epsilon \mid \lambda V. V(\lambda e. \mathbf{true}) \rangle$ b. $M_s = \lambda Q \lambda V. \lambda f. Q(\lambda x. V(\lambda e. f(e)))$ c. $M_7 = \lambda V. V(\lambda e. \mathbf{true})(M_6)$ $= \forall z [\mathbf{rat}(z) \rightarrow \exists x. [\mathbf{cat}(x) \wedge \exists t. \iota y. [y = z \wedge \mathbf{rat}(y)$ $\wedge \exists e. [\mathbf{eat}(e) \wedge \mathbf{th}(e, y) \wedge \mathbf{ag}(e, x) \wedge \tau(e) > t]]]]]$

4.6 Previous Work

Incremental Semantic Interpretation: BOTT & STERNEFELD (2017) and BECK & TIEMANN (2018)

As mentioned above, there is not much previous work on incremental semantic interpretation. One notable exception is BOTT & STERNEFELD (2017), who use heavy non-standard semantic machinery like dynamic binding and unrestricted β -reduction. As our work demonstrates, such heavy machinery might not be necessary if we build our semantics on top of an incremental model of syntax. BOTT & STERNEFELD (2017), on the other hand, do not assume any syntax. In general, their approach is similar to ours, they also start from the Compositional Event Semantics of CHAMPOLLION (2015) and interpret it as a form of continuation semantics, with the continuation term $f(e)$ marking the position where the derivation continues. But in addition, they also need a scope term, while in our semantics scope is handled via syntactic movement.

A critique of BOTT & STERNEFELD 2017 and a more conservative sketch of incremental semantic interpretation is offered in BECK & TIEMANN 2018. They advocate for the position that semantic composition is only partially incremental, in the sense that elements in the same type domain modifying the same variable get composed immediately, e.g. different predicates modifying the same individual variable are composed immediately, but there is a delay in compositional processing where composition switches from one type of variable to another type, in our model for example at thematic roles, which link events and individuals. This is somewhat similar to the conclusion we reach in Section 6.1, where we say that semantic composition is complete and immediate within a given variable, but needs to be a bit loser when two variables are linked together. The main motivation we give for that is the possibility of right adjunction.

Minimalist Grammars and Compositional Event Semantics: TOMITA (2016, 2017)

A strict syntax semantics mapping between Minimalist Grammar syntax and a compositional event semantics based on CHAMPOLLION 2015 is also explored in TOMITA 2016, 2017. The main technical innovation there is that each MG feature corresponds to an independent meaning unit, so that a transitive verb like *like* is represented as three components as in (425).

(425) $\text{like} :: \langle \rangle D \mid \theta_{\text{likee}} \rangle \langle \rangle D \mid \theta_{\text{liker}} \rangle \langle V \mid \lambda f. \exists e [\text{walking}(e) \wedge f(e)] \rangle$

Assigning each feature its own meaning component is exactly the solution we proposed in Section 4.4 for features involved in \bar{A} -movement chains, but TOMITA extends this logic to all types of features⁴ including selection features as shown in (425). Assigning each selection feature its own meaning component has a similar effect to our notion of inverse head movement (see Chapter 3), especially once we introduce a rather elaborate set of empty heads along the clausal spine as in the derivation in Section 4.5. So in some sense assigning each feature its own meaning component unifies two independent mechanisms employed in this dissertation: inverse head movement and assigning a separate meaning component to the top and the base in an \bar{A} chain.

Like in our semantics, TOMITA (2016, 2017) uses only function application for semantic composition, and a selected element is always the semantic argument to the selecting element, again like in our semantics. But critically, there is no selection relation (and no features) specified between the individual components in (425), instead the way the individual components in (425) combine with each other is hard-coded into the definition of MERGE. In this respect, our approach is more in line with traditional syntactic and

⁴ To be precise, in TOMITA 2016 the one-feature-one-meaning logic seems to apply only to selection features, but in TOMITA 2017 it is extended to licensing features.

semantic theorizing, according to which the external argument is introduced by a v(oice) head, which selects a verb phrase (and the external argument).

In addition, one may wonder if (and how) the approach of having one meaning per feature can be generalized to all types of MG expressions, especially expressions outside the domain of verbs and their thematic arguments.

CHAPTER 5

INCREMENTAL ISLANDS AND CLAUSAL PENINSULAS

Syntactic movement is often conceptualized as unbounded (CHOMSKY 1977): in sentences like (426), *wh*-movement can span across multiple clauses.

- (426) a. Who do you think Nick said that he saw in the park today?
 b. What did your your neighbor say that he thinks the mailman believes that your dog destroyed?

But in other sentences, like those in (427), a movement dependency seems to be blocked, even though the sentences in (427) are shorter and less complex than those in (426).

- (427) a. *What did a movie about cause a scandal?
 A movie about aliens caused a scandal.
 b. *Which book did you apply to grad school after you read?
 I applied to grad school after I read Aspects.

So syntactic movement may very well be unbounded in some cases, but it is blocked out of certain syntactic environments, like subjects or adjuncts in (427). Environments that block movement were first catalogued by Ross (1967, 1986) and christened ISLANDS, because — as he put it — they are separated from the rest of the sentence. Some island environments

noted by Ross (1967) are sentential subjects, complex noun phrases, and coordinate structures. Since then, the list has grown to also include adjuncts, (non-sentential) subjects, and certain types of complement clauses (see SZABOLCSI & DEN DIKKEN 1999, BOECKX 2008 for an overview and a historical perspective).

But syntax would be so much simpler (or more minimalist) without islands. And I am going to argue that some island phenomena, which have traditionally been viewed as part of syntax, are a consequence of the necessarily incremental nature of human language processing, while other islands are (at least partly) semantic in nature. Specifically, I will argue that specifiers are necessarily islands if we assume an incremental parsing procedure like the one developed in Chapter 2. Complements, on the other hand, are predicted not to be islands. As a first approximation this is a desirable result. I then argue that the main other constraint needed to derive most of non-specifier islands (like factive islands) is that some clauses are PENINSULAS: they are almost islands and block general \bar{A} -movement, but they allow scope-taking DP-movement. Peninsulas are thus a type of weak or selective islands.

We assume that weak or island effects are an instance of the broader class of (focus) intervention effects, and thus at least partly semantic in nature: if intervention effects are due to focus alternatives (BECK 2006), this follows directly under a focus-alternative semantics for questions involving the Q-particle, as proposed by KOTEK (2014): a movement chain introduced (or headed) by Q cannot span across a focus intervener. However, DP-movement and its chain headed by a DP scope head is not subject to intervention effects and thus not affected by the presence of interveners (cf. POOLE 2017). This immediately gives us all the characteristic properties of weak or selective islands: they permit extraction of DP arguments, but block all other movement.

Next we establish that there are two types of embedded clauses: definite clauses and open clauses. Definite clauses will be shown to be peninsulas, while open clauses show no

island effects. Following the proposals of MELVOLD (1991) and KASTNER (2015), definite clauses are analyzed as having a (mostly silent) definite determiner which introduces a propositional object encoding the truth conditions of its CP argument. So definite clauses are DPs, and they are selected by verbs that select DPs, like factive verbs (KIPARSKY & KIPARSKY 1970) and response-stance verbs (CATTELL 1978). The factive or familiarity presupposition associated with the clausal complements of these verbs is at least in part the familiarity or givenness presupposition triggered by the definite determiner.

Additionally, the definite determiner also acts as an intervener, but critically only with respect to its restriction and not with respect to its scope when conceptualized as a generalized quantifier: so we expect intervention effect for movement out of a definite DP, but not for movement across a definite DP. Since definite clauses are DPs with the actual clause sitting within (the restriction) the definite quantifier, the definite quantifier acts as an intervener for *wh*-chains (or QP-movement), rendering definite clauses as peninsulas: general \bar{A} -movement is blocked, but DP-movement is still possible. So peninsulas are weak islands.

Open clauses on the other hand are not DPs. Instead their semantic denotation is a predicate $f(e)$ that holds over sets of events, which is the denotation we assumed for all verbal projections in Chapter 4. This means that both finite and non-finite clauses can be open. Open clauses are selected by a class of embedding verbs known as bridge verbs, and they show no island effects.

Finally, I discuss adjunction and coordinate structures which are often claimed to be islands for extraction. I argue that adjuncts are not islands *per se*, but clausal adjuncts are definite clauses and thus block extraction. In Section 2.4, coordinate structures have been argued to be a variant of adjunction structures, and so I suggest that in some coordinate structures one of the conjuncts is subordinate to the other, like an adjunct is subordinate in an adjunction structure. Such coordination-as-adjunction structures carry a marked

coherence relation and allow asymmetric extraction out of the non-subordinate conjunct, i.e. out of the conjunct that acts like the host in an adjunction structure.

5.1 Specifier Islands

In this section, it is shown how one prominent island constraint, namely the Specifier-Island Constraint can be derived from the incremental parsing algorithm developed in Chapter 2. The argument is very similar to the arguments given in favor of MULTIPLE SPELL-OUT (URIAGEREKA 1999), but critically there is no need to posit a special (multiple) spell-out mechanism. Instead the Specifier-Island Constraint follows from the observation that during incremental parsing a specifier S and its selecting/licensing head H correspond to two distinct items on the parse stack. This follows directly from the requirement that a specifier S can only be merged after its selecting/licensing head H has been processed, and the fact that specifiers precede their heads within the linear order of the sentence.

5.1.1 The Specifier-Island Constraint

The Specifier-Island Constraint (SPIC, STABLER 1999) states that sub-extraction out of specifiers is impossible, i.e. no proper subconstituent of a specifier can be moved out of that specifier. Under the assumption that all phrasal movement targets specifier positions, the SPIC is a generalization of the notion of freezing a moved constituent (WEXLER & CULICOVER 1980), and hence banning sub-extraction from it. A common example to illustrate both freezing effects and the SPIC is the observation that English subjects are islands for extraction: forming a constituent question into the subject as in (428) is seriously degraded, while a comparable question into the direct object as in (429) is just fine.

- (428) a. *Who did a friend of buy a book about Chomsky?
 b. A friend of someone bought a book about Chomsky.
- (429) a. Who did a friend of Chomsky buy a book about?
 b. A friend of Chomsky bought a book about someone.

Since subjects are moved into the specifier of TP, sub-extraction out of subjects is predicted to be impossible.

Within the framework of Minimalist Grammar, the SPIC is an additional assumption commonly made (STABLER 1999). It does not follow from other common principles of bottom-up structure building. But I will be showing that it is a direct consequence of incremental structure building based on a transformational grammar formalism like Minimalist Grammar.

5.1.2 Deriving the SPIC from Incremental Parsing

The derivation of the SPIC from incremental parsing is based on two observations: first, specifiers need to be fully assembled before they can be merged or otherwise integrated into the main spine; and second, any moved element becomes a mover within the tree it is first integrated (or Moved) into.

The first point is also observed in traditional bottom-up structure building and has led to the postulation of devices like workspaces (CHOMSKY 1995) and principles like MULTIPLE SPELL-OUT (URIAGEREKA 1999): when building the derived tree for a sentence like (430) in a bottom-up manner, the subject DP *our visitors from a small town in North Carolina* needs to be fully assembled before it can be merged with the VP *drank beer*.

- (430) Our visitors from a small town in North Carolina drank beer.

This observation is one of the main motivations for the idea of MULTIPLE SPELL-OUT (URIAGEREKA 1999): if the subject is assembled separately, it may as well be spelled out separately. And the separate spell-out of the subject in (430) implies that no element can move out of it after it was spelled out, so a constituent question like (431), which asks about a sub-constituent of the subject DP is ungrammatical.

(431) *Which state did our visitors from a small town in drink beer?

The same intuition is true under the incremental parsing procedure proposed in Chapter 2, but critically we need not posit a special (multiple) spell-out mechanism. Instead the different spell-out units of URIAGEREKA (1999) correspond to different items on the parse stack. When parsing (431) specifically the *wh*-phrase *which state* is first combined with the complementizer *did* and so is added to the stack as part of the item representing the matrix CP. The subject DP *our visitors from a small town in*, on the other hand, is assembled as a separate unit on the stack, and critically it does not contain any movers. So the *wh*-phrase is not visible from within the subject, instead the incomplete subject DP is just an incomplete DP expecting another DP in the upcoming input string, as indicated by its MG expression in (432). But once the incomplete subject DP in (432) gets merged (if it can be) after the verb *drink* has been shifted, the ‘edge’ where new incoming items can be integrated has already moved on to the complement position of the verb.

(432) >D. D :: our visitors from a small town in

So from an incremental perspective (in a head-initial language), the complement position is special, because it is to the right of its head and is thus the position where the derivation continues and new items are integrated. Incrementally, the complement position is at the edge of a derivation, but specifiers are not: a specifier *S* is selected by a head *H* that

follows it, and so when the specifier S is selected by the head H , the selecting head H must have already been processed. This means that the edge of the derivation has already moved past the head H : the next item in the input will appear to the right of H .

5.1.3 Subject Islands

As this all sounds rather abstract, let's try to derive sentence (433) containing a subject island violation, and see what happens.

(433) *Which actor do fans of think that the earth is flat?

(434) Fans of Tom Cruse think that the earth is flat.

In our first analysis we ignore head movement, and once again assume that the auxiliary *do* is a complementizer with a +wh licensing feature. The initial part of the derivation in (435) is standard.

(435) SHIFT which, actor

N :: actor

>N. -wh D :: which

>C. R

MERGE_> N

-wh D :: which actor

>C. R

SHIFT do

+wh >V. C :: do

-wh D :: which actor

>C. R

MOVE_E wh

>V. C :: which actor do, D :: ε

>C. R

SHIFT fans, of

>D. P :: of

>P. D :: fans

>V. C :: which actor do, D :: ε

>C. R

MERGE_» P

>D. D :: fans of

>V. C :: which actor do, D :: ε

>C. R

Next we shift the matrix verb think to get the stack in (436). But now it is not clear what to do.

(436) SHIFT think

<D >C. V :: think
 >D. D :: fans of
 >V. C :: which actor do, D :: ϵ
 >C. R

Given the structure building operations in Figure 17, there is none that could apply here. Critically, there is no ‘skipping variant’ of $\text{MERGE}_{<}$, i.e. it is not possible to select an incomplete item if it is to be linearized to the left. This encodes the observation that when a specifier is merged its selecting head must already be on the stack and so the edge of the derivation has already moved on to the complement position of the selecting head. But let’s engage in some hypothetical reasoning and assume that there was such a MERGE_{\ll} operation, then the derivation could continue as in (437).

(437) MERGE_{\ll} D

>D >C. V :: fans of think
 >V. C :: which actor do, D :: ϵ
 >C. R

The MERGE_{\ll} step in (437) may look reasonable, and after (437) the derivation could successfully continue as shown in Figure 23. But the problem is that the result of MERGE_{\ll} is not what it pretends to be: the feature specification of the expression in (438) suggests that it is a VP which selects a DP and a CP to its right. Setting aside the objection that in Minimalism it should not be possible for an item to select two complements, selecting two items to its right is not what the feature specifications of the constitutive items of (438) in (436) suggest that (438) is meant to be. Instead the expression in (438) is meant to be just a VP selecting a CP, and a constituent of the VP, the subject *fans of*, is missing a DP.

(438) $\text{>D >C. V :: fans of think}$

The reason why (438) is not what it is meant to be is that the hypothetical MERGE_{\ll} operation does not preserve the directionality of selection features when combining incomplete items: in creating (438), MERGE_{\ll} flipped the directionality of the D feature of the VP as shown in (439). Compare (439) to an application of MERGE_{\gg} in (440): there the directionality of the selection features of the VP is preserved.

(439) * $\text{MERGE}_{\ll} \text{ D}$

<D >C. V :: think

>D. D :: fans of

\Rightarrow

$\text{>D >C. V :: fans of think}$

(440) $\text{MERGE}_{\gg} \text{ D}$

>D. V :: meet

>D. D :: fans of

\Rightarrow

$\text{>D. V :: meet fans of}$

The forward variant MERGE_{\gg} preserves the directionality of its selection features, because structure building proceeds through the right edge, which is a direct consequence of the left-to-right nature of incremental processing. And the specifics of the incremental parsing procedure developed in Chapter 2, namely that specifiers can only be integrated after their selecting head has been processed, require that specifiers be completed when they are selected, because at the time of their selection by the head H the edge of the derivation

has already moved past H . And so the hypothetical variant MERGE_{\ll} is not compatible with the incremental parsing procedure developed in Chapter 2.

Adding (inverse) head movement and case checking to the picture, complicates the derivation (shown in Figure 24), but does not change the main issue. In fact, for the derivation to succeed we would now require two ill-defined \ll -operations: MOVE_E^{\ll} and MERGE_I^{\ll} .

1. SHIFT which, actor	N :: actor >N. -wh D :: which >C. R
2. MERGE _{>} N	-wh D :: which actor >C. R
3. SHIFT do	+wh >V. C :: do -wh D :: which actor >C. R
4. MOVE _E wh	>V. C :: which actor do, D :: ε >C. R
5. SHIFT fans, of	>D. P :: of >P. D :: fans >V. C :: which actor do, D :: ε >C. R
6. MERGE _{>>} P	>D. D :: fans of >V. C :: which actor do, D :: ε >C. R
7. SHIFT think	<D >C. V :: think >D. D :: fans of >V. C :: which actor do, D :: ε >C. R
8. MERGE _{<<} D	>D >C. V :: fans of think >V. C :: which actor do, D :: ε >C. R
9. MERGE _I ^{>} D + MERGE _{>>} V	>C. C :: which actor do fans of think >C. R
10. SHIFT so	C :: so >C. C :: which actor do fans of think >C. R
11. MERGE _{>} C, C	R :: which actor do fans of think so

FIGURE 23: (Impossible) incremental derivation with subject island violation: *Which actor do fans of think so?*

1. SHIFT which, actor
 - N :: actor
 - >N. -wh -acc D :: which
 - >C. R
2. MERGE_> N
 - wh -acc D :: which actor
 - >C. R
3. SHIFT do
 - +wh >T. C :: do, +nom >V. T :: ϵ
 - wh -acc D :: which actor
 - >C. R
4. MOVE_E wh
 - >T. C :: which actor do, +nom >V. T :: ϵ , -acc D :: ϵ
 - >C. R
5. SHIFT fans, of
 - >Acc. P :: of, +acc >D. Acc :: ϵ
 - >P. -nom D :: fans
 - >T. C :: which actor do, +nom >V. T :: ϵ , -acc D :: ϵ
 - >C. R
6. MERGE_{>>} P
 - >Acc. -nom D :: fans of, +acc >D. Acc :: ϵ
 - >T. C :: which actor do, +nom >V. T :: ϵ , -acc D :: ϵ
 - >C. R
7. MV_E[←] nom + MR_{>>} T
 - >v. C :: which actor do fans of, >Acc. D :: ϵ ,
 - +acc >D. Acc :: ϵ , -acc D :: ϵ
 - >C. R
8. SHIFT think
 - <D >V. v :: think, >C. V
 - >v. C :: which actor do fans of, >Acc. D :: ϵ ,
 - +acc >D. Acc :: ϵ , -acc D :: ϵ
 - >C. R
9. MR_I[←] D + MR_{>>} v
 - >Acc >V. C :: which actor do fans of think, >C. V,
 - +acc >D. Acc :: ϵ , -acc D :: ϵ
 - >C. R
10. MV_E[←] acc + MR_{>>} Acc
 - >D >V. C :: which actor do fans of think, >C. V, D :: ϵ
 - >C. R
11. MERGE_> D
 - >V. C :: which actor do fans of think, >C. V
 - >C. R
12. MERGE_{>>} V
 - >C. C :: which actor do fans of think
 - >C. R

FIGURE 24: (Impossible) incremental derivation with subject island violation and head movement: *Which actor do fans of think so?*

5.1.4 A Different Specifier: Indirect Objects

With this treatment of (inverse) head movement, we are also in a position to consider another specifier type: indirect objects in English double-object constructions, which are islands as the contrast in (441) shows.

- (441) a. *Who did Foucault give a friend of a book about Chomsky?
 b. Foucault gave a friend of someone a book about Chomsky.

- (442) *Who did he give fans of beer?

As an example, let's see how a derivation of (442) would proceed and where it crashes. Again ignoring unnecessary detail and starting with the shifting of *he*, the derivation goes as follows

And now we are at a similar point as we were with subjects containing a gap: if there was a possibility to ignore the selection feature of the indirect object we could merge it and at the same time merge the VP head/projection. But then there is no way the ignored feature could be checked by the moved *wh*-element.

1. SHIFT he
 - D :: he
 - >v. C :: who did, D :: ϵ
 - >C. R
2. SHIFT give
 - <D >V. v :: give, <D >D. V
 - D :: he
 - >v. C :: who did, D :: ϵ
 - >C. R
3. MERGE_< D
 - >V. v :: he give, <D >D. V
 - >v. C :: who did, D :: ϵ
 - >C. R
4. SHIFT fans, of
 - >D. P :: of
 - >P. D :: fans
 - >V. v :: he give, <D >D. V
 - >v. C :: who did, D :: ϵ
 - >C. R
5. MERGE_» P
 - >D. D :: fans of
 - >V. v :: he give, <D >D. V
 - >v. C :: who did, D :: ϵ
 - >C. R
6. MERGE_« D + MERGE_» V
 - >D >D. v :: he give fans of, <D >D. V
 - >v. C :: who did, D :: ϵ
 - >C. R

FIGURE 25: (Impossible) incremental derivation with indirect object: *Who did Foucault give fans of beer?*

5.1.5 Violations of Specifier Island Constraint

The Specifier island Constraint is only supposed to work in English and similar languages, definitely not in head/verb-final languages like Japanese. So we can disregard the fact that extraction is possible out of some Japanese subjects (TAKAHASHI 1994).

But even in English and similar languages, extraction out of subjects is sometimes possible. A particularly clear case are sentences with presentational subjects as in (443).

(443) Who are there pictures of on the wall?

(444) There are pictures of Foucault on the wall.

But interestingly, the subject *pictures of* in (443) cannot be located in Spec,TP like normal subjects, that position is occupied by the expletive *there* as shown in (444). Instead the subject must be located in a lower position. In fact, before the adjunct PP *on the wall* is parsed, the subject in (443) is at the right edge of the derivation just like a prototypical complement, and so the mover of *who* can be merged at that point. For such a derivation to be possible, the copula *are* in complementizer position must have the lexical entry in (445), with it the derivation can proceed as in Figure 26, where we leave out case-based movement for simplicity.

(445) +wh >T. C :: is, <Expl >D. T :: ϵ

A similar analysis can be given for low subjects of intransitive verbs (in particular of unaccusative verbs) in Romance languages (BIANCHI & CHESI 2014).

1. SHIFT who, is
 -wh D :: who
 +wh >T. C :: are, <Expl >V. T :: ϵ , >D. V :: ϵ
 >C. R
2. MOVE_E wh
 >T. C :: who are, <Expl >V. T :: ϵ , >D. V :: ϵ , D :: ϵ
 >C. R
3. SHIFT there
 Expl :: there
 >T. C :: who are, <Expl >V. T :: ϵ , >D. V :: ϵ , D :: ϵ
 >C. R
4. MR_< Expl + MR_I T
 >V. C :: who are there, >D. V :: ϵ , D :: ϵ
 >C. R
5. MERGE_I V
 >D. C :: who are there, D :: ϵ
 >C. R
6. SHIFT pictures, of
 >D. P :: of
 >P. D :: pictures
 >D. C :: who are there, D :: ϵ
 >C. R
7. MERGE_{>>} N
 >D. D :: pictures of
 >D. C :: who are there, D :: ϵ
 >C. R
8. MERGE_{>>} D
 >D. C :: who are there pictures of, D :: ϵ
 >C. R
9. MERGE_I D
 C :: who are there pictures of
 >C. R

FIGURE 26: (Incremental derivation with extraction from a low subject: *Who are there pictures of (on the wall)?*)

5.1.6 Interim Summary

Where does this leave us with islands? According to the proposal here, specifiers are islands for sub-extraction. Complements, on the other hand, are predicted not to be islands. As a first approximation this is a desirable result. Obviously, there are many types of islands that are not (necessarily) specifiers: adjuncts, factive islands, coordinate structures. I will now argue that the main other constraint needed to derive most of these island effects is that some clauses are islands, or at least *PENINSULAS*: they are almost islands and block general \bar{A} -movement, but they allow scope-taking DP-movement. Peninsulas are thus a type of weak or selective islands.

5.2 Clauses and Peninsulas

In this section I explore the idea that some finite clauses are *PENINSULAS*¹: they are almost islands and block general \bar{A} -movement, but they allow scope-taking DP-movement. Peninsulas are thus a type of weak or selective islands.

We assume that weak or selective island effects are an instance of the broader class of intervention effects: if intervention effects are due to focus alternatives (BECK 2006), this follows directly under a focus-alternative semantics for questions involving the Q-particle, as proposed by KOTEK (2014): a movement chain introduced (or headed) by Q cannot span across a focus intervener. However, DP-movement and its chain headed by a DP scope head is not subject to intervention effects and thus not affected by the presence of interveners (cf. POOLE 2017). This immediately gives us all the characteristic properties of selective islands.

¹ The first use of the term *peninsula* in this meaning appears to be by BARONE (1972), who used it as a suggestive label for the supposed exceptions to anaphoric islands (POSTAL 1969) observed by LAKOFF (1971). It appears that the term was in common use at early meetings of the Chicago Linguistic Society (cf. CORUM 1973) and it is a shame that it fell out of use.

Next we establish that there are two types of embedded clauses: definite clauses and open clauses. Definite clauses will be shown to be peninsulas, while open clauses show no island effects. Following the proposals of MELVOLD (1991) and KASTNER (2015), definite clauses are analyzed as having a (mostly silent) definite determiner which introduces a propositional object encoding the truth conditions of its CP argument. So definite clauses are DPs, and they are selected by verbs that select DPs, like factive verbs (KIPARSKY & KIPARSKY 1970) and response-stance verbs (CATTELL 1978). The factive or familiarity presupposition associated with the clausal complements of these verbs is at least in part the familiarity or givenness presupposition triggered by the definite determiner.

Additionally, the definite determiner also acts as an intervener, but critically only with respect to its restriction and not with respect to its scope when conceptualized as a generalized quantifier: so we expect intervention effects for movement out of a definite DP, but not for movement across a definite DP. Since definite clauses are DPs with the actual clause within the DP, the definite quantifier acts as an intervener for QP-movement, rendering definite clauses as peninsulas: general \bar{A} -movement is blocked, but DP-movement is still possible. This makes peninsulas weak islands.

Open clauses on the other hand are not DPs. Instead their semantic denotation is a predicate $f(e)$ that holds over sets of events, which is the denotation we assumed for all verbal projections in Chapter 4. This means that both finite and non-finite clauses can be open. Open clauses are selected by a class of embedding verbs known as bridge verbs, and they show no island effects.

5.2.1 Weak Island Effects as Intervention Effects

From a syntactic perspective, weak islands are rather puzzling: the syntactic environments that act as weak islands are not a homogeneous class: in addition to the clausal

complements of factive verbs (factive islands) as in (446) and interrogative complements (*wh*-islands) they also include matrix clauses containing negation or a negative quantifier (negative or inner islands) as in (447).

- (446) a. *How tall has Mary denied that she is?
 b. Mary has denied that she is five foot eight.

- (447) a. *How didn't John sing at the party?
 b. John didn't sing out of tune at the party.

A more descriptive term for weak islands is selective islands, because unlike 'strong' islands they do not block all types of movement: while the questions in (446) and (447) are unacceptable, the corresponding questions in (448) and (449) are much better.

- (448) What has Mary denied that she did?
 (449) a. Which song didn't John sing at the party?
 b. Who didn't Mary invite to the party?

In addition, weak islands can also affect scope: if a *how many*-phrase is moved out of an embedded clause as in (450), the resulting question has two readings: a narrow-scope reading, where *how many* takes scope under the verb *think* and a wide-scope reading where it takes scope above *think* (see Section 2.5 for discussion). But if the embedded clause is a weak island as in (451), only the wide-scope reading is available, i.e. (451) can only be used in a context where there is a fixed set of books to be reviewed, not if there is only a fixed number of books.

- (450) How many books do the authors think that the critics reviewed?

(451) How many books did the authors realize that the critics reviewed?

Determining which types of movement (or movers) are blocked by weak islands and which are allowed has been remarkably challenging: as a first approximation, one may suspect an argument-adjunct distinction, but *how tall* in (446a) may be an argument of *is* and not an adjunct. Another syntactic characterization was offered by CINQUE (1990): he suggests that only DPs can be extracted out of weak islands. This proposal is very successful in characterizing which types of movement are allowed or blocked, but as it stands it cannot explain, why the narrow-scope reading in (451) is unavailable.

For my own characterization of weak islands, I follow CINQUE (1990) and assume that only DPs can move out of or across weak islands, but I suggest a different mechanism and will also offer an explanation for the scope facts in (451). In Section 4.4 we argued that *wh*-phrases consisting of DPs can have both a $-\text{wh}$ feature and $-\text{s}$ feature: since DPs denote generalized quantifiers, they can receive a scope feature $-\text{s}$ for QR-like scope taking, as illustrated in (452).² All other *wh*-phrases, like *where* in (453), only have a $-\text{wh}$ feature.

(452) a. $-\text{wh} -\text{s} D :: \text{how many}$
 b. $-\text{wh} -\text{s} D :: \text{what}$

(453) a. $-\text{wh} P :: \text{where}$

We can now make our characterization of weak islands more precise: only chains started by a $-\text{s}$ feature (i.e. DP-movement chains) can span across weak islands, while chains started by a $-\text{wh}$ feature or Q(uestion) head/particle (i.e. QP-movement chains) are blocked

² The discussion in Section 4.4 was centered around *how many* and we argued that the $-\text{s}$ feature is carried by *many*, while the $-\text{wh}$ feature is carried by *how*. This was possible, because in the phrase *how many* the word *many* is the determiner and the word *how* is a *wh*-word. In contrast, the *wh*-word *which* is both a determiner and a *wh*-expression, and so it carries both a $-\text{s}$ feature and a $-\text{wh}$ feature. Similarly for *wh*-DPs like *what* or *who*.

by weak islands. But why should QP-movement be banned across weak islands? To explain this we make two substantial claims: first we assume that weak island effects are an instance of a broader class of intervention effects (cf. HONCOOP 1998). Second, following KOTEK (2014) and POOLE (2017) we conclude that only QP-movement chains are subject to intervention effects.

Intervention effects are typically associated with *wh-in-situ* constructions like the German multiple-*wh*-questions in (454): the common generalization is that the negative quantifier *niemand* ‘nobody’ acts as an intervener and blocks a (covert movement) dependency between the *in-situ-wh wo* ‘where’ and the interrogative C-head, rendering the question ungrammatical. If the intervener is replaced with a non-intervening expression like a proper name or if the *in-situ-wh* scrambles across the intervener, the question is grammatical.

- (454) a. *Wen hat niemand wo gesehen?
whom has nobody where seen
- b. Wen hat Maria wo gesehen?
whom has Maria where seen
- c. Wen hat wo niemand gesehen?
whom has where nobody seen
‘Who has nobody seen where?’

Intervention effects like the one in (454) are typically associated with focus, and BECK (2006) proposes a focus-alternative semantics for constituent questions to account for these facts. But KOTEK (2014) shows that the relevant dependency for intervention effects is not one between the *wh*-element and the C-head, but instead between the *wh*-element and a Q-head (CABLE 2007, 2010). And so KOTEK proposes a focus-alternative semantics for what POOLE (2017) calls QP-movement, and according to that semantics QP-movement (i.e. a movement chain started by a Q-head) is subject to intervention effects, and thus

cannot span across a (focus) intervener. Details of the focus-alternative semantics for QP-movement and how interveners lead to ungrammaticality are beyond the scope of this work and can instead be found in POOLE 2017, KOTEK 2014, and BECK 2006. All that matters for us is that QP-movement (or a *wh*-chain) is not possible across (focus³) interveners.

A common class of interveners are negation, negative quantifiers and elements associating with focus such as English *only*. And the same set of interveners also induces weak island effects: the minimal pair in (455) shows an example of a weak island effect induced by the focus-sensitive particle *only* (cf. ABRUSÁN 2014).

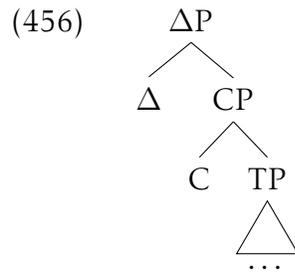
- (455) a. *How does only Mary think that John sang at the party?
 b. How does Mary think that John sang at the party?

Now that we have argued that weak island effects are (focus) intervention effects, we need to explain what causes the intervention effects associated with the complement clauses of factive and other presuppositional verbs. To this end we first argue the clausal complements of these verbs are different from the clausal complements of other propositional attitude verbs.

5.2.2 Definite and Open Clauses

We claim that there are two types of finite embedded clauses: definite clauses and open clauses. Definite (or presuppositional) clauses follow the syntactic proposals of MELVOLD (1991) and KASTNER (2015): they contain a clausal definite determiner Δ on top of their CP projection as sketched in (456).

³ In fact, we are not even committed to intervention effects being associated with focus.



The clausal determiner Δ looks very similar to a regular definite determiner. One possible semantics for Δ is given in (457), where we assume an ι -operator to achieve definiteness.

$$(457) \quad \llbracket \Delta \rrbracket = \lambda V. \lambda P. \iota o [\mathbf{cont}(o) = V(\lambda e. \mathbf{true}) \wedge P(o)]$$

More importantly, the variable o represents a propositional object that encodes the truth-making conditions for the clause it represents (MOLTMANN 2020). The propositional object receives its propositional content via the content function **cont** (PIETROSKI 2000, MOULTON 2015), which acts like a thematic role. If a definite clause is embedded under a propositional attitude verb, the propositional object represents the object of the attitude: in the case of a sentence like (458), the propositional object denotes the regret as an attitudinal object.

(458) The critics regret that they wrote a bad review.

Critically, propositional objects are entities of type e , just like individuals and events. And just like individuals and events, propositional objects can be referred to by deictic anaphoric expressions like *this* or *that*. So definite clauses are DPs, and they are selected by verbs that select DPs, like factive verbs (KIPARSKY & KIPARSKY 1970) and response-stance verbs (CATTELL 1978). In fact we may be tempted to identify the clausal determiner Δ with

the optional pronoun *it*⁴ in (459), especially if pronouns are conceptualized as anaphoric definite descriptions (DE GROOTE 2006, ELBOURNE 2013)

(459) The critics regret *it* that they wrote a bad review.

Together with their clausal complements, factive and response-stance verbs give rise to a presupposition or similar inference: in the case of factive verbs, the embedded proposition is presupposed to be true (460), in the case of response-stance verbs, the embedded proposition need not be true (461), but it is assumed to be discourse-given or otherwise familiar.

(460) The critics regret/realized that the book is out of print.

→ the book is out of print

(461) The critics confirmed that the book is out of print.

→ the book being out of print was mentioned/discussed before

Such inferences or presuppositions of familiarity and givenness are commonly associated with definiteness, and so we assume that the factive or familiarity presuppositions associated with factive and response-stance verbs are (in large part) due to the clausal definite determiner Δ : they are the familiar familiarity or givenness presupposition of the definite determiner.

The second type of finite embedded clauses are open clauses, which are just as in Chapter 4 where we stipulated that all clausal projections denote a predicate $f(e)$ that

⁴ Another suggestive option is to identify the definite determiner with the complementizer *that*. Evidence in favor of such a move is that verbs selecting definite clauses as complements do not allow complementizer drop in English or German (where it also implies embedded verb-second word order). Additionally, the morphological form of the complementizer is also suggestive: in many Indo-European languages the declarative complementizer has the same phonological exponent as a demonstrative: *that* in English, *dass* in German, *que* or *che* in Romance languages.

holds over sets of events. We assume that bridge verbs take open clauses as their complements and include the content thematic role **cont** as part of their lexical entry, just like DP-selecting transitive verbs project a theme role **th**. Interestingly, open finite clauses and non-finite clauses have the same semantic denotation, both denote a predicate $f(e)$.

5.2.3 Definite Clauses as Peninsulas

Definite clauses appear as the complements of factive and other ‘presuppositional’ verbs, and the clausal complements of these verbs are known to be weak or selective islands, as shown in (462) and (463)

(462) *When do the critics regret that the book went out of print?

(463) When did the critics confirm that the book went out of print?

a. When did the critics confirm *X*?

b. *When did the book go out of print?

Above we argued that weak islands effect are intervention effects: a weak island is induced by an intervener, since *wh*-chains (of QP-movement) are blocked across interveners. But what is the intervener in the case of factive complements? We assume that the clausal definite determiner Δ acts as an intervener. At first this might appear rather odd, since definite expressions typically do not induce focus intervention effect, as shown in (464), and as (465) shows they do not induce inner islands either.

(464) Wen hat der Mann wo gesehen?
 who has the man where seen
 ‘Who has the man seen where?’

(465) How did the man sing at the party?

However, in the case above the movement dependency is into the scope S of the definite generalized quantifier in (466), while the embedded CP of definite clauses is the argument of the definite determiner Δ and thus gets substituted for the restriction R , just like an NP within a regular DP.

(466) $\lambda R.\lambda S.\iota x[R(x) \wedge S(x)]$

So we assume that a definite determiner is an intervener with respect to its restriction term, but not with respect to its scope term (cf. DIESING 1990, 1992), and we expect intervention effects for movement out of a definite DP, but not for movement across a definite DP.

Since definite clauses are DPs with the CP within the restrictor of the DP, the definite quantifier acts as an intervener for QP-movement out of definite clauses, rendering definite clauses as peninsulas: Q-based \bar{A} -movement is blocked out of definite clauses. But scope-taking DP-movement is still possible. This is exactly the same behavior we observe with regular, nominal DPs, which may also block sub-extraction, as illustrated by the pair of examples in (467), originally discussed in FIENGO & HIGGINBOTHAM 1981. But while there is a contrast in (467), the second example is typically not perceived as fully unacceptable (ERTESCHIK-SHIR 1973, SIMONENKO 2015).

- (467) a. Who did you see pictures of?
 b. ?Who did you see the picture of?

The element being sub-extracted in (467) is a DP, and we just argued that definite determiner interveners only affect QP-movement, but not DP-movement. The fact that DP-movement in (467) still incurs some acceptability penalty mirrors similar patterns for factive islands. For possible explanations, see SIMONENKO 2015 and Section 5.4

The obligatorily scope-taking behavior of DP-movement can explain why peninsulas ban scope reconstruction and force a wide scope reading on scope taking *wh*-phrases like the *how many*-phrase in (468).

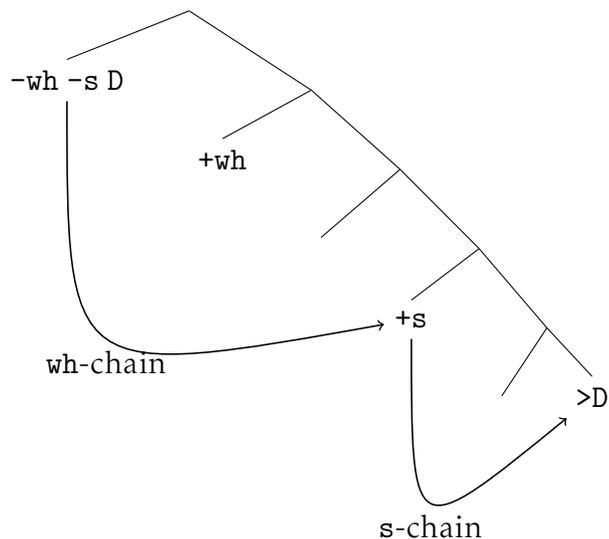
(468) How many books did the authors realize that the critics reviewed?

In Section 4.4, we argued that *wh*-expressions consisting of a DP like the phrase *how many books* carry two \bar{A} -features as in (469): $-\mathbf{wh}$ is introduced by a Q-head and leads to QP-movement (or a \mathbf{wh} -chain) and $-\mathbf{s}$ is introduced by a DP-scope head and leads to DP-movement (or a \mathbf{s} -chain).

(469) $-\mathbf{wh} \ -\mathbf{s} \ D ::$ how many books

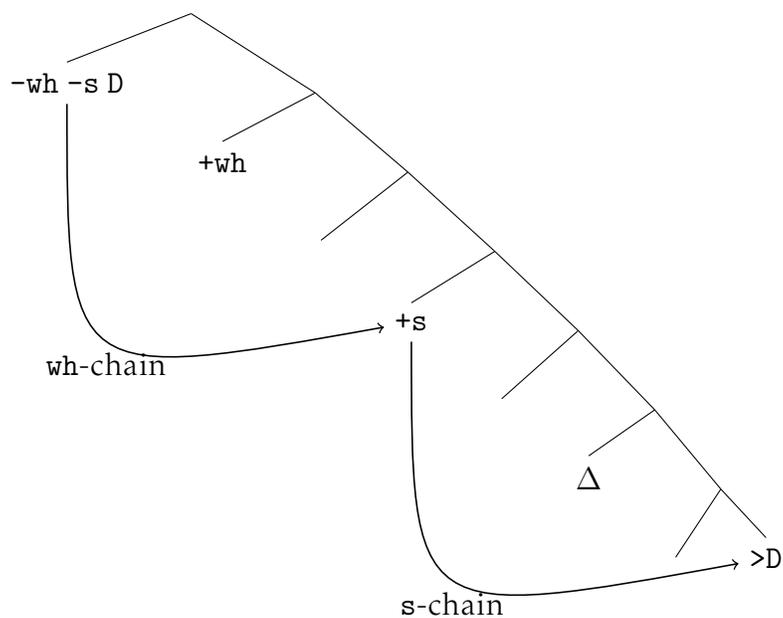
The two licensing features can be discharged in different positions: the $-\mathbf{wh}$ -feature is always checked by the question complementizer. But the scope feature $-\mathbf{s}$ can be checked in any possible scope position. So a constituent question with a DP-*wh* like (469) always involves QP-movement, there is always a \mathbf{wh} -chain started by checking the $-\mathbf{wh}$ feature, but the \mathbf{wh} -movement chain ends at the position in which the $-\mathbf{s}$ feature is checked and the DP *many books* takes scope. Checking the $-\mathbf{s}$ feature starts a \mathbf{s} -movement chain, which ends at the base position, where the moved DP is merged. The basic picture is illustrated in (470).

(470)



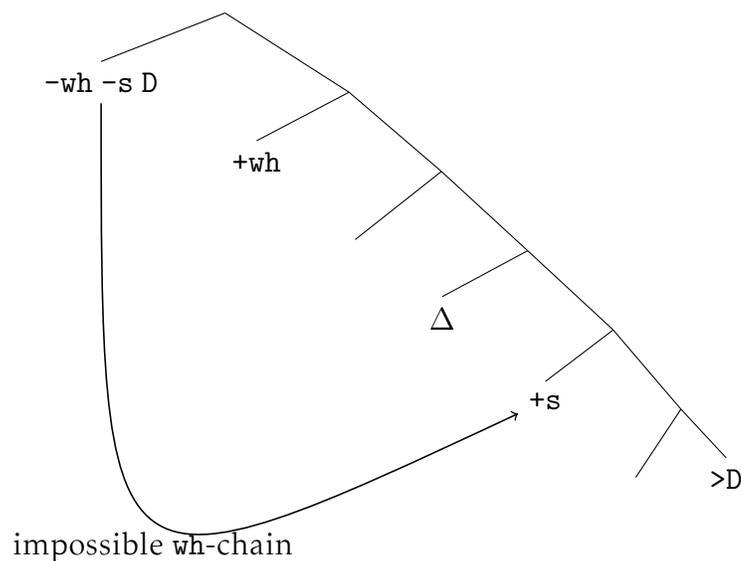
Applied to the weak island example in (468) this means: a *wh*-chain (or QP movement) across the definite determiner Δ is blocked, but a *s*-chain (DP movement) is possible, and so the *wh*-phrase *how many books* has to take scope above the definite determiner Δ and so above the embedding verb *realize*, giving us the wide-scope reading in (471).

(471)



The narrow scope reading in (472) is blocked, because the only way for *how many books* to take scope within the embedded clause would be to check its scope feature $-s$ below the intervener Δ . But this would require the wh -chain of QP-movement to span across Δ , which is impossible, because Δ acts as an intervener.

(472) *



So in summary, only s -chains (DP-movement) can span across a weak-island inducing intervener, while wh -chains (QP-movement) are subject to intervention effects. This characterization of weak islands is essentially the one of CINQUE (1990): weak islands allow extraction of DPs. But instead of just stipulating that only DPs can be extracted out of weak islands, we offered a sketch of an explanation why DP-movement is possible across weak islands, while wh -chains are blocked.

So our account of weak island is mostly syntactic, but interestingly it captures some of the nuances of weak islands, which have been argued to require a semantic account (ABRUSÁN 2014): our rather strict definition of DP-movement requires that the domain quantified over by the moved DP be the domain of individuals. This is the same conclusion

reached by SZABOLCSI & ZWARTS (1993) and ABRUSÁN (2014): a *wh*-expression can escape a weak island if it quantifies over individuals. In addition, we got the scope effect in (472) for free without invoking any additional machinery, like referentiality (RIZZI 1990) or d-linking (PESETSKY 1987), or the requirement that certain questions have a maximal answer (RULLMANN 1995) or a maximally informative answer (RULLMANN & BECK 1998).

5.2.4 Open Clauses

Open clauses are the type of clausal complements selected by a class of propositional attitude verbs informally known as ‘bridge verbs’. These include the most common propositional attitude verbs like *think* and *believe*. We will not attempt to offer an exhaustive list of bridge verbs, instead we note that the clausal complements of bridge verbs show a number of unique features when compared to clausal complements of non-bridge verbs. These differences should justify our assumption that bridge verbs select a different type of clausal complements, namely open clauses as opposed to definite clauses.

Bridge verbs are special, and only the complements of bridge verbs allow complementizer drop in English, while non-bridge verbs like factive *regret* do not, as illustrated in (473).

- (473) a. Mary thinks John left.
 b. *Mary regrets John left.

Similarly, in German the complements of bridge verbs allow verb-second word order, which is normally reserved for main clauses, while complements of factive verbs only allow verb-final word order, as illustrated in (474).

- (474) a. *Maria bedauert Hans is gegangen.
 Maria regrets Hans is gone

'Maria regrets that Hans left'

- b. Maria glaubt Hans is gegangen.
 Maria believes Hans is gone
 'Maria believes Hans left.'

In English, similar embedded main clause phenomena can be observed with the complements of bridge verbs, but not with the complements of factives.

In a fair number of languages with overt *wh*-movement, long-distance questions like (475) can or must be formed using a strategy known as *wh*-scope-marking (DAYAL 1996), where the *wh*-phrase moves only to the edge of the embedded clause, while its question scope in the matrix clause is marked by a default *wh*, the scope marker, which is typically a form of *what* or *how*.

(475) Which car do you think Mary bought?

Example (476) shows an instance of *wh*-scope marking in German, where it is optional.

(476) Was denkst du, welches Auto Maria gekauft hat?
 what think you which car Maria bought has
 '*What do you think which car Maria bought'

Interestingly, *wh*-scope marking seems to enjoy the company of bridge verbs: in German *wh*-scope marking is only possible with bridge verbs, but not with factive verbs as in (477), while in Russian bridge verbs require *wh*-scope marking to form long-distance questions (STEPANOV & STATEVA 2006).

(477) *Was bedauerst du, welches Auto du gekauft hast?
 what regret you which car you bought have

Bridge verbs are also special in their selection behavior: most clausal embedding verbs are responsive verbs as they select both declarative and interrogative complements, and responsive verbs also select nominal DP complements (WHITE & RAWLINS 2018). Compared to this baseline (for which factive verbs are a prime example), bridge verbs are more restrictive: they typically do not accept arbitrary nominal DPs as complements, and many of them only select declarative, but not interrogative clausal complements.

Finally, the phenomenon of negative transportation or neg-raising (FILLMORE 1963, HORN 1971), according to which negation in the matrix clause can be interpreted as a negation of the embedded clause as in (478), seems to be restricted to (a subset of) bridge verbs: all neg-raising predicates listed in HORN 1989 appear to be bridge verbs, but the converse does not hold: verbs of saying are not neg-raising, but they appear to be bridge verbs. Neg-raising verbs are also anti-rogative, they do not select interrogative complements, as first observed by ZUBER (1982).

- (478) The senator doesn't think the bill will get enough votes.
 So she is not willing to discuss it publicly.

With respect to movement and islands, the main topic of this chapter, the complements of bridge verbs are special in that they do not show any island effects, selective or otherwise: only bridge verbs allow adjunct *wh*-phrases like *how* to be extracted out of their complements as in (479), thus allowing the adjunct to be interpreted in the lower clause.

- (479) a. How do you suppose Mary fixed the car?
 b. I suppose Mary fixed the car by replacing a broken spark plug.

Such a low interpretation of adjunct *wh*-phrases is not available for non-bridge verbs like factive *know*:

- (480) a. *How do you know that Mary fixed the car?
 b. I know that Mary fixed the car by replacing a broken spark plug.

While the complements of bridge verbs are different from other finite clausal complements, they behave exactly like infinitival complements, which also allow extracted adjunct *wh*-phrases to be interpreted low as shown in (481) for two types of infinitives.

- (481) a. How did Mary begin to fix the car?
 b. How does Mary want to fix the car?

The similar extraction behavior of infinitives and complements of bridge verbs can be explained by assuming that the latter are open clauses, which have the same semantic denotation as infinitives: both denote a predicate $f(e)$, and so we expect open clauses and infinitives to behave similarly. More importantly, unlike definite clauses, open clauses and infinitives do not project a (focus) intervener, and so they allow any type of movement, in particular QP-movement (or *wh*-chains). With open clauses and infinitives (but not definite clauses) denoting a predicate $f(e)$ that holds over sets of events, we can also implement a proposal by HEGARTY (1992) according to which reconstruction of moved adjuncts across a clause-boundary requires access to event variable of the embedded clause: for open clauses and infinitives the embedded event variable is accessible to the matrix predicate and so adjunct reconstruction is possible, while for definite clauses the embedded event variable is inaccessible and so adjunct reconstruction is not possible.

5.2.4.1 Propositional Content *Explained*

We assume that all propositions must be represented as propositional objects (instead of truth values), since those propositional objects represent the truth-making conditions (or situations) for the proposition. A clause represented as predicate $f(e)$ over sets of

events is assigned as the content of a propositional object via the content role **cont**. For definite clauses, the contents role is part of the definite determiner, which also ι -binds the propositional object variable. Since complements of bridge verbs are open clauses, the content role **cont** must be projected by the lexical entry of the bridge verb, just like DP-selecting transitive verbs project a theme role. However, in contrast to definite clauses the propositional object variable predicated over by the content role is not immediately bound. This allows for the argument of the content role and the verb to be the same. So instead of being a predicate of events, a bridge verb can be a predicate of propositional objects, and its complement clause provides the contents of that propositional object. The sentence (482) may thus be represented as (483), where we changed the thematic role of the external argument to an experiencer (or holder) role **exp**.

(482) Bill believes it is raining.

(483) $\exists o. [\mathbf{believe}(o) \wedge \mathbf{exp}(o) = \mathbf{bill} \wedge \mathbf{cont}(o) = \exists e. \mathbf{rain}(e)]$

In (483) there is no event or state of believing, instead the propositional object itself is the content of the belief.⁵

With the analysis in (483), we may also attempt to explain *explain*, i.e. explain why the two sentences in (484) and (485) are not synonymous (PIETROSKI 2000). In fact, neither one entails the other.

(484) Mary explained that it is raining.

(485) Mary explained the fact that it is raining.

⁵ HEGARTY (2016) adopts a similar analysis of propositional attitude verbs as ‘simply describing’ or predicating over the doxastic situation of the experiencer, but strictly within KRATZER’S (1981) framework of modal interpretation. Using his modal analysis of bridge verbs, he then develops a semantic account of neg-raising that does not rely on the excluded middle. Since propositional objects are the same as modal objects, it would be interesting to see if HEGARTY’S analysis of neg-raising can be translated into the framework assumed here.

In the case of an embedded CP as in (484), the verb *explain* can directly predicate over the propositional object *o*, which represents the truth-making conditions of the complement clause, as shown in (486)

$$(486) \quad \exists o. [\mathbf{explain}(o) \wedge \mathbf{ag}(o) = \mathbf{mary} \wedge \mathbf{cont}(o) = \exists e. \mathbf{rain}(e)]$$

But with a DP complement or a definite clause as complement, as in (485), the verb *explain* cannot directly predicate over a propositional object. Instead *explain* must predicate over an event of explaining and the complement DP, which contains the propositional contents of the embedded clause within it, needs to fill the theme role of said event of explaining *explain*, as in (487).

$$(487) \quad \iota x [\mathbf{fact}(x) \wedge \mathbf{cont}(x) = \exists e. \mathbf{rain}(e) \wedge \exists e. [\mathbf{explain}(e) \wedge \mathbf{th}(e) = x \wedge \mathbf{ag}(e) = \mathbf{mary}]]$$

5.2.5 Anti-Pronominal Contexts, Movement Types, and Weak Islands

In Section 4.4.4.1, we pointed out how our distinction between *wh*-chains (or QP movement) and *s*-chains (or DP-movement) is also made by POOLE (2017) who in turn maps them onto the distinction between ‘A-TYPE extractions’ and ‘B-TYPE extractions’, as formulated by POSTAL (1998). The main explanandum of that work is restrictions on movement out of POSTAL’S (1998) ANTI-PRONOMINAL CONTEXTS, which POOLE (2017) calls Π -positions. POOLE then follows JOHNSON (2012) in assuming that there are two types of \bar{A} -movement: QP-movement and DP-movement. And like JOHNSON (2012) he assumes a multi-dominance structure to allow for a DP to be within a QP, and thus have chained QP+DP movement, which is interpreted as scope-taking *wh*-movement; but in Chapter 4 we show that scope-taking *wh*-movement can easily be accounted for using inverted movement chains and the

introduction of multiple licensing features (-*wh* and -*s*) by separate functional heads (Q head and a scope-taking head).

POOLE (2017) then shows convincingly that POSTAL's (1998) anti-pronominal contexts only allow QP-movement (or *wh*-chains), but not DP-movement (or *s*-chains). In this respect, anti-pronominal contexts are the mirror image of weak islands, which allow *s*-chains, but not *wh*-chains. In terms of POSTAL's (1998) (rather non-descriptive terminology) *s*-chains (DP-movement) correspond to B-extractions, and *wh*-chains (QP-movement) correspond to A-extractions. POSTAL further distinguishes two types of A-extractions: A1-extractions and A2-extractions, which we can translate as follows: an A2-extraction is simple a *wh*-chain (QP-movement), while an A1-extraction is a *wh*-chain followed by an *s*-chain (QP+DP-movement).

With our characterization of weak islands, we can now add one missing piece to POOLE (2017), namely the differential weak island sensitivity of A-extraction when contrasted with B-extractions:⁶ POSTAL (1998) observes that "B-extractions and A1-extractions but not A2-extractions should in general be able to extract from selective islands. In contrast, A1- and A2-extractions but not B-extractions should be able to have extraction sites in [anti-pronominal contexts]" (p. 8).

This full patterns is easily explained: only *s*-chains (DP-movement) can cross weak island interveners, and so B-extractions and A1-extraction, which both involve *s*-chains, are possible out of weak islands. In contrast, *s*-chains (DP-movement) are not possible out of (or into) anti-pronominal contexts, and so B-extractions are banned out of anti-pronominal contexts. The interesting case here are A1-extractions, because for them to be possible out of anti-pronominal contexts, they cannot actually involve a DP-movement step (or *s*-chain). So although A1-extractions can be scope-taking in general, they can not

⁶ POOLE (2017) does discuss intervention effects and how they only occur with QP-movement, but he does not subsume weak islands under the umbrella of intervention effects. Instead he only uses weak islands as a diagnostic to disambiguate scope ambiguities.

be scope taking when out of anti-pronominal contexts. We leave testing this interesting prediction for future research.

5.3 Adjuncts and Coordinate Structures

Adjunction and coordinate structure are often claimed to be islands for extraction. I claim that there is nothing special about adjuncts or coordinate structures, at least not much.

Adjuncts are not per se islands, as will be shown below. But clausal adjuncts are islands. Unlike most clausal complements, clausal adjuncts do not seem to be peninsulas, but actual islands (just like the complements of manner-of-speaking verbs): they seem to block any type of (overt) movement. So we claim that clausal adjuncts are islands because of their clausal nature, not because they are adjuncts. And so we expect small adjuncts to not be island, as confirmed by the data in (488), which is discussed in TRUSWELL 2011.

- (488) a. Who were you sitting next to _ ?
 b. What did you come here to accomplish _ ?

However, subsuming clausal adjuncts under a general principle of clausal islands is not meant as an argument against the possibility that there is something inherent to adjuncts that makes them islands (HUNTER 2011, GRAF 2015).

5.3.1 Coordinate Structure Constraint

Ross (1967) formulated the Coordinate Structure Constraint (CSC), according to which coordinate structures are islands. In addition to barring the extraction of entire conjuncts as in (489), coordinate structures seem to also resist sub-extraction as in (490).

- (489) *This is the magazine which John bought the book and.

- (490) a. *What book did John buy and read the magazine?
 b. *What book did John read the magazine and buy?

However, the latter constraint (which I will be referring to as the Coordinate Structure Constraint) is not without exceptions. The most important class of exceptions is across-the-board (ATB) extraction as in (491), where the same element is extracted from both conjuncts.

- (491) What book did John buy and read?

While the CSC and the possibility of ATB-extraction may easily be handled in syntax (e.g. SCHACHTER 1977, GAZDAR ET AL. 1985, STEEDMAN 1985, GOODALL 1987), the following examples in (492) of grammatical asymmetric extraction from coordinate structures are more challenging for the CSC and arguably impossible to account for in purely syntactic terms LAKOFF (1986).

- (492) a. What did Harry go to the store and buy?
 b. How much can you drink and still stay sober?
 c. That's the stuff that the guys in the Caucasus drink and live to be a hundred.

In Section 2.4 we analyze coordinate structures as a variant of adjunction structures. But unlike adjunction structures, where the adjunct is subordinate to its host phrase, the two (or more) coordinate phrases in a coordinate structure are syntactically (and logically) on equal footing. Or at least they can be, as shown in (493).

- (493) Mary bought *The Lord of the Rings*, and John bought *Pride and Prejudice*.

But in examples like (494) and (495), it is not so clear if the two conjuncts really are on equal footing. Instead in (494), the second conjunct seems to be logically subordinate to the first one, while in (495) the first conjunct seems to be logically subordinate to the second one.

(494) He can drink ten beers and still stay sober.

(495) Harry went to the store and bought a lamp.

These different levels of logical subordination reflect different discourse coherence relations between the two conjuncts: the conjuncts in (495) express a temporal sequence of events, while the conjuncts in (494) express a cause-effect relation. And finally, the two conjuncts in (493) are in a parallel relation. According to KEHLER (2002) these relations between two conjuncts correspond to the discourse coherence relations of CONTIGUITY, CAUSE-EFFECT and RESEMBLANCE, and he observes that the three types of coherence relations between conjuncts allow different types of extraction from coordinate structures.

Coordinate structures in Cause-Effect and Contiguity relations both allow asymmetric extraction from only one conjunct: if the two conjuncts are in a Cause-Effect relation, asymmetric extraction is possible from the first conjunct as in (496), while in the case of a Contiguity relation extraction is possible from the last conjunct, as in (497).

(496) How much can you drink and still stay sober?

(497) What did Harry go to the store and buy?

The examples in (497) and (496) are representatives of two classes of counter-examples to the Coordinate Structure Constraint (CSC), according to which extraction out of coordinate structures is only possible across the board (ATB) from all conjuncts as in (498). But

according to KEHLER (2002), the ban of asymmetric extraction is instead due to the Resemblance, or parallel and contrastive, relation between two conjuncts in (498).

- (498) a. *Which book did Mary buy and John read *Pride and Prejudice*?
 b. Which book did Mary buy and John read?

But how exactly is discourse coherence supposed to affect the (im)possibility of syntactic movement? The easiest explanation is that different coherence relations correspond to different syntactic structures. We observed above that coordinate structures in CONTIGUITY and CAUSE-EFFECT relations seem more like subordinate structures. And our analysis of coordinate structures as a variant of adjunction structures allows us to implement this intuition: in addition to a coordinating *and*, there may also be a subordinating *and*, which behaves more like a subordinating conjunction.

So in some coordinate structures, one of the two conjuncts is in fact an adjunct and this adjunct structure gives rise to either a Cause-Effect or a Contiguity relation: in a coordinate structure in a Cause-Effect relation like (496) the second conjunct is an adjunct, and so extraction out of the first conjunct should be possible, since it is just a complement. In a coordinate structure in a Contiguity relation like (497), the first conjunct is an adjunct, and so extraction out of the second conjunct should be possible.

So I claim that different discourse coherence relations between conjuncts correspond to slightly different syntactic structures of the underlying coordinate structure, and that asymmetric extraction is possible when the coordinate structure itself is asymmetric and thus resembles an adjunction structure. In such adjunction-like asymmetric coordinate structures, asymmetric extraction is possible out of the logically super-ordinate conjunct, but not out of the logically subordinate or adjunct-like conjunct.

Having explained asymmetric extraction out of coordinate structures as extraction out of the host of an adjunction structure, we now need to explain what happens in the case of truly coordinating coordinate structures, where only across-the-board extraction out of both conjuncts is possible as in (499).

(499) Which book did Mary buy and John read?

In terms of incremental syntax, (499) can be treated just like right-node raising, i.e. the string in (500) forms an incomplete expression (assembled via MERGE_{\gg}), which expects a DP to the right, and that expected DP is a mover

(500) $\text{>D. } v :: \text{ Mary buy and John read}$

So overall, the expression for (499) before the final MERGE is as in (501).

(501) $\text{>D. } C :: \text{ which book did mary buy and john read, } D :: \epsilon$

In conclusion, across-the-board movement is not so much a special case of movement, but instead a consequence of the great flexibility of coordination, which can combine two expressions of any type, even complex types. This distinguishes true coordination from adjunction, which can only target host of simple types, better known as maximal projections. So we would predict that if a coordinate structure is asymmetric because it underlyingly is an adjunction structure, only asymmetric extraction should be possible, and across-the-board extraction should be banned.

This sketch of an analysis of extraction from coordinate structures builds on KEHLER's (2002) coherence-based explanation, but it offers a more structural basis, suggesting that different coherence relations are realized by different syntactic structures. In this respect, my analysis is similar to the analyses proposed by POSTAL (1998) and WEISSER (2014), who

in order to maintain the Coordinate Structure Constraint as a syntactic constraint try to explain the reported cases of asymmetric extraction from coordinate structures as cases of extraction from adjunction structures.

5.4 Summary and Other Factors Influencing Islands

One of the goals of this chapter was to argue that (some) island phenomena are not necessarily syntactic in nature. Specifically, we argued that

- specifier islands (including freezing effects) are a necessary consequence of incremental structure building
- factive islands are due to (focus) intervention effects out of definite clauses
- clausal adjuncts are islands because they are definite
- coordinate structures can be asymmetric and behave like adjunction structures, or symmetric and only allow across-the-board movement

In addition, there are other factors that can influence the acceptability of sentences containing island constructions, most notably processing restrictions, plausibility and semantic framing, and information structure.

As far as processing explanations of island effects are concerned I assume that KLUENDER (1992, 2004)⁷ is basically right and that there are non-structural factors influencing the acceptability of sentences with island violations. KLUENDER starts from the established claim that the perceived unacceptability of multiple-center embedding constructions as in (502) is the result of the human sentence processor's inability to parse them due to resource limitations, in particular the human working memory capacity being limited to 7 ± 2 items (MILLER 1956).

⁷ See also KLUENDER & KUTAS (1993).

(502) ?The novel that the critic that the author that the publisher had recently fired had refused to talk to had written a scathing review for has become a bestseller.

KLUENDER extends that line of argumentation to various island phenomena, and I fully agree. But it seems difficult to argue that there is nothing structural about islands, not even a residue. My goal in this chapter has been to attempt an explanation of this structural residue, and so my work can be viewed as complimentary to that of KLUENDER. In contrast, the parsing procedure in Chapter 2 and its application to explain certain islands phenomena in this chapter abstracts away from resource limitations, and my claim is that certain island violating sentences cannot be parsed incrementally, no matter what the resources are. The fact that some island-violating sentences can be structurally very simple, and the results of various sentence processing experiments showing little to no correlation between certain island effects and working memory capacity (SPROUSE, WAGERS & PHILLIPS 2012), corroborate this claim.

Additionally, island effects can also be affected by semantic framing and stereotypicality, constituent questions in particular seem to be sensitive to semantic framing and stereotypicality. DEANE (1991) notices the acceptability contrast in (503) and concludes that stereotypicality affects the acceptability of syntactic movement.

- (503) a. Who did you obtain votes for the impeachment of?
 b. *Who did you describe votes for the impeachment of?

But notice that there is also a contrast in (504), where the constituent questions are replaced with polar questions and a salient element in focus position.

- (504) a. Did you obtain votes for the impeachment of Donald Trump?
 b. #Did you describe votes for the impeachment of Donald Trump?

Taken together, (504) and (503) suggest that semantic framing and stereotypicality may not act as constraints on syntactic movement, but instead as constraints on question formation more generally: there are good and bad questions, irrespective of whether those questions involve *wh*-movement.

Another factor that can influence the acceptability of island constructions is information structure: constituent question formation is associated with focus (as we assumed above) and makes the moved constituent a focused element. Focus typically cannot be placed in backgrounded constituents, and so we expect that extraction out of backgrounded material is degraded. See GOLDBERG 2006 for arguments and examples, and AMBRIDGE & GOLDBERG 2008 for initial experimental evidence.

Finally, the parser most likely has a repair mechanism to make sense of unparsable sentences. If such repair mechanisms exist, it would make sense that participants in experiments can be trained to employ them, which predicts the effects reported in CHAVES & DERY (2018), where participants found subject island violations less severe at the end of an experiment than they did at the beginning.

CHAPTER 6

IMPLICATIONS AND CONCLUSIONS

Looking at our incremental derivations from a purely formal perspective, the changes may seem rather minimal: MERGE and the feature calculus of selection work pretty much the same as in the bottom-up case, with some small adjustments like MERGE for incomplete items, which ultimately bring Minimalist Grammars closer to Combinatory Categorical Grammar. And similarly, movement seems strangely familiar, except that everything is inverted. But inverted movement chains have some interesting conceptual consequences or challenges, which led us to re-assess some of our assumptions about syntactic movement and the lexicon.

In this final chapter, we first offer a very rough sketch of how incremental structure building may work to allow for right adjunction. This will also lead us to a new conception of ‘phases’. Next, we turn to syntactic movement and compare inverted movement chains to the standard copy theory of movement. We conclude this chapter by discussing the relationship between the parser and the grammar, and previous research, which equates the two.

6.1 Putting it all together in Phases

So far we have assume that the goal of incremental structure building is to build a fully connected syntactic structure with a single PF and LF component as soon as possible,

and in the course of our derivation we deleted all intermediate category features. But as briefly discussed in Section 2.3.2 deleting intermediate category features poses a problem for right adjuncts (and coordinate structures): they need access to intermediate category features to adjoin to, but importantly, this is only a problem on the LF side of things. As far as PF is concerned, right adjunction (and coordination) is just string concatenation. On the LF side, however, we minimally need access to the specific semantic variable that the adjunct is supposed to modify, and this is true whether we use Function Application for semantic combination or Predicate Modification.

One way to approach the problem of right adjunction is to give up on the idea that structure building produces a single fully composed meaning component. Instead we assume that during parsing units on the stack are assembled around a single semantic variable, like an event variable or an individual variable. And that these units are not combined via Function Application (see also BECK & TIEMANN 2018 for a similar proposal), instead they remain independent units with their linkage specified in a top-down (or discourse model) component. We may call these units phases.

Importantly, a phase is now strictly a semantic unit, and the only motivation for this notion of phases is semantic modification, not syntactic movement. The organization between phases may still be hierarchical, but a nesting structure like the one used in DRT may also be appropriate. This suggests that a representation of meaning should not be a logical formula as we have constructed so far, but a representation from which a logical formula can be read off. This is exactly the point of Logical Form as used in generative syntax, but notice that despite the use of phases and multiple spell-out, in generative syntax it is still assumed that a meaning can only be constructed once the whole sentence is assembled.

Instead we may assume that the meaning representation being constructed during parsing is hierarchical while the sentence is being processed and then gets flattened

out. And truth evaluation (or closure of event variables) is a process that can be applied whenever necessary, but is also associated with certain linkages between phases, like the one between a matrix clause with a factive predicate and the selected complement clause: when building a sentence like (505) incrementally, the embedded clause can be truth evaluated after integrating *Bill* and once again after integrating the instrument.

(505) John knows that Mary killed Bill with a knife.

From the perspective of syntax and parsing, we can assume that phases remain on the stack as individual units, but are also linked together. This linkage allows the transmission of movers, so movement can in theory proceed across phase boundaries. In practice, however, movement crossing a phase boundary may now be subject to semantic restrictions: any movement restrictions that occur at a clause boundary are now due to the fact that the embedded clause needs to be closed semantically. From a syntactic perspective, (at least some) intermediate phrases are now phases and thus accessible for adjunction: so there should at least be a verbal phase for each event introduced and a nominal phase for each DP. All verbal adjuncts then select the category of the verbal phase, and all nominal adjuncts select the category of the nominal phase.

6.2 (Copy) Theory of Movement

The theory of inverted movement chains advanced in this dissertation is not too far removed from standard theories of syntactic movement, like the copy theory of movement (CHOMSKY 1995). Under the copy theory of movement, all positions of a movement chain contain a copy of the mover, but at SPELL-OUT the different copies may be handled differently. In the prototypical case of overt *wh*-movement as in (506) a copy of the moved

element *which book* occurs at the top of the chain and at the base of the chain. At LF, both copies are legible, but at PF, the lower copy is not spelled out.

- (506) a. Which book do the authors think that the critics like?
 b. [which book] do the authors think that the critics like [which book]

As far as meaning is concerned, inverted movement chains may seem similar to the copy theory of movement: we also assume that a moved expression makes a meaning contribution at all positions of the movement chain, but critically we do not assume that those meaning contributions are the same. In fact, we assume that the core meaning of the moved expression (i.e. the meaning it has when not moved) is only discharged at the base of the chain. The meaning discharged at the top of the chain may be related to the core meaning, but it is introduced by the head, which also introduces the feature licensing the movement chain. The fact that the two meanings need not be the same allows us to handle reconstruction phenomena more flexibly, in particular what is typically called partial reconstruction is often just intermediate scope taking of a scope feature *-s*, which is unrelated to the feature licensing the original movement chain.

As far as pronunciation is concerned, inverted movement chains are very simple: a moved element is inserted where it is pronounced. In the copy theory of movement the fact that only the highest copy of the mover in (506) gets realized at PF needs to be stipulated: the principle of *PRONOUNCE HIGHEST* requires that a mover gets pronounced at its final landing site, unless movement is covert, in which case it doesn't.

Explicitly specifying where a mover is pronounced is less parsimonious than what is being proposed here, but it also offers the flexibility to stipulate that some movers be pronounced in locations other than the highest landing site. And they may even be pronounced in multiple locations, like at some (or all) of the intermediate landing sites

of successive cyclic movement. One phenomenon supposedly suggesting that a mover can be spelled out in multiple locations is *wh*-copying, as illustrated in (507) with an ungrammatical English sentence and its grammatical German translation. The English sentence in (507) is perceived as ungrammatical by most adult native speakers of English, it is common in child English and L2 English, and cross-linguistically.

- (507) a. *Who do you think who John met?
 b. Wen denkst du, wen Hans getroffen hat?

While (507) may suggest that the *wh*-element *who* is being pronounced at its intermediate and its final landing site (e.g. BARBIERS, KOENEMAN & LEKAKOU 2010), such an analysis is not very promising, since it does not generalize to complex *wh*-phrases, where *wh*-copying is generally impossible, as shown in (508).

- (508) *Welche Gäste denkst du welche Gäste man einladen sollte.
 which guests think you which guests one invite should
 'Which guests do you think one should invite'

Instead *wh*-copying may be better analyzed as a special case of the cross-linguistically common phenomenon of *wh*-scope marking (illustrated in (509), see also Section 5.2.4), where the *wh*-phrase in question only moves to the right edge of the embedded clause, while its scope position in a higher clause is marked by a special *wh*-expression (often the equivalent of *what* or *how*).

- (509) Was denkst du, wen Hans getroffen hat?
 what think you who Hans met has
 'Who do you think Hans has met?'
- (510) Was glaubst du, wann Hans an welcher Universität studiert hat?
 What believe you when Hans at which university studied has

‘When do you believe Hans studied at which university?’

Examples like (510), where the scope marker *was* is associated with two embedded *wh*-phrases, strongly suggest that in *wh*-scope marking the dependency between the lower *wh*-phrase(s) and the scope marker is most likely not a direct syntactic one (DAYAL 1994). For more arguments, see DEN DIKKEN (2017), where the alleged direct evidence for intermediate landing sites and successive cyclic movement is critically assessed.

6.3 Parsing and Grammar

Summarizing the crucial steps and motivation of the development of our incremental parsing procedure in Chapter 2 and Chapter 3 may help shedding some light onto the relationship between the parser and the grammar. In the first half of Chapter 2 our goal was to develop an incremental parsing algorithm for an established transformational grammar formalism, Minimalist Grammars (MGs). Our intention was not to develop a left-to-right grammar (as in CHESI 2015) and we were not claiming that structure building should only be viewed from an incremental perspective (as in DEN DIKKEN 2018). Instead we tried to stay as close to standard bottom-up Minimalist Grammars as possible, but in the process of developing the parsing algorithm for MGs, we were forced to extend the structure building operations of MGs to allow for incremental parsing, and some of the incremental variants of MERGE and MOVE may also be useful for bottom-up structure building. This is comparable to Combinatory Categorical Grammar (CCG), where some operations, in particular type-lifting, mostly exist to allow for flexible structure building.

Furthermore, I am not claiming that the parser is the grammar (PHILLIPS 1996). Instead I’m operating under the traditional assumption that a parser needs a grammar according to which it does the parsing, and that for any given grammar there are many different

parsing strategies. What we are claiming instead is language processing as done by humans is incremental and that any linguistic evidence, such as acceptability judgments, reflects not only the grammar, but also the specific incremental parsing procedure. So it is possible that some phenomena that have traditionally been analyzed as constraints within the grammar, may instead be the consequence of a specific incremental parsing algorithm used by the human language processor: in Chapter 5 we argued that specifier islands may be such a phenomenon: sentences involving movement out of specifiers such as (511) may be grammatical according to the grammar of English, but they cannot be parsed, because they are incompatible with the incremental parsing algorithm employed by the human sentence processor.

(511) *Who did a friend of buy a book about Chomsky?

Adopting an incremental parsing perspective to determine which grammatical constraints may not need to be handled within the grammar, can lead to more minimalist grammars, and it may be possible to remove or re-conceptualize some grammatical concepts and devices that have accumulated over the years. Multiple Spell-Out (URIAGEREKA 1999) and the concept of phases are strong candidates for such a re-conceptualization, whose details I leave for future research to be worked out.

From the perspective of the theory of formal languages and parsing we can make the following important observation: Normally, the desired relationship between a given grammar G and a parsing algorithm P is such that P can recognize all and only the sentences generated by G , i.e. all elements in $L(G)$. But this fundamental property does not apply for the parsing algorithm developed in this chapter. There are sentences s which can be generated by the grammar G , but which are not parsable by our parsing algorithm:

sentences containing certain types of island constructions like (511) are examples of such sentences: they can be generated by the grammar G , but they cannot be parsed by P .

Of course, we could follow standard practice and encode the standard island constraints in the grammar. But these constraints are often either too broad or so specific as to be mere restatements of distributional facts. So it may make sense to leave island constraints out of the grammar and attempt to explain island effects through other means.

6.4 Related Work

PHILLIPS (1996, 2003)

Explaining grammar phenomena through parsing is also the goal of PHILLIPS (1996, 2003). In fact, he goes as far as to say that the grammar and the parser are identical. While I do not subscribe to this claim. I'm not arguing against it either, and most of what I am proposing in this dissertation is either compatible with PHILLIPS's claims or even directly adapted from his work. My goal in this dissertation was to propose an incremental parsing algorithm that is based on a well-established bottom-up grammar framework, and I leave open the question how to distinguish parsing from grammatical derivations and whether it makes sense to speak of left-to-right grammars.

PHILLIPS is also trying to explain grammatical phenomena through parsing, but he does not address islands, instead he focuses on constituency and situations where constituency tests yield conflicting results. Many of the grammatical phenomena he takes to show right-branching constituency conflicting with other constituency tests, seem to be based on the notion of *c-command*, as they are often binding phenomena.¹ In this respect, my proposal is both more specific and more radical, as I am claiming (like BRUENING 2014) that

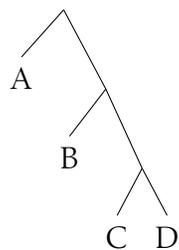
¹ The major exception here being rightward 'movement' phenomena like right-node raising and extraposition, and the non-standard constituent associated with those phenomena.

c-command is a consequence of left-to-right structure building and thus not a necessary part of the grammar.

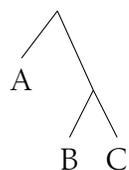
Interpreting PHILLIPS as proposing ‘just’ an incremental parsing algorithm, there are many commonalities: our account of inverse head movement in Chapter 3 is formally equivalent to the one proposed in PHILLIPS 1996. The major innovation we are adding is that head movement can be treated just like phrasal movement if we allow lexical items to be complex Minimalist Grammar expressions. PHILLIPS also discusses that for incremental parsing movement chains need to be inverted, but he does not specify that point very much.

The main point of disagreement between PHILLIPS and me is in how MERGE and constituents are handled: for him, constituents can be destroyed during the course of an unambiguous derivation: in the course of deriving a final tree like the one in (512), PHILLIPS assumes that there is a tree like (513).

(512)



(513)

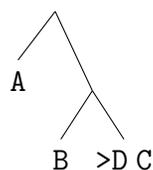


So according to PHILLIPS, C is a complement for part of the derivation until it is shifted into a specifier or even head position, once D comes in. I strongly oppose this (and only this) particular aspect of his proposal. On theoretical grounds, his proposal is not compatible

with X-bar structure and blurs the distinction between heads, specifiers and complements. And while there may be some reason to question the distinction between complements and specifiers, heads seem substantially different to not be treated the same.

To some extent this aspect of his proposal may be based in a confusion due to the fact that in less formal variants of Minimalism the selectors of a head are not notated, so in (513) we do not know that C expects a complement, but in more explicit formalisms like Minimalist Grammars this would be indicated in the notation by writing (513) as in (514).

(514)



However, PHILLIPS full on embraces this notational confusion and generalizes it to adjuncts, so that even if C in (513) does not select anything, the derivation can continue as in (512) with D being an adjunct. This implements a radical version of LARSON'S (1988) VP shell analysis without the need for empty heads.

More problematically, while PHILLIPS'S (1996) parser-grammar allows for very incremental structure building, it may actually delay semantic interpretation. At least if we assume that syntactic positions play an active role in semantics, because then semantics would either constantly need to be backtracking from an interpretation just attempted, like combining B and C in (513) via Function Application, or it would have to wait and see if and until the structure changes, which prevents incremental comprehension. And ultimately, all the experimental evidence for incremental processing shows that comprehension and not just parsing is incremental, and comprehension includes semantic interpretation.

The role of semantics is also problematic in the definition of BRANCH RIGHT (PHILLIPS 1996: p. 19), reproduced in (515). According to this definition a parsing decision is based on a given interpretation. But how can a given interpretation be known before a sentence is parsed?

(515) BRANCH RIGHT

Metric: select the attachment that uses the shortest path(s) from the last item in the input to the current input item.

Reference set: all attachments of a new item that are compatible with a given interpretation.

Aside from the circularity of the proposed parsing principle, which relies on an intended interpretation to derive the intended derivation, this principle make it hard to understand how PHILLIPS imagines that actual left-branching structures are to be built. In particular, how sentence final adjuncts that are not located within the extended VP shell can be integrated. While PHILLIPS (1996) provides good evidence that many adjuncts should be part of a right-branching VP shell extending beyond the verb argument and sketches how such a structure should be built, there are adjuncts, which are most likely not part of such an extended VP shell. Post-verbal temporal adverbs may an example, and (516) shows that the temporal adverb *yesterday* cannot easily be fronted with the VP it follows.

- (516) a. ?Read the book yesterday Mary did.
 b. Read the book Mary did yesterday.

CHESI (2007, 2015)

Investigating grammatical phenomena through the lens of incremental structure building is also the main focus of CHESI and collaborators (e.g. CHESI 2007, BIANCHI & CHESI

2014, CHESI 2015, among others): inspired by the work of PHILLIPS (1996, 2003) and the authors often claim to be applying or implementing his parser/grammar to explain more grammatical phenomena. In particular, CHESI and colleagues also try to explain (non)island phenomena as parsing phenomena, and their treatment of Romance low subjects in BIANCHI & CHESI 2014 provided the basis for the short discussion of possible extractions from low subjects in Section 5.1.5. However, in our treatment of extraction from subjects we did not make reference to BIANCHI & CHESI's (2014) distinction between categorical andthetic structures, and more generally I tend to disagree with some of the specifics of their proposals: in particular, the notion of phase is still a primitive in their grammars, while I am denying the usefulness of having phases as a part of syntax.

In a recent paper, BRATTICO & CHESI (2020) claim to be deriving the constraint that left-branches are phases in a parsing model that they claim is based on the one reported in PHILLIPS 1996. While that amounts to the same thing as deriving the specifier island constraint from left-to-right parsing as is done in Chapter 5, I am not sure if they actually achieve their goal: first it should be noted that specifiers or left-branches cannot be islands in a parsing system like the one proposed by PHILLIPS, because in that system most left branches start out as right branches and could have their gaps filled, before they become left branches. So while BRATTICO & CHESI (2020) are claiming to be implementing the left-to-right parser proposed by PHILLIPS, they are actually doing something different: they build the structure in a strictly right-branching fashion until an item comes in that selects a specifier. Then that element, in what amounts to something like our adjunction operation, picks the appropriate node to attach to and turns the subtree dominated by that node into a specifier.

However, a gap within a subject could still be filled, before the subject is shifted from its MERGERIGHT complement position into a specifier. To rule out this possibility, the authors could argue that gap-filling is somewhat delayed until the gap is confirmed by

the next incoming word², and if that next word turns the subject into a specifier, it would be a phase. But this is in conflict with an active gap filling strategy and the established filled-gap effect. And even if we were to assume a less than active gap filling strategy, their proposal does not actually derive left-branches as islands, all it does is prevent a gap in a left-branch to be filled if that gap is the last element of the future left branch. But it seems to be the case that subjects are islands no matter where their gap is located as shown in (517), and BRATTICO & CHESI (2020) would predict that (517) is grammatical, contrary to fact.

(517) *Who did a friend of _ from Spain kick Mary?

² Their parser is implemented, but I have not been able to determine the exact strategy used in this case, or to test my own examples.

REFERENCES

- ABELS, KLAUS, & AD NEELEMAN. 2012. Linear asymmetries and the LCA. *Syntax* 15(1). 25–74. [p. 37]
- ABNEY, STEVEN P. 1987. *The English Noun Phrase in its Sentential Aspect*. Cambridge, MA: MIT dissertation. [p. 28]
- ABRUSÁN, MÁRTA. 2014. *Weak Island Semantics*. Oxford, UK: Oxford University Press. [pp. 272, 279, and 280]
- ALTMANN, GERRY TM, & YUKI KAMIDE. 1999. Incremental interpretation at verbs: Restricting the domain of subsequent reference. *Cognition* 73(3). 247–264. [p. 14]
- ALTMANN, GERRY TM, & YUKI KAMIDE. 2007. The real-time mediation of visual attention by language and world knowledge: Linking anticipatory (and other) eye movements to linguistic processing. *Journal of Memory and Language* 57(4). 502–518. [p. 14]
- AMBRIDGE, BEN, & ADELE E GOLDBERG. 2008. The island status of clausal complements: Evidence in favor of an information structure explanation. *Cognitive Linguistics* 19(3). 357–389. [p. 294]
- ARAI, MANABU, & CHIE NAKAMURA. 2016. It's harder to break a relationship when you commit long. *PLOS One* 11(6). 1–13. [p. 169]

- AUER, PETER, PETER BAUMANN, & CHRISTIAN SCHWARZ. 2011. Vertical vs. horizontal change in the traditional dialects of southwest Germany: A quantitative approach. *Taal en Tongval* 63(1). 13–41. [p. 333]
- BARBIERS, SJEFF, OLAF KOENEMAN, & MARIKA LEKAKOU. 2010. Syntactic doubling and the structure of wh-chains. *Journal of Linguistics* 46(1). 1–46. [p. 299]
- BARKER, CHRIS. 2002. Continuations and the nature of quantification. *Natural Language Semantics* 10(3). 211–242. [p. 199]
- BARKER, CHRIS. 2015. Scope. In Shalom Lappin & Chris Fox (eds.), *The Handbook of Contemporary Semantic Theory*, 40–76. Malden, MA: Wiley-Blackwell 2nd edn. [pp. 156 and 222]
- BARONE, JOSEPH M. 1972. Anaphoric peninsulas? *Language and Social Interaction* 5(1). 138–140. [p. 267]
- BAUMANN, PETER. 2013. Syntactic category disambiguation within an architecture of human language processing. In *35th Annual Conference of the Cognitive Science Society*, 1833–1838. [pp. 46, 169, and 333]
- BAUMANN, PETER. 2014. Dependencies and hierarchical structure in sentence processing. In *36th Annual Conference of the Cognitive Science Society*, 152–157. [pp. 50 and 332]
- BAUMANN, PETER. 2015. The special status of color in pragmatic reasoning: evidence from a language game. In *37th Annual Conference of the Cognitive Science Society*, 178–183. [p. 332]
- BAUMANN, PETER. in prep. Unifying dynamic and structural continuations. [p. 235]

- BAUMANN, PETER, BRADY CLARK, & STEFAN KAUFMANN. 2014. Overspecification and the cost of pragmatic reasoning about referring expressions. In *36th Annual Conference of the Cognitive Science Society, 1898–1903*. [p. 332]
- BAUMANN, PETER, LARS KONIECZNY, & BARBARA HEMFORTH. 2011. Pragmatic expectations and coreference: How alternative constructions and referring expressions can serve as cues. In *33rd Annual Conference of the Cognitive Science Society, 3293–3298*. [p. 333]
- BAUMANN, PETER, LARS KONIECZNY, & BARBARA HEMFORTH. 2014. Conversational implicatures in anaphora resolution: Alternative constructions and referring expressions. In Barbara Hemforth, Barbara Mertins, & Cathrine Fabricius-Hansen (eds.), *Psycholinguistic Approaches to Meaning and Understanding across Languages, 197–212*. Heidelberg: Springer. [p. 333]
- BAUMANN, PETER, & JANET PIERREHUMBERT. 2014. Using resource-rich languages to improve morphological analysis of under-resourced languages. In *Ninth International Conference on Language Resources and Evaluation (LREC'14), 3355–3359*. [p. 333]
- BECK, SIGRID. 2006. Intervention effects follow from focus interpretation. *Natural Language Semantics* 14(1). 1–56. [pp. 251, 267, 271, and 272]
- BECK, SIGRID, & SONJA TIEMANN. 2018. Towards a model of incremental composition. In *Sinn und Bedeutung, vol. 21 1, 143–162*. [pp. 189, 247, and 296]
- BIANCHI, VALENTINA, & CRISTIANO CHESI. 2014. Subject islands, reconstruction, and the flow of the computation. *Linguistic Inquiry* 45(4). 525–569. [pp. 265, 305, and 306]
- BOCK, KATHRYN. 1995. Sentence production: From mind to mouth. In Joan Miller & Peter Eimas (eds.), *Handbook of Perception and Cognition: Speech, Language, and Communication, vol. 11, 181–216*. New York, NY: Academic Press. [p. 15]

- BOECKX, CEDRIC. 2008. Islands. *Language and Linguistics Compass* 2(1). 151–167. [p. 251]
- BOTT, OLIVER, & WOLFGANG STERNEFELD. 2017. An event semantics with continuations for incremental interpretation. *Journal of Semantics* 34(2). 201–236. [pp. 189 and 247]
- BOUMA, GOSSE, ROBERT MALOUF, & IVAN A SAG. 2001. Satisfying constraints on extraction and adjunction. *Natural Language & Linguistic Theory* 19(1). 1–65. [p. 114]
- BRATTICO, PAULI, & CRISTIANO CHESI. 2020. A top-down, parser-friendly approach to pied-piping and operator movement. *Lingua* 233. 102760. [pp. 306 and 307]
- BRESNAN, JOAN. 2001. Explaining morphosyntactic competition. In Mark Baltin & Chris Collins (eds.), *Handbook of Contemporary Syntactic Theory*, 11–44. Malden, MA: Blackwell. [p. 188]
- BRESNAN, JOAN, ASH ASUDEH, IDA TOIVONEN, & STEPHEN WECHSLER. 2016. *Lexical-Functional Syntax*. Malden, MA: John Wiley & Sons. [pp. 186, 187, and 188]
- BROWN-SCHMIDT, SARAH, & MICHAEL K TANENHAUS. 2006. Watching the eyes when talking about size: An investigation of message formulation and utterance planning. *Journal of Memory and Language* 54(4). 592–609. [p. 15]
- BRUENING, BENJAMIN. 2014. Precede-and-command revisited. *Language* 342–388. [p. 302]
- CABLE, SETH. 2007. *The grammar of Q: Q-particles and the nature of Wh-fronting, as revealed by the Wh-questions of Tlingit*. Cambridge, MA: MIT dissertation. [pp. 20, 22, 25, 148, 149, 221, 230, and 271]
- CABLE, SETH. 2010. *The grammar of Q: Q-particles, wh-movement, and pied-piping*. Oxford, UK: Oxford University Press. [pp. 20, 148, 149, and 271]
- CAHA, PAVEL. 2009. *The Nanosyntax of Case*: University of Tromsø dissertation. [p. 183]

- CATTELL, RAY. 1978. On the source of interrogative adverbs. *Language* 61–77. [pp. 252, 268, and 273]
- CHAMPOLLION, LUCAS. 2015. The interaction of compositional semantics and event semantics. *Linguistics and Philosophy* 38(1). 31–66. [pp. 190, 191, 192, 196, 197, 199, 215, 216, 228, 247, and 248]
- CHAVES, RUI P, & JERUEN E DERY. 2018. Frequency effects in subject islands. *Journal of Linguistics* 1–47. [p. 294]
- CHESI, CRISTIANO. 2007. Five reasons for building phrase structures top-down from left to right. *Nanzan Linguistics: Special Issue* 3. 71–105. [pp. 305 and 306]
- CHESI, CRISTIANO. 2013. Rightward movement from a different perspective. In Gert Webelhuth, Manfred Sailer, & Heike Walker (eds.), *Rightward Movement in a Comparative Perspective*, 243–279. Amsterdam, NL: John Benjamins. [pp. 152 and 156]
- CHESI, CRISTIANO. 2015. On directionality of phrase structure building. *Journal of Psycholinguistic Research* 44(1). 65–89. [pp. 300, 305, and 306]
- CHIERCHIA, GENNARO. 2020. Origins of weak crossover: when dynamic semantics meets event semantics. *Natural Language Semantics* 28(1). 23–76. [p. 235]
- CHOMSKY, NOAM. 1965. *Aspects of the Theory of Syntax*. Cambridge, MA: MIT Press. [p. 49]
- CHOMSKY, NOAM. 1970. Remarks on nominalization. In Roderick Jacobs & Peter Rosenbaum (eds.), *Readings in English Transformational Grammar*, 184–221. Waltham, MA: Ginn. [p. 62]
- CHOMSKY, NOAM. 1977. On *wh*-movement. In Peter Culicover, Thomas Wasow, & Adrian Akmajian (eds.), *Formal Syntax*, 71–132. New York, NY: Academic Press. [pp. 134 and 250]

- CHOMSKY, NOAM. 1995. *The Minimalist Program*. Cambridge, MA: MIT Press. [pp. 17, 21, 24, 26, 227, 254, and 297]
- CHURCH, ALONZO. 1940. A formulation of the simple theory of types. *The Journal of Symbolic Logic* 5(2). 56–68. [p. 64]
- CINQUE, GUGLIELMO. 1990. *Types of \bar{A} -Dependencies*. Cambridge, MA: MIT Press. [pp. 270 and 279]
- CINQUE, GUGLIELMO. 2005. Deriving Greenberg's Universal 20 and its exceptions. *Linguistic Inquiry* 36(3). 315–332. [p. 37]
- CINQUE, GUGLIELMO, & LUIGI RIZZI. 2008. The cartography of syntactic structures. *Studies in Linguistics* 2. 42–58. [p. 115]
- CORUM, CLAUDIA. 1973. Anaphoric peninsulas. In *Ninth Regional Meeting of the Chicago Linguistic Society (CLS-9)*, 89–97. Chicago, IL. [p. 267]
- CROCKER, MATTHEW W., & STEFFAN CORLEY. 2002. Modular Architectures and Statistical Mechanisms: The Case from Lexical Category Disambiguation. In Paola Merlo & Suzanne Stevenson (eds.), *The Lexical Basis of Sentence Processing: Formal, computational and experimental issues*, 157–180. Amsterdam: John Benjamins Publishing. [p. 46]
- DAVIDSON, DONALD. 1967. The logical form of action sentences. In Nicholas Rescher (ed.), *The Logic of Decision and Action*, 81–95. Pittsburgh, PA: University of Pittsburgh Press. [pp. 192, 194, 196, and 208]
- DAYAL, VENEETA. 1996. *Locality in WH Quantification*. Dordrecht, NL: Kluwer. [p. 281]
- DAYAL, VENEETA SRIVASTAV. 1994. Scope marking as indirect *wh*-dependency. *Natural Language Semantics* 2(2). 137–170. [p. 300]

- DE GROOTE, PHILIPPE. 2006. Towards a Montagovian account of dynamics. In *Semantics and Linguistic Theory*, vol. 16, 1–16. [pp. 199, 235, and 274]
- DEANE, PAUL. 1991. Limits to attention: A cognitive theory of island phenomena. *Cognitive Linguistics* 2(1). 1–64. [p. 293]
- DIESING, MOLLY. 1990. *The Syntactic Roots of Semantic Partition*. Amherst, MA: University of Massachusetts dissertation. [p. 276]
- DIESING, MOLLY. 1992. *Indefinites*. Cambridge, MA: MIT Press. [p. 276]
- DEN DIKKEN, MARCEL. 2017. Overtly marked *wh*-paths. In Martin Everaert & Henk C. van Riemsdijk (eds.), *The Wiley Blackwell Companion to Syntax*, Malden, MA: John Wiley & Sons 2nd edn. [p. 300]
- DEN DIKKEN, MARCEL. 2018. *Dependency and Directionality*. Cambridge, UK: Cambridge University Press. [p. 300]
- DOWTY, DAVID. 1991. Thematic proto-roles and argument selection. *Language* 67(3). 547–619. [p. 200]
- DOWTY, DAVID. 2003. The dual analysis of adjuncts/complements in categorial grammar. In Ewald Lang, Claudia Maienborn, & Cathrine Fabricius-Hansen (eds.), *Modifying Adjuncts*, 33–66. Berlin: Mouton de Gruyter. [pp. 110, 113, and 129]
- ELBOURNE, PAUL. 2013. *Definite Descriptions*. Oxford, UK: Oxford University Press. [p. 274]
- ERNST, THOMAS. 2004. *The Syntax of Adjuncts*. Cambridge, UK: Cambridge University Press. [p. 115]
- ERNST, THOMAS. 2020. The syntax of adverbials. *Annual Review of Linguistics* 6. 89–109. [pp. 107, 115, 122, and 123]

- ERTESCHIK-SHIR, NOMI. 1973. *On the Nature of Island Constraints*. Cambridge, MA: MIT dissertation. [p. 276]
- FANG, HAO, MARI OSTENDORF, PETER BAUMANN, & JANET PIERREHUMBERT. 2015. Exponential language modeling using morphological features and multi-task learning. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 23(12). 2410–2421. [p. 332]
- FIENGO, ROBERT, & JAMES HIGGINBOTHAM. 1981. Opacity in NP. *Linguistic Analysis* 7(4). 395–421. [p. 276]
- FILLMORE, CHARLES J. 1963. The position of embedding transformations in a grammar. *Word* 19(2). 208–231. [p. 282]
- FODOR, JANET DEAN. 1978. Parsing strategies and constraints on transformations. *Linguistic Inquiry* 9(3). 427–473. [p. 167]
- FOWLIE, MEAGHAN. 2013. Order and optionality: Minimalist grammars with adjunction. In *Mathematics of Language (MOL 13)*, 12–20. Sofia, Bulgaria. [p. 123]
- FOX, DANNY. 1999. Reconstruction, binding theory, and the interpretation of chains. *Linguistic Inquiry* 30(2). 157–196. [pp. 234 and 243]
- FOX, DANNY. 2002. Antecedent-contained deletion and the copy theory of movement. *Linguistic Inquiry* 33(1). 63–96. [p. 234]
- FOX, DANNY, & JON NISSENBAUM. 1999. Extraposition and scope: A case for overt QR. In *18th West Coast Conference on Formal Linguistics (WCCFL)*, 132–144. [pp. 144, 152, and 156]
- FRANK, ROBERT. 1998. Structural complexity and the time course of grammatical development. *Cognition* 66(3). 249–301. [p. 121]

- FRANK, ROBERT, & K VIJAY-SHANKER. 2001. Primitive c-command. *Syntax* 4(3). 164–204. [p. 121]
- FRAZIER, MICHAEL, LAUREN ACKERMAN, PETER BAUMANN, DAVID POTTER, & MASAYA YOSHIDA. 2015. Wh-filler-gap dependency formation guides reflexive antecedent search. *Frontiers in Psychology* 6(1504). [p. 332]
- FREY, WERNER, & HANS-MARTIN GÄRTNER. 2002. On the treatment of scrambling and adjunction in Minimalist Grammars. In *Formal Grammar (FGTrento)*, 41–52. [p. 110]
- GÄRTNER, HANS-MARTIN, & JENS MICHAELIS. 2003. A note on countercyclicity and Minimalist Grammars. In *FGVienna: The 8th Conference on Formal Grammar*, 95–109. Vienna. [pp. 120 and 158]
- GÄRTNER, HANS-MARTIN, & JENS MICHAELIS. 2010. On the treatment of multiple-wh-interrogatives in Minimalist Grammars. In *Language and Logos*, vol. 72, . [p. 157]
- GAZDAR, GERALD, EWAN KLEIN, GEOFFREY K. PULLUM, & IVAN A. SAG. 1985. *Generalized Phrase Structure Grammar*. Cambridge, MA: Harvard University Press. [pp. 134 and 288]
- GIBSON, EDWARD. 1991. *A Computational Theory of Human Linguistic Processing: Memory Limitations and Processing Breakdown*. Pittsburgh, PA: Carnegie Mellon University dissertation. [pp. 68 and 164]
- GIBSON, EDWARD. 1998. Linguistic complexity: locality of syntactic dependencies. *Cognition* 68. 1–76. [p. 50]
- GOLDBERG, ADELE E. 2006. *Constructions at Work: The Nature of Generalization in Language*. Oxford, UK: Oxford University Press. [p. 294]
- GOODALL, GRANT. 1987. *Parallel Structures in Syntax*. Cambridge, UK: Cambridge University Press. [p. 288]

- GRAF, THOMAS. 2013. *Local and Transderivational Constraints in Syntax and Semantics*. Los Angeles, CA: University of California dissertation. [p. 98]
- GRAF, THOMAS. 2014. Models of adjunction in Minimalist Grammars. In *Formal Grammar*, 52–68. [pp. 110 and 114]
- GRAF, THOMAS. 2015. The syntactic algebra of adjuncts. In *Chicago Linguistic Society (CLS-49)*, Chicago, IL. [p. 287]
- GREENBERG, JOSEPH. 1963. Some universals of grammar with particular reference to the order of meaningful elements. In Joseph Greenberg (ed.), *Universals of Language*, chap. 5, 73–113. MIT Press. [p. 37]
- GRIMSHAW, JANE. 2000. Locality and extended projection. In Peter Coopmans, Martin Everaert, & Jane (eds.), *Lexical Specification and Insertion*, 115–133. Amsterdam, NL: John Benjamins. [p. 203]
- GRISSOM II, ALVIN, NAHO ORITA, & JORDAN BOYD-GRABER. 2016. Incremental prediction of sentence-final verbs: Humans versus machines. In *20th SIGNLL Conference on Computational Natural Language Learning*, 95–104. [pp. 15 and 170]
- GROVE, JULIAN, & EMILY HANINK. 2016. Article selection and anaphora in the German relative clause. In *Semantics and Linguistic Theory*, vol. 26, 417–432. [p. 127]
- GRUNE, DICK, & CERIEL JH JACOBS. 1990. *Parsing Techniques: A Practical Guide*. Hemel Hempstead, UK: Ellis Horwood. [pp. 46, 49, and 167]
- HAIDER, HUBERT. 2004. Pre-and postverbal adverbials in OV and VO. *Lingua* 114(6). 779–807. [pp. 107, 122, and 123]

- HALE, JOHN. 2001. A probabilistic Earley parser as a psycholinguistic model. In *Second Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*, Pittsburgh, PA. [p. 49]
- HALE, JOHN. 2006. Uncertainty about the rest of the sentence. *Cognitive Science: A Multidisciplinary Journal* 30(4). 643 – 672. [pp. 49 and 167]
- HALE, JOHN. 2014. *Automaton Theories of Human Sentence Comprehension*. Stanford, CA: CSLI Publications. [p. 50]
- HALE, JOHN T, & EDWARD P STABLER. 2005. Strict deterministic aspects of minimalist grammars. In *International Conference on Logical Aspects of Computational Linguistics*, 162–176. [p. 169]
- HARKEMA, HENDRIK. 2001. *Parsing Minimalist Languages*. Los Angeles, CA: University of California dissertation. [p. 58]
- HE, YANZHANG, PETER BAUMANN, HAO FANG, BRIAN HUTCHINSON, AARON JAECH, MARI OSTENDORF, ERIC FOSLER-LUSSIER, & JANET PIERREHUMBERT. 2016. Using pronunciation-based morphological subwords to improve OOV handling in keyword search. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 24(1). 79–92. [p. 332]
- HE, YANZHANG, BRIAN HUTCHINSON, PETER BAUMANN, MARI OSTENDORF, ERIC FOSLER-LUSSIER, & JANET PIERREHUMBERT. 2014. Subword-based modeling for handling OOV words in keyword spotting. In *2014 IEEE International Conference on Acoustic, Speech and Signal Processing (ICASSP)*, 7864–7868. [p. 333]
- HEGARTY, MICHAEL. 1992. Adjunct extraction without traces. In *Tenth West Coast Conference on Formal Linguistics (WCCFL)*, 209–223. [p. 283]

- HEGARTY, MICHAEL. 2016. *Modality and Propositional Attitudes*. Cambridge, UK: Cambridge University Press. [p. 284]
- HEIM, IRENE, & ANGELIKA KRATZER. 1998. *Semantics in Generative Grammar*. Oxford, UK: Blackwell. [pp. 20, 190, 192, 194, 195, 197, 228, and 236]
- HEMFORTH, BARBARA, LARS KONIECZNY, CHRISTOPH SCHEEPERS, SAVÉRIA COLONNA, SARAH SCHIMKE, PETER BAUMANN, & JOEL PYNTE. 2010. Language specific preferences in anaphor resolution: Exposure or Gricean maxims. In *32nd Annual Conference of the Cognitive Science Society*, 2218–2223. [p. 333]
- HENDRIKS, HERMAN. 1993. *Studied Flexibility: Categories and Types in Syntax and Semantics*. Amsterdam, NL: ILLC dissertation. [pp. 197, 215, and 216]
- HONCOOP, MARTIN. 1998. *Dynamic Excursions on Weak Islands*. Leiden, NL: University of Leiden dissertation. [p. 271]
- HORN, LAURENCE R. 1971. Negative transportation: Unsafe at any speed. In *Chicago Linguistic Society (CLS-7)*, 120–133. Chicago, IL. [p. 282]
- HORN, LAURENCE R. 1989. *A Natural History of Negation*. Chicago, IL: University of Chicago Press. [p. 282]
- HUNTER, TIM. 2011. *Syntactic Effects of Conjunctivist Semantics: Unifying Movement and Adjunction*. Amsterdam, NL: John Benjamins. [pp. 110 and 287]
- HUNTER, TIM. 2019. Left-corner parsing of minimalist grammars. In Robert Berwick & Edward Stabler (eds.), *Minimalist Parsing*, 125–158. Oxford, UK: Oxford University Press. [pp. 25, 102, and 165]
- HUNTER, TIM, & ROBERT FRANK. 2014. Eliminating rightward movement: Extraposition as flexible linearization of adjuncts. *Linguistic Inquiry* 45(2). 227–267. [p. 127]

- HUNTER, TIM, MILOŠ STANOJEVIĆ, & EDWARD STABLER. 2019. The active-filler strategy in a move-eager left-corner minimalist grammar parser. In *Workshop on Cognitive Modeling and Computational Linguistics*, 1–10. Minneapolis, MN. [pp. 25, 36, 58, 71, 159, 165, and 166]
- JOHNSON, KYLE. 2012. Towards deriving differences in how wh movement and qr are pronounced. *Lingua* 122(6). 529–553. [pp. 230 and 285]
- JOSHI, ARAVIND K, LEON S LEVY, & MASAKO TAKAHASHI. 1975. Tree adjunct grammars. *Journal of Computer and System Sciences* 10(1). 136–163. [p. 26]
- KAMP, HANS. 1979. Events, instants and temporal reference. In Rainer Bäuerle, Urs Egli, & Arnim von Stechow (eds.), *Semantics from Different Points of View*, 376–418. Berlin: Springer. [p. 194]
- KAPLAN, RONALD M. 1972. Augmented transition networks as psychological models of sentence comprehension. *Artificial Intelligence* 3. 77–100. [p. 85]
- KAPLAN, RONALD M. 1974. *Transient Processing Load in Relative Clauses*. Cambridge, MA: Harvard University dissertation. [p. 85]
- KAPLAN, RONALD M, & JOAN BRESNAN. 1982. Lexical-Functional Grammar: A formal system for grammatical representations. In Joan Bresnan (ed.), *The Mental Representation of Grammatical Relations*, 173–281. Cambridge, MA: MIT Press. [p. 134]
- KASTNER, ITAMAR. 2015. Factivity mirrors interpretation: The selectional requirements of presuppositional verbs. *Lingua* 164. 156–188. [pp. 252, 268, and 272]
- KEHLER, ANDREW. 2002. *Coherence, Reference, and the Theory of Grammar*. Stanford, CA: Center for the Study of Language and Information (CSLI). [pp. 289, 290, and 291]

- KIPARSKY, PAUL, & CAROL KIPARSKY. 1970. Fact. In M. Bierwisch & K.E. Heidolph (eds.), *Progress in Linguistics*, The Hague, NL: Mouton. [pp. 252, 268, and 273]
- KLECHA, PETER, & MARTINA MARTINOVIĆ. 2015. Exhaustivity, predication and the semantics of movement. In *Berkeley Linguistics Society*, vol. 41 41, 267–286. [pp. 221 and 237]
- KLUENDER, ROBERT. 1992. Deriving island constraints from principles of predication. In Helen Goodluck & Michael Rochemont (eds.), *Island Constraints: Theory, Acquisition, and Processing*, 223–258. Dordrecht, NL: Kluwer. [pp. 292 and 293]
- KLUENDER, ROBERT. 2004. Are subject islands subject to a processing account. In *West Coast Conference on Formal Linguistics (WCCFL)*, vol. 23, 475–499. [p. 292]
- KLUENDER, ROBERT, & MARTA KUTAS. 1993. Subjacency as a processing phenomenon. *Language and Cognitive Processes* 8(4). 573–633. [p. 292]
- KOBELE, GREGORY M. 2006. *Generating Copies: An investigation into structural identity in language and grammar*. Los Angeles, CA: University of California dissertation. [pp. 33, 175, and 190]
- KOBELE, GREGORY M. 2010. A formal foundation for A and A-bar movement. In *Mathematics of Language (MOL 10/11)*, 145–159. [pp. 134, 135, 138, and 139]
- KOBELE, GREGORY M. 2010. Without remnant movement, MGs are context-free. In *Mathematics of Language (MOL 10/11)*, 160–173. [p. 102]
- KOBELE, GREGORY M. 2015. LF-copying without LF. *Lingua* 166. 236–259. [p. 183]
- KOBELE, GREGORY M., SABRINA GERTH, & JOHN HALE. 2013. Memory resource allocation in top-down minimalist parsing. In Glyn Morrill & Mark-Jan Nederhof (eds.), *Formal Grammar*, 32–51. Berlin: Springer. [p. 58]

- KONIECZNY, LARS. 2000. Locality and parsing complexity. *Journal of Psycholinguistic Research* 29(6). 627–645. [pp. 15 and 170]
- KONIECZNY, LARS, & PHILIPP DÖRING. 2003. Anticipation of clause-final heads: Evidence from eyetracking and srns. In *4th International Conference on Cognitive Science and 7th Conference of the Australasian Society for Cognitive Science*, 330–335. Sydney. [pp. 15 and 170]
- KONIECZNY, LARS, HELMUT WELDLE, SASCHA WOLFER, DANIEL MÜLLER, & PETER BAUMANN. 2010. Anaphora and local coherences. In *32nd Annual Conference of the Cognitive Science Society*, 1204–1209. [p. 333]
- KOTEK, HADAS. 2014. *Composing Questions*. Cambridge, MA: MIT dissertation. [pp. 251, 267, 271, and 272]
- KRACHT, MARCUS. 2011. *Interpreted Languages and Compositionality*. Dordrecht, NL: Springer. [pp. 26 and 30]
- KRATZER, ANGELIKA. 1981. The notional category of modality. In Hans Eikmeyer & Hannes Rieser (eds.), *Words, Worlds, and Contexts: New Approaches in Word Semantics*, Berlin: Walter de Gruyter. [p. 284]
- KRATZER, ANGELIKA. 1996. Severing the external argument from its verb. In Johan Rooryck & Laurie Zaring (eds.), *Phrase Structure and the Lexicon*, 109–137. Berlin: Springer. [pp. 114, 185, 199, 200, 215, and 217]
- LAKOFF, GEORGE. 1971. Deletion paths and the invasion of anaphoric islands. *Papers in Linguistics* 4. 197–198. [p. 267]
- LAKOFF, GEORGE. 1986. Frame semantic control of the coordinate structure constraint. In *Chicago Linguistic Society (CLS-22)*, Chicago, IL. [p. 288]

- LANDMAN, FRED. 2000. *Evants and Plurality: The Jerusalem Lectures*. Dordrecht, NL: Kluwer. [pp. 197 and 216]
- LARSON, RICHARD K. 1988. On the double object construction. *Linguistic Inquiry* 19(3). 335–391. [p. 304]
- LASNIK, HOWARD. 1999. Chains of arguments. In Samuel D. Epstein & Norbert Hornstein (eds.), *Working Minimalism*, 189–215. Cambridge, MA: MIT Press. [p. 227]
- LEBEAUX, DAVID. 1988. *Language Acquisition and the Form of the Grammar*. Amherst, MA: University of Massachusetts dissertation. [p. 144]
- LEBEDEVA, EKATERINA. 2012. *Expressing Discourse Dynamics Through Continuations*. Nancy: Université de Lorraine dissertation. [p. 235]
- LEVELT, WILLEM. 1989. *Speaking: From Intention to Articulation*. Cambridge, MA: MIT Press. [p. 15]
- MAIER, WOLFGANG, MIRIAM KAESHAMMER, PETER BAUMANN, & SANDRA KÜBLER. 2014. Disco-suite - a parser test suite for German discontinuous structures. In *Ninth International Conference on Language Resources and Evaluation (LREC'14)*, 2905–2912. [p. 333]
- MARR, DAVID. 1982. *Vision: A Computational Approach*. San Francisco, CA: Freeman & Co. [p. 62]
- MAY, ROBERT. 1977. *The Grammar of Quantification*. Cambridge, MA: MIT dissertation. [p. 151]
- MCCARTHY, JOHN. 1960. Recursive functions of symbolic expressions and their computation by machine, part i. *Communications of the ACM* 3(4). 184–195. [p. 42]

- MELVOLD, JANIS. 1991. Factivity and definiteness. *MIT Working Papers in Linguistics* 15. 97–117. [pp. 252, 268, and 272]
- MILLER, GEORGE A. 1956. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review* 63. 81–97. [p. 292]
- MOLTMANN, FRIEDERIKE. 2020. Truthmaker semantics for natural language: Attitude verbs, modals, and intensional transitive verbs. *Theoretical Linguistics* 46(3-4). 159–200. [pp. 194, 203, and 273]
- MOMMA, SHOTA, L ROBERT SLEVC, & COLIN PHILLIPS. 2016. The timing of verb selection in japanese sentence production. *Journal of Experimental Psychology: Learning, Memory, and Cognition* 42(5). [p. 15]
- MONTAGUE, RICHARD. 1970. Universal grammar. *Theoria* 36(3). 373–398. [pp. 190, 192, 194, 196, and 208]
- MONTAGUE, RICHARD. 1973. The proper treatment of quantification in ordinary English. In Jaakko Hintikka, Julius Moravcsik, & Patrick Suppes (eds.), *Approaches to Natural Language*, 221–242. Dordrecht, NL: Reidel Publishing. [pp. 15, 190, 192, 194, 196, and 208]
- MOULTON, KEIR. 2015. CPs: Copies and compositionality. *Linguistic Inquiry* 46(2). 305–342. [pp. 194, 203, and 273]
- MUNN, ALAN BOAG. 1993. *Topics in the Syntax and Semantics of Coordinate Structures*. College Park, MD: University of Maryland dissertation. [p. 128]
- MUYSKENS, PIETER. 1982. Parameterizing the notion 'head'. *Journal of Linguistic Research* 2. 57–75. [p. 28]

- PARSONS, TERENCE. 1990. *Events in the Semantics of English: A Study in Subatomic Semantics*. Cambridge, MA: MIT Press. [pp. 192, 194, 196, and 208]
- PAUL, HERMANN. 1880. *Prinzipien der Sprachgeschichte*. Halle: Max Niemeyer. [p. 15]
- PESETSKY, DAVID. 1987. Wh-in-situ: Movement and unselective binding. In Eric Reuland & Alice ter Meulen (eds.), *The Representation of (In)Definiteness*, 98–129. Cambridge, MA: MIT Press. [p. 280]
- PESETSKY, DAVID. 1995. *Zero Syntax: Experiencers and Cascades*. Cambridge, MA: MIT Press. [p. 123]
- PHILLIPS, COLIN. 1996. *Order and Structure*. Cambridge, MA: MIT dissertation. [pp. 25, 63, 72, 173, 175, 176, 300, 302, 303, 304, 305, and 306]
- PHILLIPS, COLIN. 2003. Linear order and constituency. *Linguistic Inquiry* 34(1). 37–90. [pp. 63, 72, 302, and 306]
- PIETROSKI, PAUL M. 2000. On explaining that. *The Journal of Philosophy* 97(12). 655–662. [pp. 273 and 284]
- PIETROSKI, PAUL M. 2005. *Events and Semantic Architecture*. Oxford, UK: Oxford University Press. [pp. 192, 194, 196, and 208]
- POLLARD, CARL, & IVAN SAG. 1994. *Head-Driven Phrase Structure Grammar*. Chicago, IL: University of Chicago Press. [p. 134]
- POOLE, ETHAN. 2017. *Movement and the Semantic Type of Traces*. Amherst, MA: University of Massachusetts dissertation. [pp. 230, 232, 233, 251, 267, 271, 272, 285, and 286]
- POSTAL, PAUL. 1969. Anaphoric islands. In *Fifth Regional Meeting of the Chicago Linguistic Society (CLS-5)*, 205–239. Chicago, IL. [p. 267]

- POSTAL, PAUL. 1998. *Three Investigations of Extraction*. Cambridge, MA: MIT Press. [pp. 233, 285, 286, and 291]
- RIZZI, LUIGI. 1990. *Relativized Minimality*. Cambridge, MA: MIT Press. [p. 280]
- ROBERTS, IAN. 2010. *Agreement and Head Movement: Clitics, Incorporation, and Defective Goals*. Cambridge, MA: MIT Press. [p. 175]
- ROSS, JOHN ROBERT. 1967. *Constraints on Variables in Syntax*. Cambridge, MA: MIT dissertation. [pp. 23, 250, 251, and 287]
- ROSS, JOHN ROBERT. 1986. *Infinite Syntax!* Norwood, NJ: Ablex Publishing Corporation. [p. 250]
- RULLMANN, HOTZE. 1995. *Maximality in the Semantics of Wh-Constructions*. Amherst, MA: University of Massachusetts dissertation. [p. 280]
- RULLMANN, HOTZE, & SIGRID BECK. 1998. Presupposition projection and the interpretation of *which*-questions. In *Semantics and Linguistic Theory (SALT VIII)*, 215–232. Cambridge, MA. [p. 280]
- SAUERLAND, ULI. 1998. *The Meaning of Chains*. Cambridge, MA: MIT dissertation. [p. 154]
- DE SAUSSURE, FERDINAND. 1916. *Cours de Linguistique Générale*. Lausanne, CH: Payot. [pp. 19, 26, 30, and 77]
- SCHACHTER, PAUL. 1977. Constraints on coördination. *Language* 53(1). 86–103. [p. 288]
- SCHRIEFERS, HERBERT, ENCARNACION TERUEL, & RAIK-MICHAEL MEINSHAUSEN. 1998. Producing simple sentences: Results from picture–word interference experiments. *Journal of Memory and Language* 39(4). 609–632. [p. 15]

- SCHUMACHER, R. ALEXANDER. 2018. *Prepositions and Verbs in the Syntax and the Lexicon*. Evanston, IL: Northwestern University dissertation. [p. 218]
- SCHÜTZE, CARSON T. 2001. On the nature of default case. *Syntax* 4(3). 205–238. [p. 98]
- SIMONENKO, ALEXANDRA. 2015. Semantics of dp islands: the case of questions. *Journal of Semantics* 33(4). 661–702. [p. 276]
- SPROUSE, JON, MATT WAGERS, & COLIN PHILLIPS. 2012. A test of the relation between working memory capacity and syntactic island effects. *Language* 88. 82–123. [p. 293]
- STABLER, EDWARD. 1997. Derivational minimalism. In Christian Retoré (ed.), *Logical Aspects of Computational Linguistics*, 68–95. Berlin: Springer. [pp. 17, 21, 24, and 26]
- STABLER, EDWARD P. 1999. Remnant movement and complexity. In G. Bouma, G. Kruijff, E. Hinrichs, & R. Oehrle (eds.), *Constraints and Resources in Natural Language Syntax and Semantics*, 299–326. Stanford, CA: Center for the Study of Language and Information (CSLI). [pp. 253 and 254]
- STABLER, EDWARD P. 2001. Recognizing head movement. In *International Conference on Logical Aspects of Computational Linguistics*, 245–260. [p. 175]
- STABLER, EDWARD P. 2006. Sideways without copying. In *11th Conference on Formal Grammar*, 157–170. [pp. 31, 120, and 158]
- STABLER, EDWARD P. 2011. Computational perspectives on minimalism. In Cedric Boeckx (ed.), *The Oxford Handbook of Linguistic Minimalism*, 617–643. Oxford, UK: Oxford University Press. [pp. 27, 30, 31, 34, 36, 37, 42, 157, and 158]
- STABLER, EDWARD P. 2013. Two models of minimalist, incremental syntactic analysis. *Topics in Cognitive Science* 5(3). 611–633. [pp. 58 and 167]

- STANOJEVIĆ, MILOŠ, & EDWARD STABLER. 2018. A sound and complete left-corner parsing for minimalist grammars. In *Eighth Workshop on Cognitive Aspects of Computational Language Learning and Processing*, 65–74. [pp. 25, 36, 58, 71, 159, 165, and 166]
- STANOJEVIĆ, MILOŠ, JOHN HALE, & MARK STEEDMAN. 2020. Predictive processing of coordination in CCG. In *33rd Annual CUNY Conference on Human Sentence Processing*, Amherst, MA. [pp. 118, 120, and 130]
- STARKE, MICHAL. 2009. Nanosyntax: A short primer to a new approach to language. *Nordlyd* 36(1). 1–6. [pp. 174 and 186]
- STEEDMAN, MARK. 1985. Dependency and coördination in the grammar of Dutch and English. *Language* 61(3). 523–568. [pp. 26 and 288]
- STEEDMAN, MARK, & JASON BALDRIDGE. 2011. Combinatory Categorical Grammar. In Robert D. Borsley & Kersti Börjars (eds.), *Non-Transformational Syntax: Formal and Explicit Models of Grammar*, 181–224. Malden, MA: Blackwell. [p. 26]
- STEPANOV, ARTHUR, & PENKA STATEVA. 2006. Successive cyclicity as residual wh-scope marking. *Lingua* 116(12). 2107–2153. [p. 281]
- STOWE, LAURIE A. 1986. Parsing wh-constructions: Evidence for on-line gap location. *Language and Cognitive Processes* 1(3). 227–245. [p. 167]
- SZABOLCSI, ANNA, & MARCEL DEN DIKKEN. 1999. Islands. *Glott International* 4(6). 3–8. [p. 251]
- SZABOLCSI, ANNA, & FRANS ZWARTS. 1993. Weak islands and an algebraic semantics for scope taking. *Natural Language Semantics* 1(3). 235–284. [p. 280]
- TAKAHASHI, DAIKO. 1994. *Minimality of Movement*. Storrs, CT: University of Connecticut dissertation. [p. 265]

- TANENHAUS, MICHAEL K, MICHAEL J SPIVEY-KNOWLTON, KATHLEEN M EBERHARD, & JULIE C SEDIVY. 1995. Integration of visual and linguistic information in spoken language comprehension. *Science* 268(5217). 1632–1634. [p. 14]
- TOMITA, YU. 2016. Solving event quantification and free variable problems in semantics for minimalist grammars. In *30th Pacific Asia Conference on Language, Information and Computation (PACLIC 30)*, 219–227. Seoul. [pp. 190, 199, 201, and 248]
- TOMITA, YU. 2017. Towards a pied-piping account for wh-scope within compositional event semantics in minimalism. In *Chicago Linguistic Society (CLS-53)*, 381–395. Chicago, IL. [pp. 201 and 248]
- TORR, JOHN, & EDWARD P. STABLER. 2016. Coordination in minimalist grammars: Excorporation and across the board (head) movement. In *Proceedings of the 12th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+12)*, 1–17. Düsseldorf, Germany. [pp. 36, 37, 110, 111, and 117]
- TRUESWELL, JOHN C., MICHAEL K. TANENHAUS, & SUSAN M. GARNSEY. 1994. Semantic influences on parsing: Use of thematic role information in syntactic ambiguity resolution. *Journal of Memory and Language* 33. 285–318. [p. 14]
- TRUSWELL, ROBERT. 2011. *Events, Phrases, and Questions*. Oxford, UK: Oxford University Press. [p. 287]
- URIAGEREKA, JUAN. 1999. Multiple spell-out. In Samuel D. Epstein & Norbert Hornstein (eds.), *Working Minimalism*, 251–282. Cambridge, MA: MIT Press. [pp. 253, 254, 255, and 301]
- VAN URK, COPPE. 2015. *A Uniform Syntax for Phrasal Movement: A Case Study of Dinka Bor*. Cambridge, MA: MIT dissertation. [pp. 133 and 139]

- VASISHTH, SHRAVAN, & RICHARD L LEWIS. 2006. Argument-head distance and processing complexity: Explaining both locality and antilocality effects. *Language* 82(4). 767–794. [pp. 15 and 170]
- WANNER, ERIC, & MICHAEL MARATSOS. 1978. An ATN approach to comprehension. In Morris Halle, Joan Bresnan, & George A Miller (eds.), *Linguistic Theory and Psychological Reality*, chap. 3, 119–161. Cambridge, MA: MIT Press. [p. 85]
- WEISSER, PHILIPP. 2014. *Derived Coordination: A Minimalist Perspective on Caluse Chains, Converbs and Asymmetric Coordination*: University of Leipzig dissertation. [p. 291]
- WELDLE, HELMUT, LARS KONIECZNY, DANIEL MÜLLER, SASCHA WOLFER, & PETER BAUMANN. 2009. Connectionist modeling of situated language processing: Language and meaning acquisition from an embodiment perspective. In *31st Annual Conference of the Cognitive Science Society*, 827–832. [p. 333]
- WEXLER, KEN, & PETER CULICOVER. 1980. *Formal Principles of Language Acquisition*. Cambridge, MA: MIT Press. [p. 253]
- WHITE, AARON STEVEN, & KYLE RAWLINS. 2018. The role of veridicality and factivity in clause selection. In *Northeastern Linguistic Society (NELS 48)*, Amherst, MA. [pp. 203 and 282]
- WILLIAMS, ALEXANDER. 2009. Themes, cumulativity, and resultatives: Comments on Kratzer 2003. *Linguistic Inquiry* 40(4). 686–700. [pp. 200 and 215]
- WILLIAMS, ALEXANDER. 2015. *Arguments in Syntax and Semantics*. Cambridge, UK: Cambridge University Press. [pp. 199 and 200]

- WOLK, CHRISTOPH, SASCHA WOLFER, PETER BAUMANN, BARBARA HEMFORTH, & LARS KONIECZNY. 2011. Acquiring English dative verbs: Proficiency effects in German L2 learners. In *33rd Annual Conference of the Cognitive Science Society*, 2401–2406. [p. 333]
- YOSHIDA, MASAYA, MICHAEL WALSH DICKEY, & PATRICK STURT. 2013. Predictive processing of syntactic structure: Sluicing and ellipsis in real-time sentence processing. *Language and Cognitive Processes* 28(3). 272–302. [p. 15]
- ZUBER, RYSZARD. 1982. Semantic restrictions on certain complementizers. In *XIIIth International Congress of Linguists*, Tokyo, Japan. [p. 282]

VITA

Education

- | | | |
|------|---|--|
| 2014 | M. A. | Linguistics
Northwestern University, Evanston, IL |
| 2011 | B. A. + M. A.
(<i>Magister Artium</i>) | Portuguese, Cognitive Science, & Anthropology
University of Freiburg, Germany |
| 2009 | B. S. + M. S.
(<i>Diplom</i>) | Physics
University of Freiburg, Germany |

Publications

- HE, BAUMANN, FANG, HUTCHINSON, JAECH, OSTENDORF, FOSLER-LUSSIER & PIERREHUMBERT (2016). Using pronunciation-based morphological subwords to improve OOV handling in keyword search. *IEEE/ACM Transactions*.
- BAUMANN (2015). The special status of color in pragmatic reasoning: evidence from a language game. *37th Conference of the Cognitive Science Society*.
- FANG, OSTENDORF, BAUMANN & PIERREHUMBERT (2015). Exponential language modeling using morphological features and multi-task learning. *IEEE/ACM Transactions*.
- FRAZIER, ACKERMAN, BAUMANN, POTTER & YOSHIDA (2015). Wh-filler-gap dependency formation guides reflexive antecedent search. *Frontiers in Psychology*.
- BAUMANN (2014). Dependencies and hierarchical structure in sentence processing. *36th Conference of the Cognitive Science Society*.
- BAUMANN, CLARK & KAUFMANN (2014). Overspecification and the cost of pragmatic reasoning about referring expressions. *36th Conference of the Cognitive Science Society*.

- BAUMANN & PIERREHUMBERT (2014). Using resource-rich languages to improve morphological analysis of under-resourced languages. *Ninth Conference on Language Resources and Evaluation (LREC'14)*.
- MAIER, KAESHAMMER, BAUMANN & KÜBLER (2014). Discosuite - a parser test suite for German discontinuous structures. *Ninth Conference on Language Resources and Evaluation (LREC'14)*.
- BAUMANN, KONIECZNY & HEMFORTH (2014). Conversational implicatures in anaphora resolution: Alternative constructions and referring expressions. *Psycholinguistic Approaches to Meaning and Understanding across Languages*.
- HE, HUTCHINSON, BAUMANN, OSTENDORF, FOSLER-LUSSIER & PIERREHUMBERT (2014). Subword-based modeling for handling OOV words in keyword spotting. In *2014 IEEE International Conference on Acoustic, Speech and Signal Processing (ICASSP)*.
- BAUMANN (2013). Syntactic category disambiguation within an architecture of human language processing. *35th Conference of the Cognitive Science Society*.
- AUER, BAUMANN & SCHWARZ (2011). Vertical vs. horizontal change in the traditional dialects of southwest Germany: A quantitative approach. *Taal en Tongval*.
- BAUMANN, KONIECZNY & HEMFORTH (2011). Pragmatic expectations and coreference: How alternative constructions and referring expressions can serve as cues. *33rd Conference of the Cognitive Science Society*.
- WOLK, WOLFER, BAUMANN, HEMFORTH & KONIECZNY (2011). Acquiring English dative verbs: Proficiency effects in German L2 learners. *33rd Conference of the Cognitive Science Society*.
- HEMFORTH, KONIECZNY, SCHEEPERS, COLONNA, SCHIMKE, BAUMANN & PYNTE (2010). Language specific preferences in anaphor resolution: Exposure or Gricean maxims. *32nd Conference of the Cognitive Science Society*.
- KONIECZNY, WELDLE, WOLFER, MÜLLER & BAUMANN (2010). Anaphora and local coherences. *32nd Conference of the Cognitive Science Society*.
- WELDLE, KONIECZNY, MÜLLER, WOLFER & BAUMANN (2009). Connectionist modeling of situated language processing: Language and meaning acquisition from an embodiment perspective. *31st Conference of the Cognitive Science Society*.