UNIVERSITY OF CALIFORNIA

Los Angeles

'what' did not move: Sluicing in Minimalist Grammars

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Linguistics

by

Deborah Jia Ming Wong

2020

ABSTRACT OF THE DISSERTATION

'what' did not move: Sluicing in Minimalist Grammars

by

Deborah Jia Ming Wong
Doctor of Philosophy in Linguistics
University of California, Los Angeles, 2020
Professor Timothy Hunter, Chair

Sluicing is a type of ellipsis where only a *wh*-phrase, the remnant, is pronounced while the entire TP node of the embedded interrogative clause is unpronounced (e.g. 'John met someone, but I don't know who'). Traditional analyses of sluicing (Ross 1969, Merchant 2001) involve *wh*-movement of the remnant, followed by the deletion of TP (e.g. 'John met someone yesterday, but I don't know who ~~John met yesterday~~'). The remnant is taken to move overtly in a similar fashion to regular *wh*-words in questions. This dissertation argues against *wh*-movement in sluicing and proposes an alternative analysis that does not involve overt movement of the remnant. The analysis is explored with Malay (Austronesian, Malaysia), which has both movement and no movement for question formation. Evidence from Malay suggests that sluicing uses the no movement option rather than the movement option for sluicing. In the proposed analysis, the remnant does not move, while the surrounding material is elided (e.g. 'John met someone yesterday, but I don't know ~~John met~~ who ~~yesterday~~'). The remnant is argued to have the semantic characteristics of in situ *wh*-phrases and indefinites, both of which do not undergo movement. These characteristics are argued to give rise to the patterns seen in sluicing. This analysis is then implemented within the Minimalist Grammars framework, where the architecture of the grammar allows for the possibility of non-constituent deletion. Movement in Minimalist Grammars is thought to involve multi-component expressions that allows individual components to be held out of the

main derivation. The analysis makes use of this notion of separation between components to implement sluicing.

The dissertation of Deborah Jia Ming Wong is approved.

Dylan Thomas Bumford

Edward L. Keenan

Timothy A. Stowell

Timothy Hunter, Committee Chair

University of California, Los Angeles

2020

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

Writing a dissertation is hard. This journey has undoubtedly been one of the most challenging experiences in my life and I could not have done it without the many people who have contributed, especially my committee members.

I owe my biggest thanks to Tim Hunter for his guidance and support throughout the entire process. Tim has been exceedingly patient and generous with his knowledge, and I have enjoyed all our (often lengthy) meetings, even the COVID-era cross-continental Zoom meetings at 4am. Tim's enthusiasm and dedicated support have never waned, and for that I am truly grateful. So, thanks Tim, and I promise I will email Larry Moss one day.

I thank Ed Keenan for shaping my academic life. When I came to UCLA, I did not quite know what I wanted to do until I took Ed's mathematical linguistics course, which somehow also led to fieldwork in Borneo. I am very lucky to have found such a good teacher and mentor. Also, thanks for making me your RA all these years and for all the stories about Madagascar; I will forever remember the one about the Norwegian baby.

This dissertation has greatly benefited from Dylan Bumford's meticulous eye for details. Thanks for seeing what I could not see in formalisms. Meetings with Tim Stowell were helpful and his feedback on this dissertation definitely stimulated new thoughts. Beyond my committee, I thank my first syntax teacher, Klaus Abels, who started me on this sluicing journey as an undergrad.

Richard Stockwell gets a special mention for being the best collaborator and PhrienD one could ask for (*where is my bottle of whiskey?*). Maddy Booth deserves special thanks for being such a great lockdown housemate, introducing me to sherry, and starting Bar Bentlé at #Wongbooth with me. Thanks Meng Yang for being my photographer and chauffeur all these years. Thanks to Tania Ling for being the best writing buddy in the depths of the night. I also thank Yun Lu and Ashwini Nene for helping me with formalisms during the early stages of this dissertation. A big thanks goes out to Angelica Pan for combing through and fixing this entire dissertation because I have never learnt how to do English agreement properly.

I thank my family back home in Malaysia for their constant love and support throughout all the years I have been abroad. This dissertation is dedicated to them.

Finally, thanks to all my friends (academic and non-academic) who have taught me, studied with me, or enjoyed life with me. This year has been quite the ride and I hope we will all be having fun in person again soon.

VITA

2009–2012	BA, Linguistics , University College London
2012–2013	MPhil, Theoretical and Applied Linguistics, University of Cambridge, Newnham College
2014–2020	PhD candidate, Linguistics, University of California, Los Angeles

PUBLICATIONS

Stockwell, Richard & Deborah J.M. Wong. in press. Sprouting and the structure of *except*-phrases. *Proceedings of the 50th Annual Meeting of the North East Linguistic Society.*

Yang, Meng & Deborah J.M. Wong. 2020. Malay verbal reduplication with the *məN*-prefix. In H. Y. Chang, H.J. Huang (eds.) *Papers from the Austronesian Formal Linguistics Association 25.* Journal of the Southeat Asian Linguistics Society.

Wong, Deborah J.M.. 2017. Negative Polarity Items In Malay: An Exhaustification Account. *Proceedings of GLOW in Asia XI.*

Wong, Deborah J.M. & Iara Mantenuto. 2017. Quantifiers in Kenyah Uma Baha. In D. Paperno, E.L. Keenan (eds.) *Handbook of Quantifiers in Natural Language: Volume II, Studies in Linguistics and Philosophy 97.*

CHAPTER 1

Introduction

This dissertation is about something, and after reading it, you will know what. 'Sluicing', as coined by Ross (1969), is a widely attested type of ellipsis where only a *wh*-phrase is pronounced while the entire TP node of the embedded interrogative clause is elided. (1) exemplifies sluicing in English.

(1) John ate something, but I don't know what.

The terminology that will be used in this dissertation is as follows: the *wh*-phrase at the end of a sluiced sentence, *what* in (1), is referred to as the *wh*-remnant; the clause which carries the meaning of the elliptical question, *John ate something*, is the antecedent; the indefinite, *something*, is the correlate.

The chapter is organized as follows: In the next section, I state the goal of this dissertation. I first discuss Merchant's (2001) *wh*-movement analysis of sluicing and argue against it. I also give a sketch of the analysis that I am pursuing. In the second section, I provide the structure of this dissertation.

1.1 Goal of this Dissertation

My goal in this dissertation is to investigate the idea of *wh*-movement in traditional analyses of sluicing. Traditional analyses of sluicing involve movement of the *wh*-remnant to the complement of the verb that takes the TP clause, followed by the deletion of the TP clause (Ross 1969, Merchant 2001). This is illustrated in (2).

(2) John ate something, but I don't know what$_1$ ~~[John ate t_1]~~.

The *wh*-remnant, *what*, in (2) is taken to move overtly in a similar fashion to regular *wh*-words in a question sentence. The ellipsis operation is then licensed under some condition (either semantic, syntactic or both) that requires identity with the antecedent. However, based on facts presented in Chapter 2, this dissertation argues against the *wh*-movement analysis and provides an alternative no movement analysis illustrated in (3).

(3) John ate something yesterday, but I don't know ~~John ate~~ what ~~yesterday~~.

In this analysis, the *wh*-remnant, *what*, does not undergo *wh*-movement while its surrounding material is deleted. The following subsection discusses Merchant's (2001) movement analysis and the subsequent subsection gives a sketch of the proposal in this dissertation.

1.1.1 Merchant (2001): Sluicing with *wh*-movement plus deletion

Merchant (2001) argues that sluicing is derived by *wh*-movement of the *wh*-phrase from within the full clausal structure to its scopal position, followed by deletion of the TP it was extracted from. This account is illustrated in (2). The *wh*-remnant is taken to have moved to the Spec CP position in the same manner and for the same reason as *wh*-words in non-elliptical *wh*-questions.

The greatest support for this movement account stems from 'connectivity effects' where the *wh*-remnant shows similar morphosyntactic constraints to its correlate in the antecedent. These connectivity effects include morphological case matching effects as illustrated for German in (4) and preposition-stranding parallelisms (5 and 6).

(4)

Er	will	jemandem	schmeicheln,	aber	sie	wissen	nicht,	wem/	*wen
he	wants	someone.DAT	flatter	but	3PL	know	NEG	who.DAT/	*who.ACC

'He wants to flatter someone, but they don't know who' (Ross 1969)

(5) English

a. John talked to someone, but I don't know (to) who.

b. Who was John talking with?

(6) German

a. Anna hat mit jemandem gesprochen, aber ich weiß nicht *(mit) wem.
Anna has with someone spoken, but I know not with who
'Anna has spoken with someone, but I don't know with who'

b. *Wem hat sie mit gesprochen?
who has she with spoken
'Who has she spoken with?' (Merchant 2001)

However, this movement analysis runs into problems when the *wh*-remnant breaks the morphosyntactic constraints imposed by the surrounding material of the correlate in the antecedent. These cases of "disconnectivity" between the *wh*-remnant and antecedent include island insensitivity and languages that violate the preposition-stranding generalization. Given the movement of the *wh*-remnant under Merchant's analysis, island sensitivity would prove without a doubt the analysis to be right. However, the fact is that we do not see island effects under sluicing, shown in (7).

(7) They want to hire [someone who speaks a Balkan language], guess which.

Hence, to maintain the movement analysis, there needs to be some kind of repair strategy, usually by proposing a different source in the ellipsis site (Merchant 2001, Barros et al. 2014). Merchant divides syntactic islands into PF islands and "propositional islands", and has two different solutions regarding the island sensitivity. For PF islands, the ungrammaticality is not due to the derivation of the structure but due to an illicit phonological representation instead. Since there is no phonological material under ellipsis, sluicing does not incur ungrammaticality as the island itself is not pronounced. For "propositional islands" such as complex NP and adjunct islands, Merchant proposes a semantic identity condition to license the ellipsis. This condition allows for structural mismatches between ellipsis site and antecedent, provided that their respective structures encode the same meaning. Following Romero (1998) in applying

Schwarzschild (1999)'s theory of focus to sluicing, deletion in Merchant's approach is only possible when the elided constituent is *E-given*:

(8) An expression E is *E-given* iff E has a salient antecedent A and, modulo ∃-type shifting,

i A entails the focus-closure of E, and

ii E entails the focus-closure of A

(8) says that sluicing is permitted if there is mutual entailment between the non-focused part of the antecedent and the ellipsis site. Since the elided portion does not have to be identical to its antecedent in this analysis, the derivation of *wh*-remnant to scopal position is allowed to use E-type pronouns as shown in (9).

(9) a. They want to hire someone who speaks a Balkan language, guess which.

b. They want to hire someone who speaks a Balkan language, guess which ~~[she should speak]~~.

c. *They want to hire someone who speaks a Balkan language, guess which ~~[they want to hire someone who speaks]~~.

Merchant argues that a sentence like (9a) is derived without the *wh*-remnant crossing a syntactic island because the source sentence for sluicing has an E-type pronoun (she) as in (9b).

1.1.2 Proposal: No Movement of the Remnant

As seen in the section before, to explain island insensitivity, Merchant has to come up with a complicated story to keep the *wh*-movement analysis. Based on facts presented in Chapter 2, this dissertation argues against the traditional *wh*-movement analysis and provides an alternative no movement analysis, shown in (3) repeated below.

(3) John ate something yesterday, but I don't know ~~John ate~~ what ~~yesterday~~.

The analysis of sluicing is built within the Minimalist Grammars (MG) framework where the architecture of the grammar allows for the possibility of non-constituent deletion once we look at the way movement is executed in the framework. The rough idea is that movement in MG is thought to involve multi-component expressions that allows individual components to be held out. The analysis makes use of this notion of separation between the components to implement sluicing. In this analysis, *wh*-remnants are argued to have characteristics of indefinites and *wh*-phrases in situ, both of which do not have overt syntactic movement. These characteristics are argued to give rise to the patterns seen in sluicing. Although the *wh*-remnant is argued to not move, it is still required to take scope over its elided material. Its ability to take scope is argued to be similar to the way in which its correlate (an indefinite) takes scope, rather than the way in which moving *wh*-phrases take scope.

The general idea of the analysis is sketched out as follows. In the semantics literature, quantifiers in object position, such as the indefinite *someone* in (10), create a syntax-semantics mistmatch.

(10) a. John [$_{VP}$ saw someone]

b. $[\![someone]\!] = \lambda Q.\exists x[\mathbf{person}(x) \wedge Qx]$

c. $[\![John\ saw\ x]\!] = \lambda y[\mathbf{see}(\mathbf{j},y)]$

d. $[\![someone]\!](\lambda y[\mathbf{see}(\mathbf{j},y)]) = \exists x[\mathbf{person}(x) \wedge \mathbf{see}(\mathbf{j},x)]$

The meaning of the sentence in (10a) requires evaluating 'John saw x' for various different values of the variable x, which means that *someone* needs to take propositional scope as seen in (10d). This clashes with the fact that the sister of *someone* is just the verb *saw* while the propositional scope of *someone* is not a constituent. Various solutions to this problem have been proposed in the literature (e.g. type-shifting, quantifier raising, continuations), all of which must somehow allow the indefinite in object position to take propositional scope in a position other than its surface syntactic position. The claim that I am making is that the *wh*-remnant in a sluiced sentence makes use of the same semantic ability as the correlate in its antecedent clause to take scope. In other words, the *wh*-remnant is like a *wh*-word in situ but is able to take scope in a position other than its surface syntactic position, in the same way that the indefinite in (10d) takes propositional scope. This view results in the elided material not being a constituent, but a *wh*-word taking appropriate scope.

As we will see in this dissertation, sluicing shows a particular mixture of similarities and differences from corresponding sentences with non-elided embedded questions. The three main "form-identity" effects concern syntactic islands, preposition-stranding and case-matching. Sluicing exhibits syntactic island insensitivity, unlike non-elided embedded questions, but is usually thought to share the same effect of preposition-stranding and case-matching as non-elided embedded questions. The no *wh*-movement approach is a way to capture this difference in "form-identity". While the *wh*-in situ approach for sluicing has been proposed in the literature (see Kimura 2010, Abe & Hornstein 2012, Abe 2015), my implementation of the idea is different. The implementation of my analysis shares the idea of no *wh*-movement for sluicing like previous approaches. However, it differs from the point of view of the *wh*-remnant staying completely in situ. The main data used to argue against movement in this dissertation is from Malay, while English will be used in certain cases for readability. But the analysis is intended to be extended for sluicing in all languages, as some of the arguments used for Malay, as discussed in Chapter 2, are seen cross-linguistically as well.

To reiterate, the main claims that I am making are:

(11) Main Claims:

a. the *wh*-remnant in sluicing does not undergo *wh*-movement

b. the ability of a *wh*-remnant to take scope is similar to the way indefinites and *wh*-phrases in situ take scope

1.2 Structure of Dissertation

Chapter 2 presents evidence for having a no *wh*-movement sluicing analysis from Malay. Malay is unusual in that it has both *wh*-movement and *wh*-in situ for question formation. With sluicing, evidence points to using the no movement option rather than *wh*-movement.

Chapter 3 introduces the Minimalist Grammars formalism, defining the syntax and semantics of the MG that will allow for sluicing with no *wh*-movement. The order of syntactic and semantic operations in this framework is represented as a derivation tree, which will be the main structure utilized to illustrate the workings of the no movement analysis of sluicing. The syntax in this system uses Stabler (1996)'s original *merge* and *move* operations. The semantic interpretation is based on a compositional semantics over derivation trees by Kobele (2006, 2012). The main discussion in this chapter involves Kobele's semantic computation of Quantified Noun Phrases (QNP) (existential and universal quantification) which involves storage and retrieval of the meaning of QNP.

Chapter 4 deals with the identity condition of the sluicing analysis I am arguing for. The conditions of ellipsis that are established here will ultimately dictate how the Elide function works. With the no *wh*-movement analysis, I argue for a hybrid account of identity with the following condition that is set within the MG framework.

(12) Condition of Hybrid Identity in MG:
The Derivational Isomorphism Condition:
The elided material and the antecedent must have the same derivation, including all lexical entries, modulo the correlate and *wh*-remnant.

Chapter 5 shows the full implementation of sluicing in MG. The *wh*-remnant is given the same semantics as indefinites, which come by their scoping capabilities by being denoted as sets of alternatives, and from being composed pointwise-functionally until a closure operator is reached. The *wh*-remnant is given the ⊛F feature, which allows it to be held out (to allow for ellipsis) but also placed in a non-c-commanding position without violating any island constraints. Then I show how the ellipsis function, Elide, works with the *wh*-remnant.

Chapter 6 shows how the implementation in Chapter 5 accounts for the Malay data patterns seen in Chapter 2.

Chapter 7 provides a conclusion for this dissertation.

CHAPTER 2

Sluicing in Malay

This chapter provides evidence for the no *wh*-movement analysis for sluicing from Standard Malay. Malay serves as a good language to test for a no *wh*-movement analysis as it exhibits both movement and no movement options for *wh*-phrases when forming questions (Cole & Hermon 1998). This is shown in (13).

(13) a. **apa** yang Ali harap Fatimah akan beli untuk.nya? (*wh*-movement)
what COMP Ali hope Fatimah will buy for.3Sg

b. Ali harap Fatimah akan beli **apa** untuk.nya? (no *wh*-movement)
Ali hope Fatimah will buy what for.3Sg
'What does Ali hope that Fatimah will buy for him?'

Both question sentences in (13) are grammatical and are equivalent in meaning. Since Malay has both options of movement and no movement for *wh*-phrases, I argue that Malay uses the no movement option under sluicing. The rough idea for no *wh*-movement under sluicing is illustrated in (14) .

(14) Ali membeli sesuatu semalam, tapi saya tak tahu ~~Ali membeli~~ **apa** ~~semalam~~
Ali bought something yesterday, but I NEG know ~~Ali bought~~ what ~~yesterday~~
'Ali bought something yesterday, but I don't know what.'

The unelided version of (14), shown in (15), is grammatical in Malay.

(15) Ali membeli sesuatu semalam, tapi saya tak tahu Ali membeli **apa** semalam
Ali bought something yesterday, but I NEG know Ali bought what yesterday

'Ali bought something yesterday, but I don't know what Ali bought.'

In the following section, I provide different evidence for the no *wh*-movement proposal. Then, I discuss Sato's (2016) work on Indonesian sluicing. Indonesian is a closely related language to Malay and Sato (2016) has also argued for a *wh*-in situ analysis for sluicing in Indonesian. While I agree with the facts that Indonesian sluicing provides evidence of no movement of the *wh*-phrase, I do not adopt his in-situ implementation for Malay as I intend to implement my analysis within the MG framework, which will be introduced in the following chapter.

2.1 Evidence for no *wh*-movement in Malay Sluicing

Data from the following phenomenon are used to argue for no *wh*-movement in Malay sluicing: scopal effects, island insensitivity, preposition-stranding, the '*meN*'-mismatch, and the sluicing-COMP generalization.

2.1.1 Scope Effects under Sluicing

Different scope patterns are observed in a *wh*-moved question sentence compared to sluicing. In Malay *wh*-questions that contain quantifiers as subjects, different responses are elicited depending on where the *wh*-phrase surfaces.

(16) a. semua pelajar itu membeli **apa**?
all student the buy what
(Readings: $\forall > wh$, $wh > \forall$)

b. **apa** yang semua pelajar itu beli?
what COMP all student the buy
(Readings: $\forall > wh^{\star}$, $wh > \forall$)

'What did every student buy?'

In (16a), the *wh*-phrase remains in its base position and the universal quantifier can take wide or narrow scope relative to the *wh*-phrase. Under the wide scope universal interpretation, a distributive answer can be obtained. Thus, in a model that contains three students {John, Mary, Bill}, a licit answer to (16a) could be 'John bought a book, Mary bought a pencil, Bill bought a ruler'. Under the narrow universal scope interpretation, a collective answer such as 'every student bought a book' is licit. However, in (16b), where the *wh*-phrase is moved, only a collective answer can be obtained. This indicates that when the *wh*-phrase undergoes overt syntactic movement, the universal quantifier falls under the scope of the *wh*-question and is thus only able to take narrow scope. The pattern observed here with questions is consistent with observations regarding the scope of quantification in Malay. Malay exhibits scope rigidity, where if one quantifier c-commands another, the first quantifier will outscope the second. This is illustrated in (17).

(17) semua pelajar itu membaca sebuah buku
all student the read CL book
'Every student read a book'

Readings:

i $\forall x[\text{student}(x) \rightarrow \exists y[\ \text{book}(y) \wedge \text{ read } (y)(x)]]$

ii $*\exists y[\text{book}(y) \wedge \forall x[\ \text{student}(x) \rightarrow \text{read } (y)(x)]]$

(18) seorang pelajar membaca semua buku
a student read every book
'A student read every book'

Readings:

i $*\forall x[\text{book}(x) \rightarrow \exists y[\ \text{student}(y) \wedge \text{ read } (x)(y)]]$

ii $\exists y[\text{student}(y) \wedge \forall x[\ \text{book}(x) \rightarrow \text{read } (x)(y)]]$

In (17), the indefinite is c-commanded by the universal quantifier and is interpreted under the scope of the universal quantifier. The inverse reading in this case is not allowed. Similarly in (18), when the indefinite c-commands the universal quantifier, only the wide scope reading of the existential quantifier is obtained.

Interestingly, under sluicing in Malay, two readings are obtained with the *wh*-remnant.

(19) semua pelajar itu membaca sebuah buku, tapi saya tak tahu buku yang mana
all student the read CL book, but 1.Sg NEG know book COMP which

'Every student read a book, but I don't know which book.'

The sentence in (19) is ambiguous. The indefinite correlate *sebuah buku* (a book) can be interpreted with a wide or narrow scope with respect to the universally-quantified DP *semua pelajar* (every student). Notice that the available readings of (19) are the same as those for (16a) which does not exhibit *wh*-movement and not (16b) with *wh*-movement. Hence, this fact from Malay lends support to the no *wh*-movement approach to sluicing rather than a *wh*-movement approach.

2.1.2 Island Insensitivity in Malay

Island insensitivity is a well-known property of sluicing and is observed cross-linguistically, even in languages where sluicing is typically assumed to be a result of a *wh*-movement operation. This hypothesized *wh*-movement operation in sluicing sentences shares properties with regular *wh*-movement in questions, except that it appears to be insensitive to syntactic islands. Thus, *wh*-movement-based analyses would have to explain this apparent island insensitivity under sluicing (e.g. repair, semantic identity etc.). On the other hand, a no-movement approach to sluicing explains the island-insensitivity straightforwardly since no *wh*-movement has occurred.

Given that Malay has a no movement option for forming question, island insensitivity for Malay can be straightforwardly accounted for. Malay questions with no *wh*-movement do not cause ungrammaticality as the *wh*-phrase stays within syntactic islands. On the other hand, questions formed with *wh*-movement do obey syntactic islands as shown in (20).

(20) *wh*-questions and Adjunct Islands

a. *Apa yang Ali dipecat [kerana dia beli t_i]?
what COMP Ali fired [because he bought t_i]

b. Ali dipecat [kerana dia beli apa]?
Ali fired [because he bought what]
'Ali was fired because he bought what?' (Cole & Hermon 1998)

The adjunct island in (20) is indicated in [ ] brackets. In (20a), movement of the *wh*-phrase *apa* out of the adjunct island causes ungrammaticality. While in (20b), there is no ungrammaticality as the *wh*-phrase *apa* remains in its base position. (21) shows the sluicing version of (20), where there is no island violation.

(21) Ali dipecat [kerana dia beli sesuatu], tapi saya tak tahu apa?
Ali fired [because he bought something], but 1.Sg NEG know what
'Ali was fired because he bought something, but I don't know what.'

(22) shows sluicing with an antecedent containing a Complex NP island with no island violation.

(22) Ali bertemu dengan [perempuan yang membeli sesuatu], tapi saya tak tahu
Ali met with woman COMP bought something but 1.Sg NEG know
apa
what
'Ali met a woman who bought something, but I don't know what.'

Even though the antecedent in (22) contains a Complex NP island, the sentence remains grammatical. Since Malay allows *wh*-phrases to remain in base position, pronouncing the ellipsis site when the *wh*-remnant does not move (23b) allows the sentence to remain grammatical. The *wh*-movement option in (23a) triggers ungrammaticality.

(23) a. *Ali bertemu dengan [perempuan yang membeli sesuatu], tapi saya tak
Ali met with woman COMP buy something but 1.Sg Neg
tahu [apa$_i$ yang Ali bertemu dengan perempuan yang membeli t_i]
know what COMP Ali met with woman COMP buy

b. Ali bertemu dengan [perempuan yang membeli sesuatu], tapi saya tak
Ali met with woman COMP buy something but 1.Sg Neg
tahu Ali bertemu dengan [perempuan yang membeli apa]
know Ali met with woman COMP buy what
'Ali met a woman who bought something, but I don't know Ali met a woman with woman who bought what'

As shown here, island insensitivity falls out of the no *wh*-movement approach and explains the Malay data without resorting to any additional mechanism.

2.1.3 P(reposition)-stranding in Malay Sluicing

The second "form-identity" effect that supports a no *wh*-movement approach comes from the p-stranding parallelism. Malay appears to be a counterexample to Merchant's (2001) preposition-stranding generalization.

(24) Preposition-stranding generalization (PSG):
A language L will allow preposition stranding under sluicing iff L allows preposition stranding under regular *wh*-movement.

Malay does not allow p-stranding under regular *wh*-movement as shown in (25), but allows for p-stranding under sluicing as shown in (26).

(25) a. *siapa yang Ali bagi buku kepada?
who COMP Ali give book to

b. kepada siapa yang Ali bagi buku?
to who COMP Ali give book

c. Ali bagi buku kepada siapa?
Ali give book to who
'Who did Ali give the book to?'

(26) Ali bagi buku kepada seseorang, tapi saya tak tahu (kepada) siapa?
Ali give book to someone, but 1.Sg Neg know (to) who
'Ali gave a book to someone, but I dont know (to) who.'

Without ellipsis, the pronounced p-stranding form of (26) with *wh*-movement is ungrammatical as shown in (27a), while the no *wh*-movement form with no p-stranding is grammatical as in (27b).

(27) a. *Ali bagi buku kepada seseorang, tapi saya tak tahu siapa Ali bagi buku kepada
Ali give book to someone, but 1.Sg Neg know who Ali give book to

b. Ali bagi buku kepada seseorang, tapi saya tak tahu Ali bagi buku kepada siapa
Ali give book to someone, but 1.Sg Neg know Ali give book to who

'Ali gave a book to someone, but I dont know who Ali gave the book to.'

Under a *wh*-movement approach, the violation of the PSG in Malay sluicing would be difficult to explain. However, with with a no wh-movement approach to sluicing, the PSG is not expected to hold, so the violation is not a surprise.

Before moving on to the next evidence, I address the possibility that p-less sluices in Malay may not be a result of sluicing operations at all. Besides Malay, there are other languages that have been found to appear to violate the PSG, Brazilian Portuguese (Almeida & Yoshida 2007, Rodrigues et al. 2009), Spanish (Rodrigues et al. 2009), Serbo-Croatian (Stjepanović 2008), Indonesian (Sato 2011), and Emirati Arabic (Leung 2014). However, in some of these cases, the violation of the PSG is argued to be superficial as p-less sluices in these languages are shown to not be a result of sluicing operations. In Brazilian Portuguese (BP) and Spanish, Rodrigues et al. (2009) argue that p-less sluices have a different source, that is, the ellipsis site is not syntactically isomorphic to the antecedent. They argue that the p-less sluices in BP have cleft sources, therefore BP does not actually violate the PSG. They use the multiple sluicing diagnostic to show that p-stranding with *wh*-movement in BP cannot occur under sluicing in the language. The multiple sluicing diagnostic has been successfully applied to numerous seemingly PSG-violating languages.

In the multiple sluicing diagnostic, only the first *wh*-phrase gives an indication whether a real sluicing operation has taken place[1]. Following Lasnik (2006), Rodrigues et al. (2009) argue that the first *wh*-phrase in a multiple sluice can only be derived through a real sluicing operation[2], and thus cannot have other sources such as clefts. Multiple sluicing rules out cleft sources because the pivots of clefts cannot accommodate multiple constituents. If a language does not allow p-stranding in regular *wh*-questions, it does not allow p-stranding with the first *wh*-phrase of a multiple sluice either, as the first *wh*-phrase must have the same source like the antecedent and not clefts. Rodrigues et al. (2009) show that this is true in Spanish and Brazilian Portuguese. Both of these languages disallow p-stranding in the first and second *wh*-phrase under multiple sluicing as seen below.

(28) a. Ella habló con alguien sobre algo, pero no se *(con) quién
she talked with someone about something but not know *(with) who
*(sobre) qué
*(about) what
'She talked with someone about something, but I dont know who about what.'

b. Ela falou sobre alguma coisa para alguém, mas eu não sei *(sobre) o
she talked about some thing to someone but I not know *(about) the
que *(para) quem
what *(to) who
'She talked about something to someone but I dont know about what to who.'

(Rodrigues et al. 2009)

---

[1]Under multiple sluicing, English prohibits the omission of the preposition in the second *wh*-phrase. According to Lasnik (2006), the first *wh*-phrase under multiple sluicing is derived through regular *wh*-movement and deletion while the second wh-phrase is derived through rightward extraposition, as shown in (1).

(1) Peter talked about something to somebody, but I cannot remember [$_{CP}$ (about) what$_1$ ~~Peter talked [(about) t_1]~~ [t_2]][*(to) whom$_2$]]

Since the first *wh*-phrase is derived through regular *wh*-movement, p-stranding is allowed. The ban on p-stranding on the second *wh*-phrase is due to constraints on rightward movement, which disallows p-stranding in general as seen in (2) (Ross 1967, Lasnik 2006).

(2) *Peter talked [$_{pp}$ about t_1] yesterday [a paper in sluicing]$_1$.

[2]For Rodrigues et al. (2009), the sluicing operation is Merchant's *wh*-movement plus deletion.

They conclude that p-less sluices in Spanish and Brazilian Portuguese are due to cleft sources, thus these language do not violate the PSG.

Here, I use the same multiple sluicing diagnostic on Malay p-less sluices. If Malay has a cleft source for p-less construction, then this diagnostic would rule out the possibility of p-stranding under multiple sluicing.

(29) Ali berbincang tentang sesuatu dengan seseorang, tapi saya tak tahu
Ali discuss about something with someone but I NEG know
(tentang) apa *(dengan) siapa.
(about) what (with) who
'Ali discussed about something with someone but I don't know (about) what *(with) who.'

Unlike Spanish and Brazilian Portuguese, a p-less construction is allowed in the first *wh*-phrase as seen in (29)[3]. The grammaticality of a p-less construction under multiple sluicing in (29) constitutes a strong argument against a cleft source for p-less sluiced construction in Malay. Given that Malay has an alternative way of forming *wh*-questions, allowing the *wh*-word to stay in base position with the preposition deleted along with other surrounding material maintains form identity[4]. The way Malay handles p-stranding under sluicing will be further discussed in Chapter 6.

---

[3] In (29), the preposition in the second *wh*-phrase cannot be stranded. To explain this, I adopt Lasnik's (2006) rightward extraposition analysis for the second *wh*-phrase in multiple sluicing. Like English, Malay appears to respect the ban on p-stranding in rightward movement as shown in (1).

(1) *Ali berbincang [$_{pp}$tentang t_1] semalam [buku yang ditulis oleh Samad Said]$_1$.
Ali discuss about t yesterday book that written by Samad Said
'*Ali discuss about yesterday a book by Samad Said.'

[4] See also Stigliano (2019) for a *wh*-in situ approach for p-less sluicing in Spanish.

2.1.4 '*meN*'-mismatch

The next "form-identity" effect is the *meN*- mismatch between antecedent and ellipsis site. The verbal prefix *meN*- is one of the most studied affixes in the Malay/Indonesian literature. Yet, its grammatical function remains widely disputed. The various objects that *meN*- has been analyzed as includes an active voice marker (Cole et al. 2008, Aldridge 2008, Nomoto 2008, 2011; Sato 2012), a transitive marker (Chung 1976, Cole & Hermon 1998), an objective case marker (Guilfoyle et al. 1992), an antipassive marker (Fortin 2006), among many others[5]. I do not intend to join the argument of the function of *meN*- in this paper but intend to use *meN*- to show that there is a mismatch between the antecedent and ellipsis site under a *wh*-movement analysis of sluicing.

Despite, the disagreement on the grammatical function of *meN*-, there is agreement on its syntactic effect, namely that it blocks A'-movement of DPs across it (Saddy 1991, Soh 1998, Cole & Hermon 1998, Nomoto 2013). This ban on DP movement is observed in *wh*-questions that show a subject-object asymmetry with respect to the presence of *meN*-. When *wh*-phrases correspond to subjects, *meN*- is optional on the verb as illustrated in (30).

(30) a. siapa beli buku itu?
who buy book that

b. siapa mem-beli buku itu
who *meN*-buy book that
'Who bought the book?'

However, when *wh*-phrases correspond to objects, *meN*- is strictly not allowed on the verb as illustrated in (31).

(31) a. apa Ali beli?
what Ali buy

[5] See Soh & Nomoto (2011) for a complete list of analysis of *meN*-.

b. *apa ali mem-beli?
what Ali *meN*-buy
'What did Ali buy?'

The contrast between (30b) and (31b) is usually described as *meN*- blocking A'-movement across it. Subject, as in (30), are unaffected as the movement path of the *wh*-phrase do not go across the verbs with *meN*-. The ban on DP movement across *meN*- is further supported with *wh*-in situ where *meN*- is optional as is the case with subjects[6].

(32) a. Ali beli apa?
Ali buy what

b. Ali mem-beli apa?
Ali *meN*-buy what
'What did Ali buy?'

In Malay sluicing, *meN*- can optionally appear in the antecedent.

(33) Ali (mem)beli sesuatu, tapi saya tak tahu apa
Ali *meN*-buy something, but I NEG know what
'Ali is buying something, but I don't know what'

Under a *wh*-movement plus deletion approach, there is a form mismatch between the antecedent and ellipsis site as exemplified when the ellipsis site is pronounced.

(34) Ali **mem-**beli sesuatu, tapi saya tak tahu apa Ali (***mem-**)beli
Ali *meN*-buy something, but I NEG know what Ali *meN*-buy
'Ali is buying something, but I don't know what Ali is buying'

The impossibility of voice mismatches in sluicing, shown in (35), has led researchers to propose that some syntactic identity is needed under a movement-deletion approach (Merchant 2013).

(35) *someone kicked John, but I don't know by who.

[6]Saddy (1991), Soh (1998) also show that neither subject nor object *wh*-phrases can be extracted when matrix verb has *meN*- attached.

The example in (34) looks like a counterexample to voice-mismatch generalizations. Besides Malay, Chamorro, another Austronesian language, is also observed to allow for antipassive voice mismatch under sluicing (Chung 2013). Chung (2013) shows that the oblique, indicated in [ ] in (36a), is associated with an antipassive verb and cannot undergo *wh*-movement. However, an antipassive clause can serve as the antecedent for sluicing, as in (36b).

(36) a. *[Håfa na klåsi-n månnuk] mam-omoksai gu'?
what? L sort-L chicken AGR.AP-raise.PROG he
'What sort of chickens is he raising?'

b. Mam-omoksai månnuk, lao ti ta tungu' [håfa na klåsi_
AGR.AP-raise.PROG chicken but not AGR know what? L sort
'He is raising chickens, but we dont know what kind.' (Chung 2013)

The possibility of these type mismatch has pushed Chung (2013) to propose a hybrid identity condition which consists of semantic and (limited) syntactic identity.

(37) Limited syntactic identity in sluicing in Chung (2013):

a. Argument structure condition: If the interrogative phrase is the argument of a predicate in the ellipsis site, that predicate must have an argument structure identical to that of the corresponding predicate in the antecedent clause.

b. Case condition: If the interrogative phrase is a DP, it must be Case-licensed in the ellipsis site by a head identical to the corresponding head in the antecedent clause.

I would like to point out again that these type mismatches are a result of having *wh*-movement in the analysis of sluicing. The line of argument here is similar to island insensitivity, where the mismatch is also a result of having *wh*-movement. Under a no *wh*-movement approach, the specifics of (37) fall out straightforwardly and the size of syntactic isomorphism of the ellipsis site does not need to be speculated.

(38) Ali (**mem-**)beli sesuatu, tapi saya tak tahu ~~Ali (**mem-**)beli~~ apa
Ali *meN*-buy something, but I NEG know ~~Ali *meN*-buy~~ what
'Ali is buying something, but I don't know ~~Ali is buying~~ what'

2.1.4.1 Sluicing COMP Generalization

In sluicing, Merchant (2001) argues that the syntactic feature [E] on the interrogative C head hosts all the syntactic, semantic, and phonological properties. This property of the C head distinguishes elliptical constrictions from non-elliptical ones. [E] is argued to have strong and uninterpretable [uwh, uQ] features that require overt checking on the C head. This ensures that sluicing only targets the TP complement of a null C and that only the *wh*-remnant appears in Spec CP. Thus, any non-operator material in C that usually appears in question sentences disappears under sluicing. This is known as the sluicing-COMP generalization.

(39) 'Sluicing COMP generalization (Merchant 2001 p.62)
In sluicing, no non-operator material may appear in COMP[7]

The generalization holds true in Germanic languages as shown in (40):

(40) Examples from Merchant (2001):

a. English
A: Max has invited someone B: Really?! Who (*has)?

b. Norwegian
noen snakker med Marit, men vi vet ikke hvem (*som)
someone talks with Marit but we know no who C
'Someone is talking to Marit, but we dont know who.'

c. Dutch
Hij heeft iemand gezien, maar ik weet niet wie (*of/ *dat/ *of dat)
he has someone seen but I know not who (*if/ *that/ *if that)

[7]Though Merchant (2001) refers to the position as COMP, what he intended was perhaps nothing other than the *wh*-remnant can remain in that position.

'He has seen someone, but I dont know who.'

Malay has a question/focus marker -*kah* that only appears on fronted *wh*-phrases and not on *wh*-phrases in situ (Fortin 2007, Eng 2008).

(41) a. apa(-*kah*) yang Ali beli?
what-(-*kah*) COMP Ali buy

b. Ali beli apa(*-*kah*)?
Ali buy what-(-*kah*)
'What did Ali buy?'

Under sluicing, -*kah* cannot appear after the *wh*-remnant.

(42) Ali beli sesuatu, tapi saya tak tahu apa-(*-*kah*)
Ali buy something, but I NEG know what-(-*kah*)
'Ali bought something, but I don't know what.

On first glance, the prohibition of -*kah* under sluicing appears to be another manifestation of the restriction on phonologically overt material appearing in C, thus obeying the sluicing-COMP generalization. Alternatively, the prohibition of -*kah* can also be argued to be the result of the *wh*-phrase staying in base position as shown in (41b).

Under a *wh*-movement story for sluicing, the sluicing-COMP generalization remains a puzzle as there is no reason as to why only the *wh*-remnant in Spec CP remains while everything else below, including the head of the phrase, cannot be pronounced. Merchant's argument for TP deletion does not reveal anything for the sluicing-COMP generalization either, as there is no reason why the C head cannot be pronounced. A possible account under a *wh*-movement analysis is to consider the deletion of the C head under the ellipsis operation. This has been proposed by Bruening (2015) who argues that sluicing is not just TP ellipsis, but is obtained by deleting all but the most prominent syntactic sub-constituent of the CP.

Under a no *wh*-movement account however, the sluicing-COMP generalization puzzle can be accounted for if the deleted material includes the entire CP phrase. Hence, nothing in the C head remains, even in the case where items in C show up in non-movement and non-deletion languages. With a no-movement account, the main issue is dealing with deleting a non-constituent, which sounds controversial in traditional syntax. However, Bruening's proposal of deleting half a phrase would be as controversial as deletion of a non-constituent. Furthermore, opting for a no-movement analysis would eliminate a sluicing stipulation, as there would be no need for the sluicing-COMP generalization.

2.2 Sato (2016): *wh*-in situ Slucing in Indonesian

In this section, I present a *wh*-in situ analysis of sluicing that is different from the one that this dissertation will develop. Sato (2016) proposes the in-situ theory of sluicing for Indonesian. The in situ theory was originally proposed by Kimura (2010), and further developed in Abe & Hornstein (2012) and Abe (2015). Contrasting Merchant's (2001) *wh*-movement and TP deletion idea of sluicing, the in-situ idea argues that sluicing is the deletion of all TP-internal material while leaving the *wh*-phrase that has not moved. (43) illustrates the idea of the in-situ sluicing theory.

(43) John bought something, but I don't know [$_{CP}$ C$_{Q[E]}$ [$_{TP}$ ~~John bought~~ **what**]]

The *wh*-phrase in (43) is not allowed to move due to a PF output economy condition (Chomsky 1986, Chomsky 1995) that bans string-vacuous application of Move. Given that the [E] feature on the C head will instruct PF to delete the TP complement, a *wh*-movement in (43) would result in a string-vacuous application of Move, violating the PF output economy condition. A non-moved *wh*-phrase on the other hand does not violate the PF output economy condition since there is no string-vacuous application of Move.

The in-situ derivation of sluicing in Indonesian works similarly to the English example in (43). (44) shows an example of sluicing in Indonesian.

(44) Esti membeli sesuatu yang mahal di sini, tapi saya tidak ingat apa
Esti buy something COMP expensive in here, but I NEG rememver what
'Esti bought something expensive here, but I don't remember what.'

Within the TP in (44) shown in (45), everything undergoes PF deletion except the *wh*-phrase *apa*, which does not move to avoid string vacuous application of Move.

(45) tapi saya tidak ingat [$_{CP}$ C$_{Q[E]}$ [$_{TP}$ ~~Esti membeli~~ apa]]

The following Indonesian data are used in Sato (2016) to support the in-situ theory of sluicing. Indonesian is closely related to Malay and shares many properties including in situ and ex situ *wh*-question formation.

(46) a. **apa**$_i$ yang kamu pikir Esti akan beli t_i?
what COMP you think Esti will buy

b. kamu pikir Esti akan beli **apa**?
you think Esti will buy what
'What do you think Esti will buy?' (Sato 2016)

Like Malay, Indonesian does not allow p-stranding under regular wh-movement, but allows p-stranding under sluicing (Sato 2011), which violates Merchant's p-stranding generalization as well.

(47) a. *siapa yang kamu berdansa dengan?
who COMP you dance with
'Who did you dance with?'

b. dengan siapa kamu berdansa?
with who you dance
'With whom did you dance?'

c. saya ingat Ali berdansa dengan seseorang, tapi saya tidak tahu (dengan) siapa
I remember Ali dance with someone but I NEG know with who
'I remember Ali danced with someone, but I dont know (with) who'

(Sato 2011:343)

Furthermore, sluicing in Indonesian is also island insensitive following the island insensitive property of a non-moved *wh*-phrase.

(48) a. ***siapa**$_i$ yang kamu suka [cerita yang t_i mengkritik itu]? (*wh*-movement)
who$_i$ COMP you like story COMP t_i criticize DEM
'who do you like the stories that criticized?'

b. David mencari [peneliti yang bekerja **dimana**]? (no *wh*-movement)
David look-for researcher COMP work in.where
'David is looking for the researcher who works where?'

c. David mau bertemu [peneliti yang bekerja di negara tertentu], tapi saya
David want meet researcher COMP work in country certain but I
sudah lupa di negara mana
already forget in country mana
'David wants to meet the researcher who works in a certain country, but I already forgot in which country.' (Sato 2016)

Indonesian has the particle *-kah* which behaves similarly as Malay *-kah* (and will not be repeated here). Given the similarities between Malay and Indonesian sluicing, I agree with Sato (2016) that sluicing in these two languages is due to a *wh*-phrase that has not moved. However, I do not subscribe to his in situ theory of sluicing described here. While I agree that the *wh*-phrase did not undergo *wh*-movement, the implementation of sluicing in Minimalist Grammars (in Chapter 5) shows the *wh*-phrase cannot truly be in-situ either. The next two chapters introduce Minimalist Grammars.

2.3 What about other languages?

This dissertation mainly argues for Malay and Indonesian sluicing as having a no *wh*-movement analysis and possibly extending the analysis to English sluicing. But what about other languages, especially the ones that have been surveyed in Merchant (2001), and that led to a movement and deletion analysis? While I do not make any claims about other languages, I discuss the possibility of extending the same arguments as seen in Malay. So far, the two form-effects that may be relevant to also argue for a no *wh*-movement account in those languages are island insensitivity and the sluicing COMP generalization.

Island insensitivity only becomes a problem if *wh*-movement occurs as it is supposedly to create a ungrammatical structure in the ellipsis site. Until now, sluicing has only ever been observed to be island insensitive. As far as I know, there are no languages that are found to be completely island sensitive under sluicing. This has led to multiple theories of island repair. If the no-movement account is extended to other languages beyond Malay, island insensitivity would no longer be an issue.

Similar to island insensitivity, the sluicing-COMP generalization is found to hold across multiple languages as seen in (40). There has not been any account as to why the C head cannot be pronounced at all despite the Spec position being filled. Merchant's account with deletion of TP does not explain this generalization either. As pointed out before, one could argue for deletion at the C' level if the movement account were to be maintained. This would be similar to having island repair in the deletion site. However, if the no-movement account is adopted instead, the ellipsis site can be much higher, right up to the CP level and the sluicing-COMP generalization is no longer needed.

CHAPTER 3

Syntax and Semantics in Minimalist Grammars

In this chapter, I introduce formalism for Minimalist Grammars (MG) and define a MG that allows for sluicing with no *wh*-movement. MG is a mildly context-sensitive grammar (Michaelis 1998) formalism developed by Stabler (1996). Being mildly context-sensitive, MG are thought to be in the right general class for human language (Joshi 1985). MG is meant to be a precise formulation of Chomsky (1995)'s Minimalist Program, which is the predominant framework in mainstream syntax.

Before delving into MGs, I briefly discuss what I mean by the formal term *"grammar"*. Following Keenan & Stabler (2003), which uses a similar definition for Bare Grammar, a defined grammar has a finite set of items (the lexicon) and a set of operations. The operations are defined recursively to apply to the lexicon and the elements of the lexicon that have been operated on. The result of applying the operations is the ***language generated by G***, L_G, where L_G is the closure of the lexicon under the operations .

For example, let G be a grammar where $\{a, b\}$ is the lexicon and **op** is the set of operations. Then,

1. $\{a, b\} \subset L_G$
2. if $x, y \in L_G$ then $\mathbf{op}(x, y)$, $\mathbf{op}(x)$, $\mathbf{op}(y) \in L_G$

op can be defined to output many different things such as strings or trees. If **op**, in the example above, is defined to output the concatenation of its arguments, $\mathbf{op}(a, b) = ab$, then the grammar outputs strings and $L_G = \{a, b\}^+$. In this case, **op** is unrestricted and is defined to apply to everything. The MG that will be defined below restricts its operations by *features*. MGs that have been defined in the literature traditionally have two structure building operations, which are binary *merge* and unary *move*. The application of the operations to expressions is dependent on the syntactic features of the expressions.

The chapter is divided into three main sections: The first section looks at syntax within the MG framework which includes a formal definition of syntactic operations. The second section introduces the semantic operations defined by Kobele (2012) and the problems with the way they handle quantifiers. In the third section, I reformulate the semantics operations for the system to deal with different quantifiers.

3.1 A Minimalist Grammar: Syntax

Here, I introduce the syntactic system of a MG. In this section, I restrict the MG formulation to only the syntax, where the grammar generates structures independant of meaning. A preliminary definition of an MG without ellipsis is given below:

Definition 3.1.1. A ***minimalist grammar*** is a 5-tuple $G = \langle\ \Sigma,\ sel,\ lic,\ Lex,\ Op\ \rangle$ such that

> Σ is a finite alphabet
>
> Let $F = \{\ +f,\ -f,\ =x,\ x \mid f \in lic,\ x \in sel\ \}$
>
> $Lex \subseteq \Sigma^* \times F^+$ where Lex is finite
>
> $Op = \{\ merge,\ move\ \}$

In an MG with alphabet Σ, $Lex \subseteq \Sigma^* \times F^+$ is the ***lexicon***. Thus Lex is a set of words or morpheme (strings) paired with a list of features[1] $L(G)$ is the language generated by taking

[1] Here, ***Lex*** is presented as a tuple, but should be a triple consisting of a string, a meaning, and a list of features, ***Lex*** $\subseteq \Sigma^* \times M \times F^+$, given that the grammar should be generating a set of string-meaning pairs. The semantic component will be added in the following section.

the closure of *Lex* under *Op*. MGs are defined to be driven by features, where the operations will only apply if certain feature configurations are met. Features enter the derivation with the lexicon and are used by the operations, thus bounding the derivation. In the original system (without ellipsis), there are two types of features and they both come in a positive and negative: ***lic***, licensing features, and ***sel***, selectional features. The features are notated as follows:

Selectional features drive the operation *merge*.

Category `X` (*is of category* `X`)

Selector `=X` (*will merge with something of category* `X`)

Licensing Features drive the operation *move*.

Licensor `+f` (*attracts item with* `-f` *feature*)

Licensee `-f` (*moves to check* `+f` *feature*)

Lexical items are notated with a string separated by two colons then the feature list. For example, `sleeps::=dV` is a lexical item that has the string `sleeps` and two features, the selector `=d` and the category `V`. `sleeps::=dV` says that the string 'sleeps' is of the category V, which can be a VP, and is looking for another lexical item that has the category `d`.

Feature lists on lexical items must be in the following configuration to allow for derivation to proceed:

(49) (`Selector` | `Licensor`)* `Category` (`Licensee`)*

The category label must be present in the feature list. Selector features and Licensor features can both appear before the category label. Any number of licensee features can follow the category label. The features are checked in a left to right order, that is, when features matched in a merge or move operation, they are deleted from the beginning of the list. Using `sleep::=dV`, if we have `John::d`, the `d` feature can check the `=d` feature on `sleep::=dV`. The section below provides examples of the structure that is built when features are checked.

3.1.1 A Tree-generating Minimalist Grammar

Minimalist Grammars can generate many different types of structures, but they are traditionally defined to generate strings or trees. Here I present a tree-generating MG as it will be more familiar to people working on mainstream syntax.

Going back to the simplified example with `sleep::=dV` and `John::d`, when `sleep::=dV` selects for `John::d`, the *merge* function takes the two expressions as input and combines them into a new expression, as shown in figure 3.1.

$$merge\ (\langle sleep\texttt{::=dV}\rangle, \langle\ John\texttt{::d}\ \rangle\ )$$

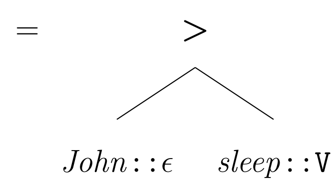


Figure 3.1: *merge* on `sleep::=dV` and `John::d`

Definition 3.1.2. Let ***expression*** (*Expr*) be the set defined by

$$Expr = \text{Arrow}\ Expr\ Expr \mid \Sigma^* \times F^*$$

$$\text{Arrow} = < \mid >$$

Following Hunter & Dyer (2013) an ***expression*** is an ordered binary tree with non-leaf nodes labeled by either $<$ or $>$, and with leaf nodes labeled by elements of $\Sigma^* \times F^*$. An *expression* built up by *merge* can be called a *derived tree*. $<$ is used when a lexical item selects a complement, and $>$ is used when the selectee precedes the selector (in specifiers or adjuncts) and also after a *move* operation, when a lexical item moves up a tree. In other words, the arrows indicate where the ***head*** of the constituent is found. In the simple case in Figure 3.1, `sleep::V` is the head of the expression.

Definition 3.1.3. ***Head*** The head of $\ell \in \Sigma^* \times F^*$ is ℓ

The head of an expression $[_< \; e_1 \; e_2 \;]$ is the head of e_1

The head of an expression $[_> \; e_1 \; e_2 \;]$ is the head of e_2

merge and ***move*** are functions that are defined over expressions. Both are used to construct larger expressions from smaller ones. *merge* is triggered by selectional features while *move* is triggered by licensing features. The definitions below follows the definition in Stabler (2011).

Definition 3.1.4. ***merge*** is a partial function that take two arguments.

$merge$ ($\langle t_1[\texttt{=x}]\rangle$, $\langle t_2[\texttt{x}]\rangle$)

$=$ $<$ (if $\langle t_1[\texttt{=x}]\rangle \in Lex$)

t_1 t_2

$=$ $>$ (otherwise)

t_2 t_1

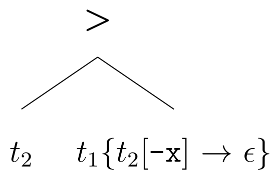
$t[\alpha]$ is an expression whose head has the feature sequence where the first element is α. The first case of *merge* creates complements while the second case of *merge* creates specifiers.

Definition 3.1.5. ***move*** is a function that takes one argument.

Let $t\{t_1 \to t_2\}$ be the result of replacing subtree t_1 with t_2 in t.

$move\ (\ \langle t_1[\texttt{+x}]\rangle\ )$

$=$



When *move* applies to an expression $t_1[\texttt{+x}]$ that contains a subtree $t_2[\texttt{-x}]$, it adds t_2 as specifier and deletes the `+x` feature on $t_1[\texttt{+x}]$ and also removes the subtree $t_2[\texttt{-x}]$ from $t_1[\texttt{+x}]$.

Let us consider a simple example. Let $G = \langle\ \Sigma,\ sel,\ lic,\ Lex,\ Op\ \rangle$ such that

$\Sigma = \{\text{did, what, John, eat}\}$

$sel = \{\texttt{d, V, C}\}$, $lic = \{\texttt{wh}\}$

$Lex = \{$`did::=V+whC`, `what::d-wh`, `John::d`, `eat::=d=dV` $\}$

$Op = \{\ merge,\ move\ \}$

The tree in figure 3.2 is generated by G.

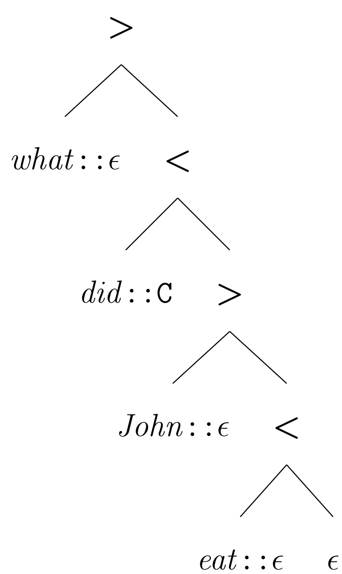
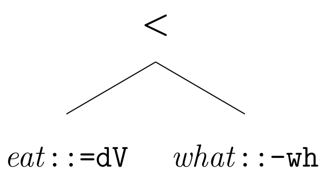


Figure 3.2: Derived tree generated by G

1. First, *merge* is applied to `eat::=d=dV` and `what::d-wh` since the first `=d` on *eat* selects for the first `d` on *what*. The `-wh` feature on *what* will remain uncheck until a `+wh` appears in the derivation.

   merge (⟨*eat*::`=d=dV` ⟩, ⟨ *what*::`d-wh` ⟩)

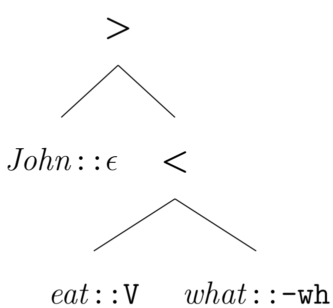
   =

2. Next, `John::d` is merged with the tree from 1. Since *John* only has one feature `d`, once it is selected, the feature list on *John* is empty. The resulting structure that is built here is a VP.

   merge (⟨*expr* in 1⟩, ⟨ *John*::`d` ⟩)

   =

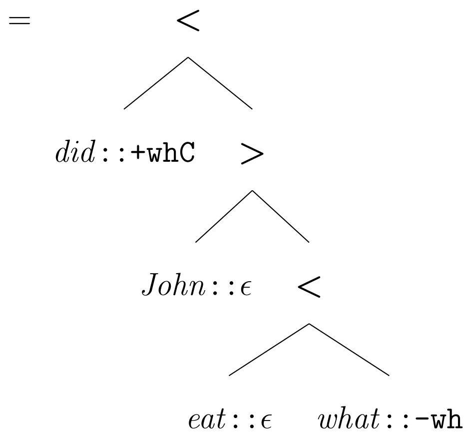
   

3. `did::=V+whC`[2] is merged similarly to *John*.

   merge (⟨ *did*::`=V+whC` ⟩, ⟨*expr* in 2⟩)

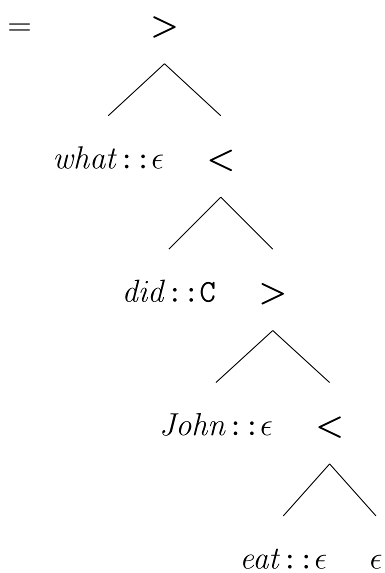
---

[2]In this simple example, I am abstracting away from the TP layer, therefore 'did' is given the feature list {=V+whC} so the *wh*-word moves right after the merging of 'did'.



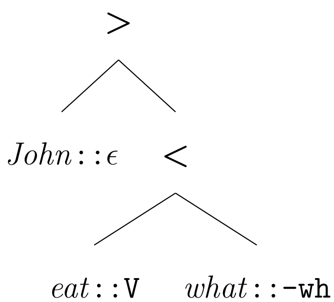
4. Since the next available feature is `+wh` on *did*, move is applied to the *Expr* in 3 and the held out subconstituent headed by `-wh`, which is *what*, is 'moved' to the specifier of the head that has the `+wh` feature.

move (⟨expr in 3⟩)



5. The derivation is now complete and `C` is the only feature left uncheck. This means that the expression that has been derived is a CP.

The derived tree in Figure 3.2 is the output of the grammar that we defined. In this example, *merge* and *move* are defined on trees, and the output can be transformed into strings for purposes of PF. However, Kobele (2006) observes that these derived tree structures are functionally inert, since neither of the syntactic operations of the grammar operate on feature-less subtrees. Therefore, we can eliminate syntactically superfluous information in the derived trees by instead representing expressions as tuples of categorized strings. For example, the resulting expression in step 2 above (repeated below)



can be represented as (*John eat*), V : (*what,*-wh). In this case, the *wh*-word 'what' is held out to the side of the main structure. This idea can also be applied to derived structures, where an item with unchecked negative feature is held out in a syntactic store. This notion of holding out lexical items creates multi-component expressions that will be crucial for the Elide function later on.

The configuration of the reduced representation is as follows: (*main string*): Features, (Moving subconstituent). Although we could easily get strings from derived structures, as seen in the example above (also see Kobele (2006) for the conversion method), we could also generate strings without derived structures. The section below discusses a string-generating MG, where *merge* and *move* are defined on strings instead of trees.

3.1.2 A String-generating Minimalist Grammar

In a string-generating MG, expressions are tuples of categorized strings, upon which *merge* and *move* operate. In this system, *Expr* is a sequence of string and feature stacks tuples, $Expr = (\Sigma^* \times F^+)^*$.

Definition 3.1.6. ***merge*** takes two *Expr*s and outputs one *Expr*. For α, $\beta \in \mathrm{F}^*$, s, t strings,

movers_s , $\text{movers}_t \in (\Sigma^* \times F^+)^*$:

$$\textit{merge}\ (\langle s,\ =\text{x}\alpha\rangle\text{:movers}_s)\ (\langle t,\ \text{x}\beta\rangle\text{:movers}_t) =$$
$$\begin{cases} \langle s{}^\frown t\rangle, \alpha : \text{movers}_s{}^\frown\text{movers}_t & \text{if } \beta = \epsilon \text{ and movers}_s = \epsilon \text{ and } \langle s, = x\alpha\rangle \in \text{Lex} \\ \langle t{}^\frown s\rangle, \alpha : \text{movers}_s{}^\frown\text{movers}_t & \text{if } \beta = \epsilon \text{ and } \langle s, = x\alpha\rangle \notin \text{Lex} \\ \langle s, \alpha\rangle : t, \beta : \text{movers}_s{}^\frown\text{movers}_t & \text{if } \beta \neq \epsilon \end{cases}$$

Definition 3.1.7. ***move*** takes an *Expr* and outputs an *Expr*. For α, β, $\gamma \in \text{F}^*$, s,t, strings, suppose $\exists!\langle t, \beta, \rangle \in$ *movers* such that $\beta = -f\gamma$. Let *movers'*= movers - $\langle t, \beta\rangle$.

$$\textit{move}\ (\langle s,\ +\text{f}\alpha\rangle\text{:movers}) = \begin{cases} \langle t{}^\frown s\rangle, \alpha : \textit{movers'} & \text{if } \gamma = \epsilon \\ \langle s, \alpha\rangle : t, \gamma : \textit{movers'} & \text{if } \gamma \neq \epsilon \end{cases}$$

Using the same example grammar (repeated below) as the derived tree,

Let $G = \langle\ \Sigma,\ sel,\ lic,\ Lex,\ Op\ \rangle$ such that

Σ = {did, what, John, eat}

sel = {`d, V, C`}, *lic* = {`wh`}

Lex = {`did::=V+whC`, `what::d-wh`, `John::d`, `eat::=d=dV` }

Op = { *merge*, *move* }

we get the resulting expression:

$\langle$*what did John eat* $\rangle$, `C` : ϵ

The derivation of the string, *what did John eat*, parallels that of the derived structure.

1. *merge* is applied to `eat::=d=dV` and `what::d-wh`. Since the `-wh` feature on *what* is a licensee waiting to be checked, it is put in the movers list, and held out of the main

derivation.

merge ($\langle$*eat*`::=d=dV` $\rangle$, $\langle$ *what*`::d-wh` $\rangle$)

= $\langle$*eat*$\rangle$, `=dV` : $\langle$*what,*`-wh`$\rangle$

2. *John* is merged in the derivation and gets concatenated with the main string. The resulting structure that is built here is a VP.

merge ($\langle$*expr* in 1$\rangle$, $\langle$ *John*`::d` $\rangle$)

= $\langle$*John eat*$\rangle$, `V` : $\langle$*what,*`-wh`$\rangle$

3. *did* is concatenated similarly to *John*.

merge ($\langle$ *did*`::=V+whC` $\rangle$, $\langle$*expr* in 2$\rangle$)

= $\langle$*did John eat*$\rangle$, `+whC` : $\langle$*what,*`-wh`$\rangle$

4. Since the next available feature is `+wh` on *did*, *move* is applied and *what* is retrieved from the movers list and concatenated with the rest of the string.

move ($\langle$expr in 3$\rangle$)

= $\langle$*what did John eat*$\rangle$, `C` : ϵ

5. The derivation is complete and the resulting string is of category `C`.

The section shows that the derivation of strings and derived trees with *merge* and *move* parallel each other. Ideally, the parallelism is maintained if any new operation is added to the syntax.

3.1.3 Derivation Tree

The derivation of the derived tree in Figure 3.2 and the string in the section above can itself be represented in tree form. The ***derivation tree*** describes the application of *merge* and *move* to expressions. The leaves of a derivation tree contain elements of *Lex* while the non-leaf nodes are labeled with elements of ***Op***. The *derivation tree* for the derived tree in Figure 3.2 and the string version is given in Figure 3.3. The numbering beside each operation on the non-leaf node corresponds to the output in the examples above.

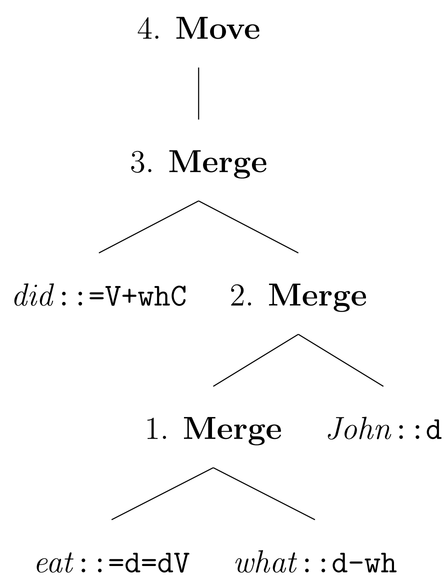


Figure 3.3: Derivation tree for 'what did John eat'

The formal definition of a *derivation tree* and the interpretation functions ($eval_T$) are given below. The purpose of *eval* is to map a *derivation tree* to an *Expr*, either as a tree structure or a string.

Definition 3.1.8. Let ***Derivation tree*** ($DevT$) be defined by

$$DevT = \Sigma^* \times F^+ \mid \text{Merge DevT DevT} \mid \text{Move DevT}$$

Definition 3.1.9. $\boldsymbol{eval_T}$ is a partial function that takes $D_1, D_2 \in DevT$ as its argument and outputs $t \in Expr$. Let $str \in \Sigma^*$ and $f \in F$.

$eval_T$ (Merge $D_1 D_2$) = $merge$ ($eval_T D_1$) ($eval_T D_2$)

$eval_T$ (Move D) = $move$ ($eval_T D$)

$eval_T$ (str, f) = (str, f)

At this point, having both derivation tree and derived structure/strings may seem redundant,as the information encoded in the derivation tree is found on the derived structure as well.

3.1.4 Constraints in Minimalist Grammars

Before introducing semantics for MG, I briefly discuss the constraints of ***move*** in MG, given that movement in traditional syntax is constrained and that *wh*-movement is involved in theories of sluicing. MG as originally formalized by Stabler (1996) come with one constraint, namely the Shortest Move Constraint (SMC). The SMC limits the number of identical negative licensing features that enter the derivation to one. For example, a derivation tree like in Figure 3.4 violates the SMC, because it contains two subtrees with the same licensee feature `-f` that would be licensed by the licensor feature `+f` that is merged later in the tree.

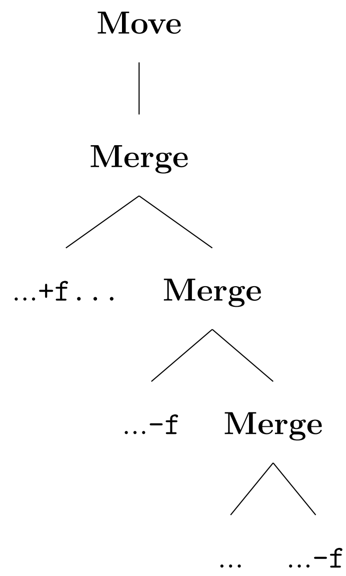


Figure 3.4: Derivation tree that violates the SMC

Thus, ***move*** carries the requirement that for each *f*, a licensing feature, there can only be one structure that is waiting to move.

However, this constraint does not prohibit the derivation of a sentence containing movement out of an island. As long as there remains an unchecked licensee, any corresponding licensor that is merged further up in the derivation will be able to check the feature through ***move***. Thus, neither of the operations defined in the section before will stop the derivation of sentences like (50) as given in the derivation tree in Figure 3.5.

(50) *Who did John believed the claim that Bill saw?

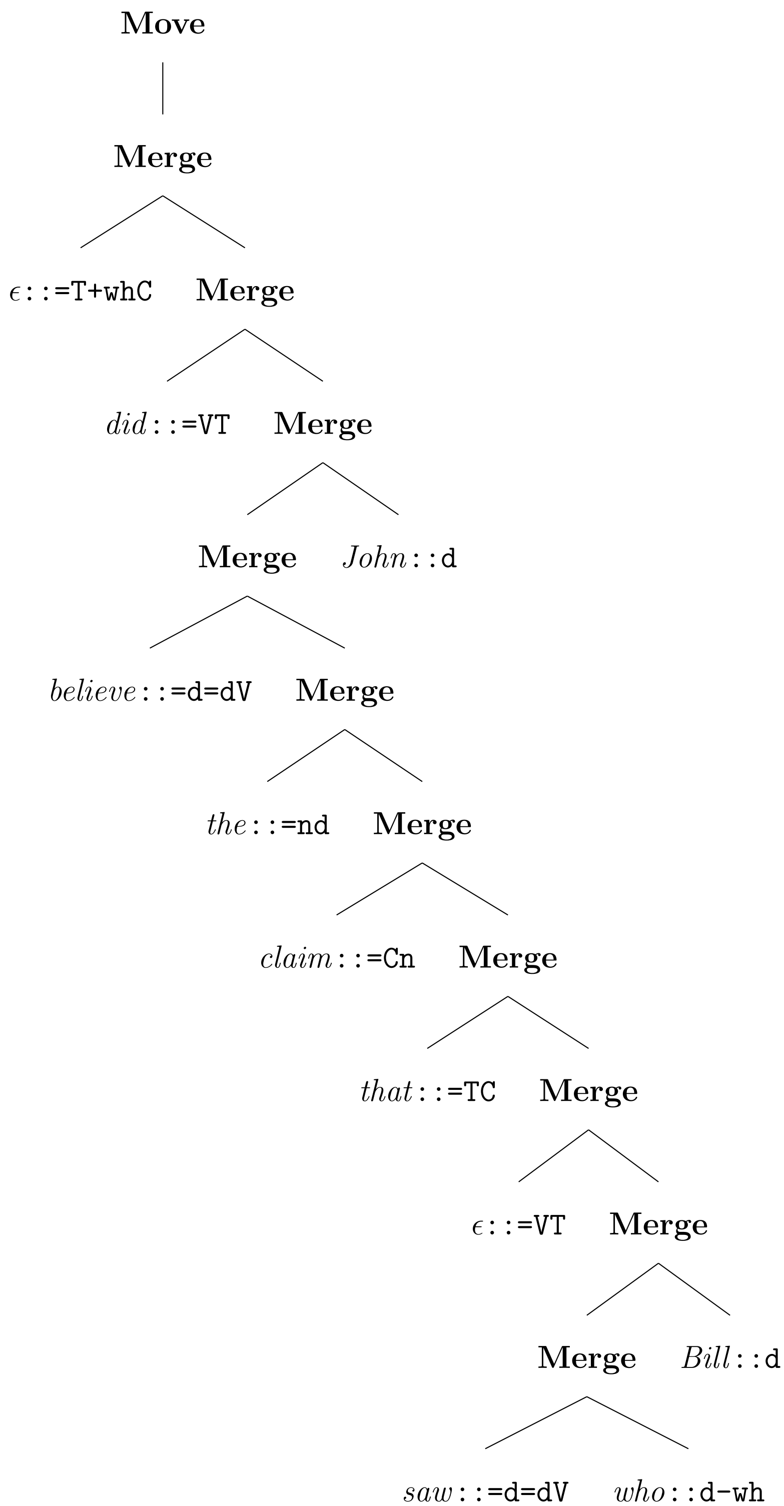


Figure 3.5: Derivation tree of an island violating sentence

Allowing derivations such as Figure 3.5 leads to the generation of ill-formed sentences. However, island insensitivity is one of the main phenomena observed under sluicing. Derivations such as in Figure 3.5 have to be ruled out by our grammar while a different derivation is needed to account for island insensitivity under sluicing. While this dissertation is not about island constraints in MG, it is possible to implement constraints on movement such as the Subjacency Condition (Chomsky 1973) to rule out sentences like (50). I do not claim that the Subjacency Condition is the correct way to constrain movement, but it is one possible implementation of constraining movement in MG.

In non-MG syntax, the Subjacency Condition is formulated by having certain phrases be blocking categories, and having movement of an item across too many of these blocking categories be prohibited. The idea of blocking categories can be implemented into MG by adding an extra feature, as well as including a count on the number of blocking categories. However, instead of implementing the Subjacency Condition in the syntax, I intend to implement a form of condition within the semantics of MG to rule out sentences with island effects due to *wh*-movement. This is due to the formulation of the Elide function which will be introduced in chapter 5.

The point of this section is to indicate that it is possible to include constraints on movement in MG movement across islands is ruled out within the MG framework. The formulation of an MG that counts blocking categories will be shown in chapter 6. In the next section, I discuss semantics within the framework and how the framework accounts for quantification, which is crucial for sluicing.

3.2 A Minimalist Grammar: Semantics

The goal of this section is to present a semantic interpretation scheme for the MG presented in the previous section. Though semantic interpretation in traditional Chomskyan linguistics is normally viewed as operating on derived structures (Kratzer & Heim 1998), a compositional semantics over derivational trees has been formulated by Kobele (2006) and updated in Kobele (2012). Since the analysis that I have argued for sluicing involves the scoping capabilities of the indefinite in the antecedent, the mechanism in which indefinites, universal quantifiers and questions take scope has to be addressed. The semantic computation for Quantified Noun Phrases (QNP) (both existential and universal quantification) in Kobele (2012) is taken to involve storage and retrieval of the meaning of the QNP. This correctly derives scopal interpretation within a single clausal sentence with multiple quantifiers, as well as the tensed-clause boundedness of quantifier raising. However, in treating universal and existential quantifiers the same way, the computation fails to derive the exceptional wide scope readings of indefinites, especially those found in the antecedent of sluiced sentences. In this section, I present and modify Kobele's (2012) simply-typed compositional semantic computation for MGs, particularly his treatment of quantifier scope to differentiate between questions, existential and universal quantifiers. I will first describe the scopal properties of questions, and existential and universal quantification before presenting Kobele's (2012) semantic computation of scope in MGs.

3.2.1 Quantifier Scope

The relation between syntactic and semantic description is central to all formal linguistics theories. In many cases, there is a direct correlation between form and meaning. English direct *wh*-questions are a typical example of this correlation. In the derivation of a *wh*-question in English, the *wh*-word appears in two positions. The first position is where the *wh*-word first merges in and the second position is where the *wh*-word syntactically moves to. A *wh*-word in an English question is assumed to take propositional scope at the position it has syntactically moved to as shown in (51)[3].

[3] In proposition-set frameworks (Hamblin 1976, Karttunen 1977, Groenendijk & Stokhof 1985 , Ciardelli et al. 2012), the semantic content of questions are represented as a set of propositions and *wh*-words are taken to denote sets of elements in their domain restrictions. However, for ease of exposition, *wh*-words here will be notated with WH instead of sets of elements.

(51) a. Who_1 did John see t_1?

b. $[\![who]\!] = \lambda P.WHx.[Px]$

c. $[\![John\ see\ ]\!] = \textbf{see}(\textbf{j},y)$

d. $[\![who]\!](\textbf{see}(\textbf{j},y)) = WHx.[\textbf{see}(\textbf{j},x)]$

In (51a), *who*, is first introduced as an argument of *see*, and then moves to the left edge of the structure and leaves a trace in its original position. In (51b), the *wh*-word is denoted as a scope taking operator and binds a variable that is found in the position where it first merges in. The y in (51c) is the variable that is bound by the *wh*-operator. Thus, the semantic meaning in (51d) mirrors the structure of the moved *wh*-word and its trace in (51a). In such cases, building a compositional semantic system is easy, as the meaning can easily be computed by tracking the syntactic operations that built the sentence.

On the other hand, quantifier scope presents a longstanding challenge to this relation between form and meaning. In many cases, a quantificational expression semantically behaves as if it appeared in a different position than its actual position in surface structure. Quantifiers in object position, such as the indefinite *someone* (52) appears in one position in a sentence. It does not syntactically move like *wh*-expressions (as seen in 51). However, it cannot be interpreted in its syntactic position, thus this creates a syntax-semantics mismatch.

(52) a. John $[_{VP}$ saw someone$]$

b. $[\![someone]\!] = \lambda Q.\exists x[\textbf{person}(x) \wedge Qx]$

c. $[\![John\ saw\ ]\!] = \textbf{see}(\textbf{j},y)$

d. $[\![someone]\!](\textbf{see}(\textbf{j},y)) = \exists x[\textbf{person}(x) \wedge \textbf{see}(\textbf{j},x)]$

The meaning of the sentence in (52a) requires *someone* to take propositional scope (just like *wh*-expressions) since the object in *John saw* is evaluated for different values. This clashes with the fact that *someone* does not appear to syntactically move at all and only appears in one position throughout its syntactic derivation. The sister of *someone* is the verb *saw* and there is no overt movement of *someone* to the position where it takes propositional scope, unlike *who* in (51a). *someone* syntactically appears in the same position as the variable that it is supposed to bind. This make the propositional scope of *someone* not a constituent. The same mismatch is found for universal quantification as well, as shown in (53).

(53) a. John $[_{VP}$ saw everyone$]$

b. $[\![everyone]\!] = \lambda Q.\forall x[\mathbf{person}(x) \rightarrow Qx]$

c. $[\![John\ saw]\!] = \lambda y[\mathbf{see}(\mathbf{j},y)]$

d. $[\![everyone]\!](\lambda y[\mathbf{see}(\mathbf{j},y)]) = \forall x[\mathbf{person}(x) \rightarrow \mathbf{see}(\mathbf{j},x)]$

This presents a challenge to any theory of compositional semantics that allows for question sentences like (52). Furthermore, the semantics of sentences with multiple quantifiers, such as (54), presents a further challenge for the form and meaning relation. (54) demonstrates that English exhibits opposite scope relations where the string in (54) is ambiguous, and has two different readings.

(54) Every student read a book.

Readings:

i Subject Wide Scope: $\forall x[\text{student}(x) \rightarrow \exists y[\text{ book}(y) \wedge \text{ read }(y)(x)]]$

ii Object Wide Scope: $\exists y[\text{book}(y) \wedge \forall x[\text{ student}(x) \rightarrow \text{read }(y)(x)]]$

In the first reading of (54), *every student* takes wide scope relative to *a book*. In this case, each student could have read a different book: John read ‘War and Peace’, Mary read ‘Anna Karenina’, Bill read ‘Resurrection’ etc. In the second reading, *a book* takes wide scope relative to *every student*. In this case, there would be a single book that every student read: all the students read ‘War and Peace’. A compositional theory of semantics would have to give an account for this scopal ambiguity of quantifiers.

To further complicate the relation between form and meaning, (54) shows that a quantifier in object position can take wide or narrow scope with respect to another quantifier in subject position. However, a universal quantifier embedded within a tensed clause is not able to take wide scope over another quantifier in subject position of a matrix clause as shown in (55).

(55) Someone ensured that everyone laughed. ($\exists > \forall$, $*\forall > \exists$)

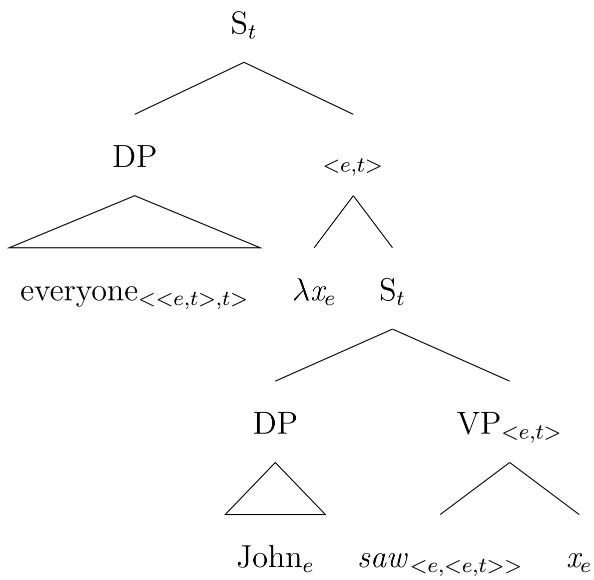
Though the sentence in (55) contains two quantifiers, it only has one reading where *everyone* takes narrow scope with respect to *someone*. In the following section, I present Kobele's (2012) semantic formalism for MGs that accounts for the scopal properties of quantifiers presented here. Since MG have no distinct level of grammar for meaning, syntax and semantic operations must occur simultaneously within the derivation, creating a fully compositional system. This creates the perfect testing ground for an analysis of ellipsis given the interaction of syntax and semantics in ellipsis.

3.2.2 Kobele's (2012) MG with Quantifier Scope

To resolve this form-meaning mismatch in traditional Chomskyan syntax, covert movement of the quantifier, called Quantifier Raising, is introduced in LF, a separate level of syntax that deals with the logical aspect of grammar. This movement of the quantifier is similar to *wh*-movement as seen in (51) in that the quantifier moves to the position where it takes propositional scope. However, instead of being overtly moved like *wh*-phrases, the quantifier moves in LF and appears in situ on the surface. The structure for (53a) in LF with QR is shown below.

(56) a. John saw everyone.

b.



In the context of MGs, there are no distinct levels of the grammar for meaning. The derivation trees are taken to also be the semantic representation of the sentence (Kobele 2009, Kobele 2012). This means that string and meaning are derived simultaneously, where each syntactic operation is associated with a semantic operation.

As a first step in creating this interpretation scheme, the denotation of a syntactic object can be thought of as a simply-typed lambda term and each *merge* operation can be thought of as function application. However, this is not sufficient to explain the scope of quantifiers since they can be interpreted in a position different from where they were merged into the derivation. To account for quantifiers, Kobele (2012) denotes syntactic object as a pair of simply-typed lambda term and a quantifier store. Thus, $\alpha = \langle a, A\rangle$, where α is the denotation of a syntactic object, a is the lambda term and A is the store component. A syntactic object with a moving subexpression has a quantifier store to store the meaning of the moving expression. The meaning of the stored item can be retrieved later during a movement step. The idea is that the meaning of a quantifier (as well as questions[4]) can be stored away and later retrieved to allow for scope taking at a position different from where the quantifier was first merged. In this

[4]Kobele (2012) never explicitly discussed how *wh*-phrases take scope but the idea of stored expressions can be extended to questions as well.

system, lexical items is treated as being paired with the empty store. This system of storage and retrieval of meaning, often called Cooper storage, was first hypothesized by Cooper (1983).

Kobele (2012) implements Hornstein's (1998) idea of Quantifier Raising in MG using this storage and retrieval technique for quantifier scope. Hornstein (1998) argues that a quantifier may take scope in any of its chain positions. However, all but one chain link in each chain must be deleted at LF and the position of the remaining chain determines the scope of the quantifier. In this system, there is no "covert movement" of QR, the movements of all quantifiers is overt, and the copy at the top of the movement chain is pronounced. For (57), there will be four configuration possibilities, shown in (58).

(57) Someone attended every seminar.

(58) a. $[_{AgrS}$ Someone [attended $[_{AgrO}$ every seminar $[_{VP}$ ~~someone~~ ~~every seminar~~]]]]

b. $[_{AgrS}$ Someone [attended $[_{AgrO}$ ~~every seminar~~ $[_{VP}$ ~~someone~~ every seminar]]]]

c. $[_{AgrS}$ ~~Someone~~ [attended $[_{AgrO}$ ~~every seminar~~ $[_{VP}$ someone every seminar]]]]

d. $[_{AgrS}$ ~~Someone~~ [attended $[_{AgrO}$ every seminar $[_{VP}$ someone ~~every seminar~~]]]]

In (58), *attended* undergoes verb movement from the VP to a higher clause between AgrS and AgrO. In (58a) to (58c), *someone* scopes over *every seminar* since the undeleted copy of *someone* c-commands *every seminar*. In (58d), the scope relations are inverted since the copy of *every seminar* c-commands *someone*.

Kobele (2012) implements this idea of QR with quantifiers taking scope through movement features. When a quantifier is merged into a derivation, it leaves behind a variable to be merged with a syntactic functor while its meaning gets stored separately. The meaning of a quantifier can be retrieved only once during a movement step and it takes scope in that position. Once the meaning is retrieved from the quantifier store, it cannot be stored again, even though the quantifier may still have other movement features to check.

Adding the quantifier store to MGs means that there are two possible semantic reflexes of *merge*. The first case is function application from one argument to another, either forward (FA) or backward (BA) application, which Kobele calls **mergeApp**[5].

(59) $\textbf{mergeApp}(\alpha,\beta) = \begin{cases} \langle a(b),\ A \cup B\rangle & \text{(FA)} \quad \text{where} \quad \alpha = (a, A) \\ \langle b(a),\ B \cup A\rangle & \text{(BA)} \qquad\qquad\quad \beta = (b, B) \end{cases}$ (Kobele 2012)

In the second case, shown in (60), one of the arguments is a moving expression, a quantifier, that introduces a variable, (x_f). The meaning of the moving expression is inserted into the store, indexed by the next licensee feature type it moves to check ($[f := b]$), and can only be retrieved during a *move* operation. The other argument, a function from individuals, is applied to the variable $((x_f))$ that was introduced by the quantifier (as it bears the same index). The semantic reflex of this merge is called **mergeStore**.

(60) $\textbf{mergeStore}(\alpha,\beta) = \langle a(x_f),\ A \cup B[f := b]\rangle$
(where β's next feature is `-f`) (Store) (Kobele 2012)

Kobele defined three possible semantic reflexes of *move*. The first is **moveEmpty**, which is used when the moving expression has already taken scope, thus there is nothing in the quantifier store. **moveLater** is used when the moving expression is going to take scope at a later position. In (62), the item with index f, has been pulled out the store and the variable with the f index is abstracted over. The variable with index g is still waiting to be removed from the store when the right feature merges in. Since there is no scoping action in **moveEmpty** and **moveLater**, the semantic reflex is taken to be the identity function (Id). **moveNow** is used when the moving expression is going to take scope at that position. In this case, the variable is abstracted over, $A(f)(\lambda x_f.a)$, and the resulting predicate is given as argument to the stored expression A/f. For the definitions below, `-f` is assumed to be the feature checked by the instance of *move*.

(61) $\textbf{moveEmpty}(\alpha) = \langle a, A\rangle$ (where A is undefined at f) (Id)

[5]A more formal definition of the semantic operations in MG will be given in the following section. Kobele (2012) has a variable free version of these definitions in his paper that will not be discussed here.

(62) $\textbf{moveLater}(\alpha) = \langle(\lambda x_f.a)(x_g),\ A_{g\leftarrow f}\rangle$ (for `-g` the next feature to check) (Id)

(63) $\textbf{moveNow}(\alpha) = \langle A(\mathtt{f})(\lambda x_{\mathtt{f}}.a), A/\mathtt{f}\rangle$ (Retrieve) (Kobele 2012)

A thing to note about semantic and syntactic storage: In many of the forthcoming examples, semantic storage might look like it correlates with syntactic storage, as both use negative licensing features to trigger the storing mechanism. However, this is not the case. There will always be syntactic storage if there is semantic storage, but the opposite is not true. With syntactic storage, such as with case (`k` features, as we will see below), there is no semantic storage. In the 'Derivation' section, I show how these different modes of semantic combination work within MGs.

3.2.2.1 Derivations

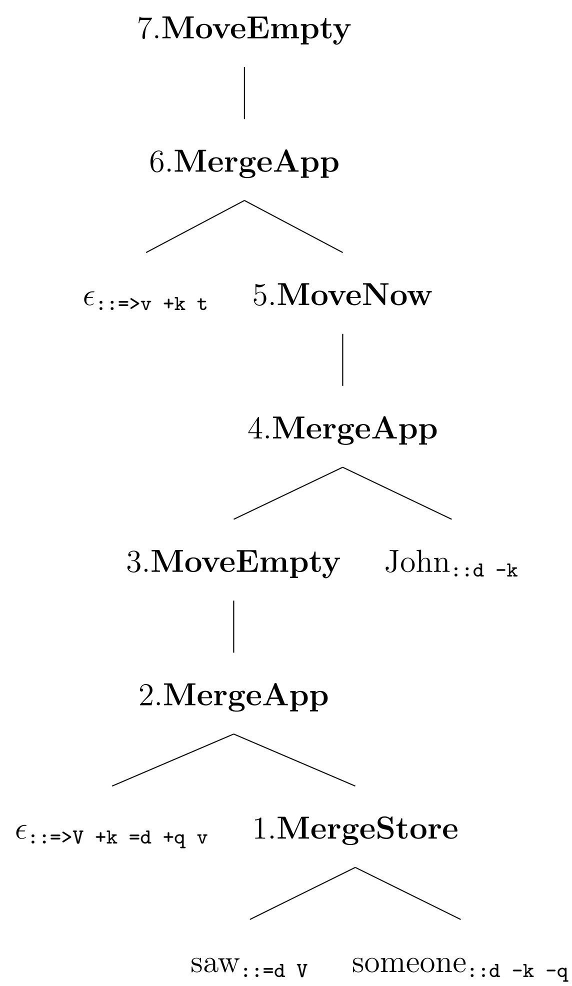
The derivation trees from now onwards will be labeled with the semantic operations instead of the syntactic operations, in order to eliminate the ambiguity caused by implementing the quantifier reconstruction proposal (Hornstein 1998). According to this proposal, the positions in which a quantifier can take scope are those to which it has been moved. This leads to ambiguity in the mapping from derivation trees to meanings. A simple solution is to change labeling to semantic operations instead. The syntax of the two semantic merge operations, *mergeApp* and *mergeStore*, still correspond to *merge* while the three semantic move operations, *moveEmpty*, *moveLater*, and *moveNow*, correspond to *move*. To ensure that the quantifier takes scope in the right position, Kobele (2012) adds a new licensing feature type, 'Q'. A determiner like *some* has the features `=n d -k -q` while the active voice head will have the features `=>V =d +q v`[6].

In a simple case with a quantifier in object position, *John saw someone*, *someone* takes propositional scope above *John saw _*. The derivation tree for the sentence is given below. In Kobele (2012), tensed clauses are created by the a lexical item with the 'prog' feature or by

[6] `=>` feature indicates head movement, while the 'k' feature type indicates case. See Kobele (2009) for more details.

a lexical item *-ed* with the ‘perf‘ feature. Both items come after a lexical item with `=>v` feature. They are omitted in the following examples, and the idea of a tensed clause will be represented with a lexical item that is ϵ`::=>v +k t`.

(64) John saw someone.



Going from the bottom of the derivation tree, the semantics of the tree in (64) is calculated as shown in Table 3.1. The semicolon in the table below is used to separate the main meaning from the stored meaning. The item to the left of the semicolon is the main meaning while any item to the right is in storage.

Table 3.1: Semantics of (64)

1.	$\mathbf{see}(x)$; $\lambda Q.\exists x[\mathbf{person}(x) \wedge Qx]$	Store
2.	$\mathbf{see}(x)$; $\lambda Q.\exists x[\mathbf{person}(x) \wedge Qx]$	FA
3.	$\mathbf{see}(x)$; $\lambda Q.\exists x[\mathbf{person}(x) \wedge Qx]$	Id
4.	$\mathbf{see}(\boldsymbol{j},x)$; $\lambda Q.\exists x[\mathbf{person}(x) \wedge Qx]$	BA
5.	$\exists x[\mathbf{person}(x) \wedge \mathbf{see}(\boldsymbol{j},x)]$	Retrieve
6.	$\exists x[\mathbf{person}(x) \wedge \mathbf{see}(\boldsymbol{j},x)]$	FA
7.	$\exists x[\mathbf{person}(x) \wedge \mathbf{see}(\boldsymbol{j},x)]$	Id

When the quantified NP *someone* is merged with *see* at step 1, the meaning of the existential is stored away and the denotation of *see* combines with a variable. At step 2, the empty head with the `+q` feature is merged in. However, the `-q` feature in this tree can only be checked after the subject of the sentence is merged at step 4. At step 4, *John* which has a `-k`, does not get its meaning stored away since it is not a quantifier. The derivation proceeds as normal until step 5, where the checking of the `-q` movement feature retrieves the meaning of the existential from storage and the quantifier takes propositional scope at that position. The rest of the derivation no longer involves the quantifier store as the only item in the store has been removed and cannot be reinserted. The same mechanics are applied to universal quantifiers in object position.

For sentences with multiple quantifiers, such as *someone devoured everyone*, the object wide scope (OWS) reading is derived by storing the semantics of the object only and not the subject. The derivation tree and the semantics for *someone devoured everyone* is as follows:

(65) Something devoured everyone. (OWS)

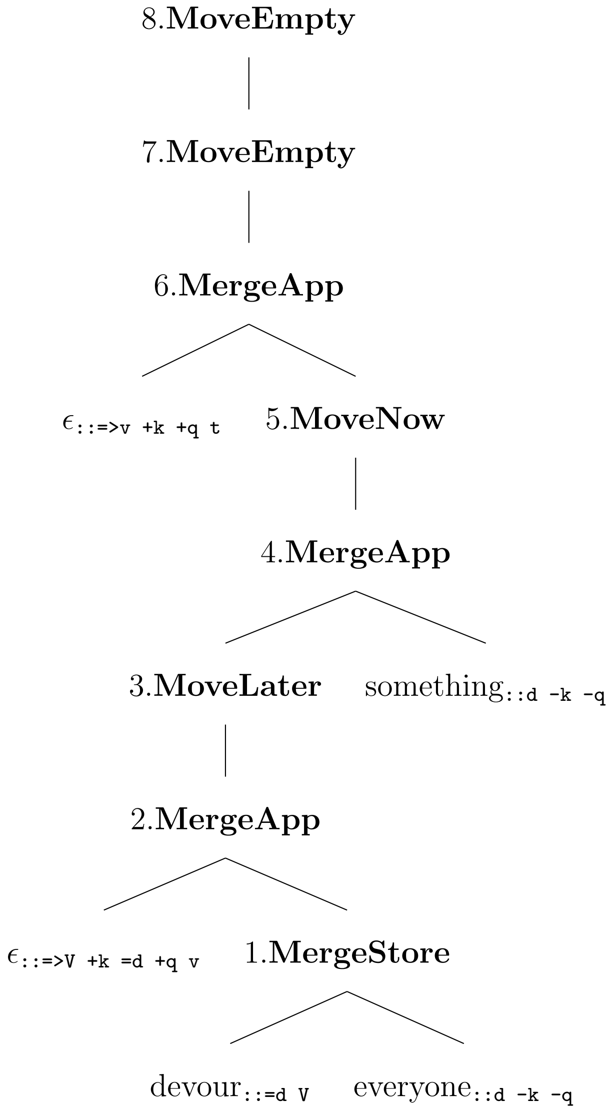

8.MoveEmpty
7.MoveEmpty
6.MergeApp
ϵ::=>v +k +q t
5.MoveNow
4.MergeApp
3.MoveLater
something::d -k -q
2.MergeApp
ϵ::=>V +k =d +q v
1.MergeStore
devour::=d V
everyone::d -k -q

Table 3.2: Object Wide Scop

1.	$\mathbf{devour}(x)$; $\lambda Q.\forall x[\mathbf{person}(x) \rightarrow Qx]$	Store
2.	$\mathbf{devour}(x)$; $\lambda Q.\forall x[\mathbf{person}(x) \rightarrow Qx]$	FA
3.	$\mathbf{devour}(x)$; $\lambda Q.\forall x[\mathbf{person}(x) \rightarrow Qx]$	Id
4.	$\exists y[\mathbf{thing}(y) \wedge \mathbf{devour}(y,x)]$; $\lambda Q.\forall x[\mathbf{person}(x) \rightarrow Qx]$	BA
5.	$\forall x[\mathbf{person}(x) \rightarrow \exists y[\mathbf{thing}(y) \wedge \mathbf{devour}(y)(x)]]$	Retrieve
6.	$\forall x[\mathbf{person}(x) \rightarrow \exists y[\mathbf{thing}(y) \wedge \mathbf{devour}(y)(x)]]$	FA
7.	$\forall x[\mathbf{person}(x) \rightarrow \exists y[\mathbf{thing}(y) \wedge \mathbf{devour}(y)(x)]]$	Id
8.	$\forall x[\mathbf{person}(x) \rightarrow \exists y[\mathbf{thing}(y) \wedge \mathbf{devour}(y)(x)]]$	Id

At step 1, *everyone* is merged with *devour* and the universal meaning of *everyone* is inserted into the store. At step 4, when the subject *something* is merged into the derivation, it combines straightaway with the denotation of *devour* and its meaning does not go into the store. Note that at this step, even though *something* has a `-q` feature, its meaning does not go into semantic storage, but it is still placed into syntactic storage. At step 5, the `-q` feature of *everyone* is checked and the universal meaning is retrieved from the store and combined with the semantics of *something devoured _*. This allows the universal object *everyone* to take scope over the subject *someone*.

On the other hand, to get the subject wide scope (SWS) reading of the sentence in (65), both meaning of subject and object are inserted into the store. The derivation tree for subject wide scope is given below.

(66) Something devoured everyone. (SWS)

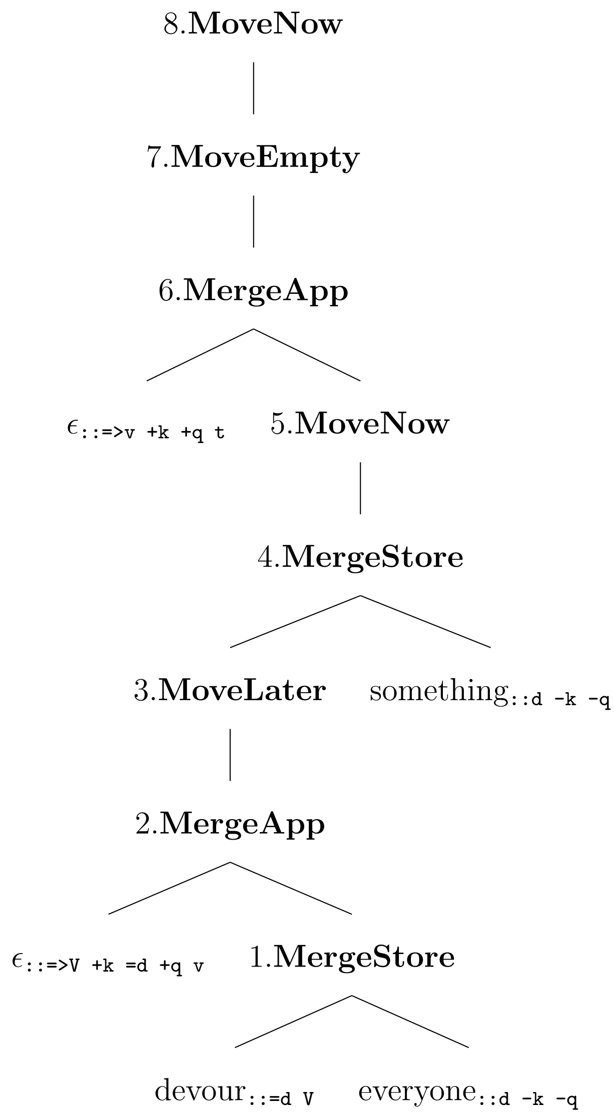

8.MoveNow
7.MoveEmpty
6.MergeApp
ϵ::=>v +k +q t
5.MoveNow
4.MergeStore
3.MoveLater
something::d -k -q
2.MergeApp
ϵ::=>V +k =d +q v
1.MergeStore
devour::=d V
everyone::d -k -q

Table 3.3: Subject Wide Scope

1.	$\mathbf{devour}(x)$; $\lambda Q.\forall x[\mathbf{person}(x) \rightarrow Qx]$	Store
2.	$\mathbf{devour}(x)$; $\lambda Q.\forall x[\mathbf{person}(x) \rightarrow Qx]$	FA
3.	$\mathbf{devour}(x)$; $\lambda Q.\forall x[\mathbf{person}(x) \rightarrow Qx]$	Id
4.	$\mathbf{devour}(x,y)$; $\lambda Q.\forall x[\mathbf{person}(x) \rightarrow Qx]$; $\lambda P.\exists y[\mathbf{thing}(y) \wedge Py]$	Store
5.	$\forall x[\mathbf{person}(x) \rightarrow \mathbf{devour}\ (y)(x)]]$; $\lambda P.\exists x[\mathbf{thing}(y) \wedge Py]$	Retrieve
6.	$\forall x[\mathbf{person}(x) \rightarrow \mathbf{devour}\ (y)(x)]]$; $\lambda P.\exists x[\mathbf{thing}(y) \wedge Py]$	FA
7.	$\forall x[\mathbf{person}(x) \rightarrow \mathbf{devour}\ (y)(x)]]$; $\lambda P.\exists x[\mathbf{thing}(y) \wedge Py]$	Id
8.	$\exists y[\mathbf{thing}(y) \wedge \forall x[\ \mathbf{person}(x) \rightarrow \mathbf{devour}\ (y)(x)]$	Retrieve

The difference between the derivations for SWS and OWS is at step 4, when the subject *someone* merges in. For SWS, the meaning of the subject *something* is stored as well. At step 5, the meaning of the object *everyone* is retrieved from storage and combines with the denotation of *devour* while the meaning of *someone* is only retrieved at step 8 when its `-q` feature is checked. This allows the existential quantifier to take scope over the universal quantifier in the sentence.

Kobele (2012) also accounts for the tensed clause boundedness of quantifier raising as discussed in (55) repeated below.

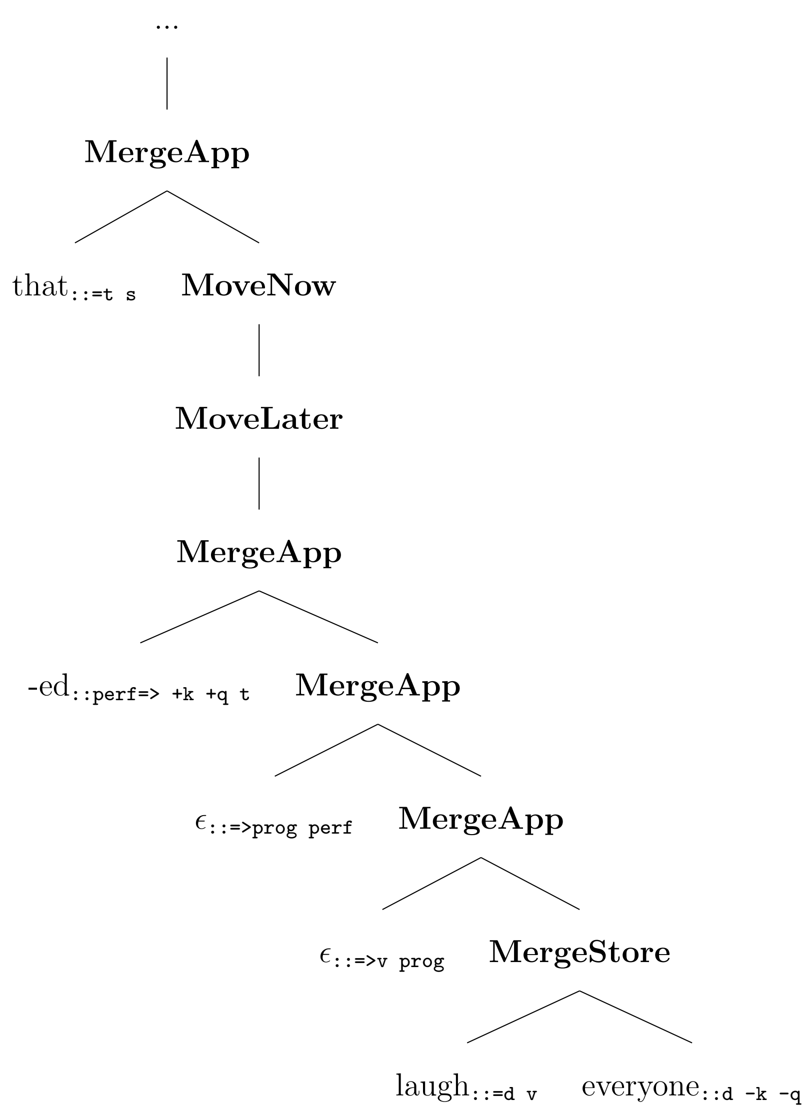
(55) Someone ensured that everyone laughed. ($\exists > \forall$, $*\forall > \exists$)

As discussed before, the quantified NP *everyone* in (55) is bound within a tensed clause and is unable to take scope over *someone.* The mechanics of MG as defined by Kobele (2012) account for the tensed-clause boundedness of quantifier raising for the simple reason that a tensed clause which has the head *-ed* with feature bundle `perf=> +k +q t` that forces a Q feature to be checked[7]. Thus, the embedded *everyone* has its Q feature checked before the

[7]The Shortest Move Constraint limits the number of the same negative licensing features from entering

matrix subject is merged into the derivation. The partial derivation tree of (55) is shown below. In the example below, I use Kobele's orignal tensed clause head, -ed`::perf=> +k +q t`, to show how quantifier scope is restricted.

(67) (Partial derivation of) Someone ensured that everyone laughed.



Kobele (2012) treats universal and existential quantifiers the same syntactically, as in they both have the feature list `d -k -q`, which allows them to take scope whenever a movement

the derivation to one. This applies to the semantic store which is correlated to syntactic store. Hence, there will not be more than one quantifier for a single +q feature to check.

step occurs. However, this proposal undergenerates as it fails to capture other properties of indefinites. In the following section, I highlight the problem that indefinites pose to Kobele's proposal.

3.2.2.2 The Problem with Indefinites

Indefinites are known to have exceptional scoping behaviour (Ruys 1992, Reinhart 1997, Brasoveanu & Farkas 2011, Charlow 2014), compared to other quantifiers. They have no trouble projecting their quantificational force out of tensed clauses (which can also be syntactic islands) as shown in the examples in (68), which is a minimal pair of (55) .

(68) Everyone ensured that someone laughed. ($\exists > \forall$, $\forall > \exists$)

(68) has two different readings (as opposed to only one reading in (55)), where the indefinite can take both narrow and wide scope:

(69) Readings of (68):

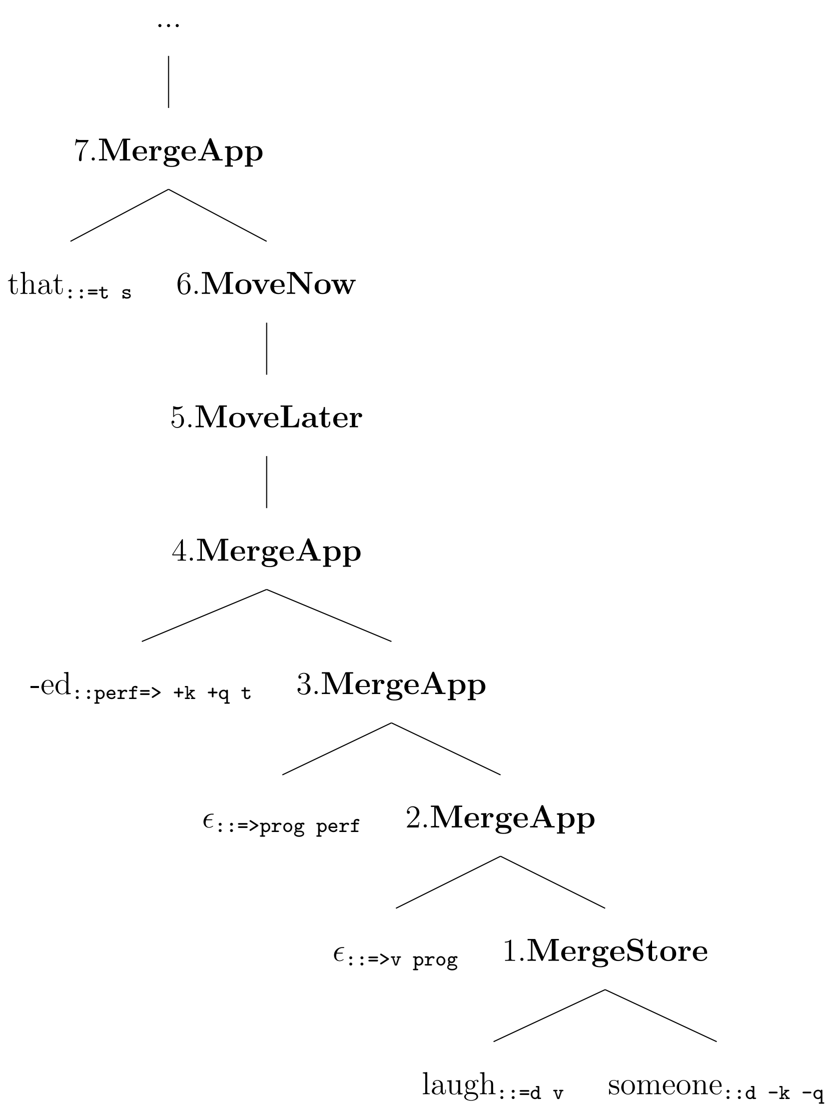
- i Narrow Scope: 'For every person x, x ensures that there is a person y, such that y laughed'
- ii Wide Scope: 'There is a person y, such that, for every person x, x ensures that y laughed'

The availability of the wide reading of the indefinite is a problem for Kobele's computation. Since all quantifiers have the same feature bundle, `d -k -q`, the wide scope reading of the indefinite is not derivable as the mechanism prevents all quantifiers from scoping outside tensed clauses. This is a problem for sluicing in general as the indefinite in the antecedent always has wide scope. (70) shows that sluicing is possible from embedded clauses. After changing (68) into a sluiced sentence (70), only the wide scope indefinite reading ($\exists > \forall$) is available.

(70) Everyone ensured that someone laughed, but I don't know who. ($\exists > \forall$)

The partial derivation tree of (68) illustrates the problem.

(71) (Partial derivation of) Everyone ensured that someone laughed.



At step 1, the meaning of *someone* is stored when it merges into the derivation. The meaning is retrieved at step 6 due to the *-ed* head with the `+q` feature. Step 6 is the final point where retrieval can happen for the indefinite *someone*. The indefinite no longer has any moving features and cannot move any furhter. This allows only for the narrow scope reading of the indefinite because it is bound within the tensed clause.

A simple solution to this problem would be to get rid of the `+q` feature on the *-ed* head (`perf=> +k`) for indefinites, and put the `+q` feature on another lexical item further up the tree. This would allow for the existential meaning to be stored further up the derivation tree. However, this would be an arbitrary solution that does not say much about why only indefinites, and not other quantifiers, have this unique ability to take scope. The reason why certain *-ed* heads have a `+q` feature while others do not would have to be stipulated as well. The availability of wide scoping indefinites suggests that it is problematic to treat indefinites and universal quantifiers the same, i.e by giving them the same feature bundle. In the next section, I reformulate the semantics for MG where indefinites denote sets of alternatives instead of a simple existential quantifier. The reformulated semantics will allow the sets of alternatives to 'expand' across tensed clauses, deriving the wide scope reading of indefinites.

3.3 A Reformulated Semantics for MG

Instead of treating indefinites like all other quantifiers, I follow Kratzer and Shimoyama's (2002) treatment of indefinites where an indefinite denotes a set of elements in their domain restrictions. This analysis follows semantics of questions in Hamblin (1976) where the root denotation of a question is the set of propositions that are possible answers of the question.

(72) a. ⟦Who slept?⟧ ={John slept, Mary slept...}

b. ⟦Which person met which⟧

On Hamblin's treatment, a *wh*-word introduces a set of alternatives, and these alternatives propagate up through the phrases until they reach a question operator. For Kratzer & Shimoyama (2002), indefinites denote a set of alternatives like a *wh*-word. For example, the denotation of *someone* is illustrated in (73).

(73) $[\![\text{someone}]\!] = [\![\text{who}]\!] = \{x|\ \text{human}(x)\}$

The idea is that the alternatives that are introduced by the indefinite percolate up the tree, resulting in a set of alternative propositions. For Kratzer & Shimoyama (2002), lexical expressions are all set-denoting. However, lexical items in this proposal still denote their standard denotations and not singleton sets. In this framework, function application is defined pointwise, as in Hamblin (1976). Function application combines an expression α, which denotes a function, with an expression β, which denotes a set of objects, by having the function apply to each of the objects in the set. The individual alternatives introduced by the indefinite give rise to alternatives of higher types via pointwise functional application. The combination of *someone* with a verb like *arrive* results in a set of propositions as in (74).

(74) $[\![\text{arrive}]\!]([\![\text{someone}]\!]) = \{\ \textbf{arrive}(x) \mid x \in \text{human}\ \} = \{\text{John arrived, Bill arrived...}\}$

Eventually, the set of alternatives reaches a suitable operator that provides it with its quantificational force. This closes off the set of alternatives. The denotation of the sentence in (74) combines with an existential closure operator in (75a), which gives the proposition in (75b).

(75) a. For any set of propositions Ω: $[\![\exists]\!](\Omega) = \bigvee\{p|p \in \Omega\}$

b. $\exists x.\ \text{human}(x) \wedge \text{arrive}(x)$ = at least one human arrived

In the following section, I incorporate this idea of indefinites into the MG framework which allows indefinites to percolate through tensed clauses.

3.3.1 Hamblin Semantics in MG

In this section, I modify Kobele's (2012) semantics system for MGs to include alternative sets for indefinites. Since indefinites are denoted as a set of alternatives that will be closed off by a suitable operator, they no longer require the quantifier store for storage nor the Q feature for movement. As a result, indefinites in this system take scope through a merge operation, and not through movement. Unlike Kobele (2012), the denotation of a syntactic object is a pair of simply-typed lambda term with a list of selector features and a quantifier store. The selector feature is included to ensure the correct retrieval from the quantifier store. The quantifier store itself is a list of triples consisting of a lambda term, a variable and a list of features. Hence, $\alpha = \langle(a,\ \text{F}^*),\ \text{A}\rangle$, where α is the denotation of a syntactic object, a is a lambda term, F is the list of selector features, and A is the store component which itself is $(m, v, \mathrm{F}^{+})^{*}$ where m is the denotation of a quantifier, v is a variable and F is a list of features.

I add a new semantic merge operation called ***mergeSet*** to Kobele's existing system. mergeSet is pointwise function application, and deals with arguments that are sets. Some of the definitions in this section are written in Haskell-like syntax, where the semi-colon : is a cons function that adds an element to the front of a list, and [ ] is an empty list. Functions that are written with Haskell-like syntax come with type constructors, indicated by `::`, that denote the type of the function. For example, a function 'add' that adds two integers together will have the following type constructor, add`::  Int -> Int -> Int`.

Definition 3.3.1. ***mergeSet*** is a partial function that takes two arguments $m_1, m_2 \in (M \times F^*) \times (M \times V \times F^+)^*$and outputs $m_3 \in (M \times F^*) \times (M \times V \times F^+)^*$

Let $(M \times V \times F^+)^* = S$

For any set $s_1, s_2 \in S,\ f_1, f_2 \in F^*$

1. if x_1 has type $\alpha \rightarrow \beta$ and x_2 has type $\{\alpha\}$ and $f_2 = \epsilon$ then
   mergeSet $((x_1, f_1), s_1)((x_2, f_2), s_2)$
   $= (\{x_1(d) | d \in x_2\}, f_1 \frown f_2), (s_1 \frown s_2)$

2. if x_1 has type $\{\alpha \rightarrow \beta\}$ and x_2 has type α and $f_2 = \epsilon$ then
   mergeSet $((x_1, f_1), s_1)((x_2, f_2), s_2)$
   $= (\{d(x_2) | d \in x_1\}, f_1 \frown f_2), (s_1 \frown s_2)$

3. if x_1 has type $\{\alpha\}$ and x_2 has type $\alpha \rightarrow \beta$ and $f_2 = \epsilon$ then
   mergeSet $((x_1, f_1), s_1)((x_2, f_2), s_2)$
   $= (\{x_2(d) | d \in x_1\}, f_1 \frown f_2), (s_1 \frown s_2)$

4. if x_1 has type α and x_2 has type $\{\alpha \rightarrow \beta\}$ and $f_2 = \epsilon$ then
   mergeSet $((x_1, f_1), s_1)((x_2, f_2), s_2)$
   $= (\{d(x_1) | d \in x_2\}, f_1 \frown f_2), (s_1 \frown s_2)$

5. if x_1 has type $\{\alpha \to \beta\}$ and x_2 has type $\{\alpha\}$ and $f_2 = \epsilon$ then

   $mergeSet\ ((x_1, f_1), s_1)((x_2, f_2), s_2)$

   $= (\{d_1(d_2) | d_1 \in x_1, d_2 \in x_2\}, f_1 \frown f_2), (s_1 \frown s_2)$

6. if x_1 has type $\{\alpha\}$ and x_2 has type $\{\alpha \to \beta\}$ and $f_2 = \epsilon$ then

   $mergeSet\ ((x_1, f_1), s_1)((x_2, f_2), s_2)$

   $= (\{d_2(d_1) | d_1 \in x_1, d_2 \in x_2\}, f_1 \frown f_2), (s_1 \frown s_2)$

Now that meaning can be denoted in terms of sets of alternatives, we need an operation that merges the appropriate operator to close off the sets. This operation is defined under ***mergeApp***, which is still function application but now also has the ability to close off a set, as seen in 3 and 4 in definition (3.3.2) .

Definition 3.3.2. ***mergeApp*** is a partial function that takes two arguments $m_1, m_2 \in (M \times F^*) \times (M \times V \times F^+)^*$and outputs $m_3 \in (M \times F^*) \times (M \times V \times F^+)^*$

Let $(M \times V \times F^*) = S$

For any $s_1, s_2 \in S,\ f_1, f_2 \in F^*,\ \alpha,\ \beta$

1. if x_1 has type $\beta \to \alpha$ and x_2 has type β, then

   $mergeApp\ ((x_1\ f_1), s_1)\ ((x_2\ f_2), s_2)$

   $= ((x_1(x_2))\ (\ plusF\ f_1 \frown f_2),\ (s_1 \frown s_2))$

2. if x_1 has type β and x_2 has type $\beta \to \alpha$, then

   $mergeApp\ ((x_1\ f_1), s_1)\ ((x_2\ f_2), s_2)$

   $= ((x_2(x_1))\ (\ plusF\ f_1 \frown f_2),\ (s_2 \frown s_1))$

   where $plusF :: F^* \to F^*$[8]

[8]The mergeApp function must keep track of positive licensor features `+f`, as these could be meanings that need to be retrieved from storage. This is accomplished by the 'plusF' function.

```
plusF [ ] = [ ]
plusF (r:rs) =
  if r = +f then f:plusF(xs)
  else plusF(xs)
```

This semantic system retains Kobele's idea of storage and retrieval for other quantificational elements with ***mergeStore*** and ***moveNow*** (which I have renamed ***moveRet***).

Definition 3.3.3. ***mergeStore*** is a partial function that takes two arguments $m_1, m_2 \in (M \times F^*) \times (M \times V \times F^+)^*$and outputs $m_3 \in (M \times F^*) \times (M \times V \times F^+)^*$

if x_1 has type $e \to \beta$ and x_2 has type $\langle\ e \to t \to t\ \rangle$,

let y be a fresh variable and $y \notin s_2,\ y \notin s_1,\ y \notin x_1,\ y \notin x_2$, then

$$mergeStore\ ((x_1\ f_1), s_2)\ ((x_2\ f_2), s_1)$$

$$= ((x_i(y)\ f_1), ((x_2, y, f_2)^\frown\ s_1^\frown s_2))$$

Definition 3.3.4. ***moveRet*** is a partial function that takes one argument $m_1 \in (M \times F^*) \times (M \times V \times F^+)^*$and outputs $m_2 \in (M \times F^*) \times (M \times V \times F^+)^*$. Let *Store* $\in (M \times V \times F^+)^*$.

$$moveRet\ (m, f : fs), s$$

$$= let\ findInStore\ (f)\ s = (y, v)\ in$$

$$(y(\Gamma), fs), (removeFromStore(m, f, s))$$

where

$$\Gamma = \begin{cases} \lambda v.m & \text{if } m \text{ is not a set} \\ \{\lambda v.m' | m' \text{ in } m\} & \text{if } m \text{ is a set} \end{cases}$$

and

$findInStore :: F \times Store \rightarrow (M, V)$ and
$removeFromStore :: M \times F \times Store \rightarrow Store$

The details of the *findInStore* and *removeFromStore* functions are not given in the definition above. *findInStore* finds the correct meaning from the store by matching the licensing feature to be checked with the one that has been kept in storage through *mergeStore*. *removeFromStore* then removes the retrieved item.

moveEmpty and ***moveLater*** are still identity functions in this system. Since the derivation tree is now named with semantic operations, the formal definition of it has to be changed slightly, as well as the interpretation function ($eval_T$). I also define a new interpretation function $eval_M$ which maps a derivation tree to a meaning.

Definition 3.3.5. ***Derivation tree (DevT)*** is defined by
$DevT = \Sigma^* \times F^+ \times M$ | MergeApp DevT DevT | MergeSet DevT DevT | MergeStore DevT DevT | MoveRet DevT | MoveEmpty DevT | MoveLater DevT

Definition 3.3.6. $eval_T$ is a function that takes $t \in DevT$ as its argument and outputs $t \in Expr$. Let $str \in \Sigma$ and $f \in F$.

$$eval_T \text{ (MergeApp } D_1 D_2 \text{) } = merge \text{ (} eval_T D_1 \text{) (} eval_T D_2 \text{)}$$

$$eval_T \text{ (MergeStore } D_1 D_2 \text{) } = merge \text{ (} eval_T D_1 \text{) (} eval_T D_2 \text{)}$$

$$eval_T \text{ (MergeSet } D_1 D_2 \text{) } = merge \text{ (} eval_T D_1 \text{) (} eval_T D_2 \text{)}$$

$$eval_T \text{ (MoveRet } D \text{) } = move \text{ (} eval_T D \text{)}$$

$$eval_T \text{ (MoveLater } D \text{) } = move \text{ (} eval_T D \text{)}$$

$$eval_T \text{ (MoveEmpty } D \text{) } = move \text{ (} eval_T D \text{)}$$

$$eval_T \text{ (} str, f, m \text{) } = \text{ (} str, f \text{)}$$

Definition 3.3.7. $eval_M$ is a partial function that takes $t \in DevT$ as its argument and outputs $m \in (M \times F^*) \times (M \times V \times F^+)^*$. Let $str \in \Sigma$ and $f \in F$.

$$eval_M \text{ (MergeApp } D_1 D_2 \text{)} = mergeApp \text{ (} eval_M D_1 \text{) (} eval_M D_2 \text{)}$$

$$eval_M \text{ (MergeStore } D_1 D_2 \text{)} = mergeStore \text{ (} eval_M D_1 \text{) (} eval_M D_2 \text{)}$$

$$eval_M \text{ (MergeSet } D_1 D_2 \text{)} = mergeSet \text{ (} eval_M D_1 \text{) (} eval_M D_2 \text{)}$$

$$eval_M \text{ (MoveRet } D \text{)} = moveRet \text{ (} eval_M D \text{)}$$

$$eval_M \text{ (MoveLater } D \text{)} = moveLater \text{ (} eval_M D \text{)}$$

$$eval_M \text{ (MoveEmpty } D \text{)} = moveEmpty \text{ (} eval_M D \text{)}$$

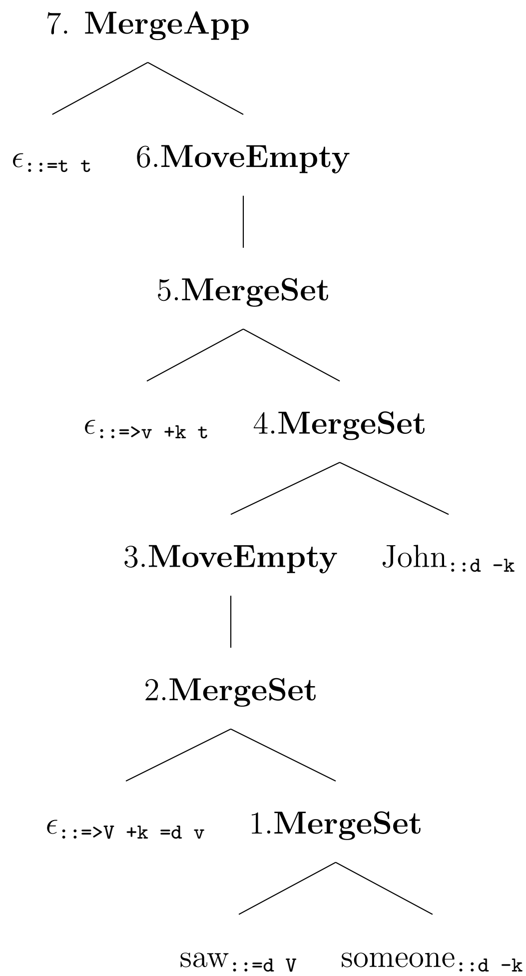
$$eval_M \text{ (} str, f, m \text{)} = \text{(} m)$$

In the next section, I illustrate the derivation of wide scoping indefinites using the modified MG defined above.

3.3.2 Derivations

A few changes to the syntax are required before delving into the derivations of the new semantic system. In Kobele's (2012) system, scoping of the quantifier is achieved through syntactic movement. Since indefinites now denote a set of alternatives that are closed off by an existential operator later, they no longer require the Q feature or the quantifier store for scope taking. Thus, in this system, indefinites have the feature bundle `d -k` while other quantificational items still have the Q feature in their feature bundle, `d -k -q`. NNow that there are two scope-taking mechanisms, indefinites take scope via the merging of an existential operator while all other quantifiers take scope via a *move* operation. The operator that closes off the set of alternatives comes in the form of an empty lexical item with feature bundle `=t t`. This lexical item is always within a TP and its semantic type is {t} → t, $\lambda_{\{t\}}. \bigvee\{t\}$. It operates under the first case of the *mergeApp* function where α is t and β is {t}. I will first show the derivation of a simple example *'John saw someone'* using the new semantic system with ***mergeSet*** for indefinites.

(76) John saw someone.



Going from the bottom of the derivation tree, the semantics of the tree in (76) is calculated in Table (3.4). The semantics is given with the type $(M \times F^{*}) \times (M \times V \times F^{+})^{*}$. Since the semantic store is not used in the example, the feature and store are empty. The store is presented after the semi-colon.

1.	$\{\lambda x.see(x,a),\ \lambda x.see(x,b)...\}$, ϵ; ϵ
2.	$\{\lambda x.see(x,a),\ \lambda x.see(x,b)...\}$, ϵ; ϵ
3.	$\{\lambda x.see(x,a),\ \lambda x.see(x,b)...\}$, ϵ; ϵ
4.	$\{see(j,a),\ see(j,b)...\}$, ϵ; ϵ
5.	$\{see(j,a),\ see(j,b)...\}$, ϵ; ϵ
6.	$\{see(j,a),\ see(j,b)...\}$, ϵ; ϵ
7.	$\bigvee\{see(j,a),\ see(j,b)...\}$, ϵ; $\epsilon \equiv \exists x.[see(j,x)]$

Table 3.4: Semantics of tree in (76)

At step 1, when the indefinite *someone* merges with *saw*, pointwise function application combines the meaning of the two to yield a set of alternatives. The alternatives continue to expand until step 7, where the existential operator merges in and closes the set.

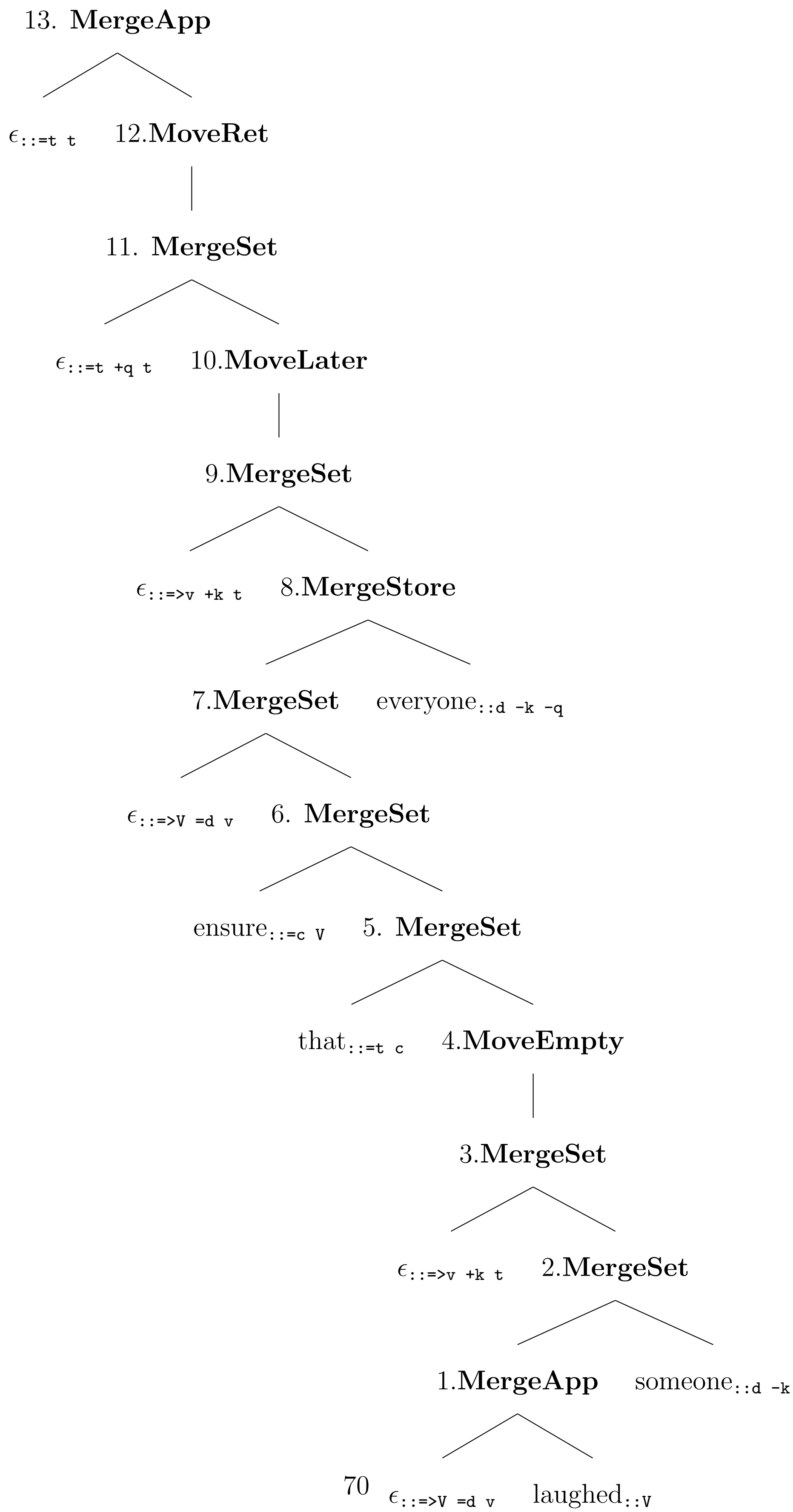
We must ensure that the interactions between the two scope-taking mechanisms do not restrict available scope readings. In Kobele (2012), the selector Q feature on the head of the tensed clauses () restricts scope of all quantifiers within tensed clauses. In this system, however, placing the Q feature in this position would restrict the the scope of all quantifiers below an indefinite. In a sentence with two quantified elements, *someone met everyone* and *everyone met someone*, the indefinite *someone* would always take widest scope as the existential operator is always merged much later in the derivation than the head of the tensed clause. This is a problem for the new system since the ambiguity of such sentences would be lost with the addition of the existential operator.

To resolve the problem, I change the location of the selector Q feature. The selector Q feature will no longer be on the head of the tensed clause, but on a null lexical item that has the feature bundle `=t +q t`, and that can either be in the embedded or the matrix clause. The changing of the location of the Q feature is necessary due to the interaction of the two scoping mechanisms and it does make the prediction that quantificational scoping always happens at the TP level. Now, indefinites can take scope across tensed clause boundaries. When there is a tensed clause, it is compulsory for ϵ`::{=t +q t}` to be in the embedded clause[9] to ensure that other quantifiers are restricted within the tensed clause. The derivation tree and semantics for sentences with *wh*-questions and multiple quantifiers can be found in the appendix.

The next derivation is of the example in (68), where the indefinite *someone* gets a wide reading over the universal quantifier: *everyone ensured that someone laughed*. Set percolation in this MG systems allows indefinites to take scope across tensed clause boundaries while still restricting the scope of other quantifiers. Due to space limitation, I will not be using Kobele's *-ed* lexical item to indicate tensed clause. The tensed clause is assumed to come with the empty lexical item ϵ`::=>v +k t`.

(68) everyone ensured that someone laughed.

[9]This is an arbitrary decision for the system but a needed one given the split between universal and existential quantification.


13. MergeApp
ϵ::=t t
12.MoveRet
11. MergeSet
ϵ::=t +q t
10.MoveLater
9.MergeSet
ϵ::=>v +k t
8.MergeStore
7.MergeSet
everyone::d -k -q
ϵ::=>V =d v
6. MergeSet
ensure::=c V
5. MergeSet
that::=t c
4.MoveEmpty
3.MergeSet
ϵ::=>v +k t
2.MergeSet
1.MergeApp
someone::d -k
ϵ::=>V =d v
laughed::V

A partial semantic calculation for the tree above is shown in Table 3.5. Again, the semantics is given with the type $(M \times F^{*}) \times (M \times V \times F^{+})^{*}$and the store is presented after the semicolon.

1.	$\lambda x.\mathbf{laugh}(x)$, ϵ; ϵ
2.	$\{\lambda x.\mathbf{laugh}(\text{a}), \mathbf{laugh}(\text{b})...\}$, ϵ; ϵ
3.	$\{\lambda x.\mathbf{laugh}(\text{a}), \mathbf{laugh}(\text{b})...\}$, ϵ; ϵ
	...
6.	$\{\lambda x.\mathbf{ensure}(\text{x}, \mathbf{laugh}(\text{a})), \lambda x.\mathbf{ensure}(\text{x}, \mathbf{laugh}(\text{b}))...\}$, ϵ; ϵ
	...
8.	$\{\mathbf{ensure}(q, \mathbf{laugh}(\text{a})), \mathbf{ensure}(q, \mathbf{laugh}(\text{b}))...\}$, `-q`
	; $\lambda Q.\forall y[\mathbf{person}(y) \rightarrow Qy]$, q, `-q`
	...
12.	$\{\forall x[\mathbf{person}(x) \rightarrow \mathbf{ensure}(x,\mathbf{laugh}(\boldsymbol{a}))]..\}$, ϵ; ϵ
13.	$\bigvee\{\forall x[\mathbf{person}(x) \rightarrow \mathbf{ensure}(x,\mathbf{laugh}(\boldsymbol{a}))]..\}$, ϵ; ϵ
	$\equiv \exists y[\mathbf{person}(y) \wedge \forall x[\mathbf{person}(x) \rightarrow \mathbf{read}(x,\forall y[\mathbf{book}(y) \rightarrow \mathbf{ensure}(x,\mathbf{laugh}(\boldsymbol{y}$

Table 3.5: Partial semantic calculation of tree in (68)

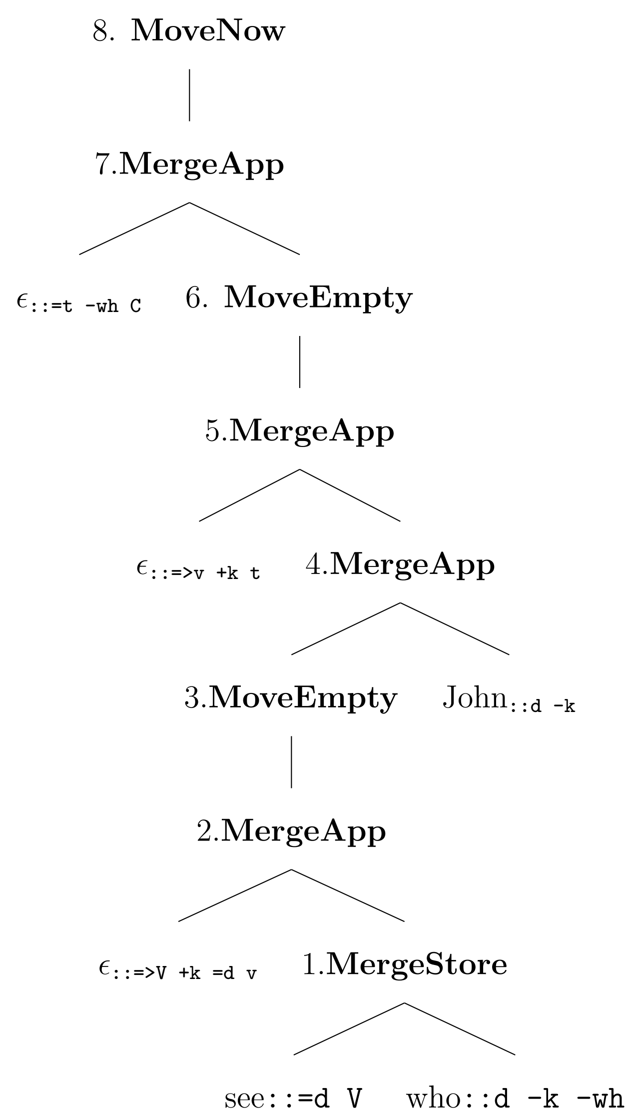
At step 2, the indefinite *someone* is merged into the derivation and introduces a set of alternatives that is composed with the denotation of *laugh*. The set of alternatives expands up the derivation tree and composes with the denotation of *ensure* at step 6. Up until this point, the quantifier store has remained empty, as no quantifier has merged in. At step 8, the universal quantifier *everyone* is merged into the derivation and its meaning is inserted into the quantifier store. It leaves behind the variable 'q' in the main meaning, as well as the feature '-q', which keeps track of the meaning it must combine with when it is retrieved from the store. In step 12, the meaning of the universal quantifier is retrieved and it combines with the set of alternatives (through pointwise functional application as well). The store returns to being empty. In step 13, the existential operator is merged into the derivation thereby allowing the indefinite to take the widest scope. Since the scope taking ability of the indefinite does not rely on any movement features, it is not bounded by licensing features on heads that can only appear in certain clauses.

This concludes the introduction of Minimalist Grammars for this dissertation. The appendix at the end of this chapter shows the derivations of questions and sentences with multiple quantifiers. In the next chapter, I discuss the identity condition that is needed for the ellipsis operation before describing the implementation of sluicing in MG in the following chapter.

Appendix: Derivations (Questions and Scopal Ambiguity with Multiple Quantifiers)

For question sentences, the new semantic operations are not needed as *wh*-words in this system retains their original meaning and not a set of alternatives. The meaning of the *wh*-word is similarly stored as with universal quantifiers and is retrieved during the movement operation where *wh*-words is syntactically moved to the left edge.

(77) Who did John see?

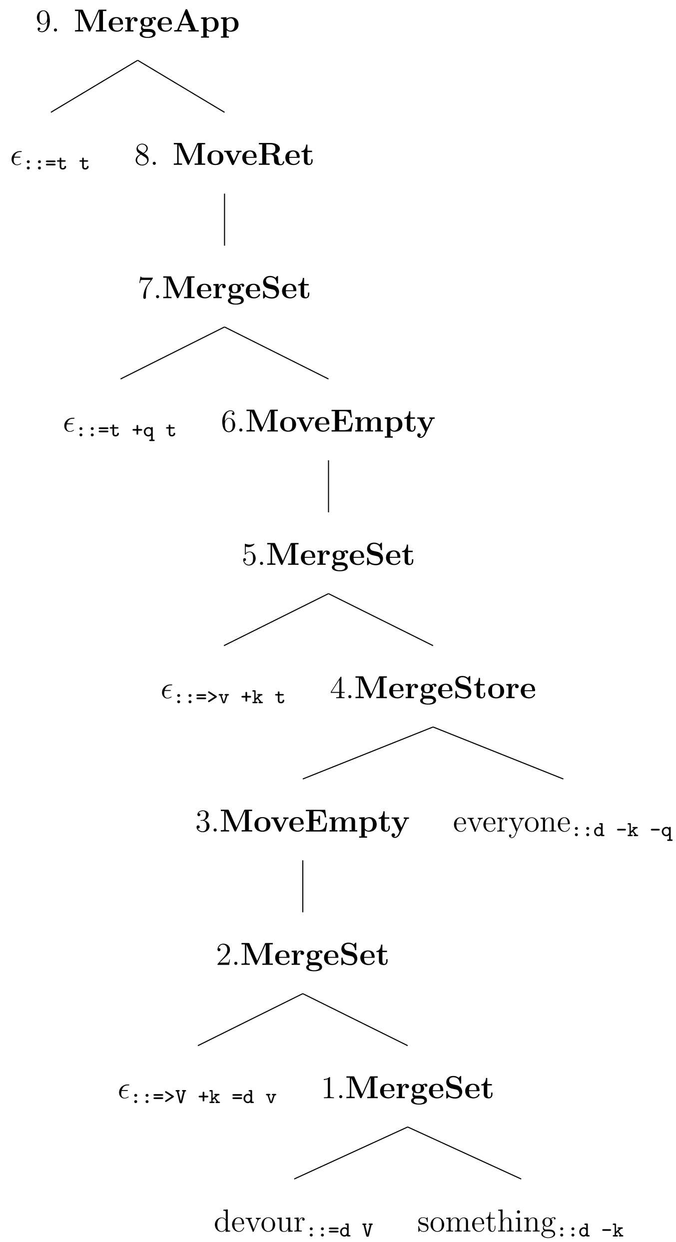


The semantics of the tree in (77) is calculated as follows:

1.	$\lambda x.\mathbf{see}(x,y)$; $\lambda P.WHy[Py]$	Store
2.	$\lambda x.\mathbf{see}(x,y)$; $\lambda P.WHy[Py]$	FA
3.	$\lambda x.\mathbf{see}(x,y)$; $\lambda P.WHy[Py]$	Id
4.	$\mathbf{see}(\boldsymbol{j},y)$; $\lambda P.WHy[Py]$	BA
5.	$\mathbf{see}(\boldsymbol{j},y)$; $\lambda P.WHy[Py]$	FA
6.	$\mathbf{see}(\boldsymbol{j},y)$; $\lambda P.WHy[Py]$	FA
7.	$\mathbf{see}(\boldsymbol{j},y)$; $\lambda P.WHy[Py]$	Id
8.	$WHy[\mathbf{see}(\boldsymbol{j},y)]$	Retrieve

The object wide scope (OWS) and subject wide scope (SWS) of an scopally ambiguous sentence like *everyone devoured something* is shown below.

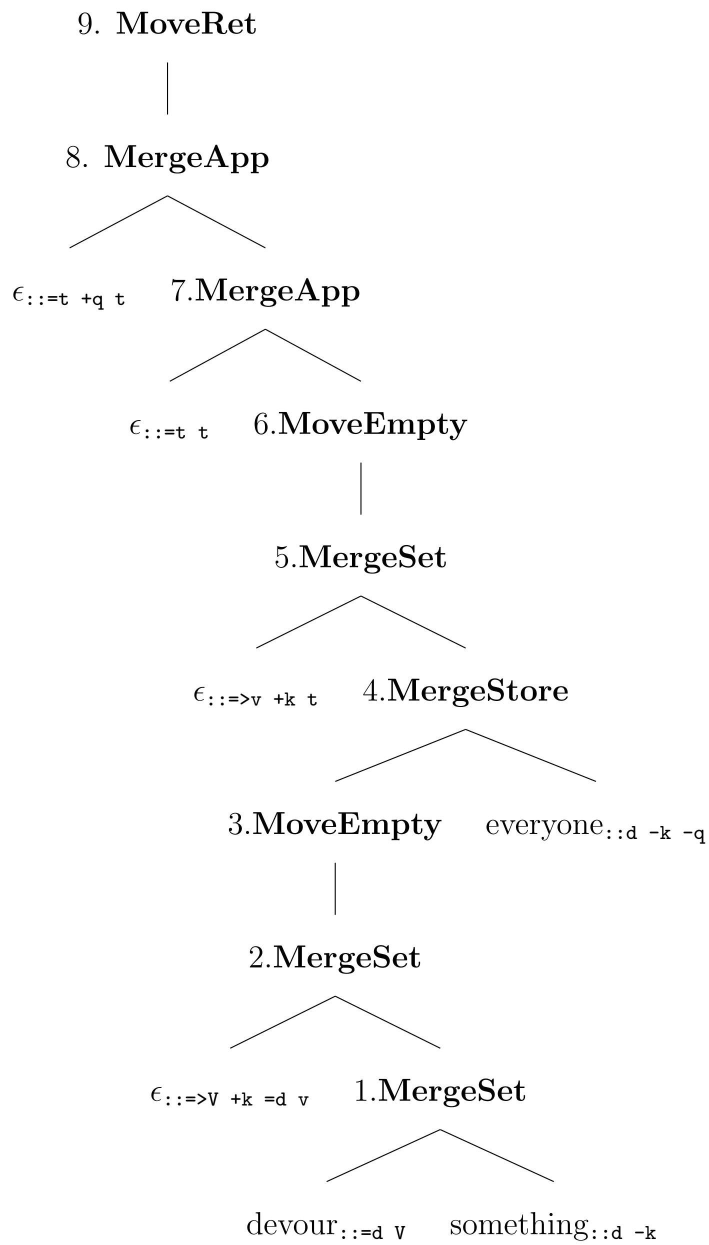
(78) Everyone devoured something. (OWS)



Object Wide Scope

1.	$\{\lambda y.\mathbf{devour}(a,y),\ \lambda y.\mathbf{devour}(b,y)..\}$	Set
2.	$\{\lambda y.\mathbf{devour}(a,y),\ \lambda y.\mathbf{devour}(b,y)..\}$	PFA
3.	$\{\lambda y.\mathbf{devour}(a,y),\ \lambda y.\mathbf{devour}(b,y)..\}$	Id
4.	$\{\mathbf{devour}(a,y),\ \mathbf{devour}(b,y)..\}\ ;\ \lambda Q.\forall x[\mathbf{person}(x) \rightarrow\ Qx]$	Store
5.	$\{\mathbf{devour}(a,y),\ \mathbf{devour}(b,y)..\}\ ;\ \lambda Q.\forall x[\mathbf{person}(x) \rightarrow\ Qx]$	PFA
6.	$\{\mathbf{devour}(a,y),\ \mathbf{devour}(b,y)..\}\ ;\ \lambda Q.\forall x[\mathbf{person}(x) \rightarrow\ Qx]$	Id
7.	$\{\mathbf{devour}(a,y),\ \mathbf{devour}(b,y)..\}\ ;\ \lambda Q.\forall x[\mathbf{person}(x) \rightarrow\ Qx]$	PFA
8.	$\{\forall x[\mathbf{person}(x) \rightarrow \mathbf{devour}(a,x)],\ \forall x[\mathbf{person}(x) \rightarrow \mathbf{devour}(b,x)]...\ \}$	Retrieve
9.	$\bigvee \{\forall x[\mathbf{person}(x) \rightarrow \mathbf{devour}(a,x)],\ \forall x[\mathbf{person}(x) \rightarrow \mathbf{devour}(b,x)]...\ \} \equiv$	
	$\exists y.[\mathbf{thing}(\mathrm{y})] \wedge \forall x[\mathbf{person}(x) \rightarrow \mathbf{devour}(y,x)]$	EC

(79) Everyone devoured something. (SWS)



Subject Wide Scope

1.	$\{\mathbf{devour}(a,y),\ \mathbf{devour}(b,y)..\}$	Set
2.	$\{\mathbf{devour}(a,y),\ \mathbf{devour}(b,y)..\}$	PFA
3.	$\{\mathbf{devour}(a,y),\ \mathbf{devour}(b,y)..\}$	Id
4.	$\{\mathbf{devour}(a,y),\ \mathbf{devour}(b,y)..\}\ ;\ \lambda Q.\forall x[\mathbf{person}(x) \rightarrow\ Qx]$	Store
5.	$\{\mathbf{devour}(a,y),\ \mathbf{devour}(b,y)..\}\ ;\ \lambda Q.\forall x[\mathbf{person}(x) \rightarrow\ Qx]$	PFA
6.	$\{\mathbf{devour}(a,y),\ \mathbf{devour}(b,y)..\}\ ;\ \lambda Q.\forall x[\mathbf{person}(x) \rightarrow\ Qx]$	Id
7.	$\bigcup\{\mathbf{devour}(a,y),\ \mathbf{devour}(b,y)..\} \equiv \exists x.\mathbf{devour}(x,y)$	
	$;\ \lambda Q.\forall x[\mathbf{person}(x) \rightarrow\ Qx]$	FA
8.	$\exists x.[\mathbf{thing} \wedge \mathbf{devour}(x,y)]\ ;\ \lambda Q.\forall x[\mathbf{person}(x) \rightarrow\ Qx]$	FA
9.	$\forall y[\mathbf{person}(y) \rightarrow \exists x\ [\mathbf{thing}(x)\wedge]\ \mathbf{devour}(x,y)]$	Retrieve

CHAPTER 4

Identity Condition

Before introducing the operations needed for forming sluiced sentences, I aim to spell out the identity condition needed for a no-movement analysis. Any discussion of ellipsis requires a proper formulation of the identity relations between elided expressions and their antecedents. Consider the VP-ellipsis example in (80):

(80) John can [play the violin], but Mary can't _.

a. but Mary can't [~~play the violin~~]

b. but Mary can't *[~~dance~~]

It is clear that the elided VP in the second conjunct has the interpretation of (80a), where the elided VP is in some sense identical (yet to be defined) with the VP in the antecedent, and not (80b). Within the sluicing literature, the identity conditions that have been proposed can be divided into three main categories: (i) syntactic identity, where antecedent and ellipsis site have the same syntactic structure (ii) semantic identity, where the two have to mean the same thing (iii) the 'hybrid' approach, where some degree of syntactic and semantic identity is taken into consideration. Under a no movement approach to sluicing, I argue that the hybrid approach to identity is needed as neither syntactic nor semantic identity alone is sufficient. In the following sections, I discuss syntactic and semantic identity and show why they are insufficient. Then I discuss the hybrid identity condition and present a version of it for a no-movement analysis under MG.

4.1 Syntactic Identity

Most formulations of syntactic identity would state hat the ellipsis site must be syntactically identical to its antecedent. But what does it mean to be "syntactically identical"? It has long been known that form mismatches exist for different kinds of ellipsis. These form mismatches include verbal morphology (81a) and pronouns (81b) as shown in the examples of VP ellipsis in (81).

(81) VPE mismatches

a. John [$_{VP}$ **slept**] and Mary will [$_{VP}$~~**sleep**~~] too.

b. A: Has John [$_{VP}$ sent **you** his book]?

B: Yes, he has [$_{VP}$ ~~sent **me** his book~~].

Similar inflectional mismatches are found in sluicing as shown in (82).

(82) Sluicing mismatches

a. **Decorating** for the holidays is fun, if you know how ~~**to decorate** for the holidays~~ (Merchant 2001, ex 30)

b. I remember **meeting** him, but I don't remember when ~~I **met** him~~. (Merchant 2001, ex 33)

Given the existence of form mismatches like (81) and (82), i if we want to pursue syntactic identity, it is reasonable to ask how large the difference between ellipsis site and antecedent can be. According to Rooth (1992), Fiengo & May (1994), and Chung et al. (1995), the difference is marginal and the form mismatches only differ in the realization of inflectional morphology. Rooth (1992) made the claim that some part of the antecedent must be "syntactically reconstructed" into the ellipsis site to capture strict and sloppy readings under ellipsis. Chung et al.'s (1995) LF copying analysis for sluicing was basically designed to capture Rooth's claim. Fiengo & May (1994) made the strictest claim to syntactic identity, where there must be lexical and also structural identity. Hence, the antecedent and ellipsis site must be composed of the same lexical expressions that are organized in the same way. Fiengo & May's (1994) notion of identity is satisfied in simple cases of VPE like (80), however, the difficulty of extending the same notion to sluicing is dependent on the analysis of sluicing.

4.1.1 No *wh*-movement & Syntactic Identity

In a movement plus deletion account of sluicing, the obvious complication in establishing syntactic identity is due to the *wh*-word extraction before deletion of the ellipsis site. There is no overt movement of the indefinite correlate in the antecedent, but there is *wh*-movement from the ellipsis site.

(83) [John met **someone**], but I don't know **who**$_i$ [~~John met t_i~~].

Under a no-movement account of sluicing, syntactic identity is much easier to establish since the antecedent and ellipsis site (identified with [ ] in (84)) are identical, modulo the *wh*-phrase and correlate (for English)[1].

(84) [John met **someone**], but I don't know [~~John met~~ **who**].

Furthermore, I argue that inflectional mismatches are not a problem when *wh*-phrase does not move. Consider the example (82b) but with the *wh*-phrase in base position.

(85) a. I remember **meeting** him, but I don't remember ~~**meeting** him~~ when.

b. I remember meeting him, but I don't remember meeting him on Tuesday.

c. *I remember meeting him, but I don't remember when meeting him.

Moving the *wh*-phrase in (85c) is ungrammatical as the present participial verb phrase, *meeting him*, is incompatible with *wh*-movement. To keep the *wh*-movement account for sluicing, the ellipsis site for (85c) in Merchant (2001) is 'I **met** him', with the inflectional mismatch. However, if we consider the sentence in (85a), where the *wh*-phrase remains in

[1]There are languages where *wh*-indefinites serve as both *wh*-phrases in questions as well as indefinite pronouns. (See Cheng et al. (1994) for Chinese and AnderBois (2012) for Yucatec Maya)

base position, we can keep the present participial verb phrase and there is not inflectional mismatch. This is supported by the non-question sentence in (85b), where 'on Tuesday' correspond to 'when'.

It is worth noting that the form mismatches that have been used in the literature to argue in favour of alternative identity conditions, as seen in (81) and (82), rely on the assumption that syntactic rules inside the ellipsis site are not in any way different. One might question the accuracy of this assumption since it could be possible that there are special properties within the ellipsis site. I have no plans of completely dismissing this assumption, as it has been proven useful for investigating ellipsis and has ultimately brought forth the isomorphism argument. However, it is noted that a no-movement account of sluicing does not comply with this assumption in certain languages. In English, *wh*-phrases are not allowed to stay in situ[2], as shown in (86). If we follow the assumption, *wh*-phrases in sluiced sentence have to move as well in accordance to it's usual behavior in a non-sluiced sentence.

(86) *John met **someone**, but I don't know John met **who**.

To counter the argument, I would like to point out that a violation such as in (86) is no different from the violation that would occur under a movement plus deletion analysis when it comes to pronouncing island constraints.

(87) *John believe that claim that Mary saw **someone**, but I don't know **who**$_i$ John believe that claim that Mary saw t_i.

Thus, to maintain syntactic isomorphism as well as the no-movement account of sluicing, I assume that the set of structures that obeys the usual syntactic rules is a subset of the set of elidable structures.

[2]The only two exceptions are echo questions and when the matrix position is already occupied by another *wh*-phrase.

4.1.2 Advantages of Syntactic Identity

The advantages of having a syntactic identity with a no movement analysis include accounting for Merchant's (2001) case-matching generalization as well as preposition generalization deviant languages like Malay. Furthermore, such an analysis does not rule out preposition generalization compliant languages, although the phenomenon itself requires a different explanation. Besides the case-matching generalization, syntactic isomorphism has also been used to explain the fixed diathesis effects (Chung et al. 2011), where the interpretation of the elided structure is affected by the lexical choices made in the antecedent.

(88) Fixed diathesis effect: 'load' alternation

a. She loaded the truck with the hay. (goal, theme)

b. She loaded hay onto the truck. (theme, goal)

c. *She loaded something with hay, but I dont know onto what ~~she loaded hay~~.

d. *She loaded something onto the truck, but I don't know with what ~~she loaded the truck~~ (Barros 2014, ex 1.5)

In (88), the verb 'load' has to take its arguments in the same order as the antecedent. (88c) is ungrammatical because 'load' in the antecedent takes the goal before the theme like in (88a), but 'load' in the ellipsis site takes the theme before the goal like in (88b). The impossibility of (88c) and (88d) follows from the lack of syntactic isomorphism between antecedent and ellipsis site. A related argument in favour of syntactic identity comes from the unavailability of voice mismatches under sluicing as shown in (89).

(89) *Someone hired John, but I don't know by whom ~~John was hired~~.

Merchant (2013) has argued that examples like (89) pose a challenge to theories of ellipsis that only posit semantic identity (more about this in the next section). He proposes a syntactic identity condition where elided Voice head must bear the same feature (active or passive) as its antecedent. However, if we assume that active and passive have different structures, the ungrammaticality of (89) falls out from the lack of syntactic isomorphism as well (see Chung (2013) for further arguments).

4.1.3 A Problem with Syntactic Identity

Though syntactic identity has been shown to be advantageous and rather intuitive, it is not sufficient to explain sluicing data involving scope parallelism between correlate and non-moving *wh*-remnant. Consider the example in (90).

(90) a. If a relative of John died, he would be upset, but I don't know which relative.

b. If any relative of John died, he would be upset, *[but I don't know which relative].

In (90a), the indefinite 'a relative of John' takes wide scope out of the *if*-clause, which is an adjunct island. Adding a sluice construction after it is fine. However, just changing the correlate into the NPI 'any' (another indefinite), as seen in (90b), renders the sluice construction ungrammatical. As shown in (91), the NPI 'any' can be a correlate to a sluiced construction.

(91) They usually ask whether the candidate reviewed any papers for the journal, but they never ask which ones. (Romero 1998)

Given (91), the ungrammaticality of (90b) is not the result of 'any' not being able to serve as a correlate to a *wh*-remnant, but rather the inability of 'any' to scope out of the island. This creates a scope mismatch between the antecedent and the ellipsis site, as the *wh*-remnant is capable of taking scope out of syntactic islands. Romero (1998) and Johnson (2001) have argued for the need of scope parallelism to be a condition for sluicing. Scope parallelism will be further discussed in the following section. In any case, a purely syntactic identity for sluicing would not be able to account for mismatches that involve scope as it does not say anything about meanings. Hence, syntactic identity alone is insufficient for sluicing.

4.2 Semantic Identity

The preference for a semantic identity for ellipsis stems from the inability to reconcile with puzzles and mismatches introduced by syntactic identity as discussed in the section above. The general idea of semantic identity is that ellipsis site and antecedent must match in meaning, which allows for structural and lexical mismatches as long as both components encode the same meaning. Since there are more representations of meaning than syntactic structure in the literature, a semantic condition (depending on the theory of ellipsis) can be stated in numerous ways. Nonetheless, theories of ellipsis with semantic identity can be roughly divided into two categories: (i) meaning computed through structure in the ellipsis site (Rooth 1992, Romero 1998, Merchant 2001) (ii) meaning of ellipsis site anaphoric to some part of the meaning in the antecedent (Ginzburg & Sag 2000, AnderBois 2011, Barker 2013, Barros 2014, Weir 2014). In the first category, the syntactic structure in the ellipsis site can be identical to the antecedent or a reduced form of the antecedent, but never be larger than the antecedent. Meaning is computed from the structure that will, at some point in the derivation, be depleted of phonetic material. In the second category, the ellipsis site can have syntactic material or be completely structureless. Given that the only condition for semantic identity is identical meaning, postulating an ellipsis site devoid of any syntactic structure is possible (see Barker (2013)), as long as there is a mechanism to provide meaning for it.

The no movement account which I am proposing has structure in the ellipsis site and I argue that meaning is built up from both structures. A semantic identity condition would be simple to establish in a no movement analysis. In this case, the antecedent and the ellipsis simply have to mean the same thing to allow for ellipsis to occur. As noted in the section above, syntactic identity is not sufficient to account for the scope parallelism data seen in (90). With a semantic identity condition, scope parallelism is a given since the meaning of both antecedent and ellipsis site are the same. However, a purely semantic identity condition is also insufficient, as discussed in the section below.

4.2.1 Problem with Semantic Identity

It is clear that an identity condition built from only semantics is not capable of explaining any syntactic generalizations. Merchant's (2001) case matching and preposition stranding generalization are usually taken to be evidence for some syntactic isomorphism. Under a purely semantic account, these generalizations would have to be independently bolted on somehow. Another example of a syntactic generalization that would be left unexplained is the fixed diathesis and the unavailability of voice mismatches as discussed in the section before.

(88) Fixed diathesis effect: 'load' alternation

a. She loaded the truck with the hay. (goal, theme)

b. She loaded hay onto the truck. (theme, goal)

c. *She loaded something with hay, but I dont know onto what ~~she loaded hay~~.

d. *She loaded something onto the truck, but I don't know with what ~~she loaded the truck~~ (Barros 2014, ex 1.5)

(92) Passive antecedent

a. John was murdered, but I don't know by whom.

b. *John was murdered, but I don't know who.

(93) Active antecedent

a. Someone murdered John, but I don't know who.

b. *Someone murdered John, but I don't know by whom.

These mismatches are not predicted by analysis with a purely semantic identity since the antecedent and ellipsis site mean the same thing.

4.3 Hybrid Identity

As shown in the sections above, neither a purely semantic nor syntactic approach is sufficient to capture the empirical picture, even if we assume a no-movement analysis. There is a growing consensus in the literature that identity for sluicing should be 'hybrid' in nature, where there is a localized syntactic condition and some version of the semantic identity condition (Merchant 2005, Chung 2006, 2013, Barros 2014, Weir 2014). The analysis of sluicing in this dissertation also incorporates a hybrid proposal, but one with a much stricter syntactic condition. I will briefly discuss the hybrid condition proposed by Chung (2013) and Barros (2014) before presenting my proposal.

4.3.1 Chung (2013) Limited Syntactic Identity

The arguments in Chung (2013) are based on the constraints of sluicing in Chamorro and English, which she attributes to mismatches in argument structure or case-assignment configuration. Chung's limited syntactic identity is presented below.

(94) Limited Syntactic Identity in Sluicing (Basic Idea)

The interrogative phrase of the sluice must be integrated into a substructure of the syntax in the ellipsis site that is identical to the corresponding substructure of the antecedent clause. (Chung 2013), ex 61

(95) Limited Syntactic Identity in Sluicing (Specifics)

a. Argument structure condition: If the interrogative phrase is the argument of a predicate in the ellipsis site, that predicate must have an argument structure identical to that of the corresponding predicate in the antecedent clause.

b. Case condition: If the interrogative phrase is a DP, it must be Case-licensed in the ellipsis site by a head identical to the corresponding head in the antecedent clause. (Chung 2013, ex.64)

The arguments for (95a) stem from the intolerable voice mismatches in sluicing discussed in the section above, with example repeated below.

(92) Passive antecedent

a. John was murdered, but I don't know by whom.

b. *John was murdered, but I don't know who.

Chung (2013) shows that Chamorro has a similar ban on voice mismatch under sluicing and concludes, following Merchant (2013), that the problem is with the different Voice head in the ellipsis site and antecedent. As for (95b), Chung (2013) provides three arguments, two of which come from sprouting cases, and are not discussed here. The third argument comes from a restriction on sluicing with nominative subject remnants when the antecedent is an infinitive.

(96) *Having to compromise is inevitable, but they have no idea who ~~has to compromise~~.

Chung (2013) argues that ellipsis in (96) is not licensed because the *wh*-remnant 'who' is case-marked by a case assigner that is not present in the antecedent, namely finite T.

4.3.2 Barros (2014) "Split" Identity Condition

In accordance with theories of hybrid identity, Barros (2014) proposes that sluicing in general has a "split" identity condition. Sluices are licensed under a very minimal structural identity condition, together with a semantic constraint requiring the sluiced question itself to be anaphoric to the Question under Discussion (QuD) (Roberts 1996) indicated by the linguistic antecedent. Barros's (2014) Split Identity condition is given in (97).

(97) Split Identity

a. The Remnant Condition:
The remnant must have a syntactic correlate, which is a semantically identical XP in the antecedent.

b. The Sluice Condition:
The sluiced question and the QuD made salient by the antecedent must have the same answer at any world of evaluation.

The Remnant Condition in (97a) calls for very minimal syntactic identity and it is also partially semantic in nature. All it says is that the *wh*-remnant in the ellipsis site must have a syntactic constituent that can serve as the correlate in the antecedent. The correlate must also be semantically identical to the *wh*-remnant. Following Karttunen (1977), *wh*-phrases are assumed to be existentially quantified, making the *wh*-remnant and the indefinite in the antecedent semantically identical.

(98) $[\![\text{someone}]\!] = [\![\text{who}]\!] = \lambda P \exists x[\text{person}(X)\ \&\ P(x)]$

While the Remnant Condition makes reference to the content of the remnant and its correlate, the Sluice Condition in (97b) serves as the semantic identity condition. According to (97b), sluicing is only possible when the question denoted by the sluice in the ellipsis site is equivalent to an implicit QuD (or *issues* in Inquisitive Semantics) raised by the antecedent. A simplified example of the QuD framework is illustrated below.

(99) Someone left, but I don't know who [~~left~~].

The antecedent in (99), 'someone left' is taken to make salient a question paraphrasable to something like *who left*. The ellipsis site will yield the same QuD, which satisfies the Sluiced Condition in (97b). The semantic identity is similar to the AnderBois (2011) IS approach. The rest of this section will be devoted to the Remnant Condition.

The Remnant Condition was proposed to capture the data motivating Chung's (2006) Generalization as well as the ban on diathesis alternations (eg. impossibility of active-passive mismatch under sluicing). Chung (2006) observes that prepositions cannot be stranded under sprouting, even in languages that usually permit preposition stranding. Chung's generalization is given and illustrated in the example below.

(100) Chungs (2006) Generalization (No New Words): The numeration of the sluice must be a subset of the numeration of the antecedent.

(101) a. They're jealous but it's unclear of who/who of.

b. *They're jealous but it's unclear who.

c. They're jealous but it's unclear who they're jealous of.

(101a) shows a case of sprouting with the preposition attached to the *wh*-remnant. (101b) shows that preposition stranding in the ellipsis site is not possible, even though preposition stranding without ellipsis is unproblematic, as shown in (101c). To support the generalization in (100), Chung (2006) argues that preposition stranding is unavailable in the ellipsis site in (101b) because the prepositional phrase lacks a twin in the antecedent. On the other hand, Barros (2014) argues that sluicing in (101b) is ruled out by the Remnant Condition through different semantics of the *wh*-remnant and the correlate. Barros (2014) assumes that implicit arguments for cases like (101b), added in () brackets in (102), are syntactically simplex and have a semantics that is distinct from that of a DP.

(102) *They're jealous (*of someone*) but it's unclear who [~~they're jealous of~~].

An implicit argument would have to compose with its containing structure in the same way as its explicit counterpart. Thus, an implicit 'of someone' would have the same semantics as a PP and since $[\![\text{of someone}]\!] \neq [\![\text{who}]\!]$, sluicing is ruled out according to the Remnant Condition. The same explanation is extended to the unavailability of the voicing mismatch case. The implicit argument is added in () brackets in (103).

(103) *John was murdered (*by someone*), but I don't know who [~~murdered John~~].

The implicit argument 'by someone' is semantically different from the *wh*-remnant 'who' which is ruled out by the Remnant Condition. Hence, the Remnant Condition manages to merge two different generalizations into one condition, Chung's (2006) generalization as well as Chung's (2013) Argument structure condition. An advantage of the Remnant Condition is that it does not refer to content of the ellipsis site, thus allows all kinds of detectable mismatches under sluicing, such as island effects. In the next section, I present another hybrid identity approach that is similar in spirit to Barros (2014). However, my proposal is based around a no-movement analysis of sluicing, where the syntactic identity will be much stricter.

4.4 Hybrid Identity for no *wh*-movement

The hybrid proposal by Chung (2013) and Barros (2014) have limited syntactic identity partially due to mismatches in the ellipsis site created by *wh*-movement. But having a no movement analysis of sluicing allows for syntactic identity to be much stricter. To briefly recap the identities that have been proposed in the ellipsis literature: syntactic identity is needed for case connectivity effects as well as the unavailability of active-passive mismatch under sluicing, whilst semantic identity is needed to account for data concerning scope parallelism. The hybrid identity that I propose here retains Barros's (2014) Remnant Condition but adds on the condition that strict syntactic isomorphism is needed as well. I will first spell out the formulation of a hybrid identity before translating it into the Minimalist Grammar framework introduced in the previous chapter. Building upon Barros's Remnant Condition, I add on the Isomorphism Condition which is purely syntactic in nature.

(104) Conditions of Hybrid Identity:

a. The Remnant Condition:
The remnant must have a syntactic correlate, which is a semantically identical XP in the antecedent.

b. The Isomorphism Condition:
The elided material has to be syntactically isomorphic, both lexically and structurally, to the antecedent modulo the correlate

Barros (2014) uses the Remnant Condition to rule out the ban on diathesis alternations, including passive/active voice mismatches. However, he must assume that implicit arguments are semantically different from the *wh*-remnant in cases like (105).

(105) *John was murdered (*by someone*), but I don't know who.

Instead of appealing to the difference in semantics of the implicit argument and the *wh*-remnant, I argue that (105) can be straightforwardly accounted for by the Isomorphism Condition instead. Without ellipsis, there will always be an additional preposition in the second conjunct when it is spelled out. Under ellipsis, the elided string itself will never be the same to the antecedent given this extra preposition. The example in (106) serves to illustrate this with a no *wh*-movement account of sluicing.

(106) *John was murdered, but I don't know [~~John was murdered **by**~~] who.

Hence, the ban on voice mismatches and diathesis alternation can be accounted for under the Isomorphism Condition. Another advantage the Isomorphism Condition has over the Remnant Condition is the ability to account for case matching effects. To recapitulate, Merchant (2001) proposes the case matching generalization where the *wh*-remnant must bear the same morphological case as its correlate, as shown in the German example.

(107) a. Sie wissen {*der Antwort/ die Antwort} nicht
they know {the answer.DAT/ the answer.ACC} not
'They dont know the answer.' (Merchant 2001)

b. Er will jemandem schmeicheln, aber sie wissen nicht, {*wer/
he wants someone.DAT flatter but they know not {who.NOM/
*wen/ wem}
who.ACC/ who.DAT}
'He wants to flatter someone, but they dont know who.' (Ross 1969)

Since the Remnant Condition makes no reference to the structure of the ellipsis site, the case matching effects as seen in (107) would be a mystery under the Remnant Condition. On the other hand, the Isomorphism Condition requires the ellipsis site and the antecedent to have the same structure, thus it is expected for the *wh*-remnant and the correlate to bear the same case.

Though Barros's (2014) original arguments for the Remnant condition has been shifted to the Isomorphism account, I maintain that the Remnant Condition is still needed. I adopt Hamblin-Karttunen alternative semantics view of *wh*-phrases and indefinites where both are existentially quantified DPs that denote sets of alternatives. As in Barros (2014), $[\![\text{someone}]\!] =$

⟦who⟧. The Isomorphism Condition does not make reference to the shape or semantics of the correlate and remnant which I argue is still needed in the case of double negation.

AnderBois (2010) notes that antecedents with doubly negated correlates do not license sluicing and provides his solution through the Inquisitive Semantics framework.

(108) a. *Sally didn't see *no one*, but I don't know who.

b. *It's not the case that no student left, but I don't know which student.

c. *Its not the case that John didn't meet with a student, but Fred still wonders who/which student. (Barros 2014)

On the contrary, I argue that this ban on double negation in sluicing can be easily attributed to the Remnant Condition. The example in (109) shows an in situ sluice in accordance to the syntactic identity given in (104).

(109) *Sally didn't see **no one**, but I don't know [~~Sally didn't see~~ **who**.]

The Isomorphism Condition by itself is not able to rule out (109) if the ellipsis site itself is similar. However, the Remnant Condition rules out double negation on the basis that the remnant 'no one' and the *wh*-remnant 'who' are not semantically equivalent. 'no one' should have the denotation of $\neg\exists$ while 'who' would just be $\exists$, it is obvious that $\neg\exists \neq \exists$.

Though the Remnant and the Isomorphism Conditions solves a number of sluicing puzzles, neither would be able to explain the need for semantic scope parallelism under sluicing. As shown in (90), repeated below, the indefinite 'a relative of John' is able to take wide scope out of the adjunct island, and sluicing is fine. On the other hand, 'any relative of John' which contains the NPI 'any' (also an indefinite) is unable to, and sluicing is problematic. This has been argued to be due to scope parallelism, where 'a relative of John' takes parallel scope as the *wh*-remnant 'which relative', while 'any relative of John' is unable to scope out of the island creating a scope mismatch between remnant and correlate.

(90) (90a) If a relative of John died, he would be upset, but I don't know which relative.

(90b) If any relative of John died, he would be upset, *[but I don't know which relative].

As the NPI 'any' can serve as a correlate to a *wh*-remnant, shown in (91) repeated below, the Remnant Condition cannot rule out (90b) because we can find examples where an 'any'-phrase can serve as correlate to a *wh*-remnant under sluicing. In (91), 'any papers' has to be semantically identical to the *wh*-remnant 'which ones' for the sentence to be grammatical.

(91) They usually ask whether the candidate reviewed any papers for the journal, but they never ask which ones. (Romero 1998)

The Isomorphism Condition fails to rule (90b) out as well as there is no reason to think that (90a) has different elided material from (90b). The main reason why (90b) is ungrammatical is because the 'any relative', though considered an indefinite as well, has limited scopal properties as compared to the indefinites like 'a relative'. Thus, the only option left is to have a condition that differentiates between the scopal properties indefinites like 'a relative of John' and the NPI 'any relative of John'. The Remnant Condition forces a syntactic correlate to mean the same thing as the *wh*-remnant, while the Isomorphism Condition forces the antecedent (besides the correlate) and ellipsis site to be the same. Now, we need a condition to force the same scopal position between the antecedent and the ellipsis site. I propose to add the following semantic condition (110c) to (104).

(110) Conditions of Hybrid Identity:

a. The Remnant Condition:
   The remnant must have a syntactic correlate, which is a semantically identical XP in the antecedent.

b. The Isomorphism Condition:
   The elided material has to be syntactically isomorphic, both lexically and structurally, to the antecedent modulo the correlate

c. The Scope Parallelism Condition:
   The correlate and the remnant must have the same scopal properties

ALthough the Remnant Condition say that the remnant and correlate must have the same denotation, the semantic operations that work between remnant and correlate may still be different, yielding different scopal properties. As a result, we have cases like (91) where the NPI 'any' can serve as a correlate but cannot escape islands such as in (90b). The Scope Parallelism Condition would allow for cases (91) like in that example, the correlate and remnant takes scope at the same height. In (91), one can stipulate that the different semantic operations between correlate and remnant yield the same scope properties. The condition rules out (90b) as well, given that the correlate has been limited by its semantic operation and is unable to take the same scope as the remnant. In the next section, the conditions in (110) are cast within the MG system presented in the previous chapter.

4.4.1 Hybrid Identity in MG

Although the MG framework is based on a traditional Chomskyan framework, the crucial difference between the two is that in an MG, the syntax and semantics are computed at the same and are both represented in the derivation tree. In the traditional framework, syntax is usually computed first and then the structure is sent off for further computation in the semantics. The conditions given in (110) are based on the traditional framework. This section translates the conditions in (110) in order to apply to an MG derivation tree. Within the MG framework, the three conditions in (110) can be collapsed into one, which I call the Derivational Isomorphism Condition.

4.4.1.1 Derivational Isomorphism Condition

The Remnant Condition states that the *wh*-remnant must have a syntactic counterpart in the antecedent that means the same thing. Thus, the indefinite in the antecedent that correlates with the *wh*-remnant must have the same denotation as the *wh*-remnant. As established in Chapter 3, indefinites in MG are existentially quantified and denoted as a set of alternatives, e.g. $[\![\text{someone}]\!] = \{x|\text{person}(x)\} = \lambda P \exists x[\text{person}(X)\ \&\ P(x)]$. Barros (2014) following Karttunen (1977) assumes that *wh*-phrases are existentially quantified as well, which makes them semantically identical to indefinites once the Hamblin-Karttunen alternative semantics view of *wh*-phrases, $[\![\text{someone}]\!] = [\![\text{who}]\!] = \{x|\text{person}(x)\} = \lambda P \exists x[\text{person}(X)\ \&\ P(x)]$.

Adopting the Hamblin-Karttunen alternative semantics view of *wh*-phrases and indefinites seems easy, aside from a single issue. While Barros (2014) does not specify the nature of the *wh*-phrase that is the remnant, it is assumed that the *wh*-remnant in his work follows Merchant (2001) in that it has been moved. In Chapter 3, I did not make any distinction between the denotation of a moved *wh*-phrase and *wh*-in situ. This will be discussed in Chaper 6. For now, I argue that *wh*-remnants in sluicing, which are similar to a *wh*-phrase in situ, must be denoted as a set of alternatives while leaving the denotation of a normal moved *wh*-phrase unspecified.

Evidence that supports *wh*-in situ being more like indefinites as opposed to a regular moved *wh*-item stems from exceptional scope. As mentioned in Chapter 3, indefinites are known to have exceptional scope, allowing them to be interpreted outside of islands. A regular *wh*-moved phrase does not have this ability, while *wh*-in situ are found to demonstrate exceptional scope like indefinites (Reinhart 1998, Huang 1998, Nishigauchi 2012, Dayal 2012). While *wh*-phrases normally move in English, in multiple *wh*-questions where there are more than one *wh*-phrase, one of them stays in situ. The *wh*-in situ phrase in multiple *wh*-questions can be interpreted outside of islands as seen in (111).

(111) a. Who will be offended [if we invite which philosopher]?

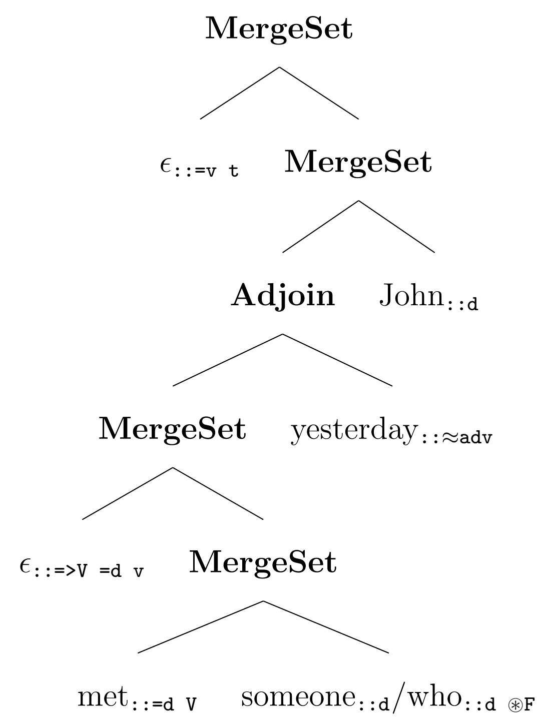
b. for which $\langle x, y \rangle$, y is a philosopher, and if we invite y, x will be offended (Reinhart 1998)

(111) shows the *wh*-phrase in situ 'which philosopher' taking scope out of the island t indicated by [ ]. This suggests that a *wh*-phrase forced to be in situ has different properties to a moved *wh*-phrase, even in English. Since this property of exceptional scope is shared with indefinites, it is not a surprise to claim that *wh*-in situ (as opposed to a moved *wh*-phrase)

has the same denotation as an indefinite. This ability is then extended to *wh*-remnants since I argue that they are similar to *wh*-phrases in situ.

The Isomorphism Condition in (110b) is fundamentally strict syntactic identity, and it says that elided material must look exactly the same as its antecedent besides the correlate. In MG, there are two structures we can choose to compare, the derived structure or the derivation structure. Given that this is syntactic identity, putting the condition on either structure would be functionally equivalent. I will be setting this condition on the derivational structure as it has more information and would allow me to combine all the three conditions into one. To get syntactic isomorphism in MG, we must simply ensure that the order of *merge* and *move* of the corresponding lexical items between antecedent and ellipsis site are the same. Thus, in a sentence such as *John met someone yesterday, bidk who*, as long as the derivation trees of the antecedent and ellipsis site look the same, modulo remnant and correlate, as in (112)[3], ellipsis is licensed.

(112)



[3]The ⊛F feature on *wh*-remnant will be discussed in the next chapter.

Scope parallelism is part of the semantic identity of this hybrid identity. The condition requires the correlate and the remnant to take equivalent scope in their domain. The semantics system as set up in Chapter 3 also requires that they have parallel derivation, illustrated in (112).

The derivation tree in (112) constructs the same meaning for both antecedent and ellipsis site since the remnant and correlate have the same meaning. This information is then used by the ellipsis operation to determine whether ellipsis can occur. Given that a derivation tree yields both syntactic structure and the semantics of a sentence simultaneously, all three conditions can be condensed into my proposed MG Derivational Isomorphism Condition. The derivation subtrees of antecedent and ellipsis site will be the same as long as the remnant and correlate have the same meaning and the same lexical items are combined to form the rest of the antecedent and ellipsis site. Thus, all three conditions can be stated as one in (113).

(113) Condition of a Hybrid Identity in MG:

The Derivational Isomorphism Condition:

The elided material and the antecedent must have the same derivation, including all lexical entries modulo the correlate and *wh*-remnant.

In the next chapter, I discuss the Elide operation and how Elide make use of the Derivational Isomorphism Condition established here to license the ellipsis operation.

CHAPTER 5

Elide

In the previous chapter, I established the identity condition needed for the ellipsis operation that I call Elide. The Derivational Isomorphism Condition makes sure that meaning and the derived structure, aside from the *wh*-remnant and the correlate, are the same before allowing ellipsis to happen. The main aim of this chapter is to explain how the Elide operation works on the classic case of sluicing (e.g. John met someone yesterday, BIDK who). First, I introduce a new feature for *wh*-remnants, $\circledast$`F`, and show how it fits into the MG framework. This new feature is needed for the implementation of a no movement analysis that would allow for seemingly non-constituent deletion. Next, I discuss the link between *wh*-remnants that have the $\circledast$`F` and normal *wh*-phrases in questions that have the `wh` feature. I argue that they should not be treated as two distinct items since they have a few similarities and I show how they relate to one another. Next, I explain some changes that need to be made in the MG system introduced in the previous chapter to allow for Elide to work within a no movement analysis. The Elide operation is broken into three parts: the first part checks the identity condition and determines whether ellipsis is allowed; the second part is the way in which ellipsis itself occurs while the *wh*-remnant does not syntactically move; the third part ties loose ends together to ensure we get the right structure and meaning. Elide is an implementation of the ellipsis process only, so certain parts of sentences like, 'BIDK', will not be discussed.

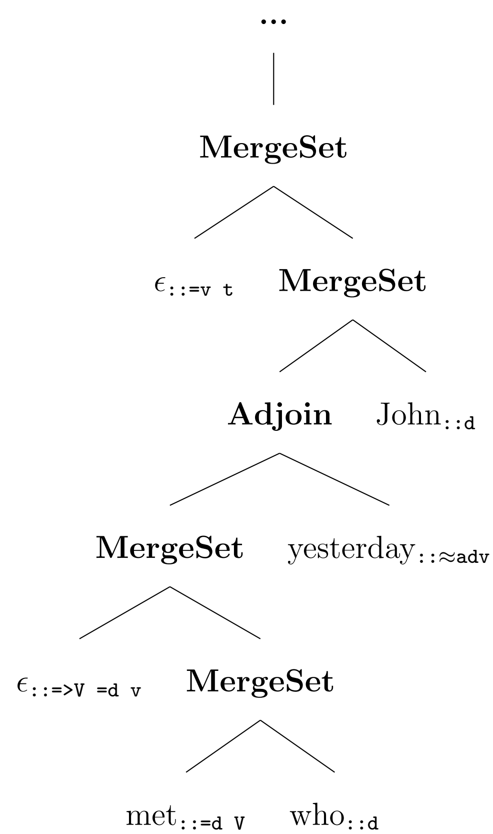
The second half of this chapter compares this implementation of sluicing in MG with the implementation in Kobele (2015). One of the differences between the two approaches is the location of the Elide operation. In my implementation, Elide is located at the very end of the derivation while in Kobele (2015), Elide (or **e** in the paper) is located on the ellipsis site itself. This is because the identity condition as established relies on both syntactic and semantic identity. This forces Elide to be at the root of the derivation tree because it needs information from the antecedent for the identity condition. The Elide in Kobele (2015) only relies on semantic identity and can be situated in the ellipsis site as it does not rely on syntactic information from the antecedent. I show that Kobele's copying approach is unable to explain certain sluicing data and that these data can be handled if structure from the antecedent is an input of Elide. The final section of this chapter discusses derivational complexity and the effect of adding Elide into MG.

5.1 Syntax and Semantics of *wh*-phrases with ⊛F

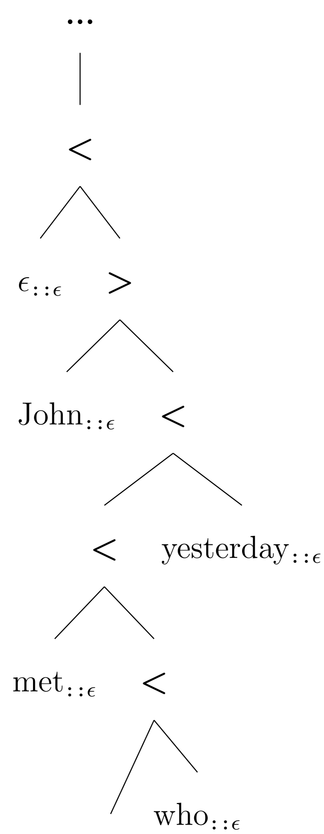
The no movement analysis that this dissertation pursues require some changes to the MG system introduced in the previous chapter. *wh*-remnants need to be defined a different syntax and semantics in order to differentiate them from *wh*-phrases that undergo movement. On first glance, the change to syntax that allows for *wh*-phrases to not move seems easy. Since the `wh` licensing feature is what allows for overt movement, the most obvious solution is to not have the feature on *wh*-phrases that do not move. Removing the *wh*- feature from a *wh*-phrases causes it to immediately become part of the derived structure after all of its features have been checked. This is exemplified in the derivation tree[1] in (114) and derived structure in (115) of the sluiced site of the sentence in (114).

(114) John met someone yesterday, but I don't know [~~John met~~ who ~~yesterday~~].

[1]The **Adjoin** operation is not defined in this dissertation, but is used in the MG literature for adjunction. See Frey & Gärtner (2002) for details on **Adjoin.**


...
MergeSet
ϵ::=v t
MergeSet
Adjoin
John::d
MergeSet
yesterday::≈adv
ϵ::=>V =d v
MergeSet
met::=d V
who::d

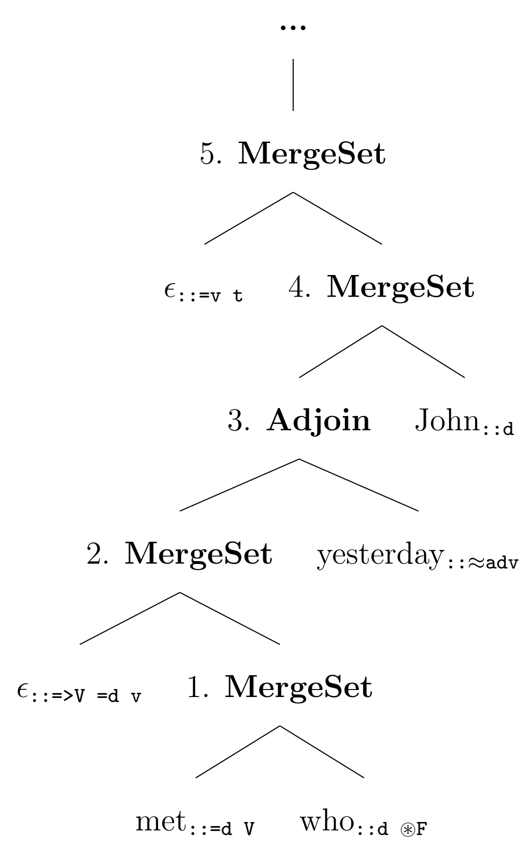
(115)



Though this fix of getting rid of the `-wh` feature might seem simple, the solution in (114) is problematic as the derived structure in (115) can no longer be modified. The *wh*-word cannot be moved beyond this point and ellipsis would not be allowed to take place if the derivation proceeds as in (114). For reasons that will be become clear when the Elide operation is explained further below, the *wh*-remnant cannot have just the `d` feature. It would still need something that holds it out like a moving *wh*-phrase with `-wh` but at the same time do not appear in a c-commanding position where a usual moving *wh*-phrase ends up. The paradox can be stated as follows: the phonological contribution of the *wh*-phrase needs to be held out (like a *wh*-phrase about to move to a higher position), but at the same time still avoid island violations.

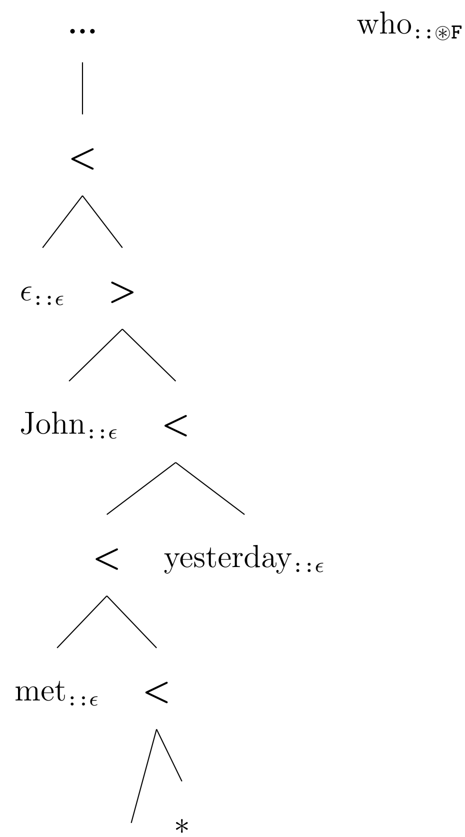
The solution that I am proposing is to add a new kind of feature into the MG system. This feature is neither a selecting nor licensing feature, but it indicates that an item is marked to be held out as well as leaves a mark where the marked item should have been. This feature is indicated by $\circledast$F[2]. Now, a *wh*-remnant has the features d, $\circledast$F. The derivation for the sluice site of (114) is almost the same as (114) besides the additional feature added to the *wh*-phrase. The derived tree in (117), on the other hand, does not have the *wh*-remnant combining into the derived structure despite having its selectional feature checked. The only indication of a *wh*-phrase in a position is the $*$ marking where the *wh*-remnant would have been. The *wh*-remnant marked by the $\circledast$F keeps it from merging into the derived structure.

(116)



[2]So far, there is no other $\circledast$F at the moment. There is a possiblity that there might be different $\circledast$F features for other uses but this option will not be explored in this dissertation.

(117)



The *wh*-phrase held out by by the ⊛F feature will eventually placed into the right position after ellipsis occurs. This will be further explained in the section below. Holding out and reattaching the *wh*-phrase later makes the *wh*-phrase both move and stay in place, in the sense that it is held out like a normal *wh*-moved phrase, but does not move to a higher c-commanding position in the derived tree. A consequence of this is that it does not violate syntactic islands[3], in the same way that a moved *wh*-phrase does.

The semantics of the *wh*-phrase that did not move is essentially identical to indefinites from the previous section. In this system, ⟦*wh*-phrases with ⊛F⟧≡⟦indefinites⟧. Thus, *wh*-phrases with a ⊛F feature also denote a set of alternatives that percolates up the tree until it reaches the existential operator that closes off the set and gives the existential interpretation. The

[3]In Chapter 6, I show an implementation of the Subjacency Condition that prevents extraction out of syntactic islands in MG. Then I show that the implementation of *wh*-phrases with the ⊛F feature does not violate island conditions while normal *wh*-phrases with `-wh` do.

semantics of (116) is calculated as follows:

1.	$\{\lambda x.met(x,a),\ \lambda x.met(x,b)...\}$, ϵ; ϵ
2.	$\{\lambda x.met(x,a),\lambda x.\ met(x,b)...\}$, ϵ; ϵ
3.	$\{\lambda x.met(x,a),\ \lambda x.met(x,b)...\}$, ϵ; ϵ
4.	$\{met(j,a),\ met(j,b)...\}$, ϵ; ϵ
5.	$\{met(j,a),\ met(j,b)...\}$, ϵ; ϵ
	...
	$\bigvee\{met(j,a),\ met(j,b)...\}$, ϵ; $\epsilon \equiv \exists x.[met(j,x)]$

Table 5.1: Semantics of derivation tree in(116)

The reason for having identical semantics between indefinites and *wh*-phrases is due to the identity condition established in the previous chapter. So far, I have been treating wh-phrases with the ⊛`F` feature and wh-phrases with the `wh` feature as two distinct items. In the next section, I show how they can be linked.

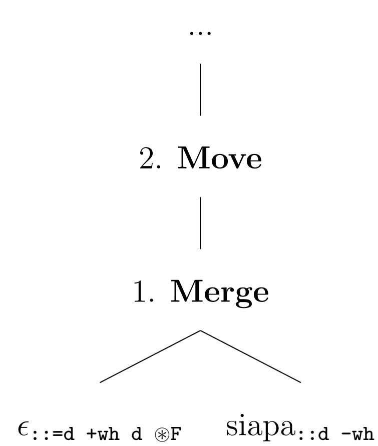
5.2 Link between -`wh` and ⊛`F`

In most theories of sluicing, there is usually a link between *wh*-phrases in normal *wh*-questions and *wh*-phrases (remnants) under sluicing. Either they are the same object (Ross 1969, Merchant 2001 etc.) or the *wh*-remnant in sluicing is an in situ copy of the moved *wh*-phrase (Abe 2015). In Kobele (2015), there is no distinction between a *wh*-phrase in a normal question and a *wh*-phrase under sluicing. However, in this dissertation, I make a distinction between the two, with normal *wh*-phrases as having the `wh` feature, while *wh*-remnants under sluicing have the ⊛`F` feature. Elide, as we will see in the coming section, can only operate on ⊛`F` and not `wh`.

Until this point, I have been treating *wh*-remnants and *wh*-words in questions as two distinct lexical items, but there should be some link between the two, given that they have the same shape and meaning. If there were no relationship between the `wh` and ⊛`F` features, the fact that Malay allows certain things under sluicing (e.g. p-stranding) but not under *wh*-movement would not be surprising to anyone. If the two features were treated differently, accounting for the differences between sluicing and *wh*-movement would be too easy and would not make any further predictions. Thus, for it to be at all interesting that sluicing and *wh*-movement show different properties, we must assume that the two constructions have at least something in common. In this section, I address the relationship I assume between the two features and how it will be implemented.

I assume that all *wh*-phrases start off with the `wh` feature and that this feature can be converted to `⊛F` at some point during the derivation. This order is motivated by p-stranding and pied-piping under Malay sluicing. The way this conversion[4] works is by using an empty lexical item (the converter) with a particular feature list. The converter has the feature list `=x +wh x ⊛F`, where `x` can be of any category. The converter essentially takes something that originally had the `-wh` feature, and "converts" it into the `⊛F` feature when it merges into the derivation. The derivation tree for the conversion of *siapa* (who) into a *wh*-remnant for sluicing is as follows:

(118)



At step 1 in (118), the converter merges into the derivation. In step 2, the `-wh` feature in the *wh*-phrase is checked and the *wh*-phrase is converted into a remnant with the `⊛F`

[4]The way this conversion works is similar to how percolation of the `-wh` feature in Kobele (2015) works. The percolator in Kobele (2015) has the feature list `=p +wh p -wh`, which brings the `-wh` feature further up the tree. This will be further discussed in the p-stranding section of chapter 6.

feature. With this conversion mechanism, a link between *wh*-remnant and regular *wh*-phrases in questions is established. This will be further examined in the p-stranding and pied-piping under sluicing in Malay section. The conversion from `-wh` to ⊛`F` reduces the stipulation on types of *wh*-phrases as it is one lexical entry that gets converted instead of two separate lexical entries. This conversion of `-wh` to ⊛`F` only happens under sluicing as only the Elide function can operate on the ⊛`F` feature. The semantics of this conversion is uncertain for now and is left for future work[5]. In the derivation trees henceforth, I will be assuming the structure in (118) for *wh*-remnants and this structure will be represented with a triangle.

5.3 Resetting MG for Elide

Before describing how Elide checks the identity condition and creates a sluiced sentence, the MG system given in Chapter 3 needs to be changed slightly in order to allow for Elide as an operation. In the part of Chapter 3 that explains the syntax of an MG, *merge* and *move* are defined over *Expr* and create a structure based on the first item on the feature list in an *Expr*. *Expr* can be thought of as a tuple consisting of a derived structure and a list of features. Given that the grammar generates a set of structure-meaning pairs, *Expr* is technically a triple with meaning (operations in Chapter 3) included, (***Expr*** $\subseteq$ derived structure$\times F^* \times M)^*$. However, Elide is not feature driven like *merge* and *move*.In this case, Elide operates on the derived structure of the *Expr*, without features. I call this derived structure *contatenation tree* (*Contree*)[6]. Thus, ***Expr*** $\subseteq (\mathit{Contree} \times F^* \times M)^*$.

Definition 5.3.1. Let ***concatenation tree*** (*Contree*) be the set defined by

$$\text{Contree} = [\text{NonLeaf}]\ \text{Contree Contree} \mid [\text{Leaf}]\ \Sigma^*$$

Contree is obviously part of *Expr* that does not include features. Since the identity

[5]In principle, the converter has to take the meaning of a *wh*-phrase and convert it into a set of alternatives as well. The semantics of the *merge* and *move* operations might need to be altered to allow this.

[6]The idea of concatenating trees shares the intuition of concatenating tuples of strings in Multiple Context Free Grammars (Seki et al. 1991).

condition has both syntactic and semantic aspects, the type of arguments (*Expr*) that are inputs to Elide will be presented as a triple *Contree, F, M*.

Part of Elide's job is to check the identity condition, so it is reasonable to define Elide as a function that takes two arguments. In a sluiced sentence like *John met someone yesterday, BIDK who*, Elide takes the antecedent *John met someone yesterday* as the first argument, and the ellipsis site plus remnant *John met who yesterday* as the second argument, and outputs the sluiced sentence. The simplified idea is illustrated in (119).

(119) Elide (John met someone yesterday) (John met who yesterday) = John met someone, bidk who

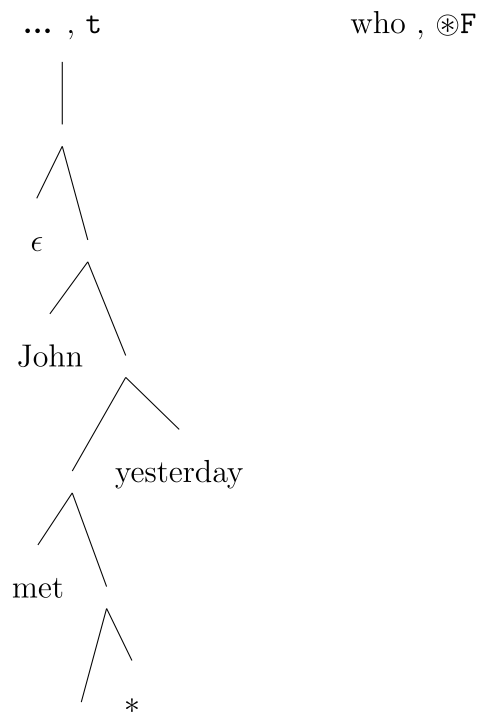
Now we have to determine the type of argument that Elide can operate on. Unlike the ***merge*** and ***move*** operations, Elide is not just a structure building operation. As its name suggests, Elide does some manipulation to the structure, meaning that is already derived. As a result, the triple (*Contree, F, M*) is not viable as the type because the only *Contree* in the triple cannot be manipulated[7]. So we need the argument of Elide to be of a more complicated type.

The solution comes with the ⊛F feature. *wh*-remnants are argued to have the features `d, ⊛F`. As mentioned before, the ⊛F feature is neither a selecting nor a licensing feature. The reason why ⊛F was introduced is to allow the *wh*-remnant to be held out from the ellipsis site so it would not be affected by what happens to the ellipsis site. ⊛F ensures that the *wh*-remnant is not merged into the derived structure, and leaves a mark in the derived structure where the *wh*-remnant is supposed to have been as shown in (120).

---

[7]In MG with just *merge* and *move* operations, we do not need look into *Expr*s to figure out what to do with them. The result is the parallel between derived structure and tuples of strings as discussed in Chapter 3. Strings in general cannot be manipulated once concatenated as the main structure. However, by adding Elide, we have to look into *Expr*s to check the identity condition. Therefore, if we assume that *Contree* is there for identity condition, we would also need to draw a line between checking identity and manipulating the derivation of expressions to keep the parallel between derived structures and strings. Hence, *Contrees* cannot be manipulated either.

(120) John met someone yesterday, BIDK (John met) who (yesterday).



⊛F holds out the *wh*-remnant and Elide puts the *wh*-remnant in its proper place once ellipsis occurs. With the *wh*-remnant being held out, the type of the second argument for Elide has to be [(*Contree, F, M*), (*Contree, F, M*)], the first part of the double being the ellipsis site and the second part being the *wh*-remnant that is held out.

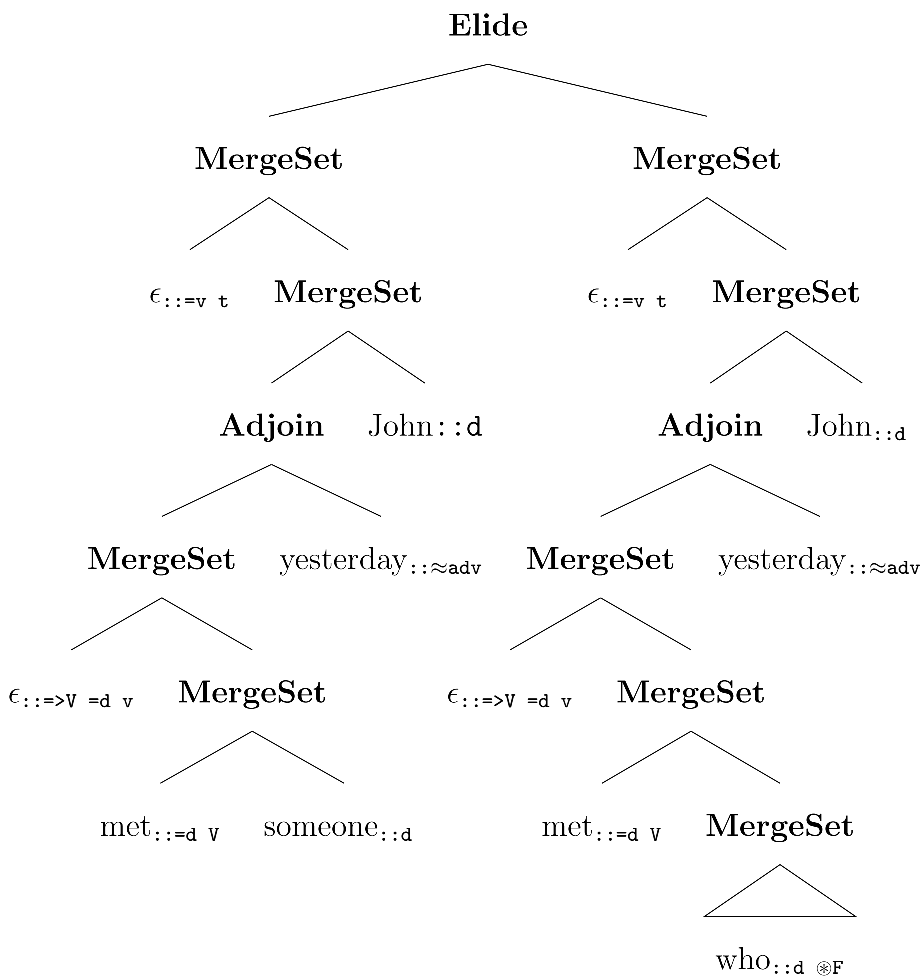
The first argument of Elide, the antecedent, is the type (*Contree, F, M*) as nothing is being held out. The indefinite correlate does not need to move and thus does not have the ⊛ feature, so it is merged into the derived structure. The idea with arguments is illustrated as strings and features in (121).

(121) Elide (John met someone yesterday, t), ((John met $*$ yesterday, t), (who, ⊛)) = (John met someone yesterday BIDK who, t)

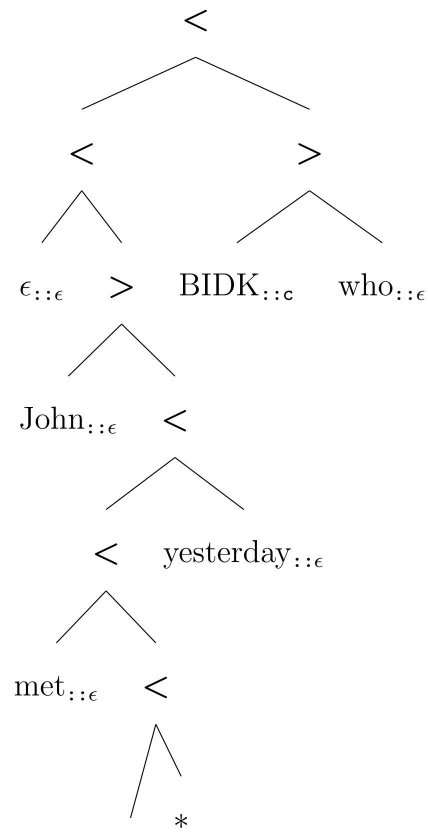
Since the indefinite correlate does not have the same features as the *wh*-remnant, they are not syntactically equivalent, which should be the case as the antecedent is not the same as the ellipsis site. However, to check the identity condition, Elide must still compare the antecedent and ellipsis site. Thus, the mark $*$, left behind by ⊛F, serves as an indicator that a *wh*-phrase should have been merged in that position. So whatever is in the antecedent that is the equivalent of $*$ must be an indefinite.

Before discussing the parts of Elide, I show the derivation tree and derived structure for the sentence, *John met someone yesterday, BIDK who.*

(122)



(123)



5.4 Elide in Three Parts

The Elide function can be divided into three essential steps, the first step is to check that the identity condition is met. If it is met, then the second and third steps of Elide can happen, where ellipsis occurs and the remaining pieces reassemble to form a sluiced sentence with the right interpretation. If the identity condition is not met, the derivation crashes resulting in ungrammaticality. The previous sections set up the types Elide uses to check the identity condition. However, the two derived structures are not completely isomorphic and additional mechanisms are needed to compare the two subtrees.

The second step of Elide asks the question: what happens to structure when ellipsis happens? In Merchant (2001), ellipsis is a PF phenomenon, where the correct structure is built and a phonological process makes the structure silent at the PF level. Minimalist

Grammars do not differentiate between different levels of derivation. Lexical items merge into the structure with or without sound, so unless a structure is originally built with silent lexical items, it would be difficult to get silence derivationally without additional operations. In this dissertation, I make use of a structure destroying process to account for ellipsis. Structure destroying is not a completely new concept in syntax and was proposed in the early days of transformational grammar in Ross (1967). Below, I explain how Elide destroys structure and reassembles the remaining pieces to form sluiced sentences.

The third step of Elide ties up loose ends for the interpretation of the sluiced sentence. The meaning that gets compared between the antecedent and ellipsis site is a set of propositions. Both sets of propositions need to be closed to get the existential and question interpretations for the antecedent and ellipsis site, respectively.

5.4.1 Part I: Checking the Derivational Isomorphism Condition

To satisfy the Derivational Isomorphism Condition, the derivational subtree of the antecedent (minus the correlate) and the ellipsis site (minus the *wh*-remnant) have to be equivalent. Thus, Elide compares the (*Contree, F, M*) triple of the antecedent, which is the first input, with the (*Contree, F, M*) triple of the ellipsis site, which is part of the second input. While the features and meaning of both triples are identical, the two *Contrees* from antecedent and ellipsis site are not identical. *Contrees* here are derived structures and not a subtree of the derivation tree, hence they are different because the *wh*-remnant has the ⊛F feature while the indefinite correlate does not. This is illustrated in the example in (124) using strings instead of actual *Contrees*. Meaning is left as M in the final output of this example as the way meaning is determined will be discussed in part III.

(124) Elide (John met someone yesterday, t, $\{met(j,x) \mid x \text{ in } \{a, b, c...\}\}$), ((John met $*$ yesterday, t, $\{met(j,x) \mid x \text{ in } \{a, b, c...\}\}$), (who, ⊛F, $\{a, b...\}$))
= (John met someone yesterday BIDK who, t, M)

The *Contree* from the first input is *John met someone yesterday* while the *Contree* from the ellipsis site is *John met* $*$ *yesterday*. Elide makes sure that besides *someone* and $*$, the rest of the structures are identical. In addition, Elide also makes sure that the equivalent of $*$ in the antecedent is an indefinite. The checking of the identity condition is roughly defined below:

Definition 5.4.1. ***Elide*** is a partial function that takes two *Expr*s as arguments; the second *Expr* must have at least two components

Elide (*Expr*) (*Expr*) =

Elide ((*Contree*$_1$, f_1, m_1), cs), [(*Contree*$_2$, f_2, m_2), (*Contree*$_3$, f_3, m_3), cs] =

if compare((*Contree*$_1$) (*Contree*$_2$)) and ($\mathrm{f}_1 = \mathrm{f}_2$) and ($\mathrm{m}_1 = \mathrm{m}_2$) **A**

then part II happens

else derivation crash

Like *merge* and *move*, not every *Expr* can be worked on by Elide, hence it is a partial function. In the rough definition above, the line marked with **A** represents the checking of the Derivational Isomorphism Condition, where the (*Contree, F, M*) triples of the antecedent and ellipsis site are checked. The *compare* function can be roughly defined below.

(125) *compare* is a function that takes two *Contrees* as input and outputs a boolean value

a. *compare* [NonLeaf] (l1, r1) [NonLeaf] (l2, r2)
= *compare* (l1, l2) and *compare* (r1, r2)

b. *compare* [NonLeaf] (x) [Leaf] (y) = False

c. *compare* [Leaf] (x) [NonLeaf] (y) = False

d. *compare* [Leaf] (a) [Leaf] (b) = a==b and match (a) (b)

(126) *match* is a function that takes two *Contrees* as input and outputs a boolean value

match (L1) (L2) =

L1 == $*$ and L2 $\in$ $\{$*indefinites*$\}$ || L1 == L2

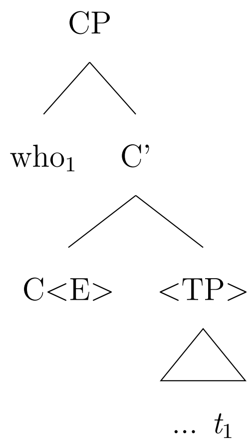
The *compare* function in (125) compares the two *Contrees* from the antecedent and ellipsis site and the *match* function (126) ensures that the equivalent of $*$ in the ellipsis site is an indefinite in the antecedent. When the identity condition is satisfied, ellipsis is allowed to happen, resulting in the sluiced sentence.

5.4.2 Part II: Eliding Structure = Destroy and Reassemble

The second part of Elide concerns creating the elliptical sentence. On the structural approach to ellipsis presented here, the syntax of the ellipsis site is built the same as the syntax of the non-elliptical antecedent. One way of making the ellipsis site 'silent', and which has been widely adopted in more recent ellipsis literature (Merchant 2001, Van Craenenbroeck 2010, Aelbrecht 2009), is to have non-pronunciation. Lexical items in the ellipsis site are rendered phonologically null through some phonological process that leaves structure otherwise intact. Merchant (2001) uses the E-feature on certain heads to signal to the phonology that the complement structure has null phonological value. This is exemplified in (127) where the E feature on the C head signals that the complement TP is not to be pronounced.

(127) a. Someone came, BIDK who.

b.



The main advantage of formulating ellipsis as non-pronunciation is preserving the syntax as it is. Hence, nothing more needs to be said about structure and all the connectivity effects from ellipsis, such as preposition stranding and case matching, follow straightforwardly.

However, implementing ellipsis in this way requires there be different levels for syntactic and phonological processes, and this is not immediately available in the MG system. Theoretically, the MG system needs an operation that takes an *Expr* and renders all the leaves into empty strings while keeping track of the gap where the *wh*-remnant was extracted from. While the implementation is not entirely impossible, it is quite hard to do.

An alternative idea t that does not require a non-pronunciation process stems from traditional formulation of a deletion transformation (Ross 1967, Ross 1969, Hankamer 1979). This idea is essentially a structure removal process, where pieces of built structure are destroyed. A more recent approach to this deletion formulation has been pursued by Müller (2017), Müller (2018). He argues for a deletion operation, *Remove*, that deletes parts of syntactic representation. While I am not going into details for Müller's *Remove* operation (see Murphy 2019 for an overview of the Structure Removal idea), I will use a form of structure removing (or destroying) for Elide since the MG formulation here has more or less forced us into that corner. However, the structure removed here is the derived structure and the connectivity effects that are observed are still preserved in the derivation tree. In that sense, we retain all the information of the derivation.

Since the *wh*-remnant is already held out and not part of the derived structure of the ellipsis site, the derived structure can be eliminated as an instance of ellipsis. Once the identity condition have been checked, Elide destroys the ellipsis site, which is the first element of the second input. By destroying the ellipsis site, I mean that the *Contree* in the ellipsis site triple (*Contree, f, m*) is not part of the output. The *Contree* of the *wh*-remnant is placed in its position instead. Only the *Contree* of the ellipsis site is changed, since we still need the feature and meaning of the ellipsis site.

(128) Elide

(John met someone yesterday, t, $\{met(j,x) \mid x \text{ in } \{a,\ b,\ c...\}\}$)

((John met $*$ yesterday, t, $\{met(j,x) \mid x \text{ in } \{a,\ b,\ c...\}\}$), (who, ⊛F, $\{a,\ b,\ c...\}$))

= (John met someone yesterday, t, $\{met(j,x) \mid x \text{ in } \{a,\ b,\ c...\}\}$))

(who , t, $\{met(j,x) \mid x \text{ in } \{a,\ b,\ c...\}\}$) **B**

= (John met someone yesterday BIDK who, t, M) **C**

In (193), the line with **B** shows the structure destroying process, in which the derived tree of the ellipsis site is replaced with the *wh*-remnant that was kept to the side. The line with **C** shows the combination of both pieces of *Expr* and stitches the *Contrees* together to form the final sluiced sentence with a single feature and meaning. The structure removing and stitching processes are not meant to be a two-step process. The lines with B and C are just meant to show that both steps happen. The stitching process also involves adding additional structure such as 'BIDK' that are external to the ellipsis process. After, adding in this structure destroying process, the definition of Elide is now as follows:

Definition 5.4.2. ***Elide*** is partial function that takes two *Expr*s as arguments, where the second *Expr* has at least two components

Elide (*Expr*) (*Expr*) =

Elide ((*Contree*$_1$, *f*$_1$, *m*$_1$), cs), [(*Contree*$_2$, *f*$_2$, *m*$_2$), (*Contree*$_3$, *f*$_3$, *m*$_3$), cs] =

if *compare*((*Contree*$_1$) (*Contree*$_2$)) and ($f_1 = f_2$) and ($m_1 = m_2$)   (see (125 and 126) for *compare*)

then

(*Contree*$_1$ *++ (BIDK ++ Contree*$_3$*)*, *f*$_1$, *sluiceMeaning(m*$_1$*)*)

else

undefined

5.4.3 Part III: Interpreting the Final Output

The final part of the Elide function concerns the meaning of the final sluiced sentence, the function noted as *sluiceMeaning* in Definition 5.4.2. A sentence with an indefinite such as, *John met someone*, and a question sentence with a *wh*-word, such as *who did John met*, do not mean the same thing eventhough *someone* and *who* might have the same meaning. In Chapter 4, indefinites and *wh*-words are a set of alternatives that percolates upwards until it reaches an operator that closes the set, $\exists$ for indefinites and Q for questions. The problem with having these sets closed by an operator before the Elide operation is that it makes it harder to compare meaning between the antecedent and ellipsis site. At the same time, the Derivational Isomorphism Condition would no longer hold. Thus, having Elide compare meaning before the set of proposition gets closed retains the idea that indefinites and *wh*-words that does not move take scope in a similar fashion (through set percolation) while also retaining the Derivational Isomorphism Condition.

However, we still need to tie up this loose end as the final sluiced sentence does not have a meaning that is a set of propositions. Hence, the last thing that Elide must do is close the two sets of propositions with the right operator and conjoin the two meanings together.

For the antecedent, the set of propositions must be closed with the $\exists$ operator, or $\bigvee$. On the other hand, the set of propositions of the ellipsis site must be closed with some kind of question operator, Q. Q is a placeholder for an operation that takes a set of propositions and turn it into a question meaning. (129) shows an example of Elide closing the sets of propositions in line **D**.

(129) Elide

(John met someone yesterday, t, $\{met(j,x) \mid x \text{ in } \{a,\ b,\ c...\}\}$)

((John met $*$ yesterday, t, $\{met(j,x) \mid x \text{ in } \{a,\ b,\ c...\}\}$), (who, $\circledast$F, $\{a,\ b,\ c...\}$))

= (John met someone yesterday, t, $\{met(j,x) \mid x \text{ in } \{a,\ b,\ c...\}\}$))

(who , t, $\{met(j,x) \mid x \text{ in } \{a,\ b,\ c...\}\}$)

= (John met someone yesterday BIDK who, t, $\bigvee\{met(j,x) \mid x \text{ in } \{a,\ b,\ c...\}\} \wedge Q\{met(j,x) \mid x \text{ in } \{a,\ b,\ c...\}\}$) **D**

Hence, the definition of *sluiceMeaning* is as follows:

(130) $sluiceMeaning(m) = \bigvee m \wedge \neg\text{know}(\text{I}, Qm)$

The final definition of Elide with all three parts of the operation is presented below:

Definition 5.4.3. ***Elide*** is partial function that takes two *Expr*s as arguments, where the second *Expr* must have at least two components

Elide (*Expr*) (*Expr*) =

Elide ((*Contree*$_1$, f_1, m_1), cs), [(*Contree*$_2$, f_2, m_2), (*Contree*$_3$, f_3, m_3), cs] =

if *compare*((*Contree*$_1$) (*Contree*$_2$)) and ($\text{f}_1 = \text{f}_2$) and ($\text{m}_1 = \text{m}_2$) (see (125 and 126) for *compare*)

then

($\mathit{Contree}_1$ ++ (BIDK ++ $\mathit{Contree}_3$), f_1, $\mathit{sliceMeaning}(m_1)$)

else

undefined

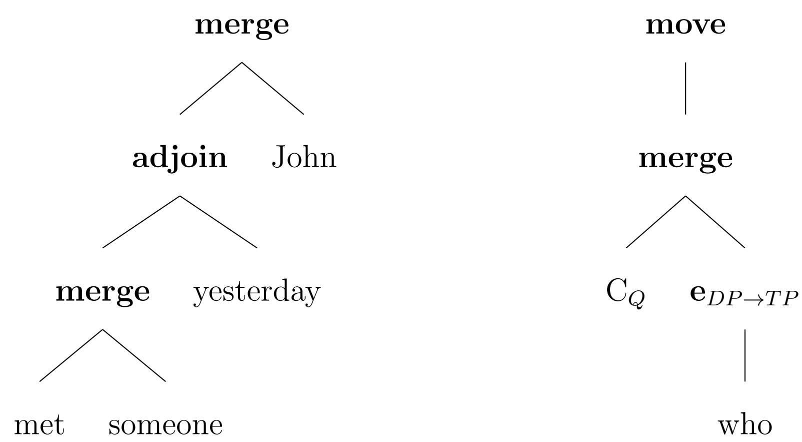
5.5 Wong (2020) vs Kobele (2015)

The analysis of sluicing with no *wh*-movement presented in this dissertation ends up being a type of structure destroying operation. The structure destroying analysis is driven by the combination of the sluicing data (form identity effects, scope parallelism) as well as the nature of the MG framework that does not have multiple levels of derivations. In Kobele (2015), he presents a different analysis to sluicing that is also couched within the MG framework. His sluicing analysis involves a copying process rather than structure destroying. In this section, I present Kobele's (2015) account of sluicing and argue that it is not able to explain the scope parallelism facts.

5.5.1 A Derivational Copying Approach (Kobele 2015)

Kobele (2015) presents a derivational copying approach to ellipsis, where the ellipsis site can be thought of as a type of proform that is anaphoric on the meaning of its antecedent. The antecedent is defined by its derivation and the ellipsis site is resolved by copying the meaning of the antecedent. The mechanism of Kobele's approach is explained with the usage of the example *John met someone BIDK who.* A simplified derivational tree for the sentence is shown in (131).

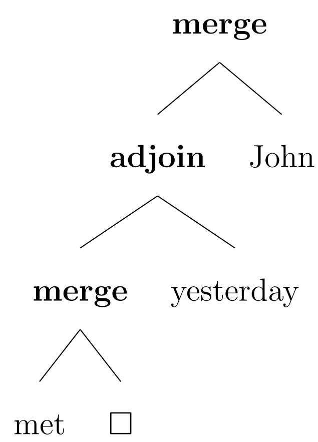
(131)



In Kobele's approach, the elide/copying operation, marked as **e** on the derivation tree, can be either a nullary or unary operation (depending on the type of ellipsis) that has a syntactic type. The syntactic type of **e** must be matched with that of its antecedent as it is where it obtains the semantics. For sluicing, the syntactic type of **e** is DP→TP as shown in (131). In the example above, **e** with DP→TP type is a unary operation that says 'given a DP, construct a TP'. Thus, given a DP like *who*, **e** constructs something that amounts to a TP in the derived structure. In this system, **e** basically represents the ellipsis site and is positioned down in the derivation tree where the derivation of the ellipsis site is. The derived structure of **e** is devoid of any phonological content as the syntactic type of **e** is sufficient for ellipsis resolution.

While the pronounced form the ellipsis site is independent from the antecedent, the meaning of **e** relies on some derivational structure (the antecedent) that is of the same type. Antecedents on this derivational approach are presented as (tree) contexts (Comon et al. 1997), which can be thought of as subtree with a missing part. A simplified derivational tree for the antecedent of *John met someone yesterday BIDK who* is presented in (132).

(132)



It is obvious that both the antecedent and a non-elliptical version of the sluiced sentence would have the same structure as (132). The □ represents the hole where either the indefinite *someone* or *wh*-word *who* would be. However, since this is a derivational copying analysis, (132) is the antecedent of the sluiced sentence and it is the same type as **e**, which is DP→TP (see Kobele (2015) for explanation of context types). Hence, **e** which represents the ellipsis site obtains its meaning from the tree context in (132).

Kobele (2015) shows that many empirical generalizations of ellipsis follow naturally from the copying approach. For sluicing, this includes the preposition-stranding generalization and the ban on active-passive mismatch. The lack of island violations under sluicing follows from this approach as well since the ellipsis site is only reliant on the semantics of the antecedent and not the internal structure that causes the island violation. The spell out operation on **e** can be defined to create structures that do not violate islands. A potential difficulty is explaining the case-matching facts of sluicing since case-matching relies on the *wh*-word and its surrounding material. However, a quick fix (albeit slightly stipulative) is to have **e** be specified for DPs with the correct case.

Kobele's copying approach to sluicing is very different from the approach argued for in this dissertation. Though the copying approach presented in Kobele (2015) has *wh*-movement, *wh*-movement itself does not play much of a role in the sluiced structure. In Merchant's

(2001) movement and deletion analysis, the *wh*-phrase moves to a higher position so that the ellipsis operation can delete a proper constituent. The copying approach does not need the ellipsis site to be a constituent, thus having a no movement approach does not change any of the mechanisms proposed by Kobele. Before moving on to the next section, I want to point out the one similarity between the approaches. Tree contexts are also found in my proposal of sluicing. The Derivational Isomorphism Condition relies on isomorphic derivational tree contexts, as the antecedent and the ellipsis site (minus the correlate and *wh*-phrase) must have the same derivation for the condition to go through. However, a more apparent usage of tree contexts is in the derived structure of the ellipsis site. When the *wh*-word is held out from the ellipsis site, the ellipsis site forms a tree context that is then compared with the antecedent before the entire context is eliminated. In the next section, I highlight the one advantage that my proposal has over the copying approach.

5.5.2 A Problem with Scope Parallelism

The structure destroying analysis presented in this dissertation has almost nothing in common with Kobele's derivational copying approach, even though both are formulated within MG. In my approach to sluicing, the antecedent and the ellipsis site form their respective structures independently but are required to satisfy the identity condition before ellipsis occurs. On the other hand, Kobele's approach does not require an identity condition because the ellipsis site is anaphoric to the antecedent. Though the two approaches are different, both are able to account for most of the facts of sluicing discussed in this dissertation, with the possible exception of scope parallelism.

Kobele's copying approach to ellipsis uses syntactic types to identify the antecedent while my analysis relies on syntactic and semantic isomorphism. Hence, a possible advantage of my analysis is when there is similarity between the semantics of the antecedent and the ellipsis site that is not reflected in the syntactic types. A possible instance of this is the scope parallelism data presented in Chapter 4 (repeated below).

(133) a. If a relative of John died, he would be upset, but I don't know which relative.

b. If any relative of John died, he would be upset, *[but I don't know which relative].

(134) They usually ask whether the candidate reviewed any papers for the journal, but they never ask which ones. (Romero 1998).

The NPI *any* is unable to take wide scope over the *if*-clause in (133b), while the *wh*-phrase *which relative* has widest scope over its ellipsis site. This causes a scope mismatch between the antecedent and the ellipsis site and the sluiced sentence in (133b) is ungrammatical. (134) is evidence that *any* can serve as a correlate as long as there is no scope mismatch between the antecedent and ellipsis site. According to the copying approach, the ellipsis site in (134) is anaphoric to its antecedent, *the candidate reviewed __ for the journal*. Kobele (2015) does not discuss how NPIs take scope or what the syntactic type of NPIs are. For Kobele to account for the data in (133), he would have to introduce different syntactic types for 'a relative' and 'any relative'. The difference in syntactic types would allow for an ellipsis site to match the antecedent context in (133a) but no ellipsis site to match in (133b). If NPIs were assumed to have the same syntactic type as normal indefinites like *someone*, the copying approach would not be able to explain the differences in grammaticality between the two sentences in (133). In any case, the scope parallelism facts come naturally from the Derivational Isomorphism Condition that is proposed in this dissertation as both syntactic and semantic information are required.

5.6 A Caveat on Elide: Derivational Complexity

One attractive property of the MG formalism (with just *merge* and *move* as operations) is that every derivation tree language of MG forms a regular tree language (Graf 2010, Graf 2013). In other words, every well-formed MG derivation tree can be recognized by a bottom-up tree automaton. A bottom-up tree automaton recognizes a tree from the leaves, and traverses upwards until it reaches the root. As it traverses the tree, it assigns each node a state that is based on the states that it assigned to the daughters of the node. If the root is assigned a designated final state, then the tree is accepted by the automaton, making the tree well-formed. If the root is not assigned a final state or the automaton cannot assign a state to a node, the tree is rejected and not well-formed. The definition of a bottom-up tree automaton is given below.

Definition 5.6.1. A bottom-up tree automaton (BTA) is a four tuple (Q, Σ, F, Δ) where:

Q is a finite set of states

Σ is a ranked alphabet

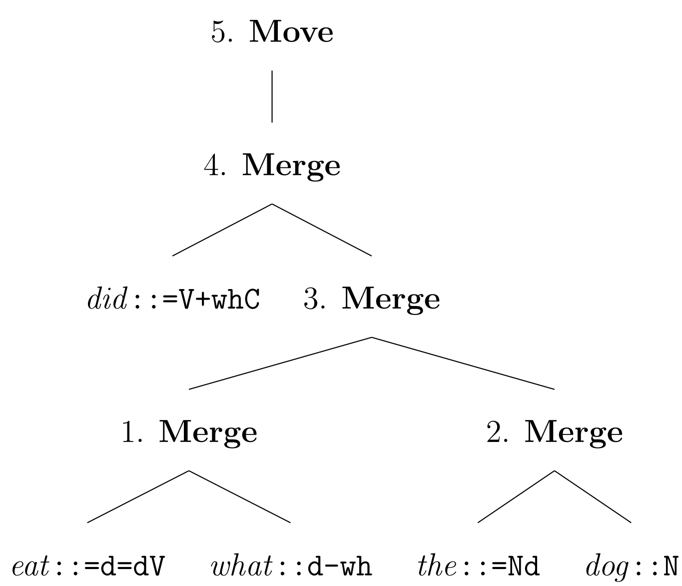
$F \subseteq Q$ is the set of ending states

$\Delta \subseteq Q^* \times \Sigma \times Q$ is the set of transitions, which must be finite

I will show an example of using a BTA to recognize a MG derivation tree with *merge* and *move* operations[8] In the case of MGs, the automaton needs to keep track of the computation of the feature calculus as it traverses up the tree to determine whether the derivation tree is well-formed. The sentence and derivation tree used to show how a BTA works are shown in (135) and (136).

(135) What did the dog eat?

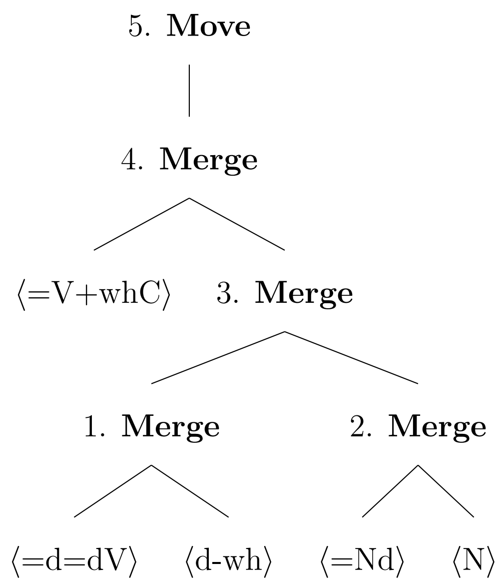
(136)



[8]For more examples of how BTA works with non-MG trees, see Graf (2013), section 2.1.1.

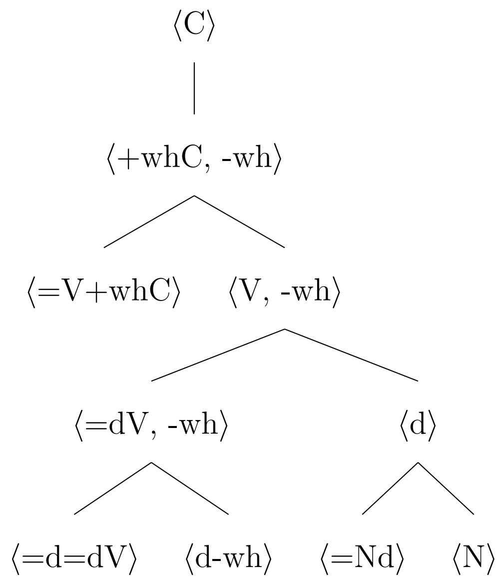
For the BTA to recognize MG derivation trees, the finite set of states, Q, is a tuples of unchecked feature sequences. This allows the BTA to keep track of the computation of feature calculus. The automaton starts from the leaves of the tree, and in (136), it starts with *eat*. *eat* has the feature sequence `=d=dV`, and since no operation has applied yet, it still has all its features unchecked. Thus, *eat* can be given the state ⟨=d=dV⟩. In general, all the leaves in a derivation tree can be mapped directly to its feature sequence as shown in (137).

(137)



As the automaton moves from leaves to nodes, it assigns a new state based on the two daughter nodes. The assignment of the new state mirrors how *merge* and *move* check features. Thus the **Merge** node in 1 is assigned the state ⟨=dV, -wh⟩ since these are the remaining active features after *merge* checks the first `d` feature. For the **Merge** node in 2, it is assigned the state ⟨d⟩, since *merge* checks the `N` feature. (138) shows the tree that has been fully traversed by the automaton, with a root that has been assigned ⟨C⟩. Since the highest projecting head is `C`, ⟨C⟩ is the final state of the automaton, making the derivation tree well-formed.

(138)



The example above shows that a BTA can be constructed to show that a MG derivation tree is regular. The implication of being regular is that the derivation tree can be computed using only a finite amount of memory as only a finite amount of states is needed (following the definition of a BTA). This can be appealing for linguistics research if the finite state assumption is thought of as corresponding to working memory configurations.

However, the addition of Elide in the MG system would be a drawback to this finite state assumption of derivation trees. That is, MG derivation trees are no longer regular if Elide is part of the operations. Elide is defined to operate on two subtrees by comparing them before allowing ellipsis to occur. The two subtrees that Elide checks can be unboundedly large and there are potentially unboundedly many ways for the two subtrees to mismatch. This means that a BTA would have to consider an infinite number of distinct figure configurations for some derivation tree, and by its definition, it is unable to do that. Thus, the catch of adding Elide into the MG system is the reconsideration of the language of derivation trees from regular into something of a higher complexity.

5.7 Conclusion

This chapter began by explaining how a *wh*-remnant, a non-moving *wh*-moved phrases, might fit into the MG framework. The syntax of a *wh*-remnant might be thought of as simple, given that there is no movement. However, having non-constituent deletion complicates the matter, thus requiring a new feature (⊛`F`) to allow for this seemingly non-constituent deletion. The semantics of this *wh*-remnant might then be expected to be more complex than a regular moved *wh*-word, as the *wh*-remnant does not take scope through movement. However, I argue that a *wh*-remnant behaves semantically like an indefinite, and takes scope through set percolation. Hence, the end result of a *wh*-remnant with ⊛`F` is a complicated syntax but a simple semantics. The ⊛`F` feature holds out the *wh*-remnant in a way similar to the way that normal *wh*-phrases get held out by `-wh`, but the *wh*-remnant does not get combined into the end structure in the same way. This allows Elide to operate and destroy the appropriate structure giving us the elliptical sentence. In the next chapter, I show how the implementation of sluicing here accounts for the Malay data seen in Chapter 2.

CHAPTER 6

Malay Sluicing Explained!

In this chapter, I apply the mechanism of sluicing in MG (defined in the previous chapter) to two of the Malay "form-identity effects" from Chapter 2 that were used to argue for a no-movement analysis of sluicing. The two "form-identity effects" are island insensitivity and p(reposition)-stranding. A brief recap: Malay exhibits both movement and no movement in regular *wh*-question formation, and the no-movement approach is argued to be the choice for sluicing. The no-movement approach is supported by a number of "form-identity effects", including island insensitivity and p-stranding. Though I have been calling the analysis 'no-movement', the *wh*-remnant in sluicing described in the previous chapters is not identical to a *wh*-phrase in situ or an indefinite. The *wh*-remnant is argued to be its own item. However, its similarities to *wh*-in situ and indefinites make this analysis of sluicing 'no-movement'.

So far, I have not described what a *wh*-phrase in situ is like in the MG system. As seen in previous chapters, *wh*-phrases that undergo movement have the `wh` feature, while *wh*-remnants have the ⊛`F` feature. A *wh*-phrase in situ in a question has neither of these features and is a lexical item with category `d`. In the following section, I give a full MG summary of the syntax and semantics of indefinites, *wh*-phrases that are in situ or moved, and remnants. Then I show how the overlapping characteristics between *wh*-remnants, *wh*-phrases in situ, and indefinites account for the island insensitivity and p-stranding patterns seen in chapter 2. In the final section of this chapter, I discuss the puzzle of the Malay scope effects data and provide a potential solution for it.

6.1 Summary: Indefinites, *wh*-(movement, in situ, remnants)

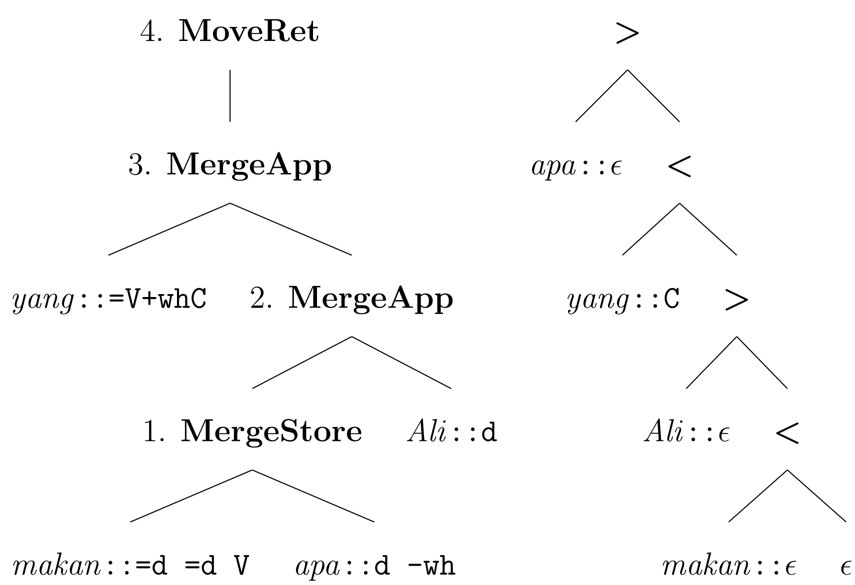
For the MG system defined in this dissertation, *wh*-phrases come in two essential forms, *wh*-phrases that move and *wh*-phrases in-situ. *wh*-phrases in situ are argued to be like indefinites in that they take scope through set percolation while a moving *wh*-phrase takes scope through *wh*-movement. In between these two *wh*-phrases lie *wh*-remnants, which are argued to be transformed from *wh*-phrases that move but behave semantically like *wh*-prhases in situ. In this section, I give a full summary of the characteristics of *wh*-movement, indefinites, *wh*-in situ, and *wh*-remnants, in that order.

6.1.1 *wh*-movement

wh-phrases that move in this MG system have the `wh` feature as part of its feature list, which is crucial for *wh*-movement to occur. This is illustrated in the derivation and derived trees below.

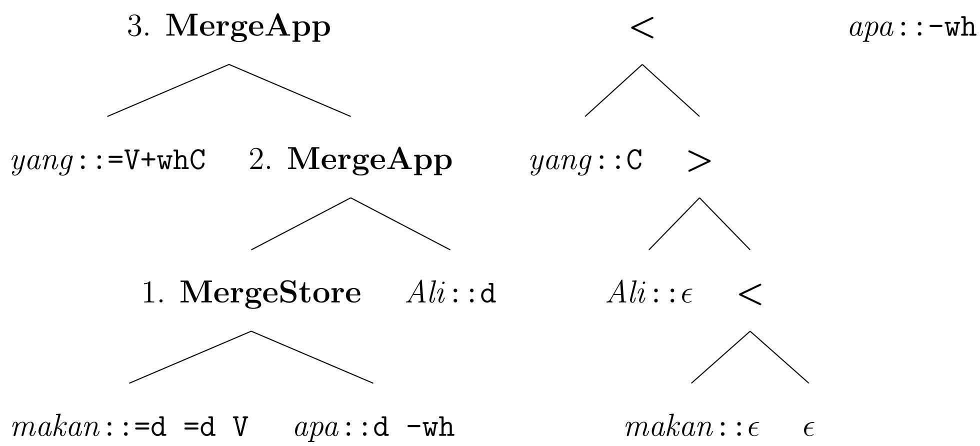
(139) apa yang Ali makan?
what COMP Ali eat
'What did Ali eat?'

(140)



(140) shows the full derivation and derived structure, and (141) shows the derivation and derived structure at Step 3, just before the movement of the *wh*-phrase. The reason for highlighting this part of the derivation, where the *wh*-phrase is held out before the final *move* operation, is to draw comparison with the *wh*-remnant which is also held out at some point of the derivation of a sluice sentence.

(141)



When the *wh*-phrase is combined into the derivation with the `-wh` feature in Step 1, it gets held out like in (141) until it gets checked by the `+wh` feature in Step 4 of (141).

Besides syntactic movement, the `wh` feature is also used by the semantics to determine where the *wh*-phrase takes scope. The semantics of the sentence in (139) is shown below. Semantic storage is used in this case where the meaning of the *wh*-phrase is stored following the `-wh` feature and retrieved following the `+wh` feature.

Steps	Main Meaning	Semantic Store
1.	$\lambda x.makan(x,a)$, `-wh`	($\lambda P.WHy[Py]$, a, `-wh`)
2.	$\lambda x.makan(A,a)$, `-wh`	($\lambda P.WHy[Py]$, a, `-wh`)
3.	$\lambda x.makan(A,a)$, `-wh`	($\lambda P.WHy[Py]$, a, `-wh`)
4.	$WH.x(makan(A,x))$	ϵ

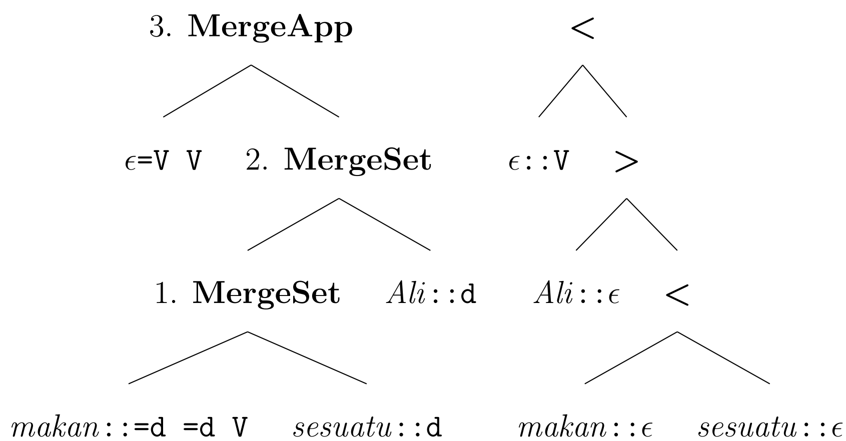
Table 6.1: Semantics of (139)

6.1.2 Indefinites

Syntactically, indefinites do not move and stay in their base position.

(142) Ali makan sesuatu.
Ali eat something
'Ali ate something'

(143)



With no movement features, indefinites do not use the semantic storage to take scope. Indefinites are argued to denote a set of individuals, notated as $\{a,\ b,\ c...\}$ in the example below. They take scope via set percolation. The set of individuals contributes to sets of meanings of other types via function application until an appropriate operator combines into the derivation. The operator collapses the set of truth conditions into a single truth condition. In the example in (145), the operator comes as an empty lexical item with the features `=V V`[1].

Steps	Main Meaning	Semantic Store
1.	$\{makan(x,a),\ makan(x,b)...\}$	ϵ
2.	$\{makan(A,a),\ makan(A,b)...\}$	ϵ
3.	$\exists x.(makan(A,x))$	ϵ

Table 6.2: Semantics of (142)

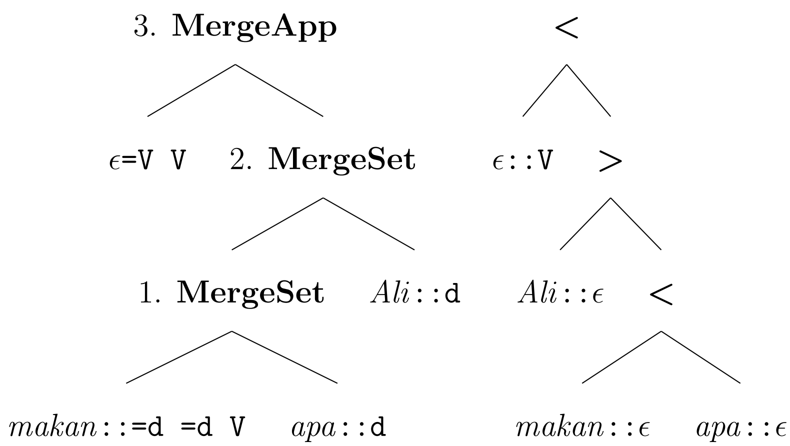
[1] Using `=V V` here is just a matter of convenience. The operator is suppose to be `=t t` and the reasoning is discussed in chapter 3 and also briefly in the scope effects section below.

6.1.3 *wh*-in situ

wh-phrases in situ do not have the `wh` feature as shown in the derivation and derived trees below. Given that they do not move, their derivation looks like indefinites.

(144) Ali makan apa?
Ali eat what
'What did Ali eat?'

(145)



With no `wh` features, *wh*-phrases in situ also cannot use the semantic storage to take scope. In this framework, they take scope like indefinites, through set percolation until an appropriate operator combines in. In Table 6.2, the operator for indefinites takes a set of truth conditions and disjoins them, giving the existential meaning. In Table 6.3, the operator takes a set of truth conditions and produces the meaning of a question, which is represented by WH[2].

Steps	Main Meaning	Semantic Store
1.	{*makan(x,a)*, *makan(x,b)*...}	ϵ
2.	{*makan(A,a)*, *makan(A,b)*...}	ϵ
3.	$WHx.$*(makan(A,x))*	ϵ

Table 6.3: Semantics of (144)

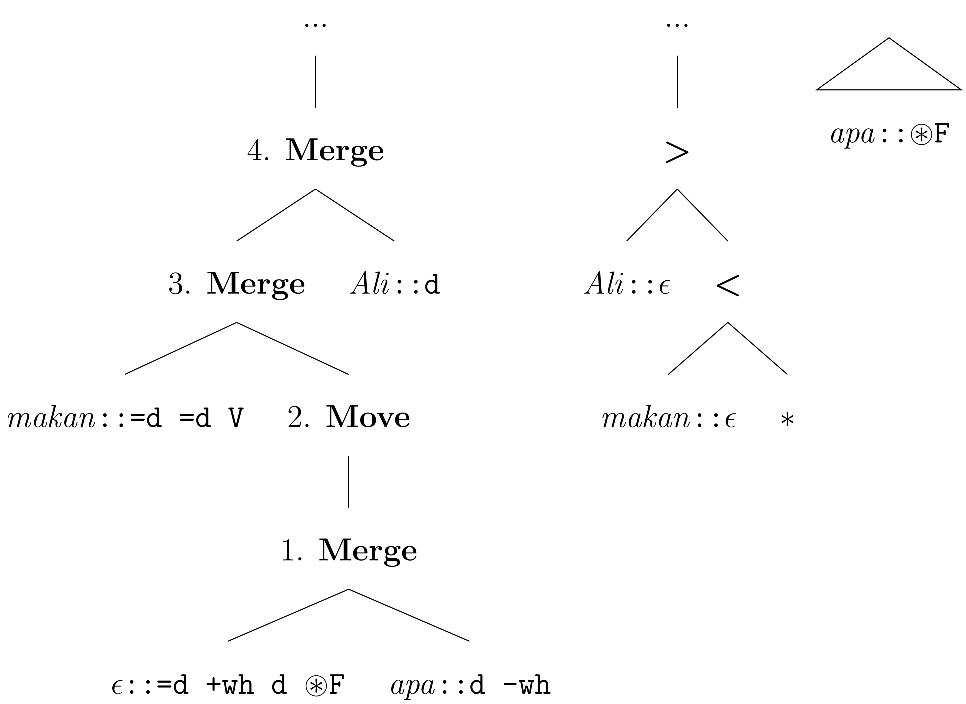
[2]I do not intend to discuss the theory of question semantics in this dissertation. The WH is meant to a placeholder of any theory of question semantics, be it Hamblin semantics, Partition semantics etc.

6.1.4 *wh*-remnants

This dissertation argues that *wh*-remnants in sluicing are like *wh*-in situ and indefinites in their scope taking property. Syntactically, Syntactically, *wh*-remnants do not have the `wh` feature but their own feature, ⊛`F`. However, *wh*-remnants do have something in common with moving *wh*-phrases. In the previous chapter, the ⊛`F` feature was argued to be transformed from the `wh` feature. This transformation produces the distictive sluicing characteristics (to be seen in the p-stranding section below). The derivation tree and derived tree with a *wh*-remnant at the ellipsis site are given below.

(146) Ali makan sesuatu, tapi saya tak tahu apa
Ali ate something, but I NEG know what
'Ali ate something, but I don't know what.'

(147)



The ⊛`F` feature keeps the *wh*-phrase aside for Elide to operate on later in the derivation. The *wh*-phrase with ⊛`F` is kept aside similar to how the `-wh` feature keeps moving *wh*-phrases aside until a lexical item with a `+wh` combines in the derivation. The derivation and derived trees in (147) are comparable to the derivation and derived trees in (141). The difference here is that there is no licensor that eventually checks off the ⊛`F` feature. Given no licensor feature, the *wh*-phrase with ⊛`F` cannot use semantic storage to take scope. Thus, like *wh*-phrases in situ and indefinites, they take scope with sets with the *wh*-phrase denoting a set of individuals.

Steps	Main Meaning	Semantic Store
1.	ϵ, `-wh`	$\lambda P.WHy[Py]$, ϵ, `-wh`
2.	$\{a\ ,\ b,\ c...\}$	ϵ
3.	$\{$*makan(x,a)*, *makan(x,b)*...$\}$	ϵ
4.	$\{$*makan(A,a)*, *makan(A,b)*...$\}$	ϵ
	...	
	$WHx.$*(makan(A,x))*	ϵ

Table 6.4: Semantics of (146)

Since *wh*-remnants begin as normal moving *wh*-phrases, their semantics reflect that of a normal moving *wh*-phrase until they get transformed into a *wh*-remnant at step 2. The semantics of the transformer with the `=d +wh d` ⊛`F` feature list is dependant on the semantics of *wh*-phrases. The *wh*-phrase at step 2 denotes a set of individuals given the ⊛`F` feature, despite the fact that *wh*-movement at step 2 usually retrieves the meaning out of the semantic store and combines it with the main meaning. For now, the transformer is stipulated to be something that would be able to take $\lambda P.WHy[Py]$ and change it to a set of individuals, seen at step 2 in Table 6.4. As with *wh*-in situ and indefinites, the set of alternatives continues to combine with meanings of other lexical items via pointwise functional application until Elide collapses the set.

6.1.5 A Summary Table for a Summary

A summary of this subsection is provided in the table below.

	Syntactic feature	Scope-taking, Semantics
wh-movement	`d -wh`, held out	semantic storage, $\lambda P.WHy[Py]$
Indefinite	`d`, not held out	alternatives, $\{a,\ b,\ c...\}$
wh-in situ	`d`, not held out	alternatives, $\{a,\ b,\ c...\}$
wh-remnant	`d` ⊛`F`, held out	alternatives, $\{a,\ b,\ c...\}$

Table 6.5: Summary table for indefinites, *wh*-(movement, in situ, remnants)

wh-remnant semantics function like indefinites and *wh*-phrases in situ in that they all use set percolation to take scope. Syntactically, the ⊛`F` feature on the remnant is derived from the `wh` feature but has its own distinct characteristics. The next two sections discusses how the three *wh*-phrases discussed here interact with two phenomena seen in chapter 2.

6.2 Island Insensitivity

Like many other languages, island insensitivity is observed under Malay sluicing. The sentence in (148) contains a complex NP island in the square brackets.

(148) Ali bertemu dengan [perempuan yang membeli sesuatu], tapi saya tak tahu apa
Ali met with woman COMP buy something, but I NEG know what
'Ali met the woman who bought something, but I don't know what.'

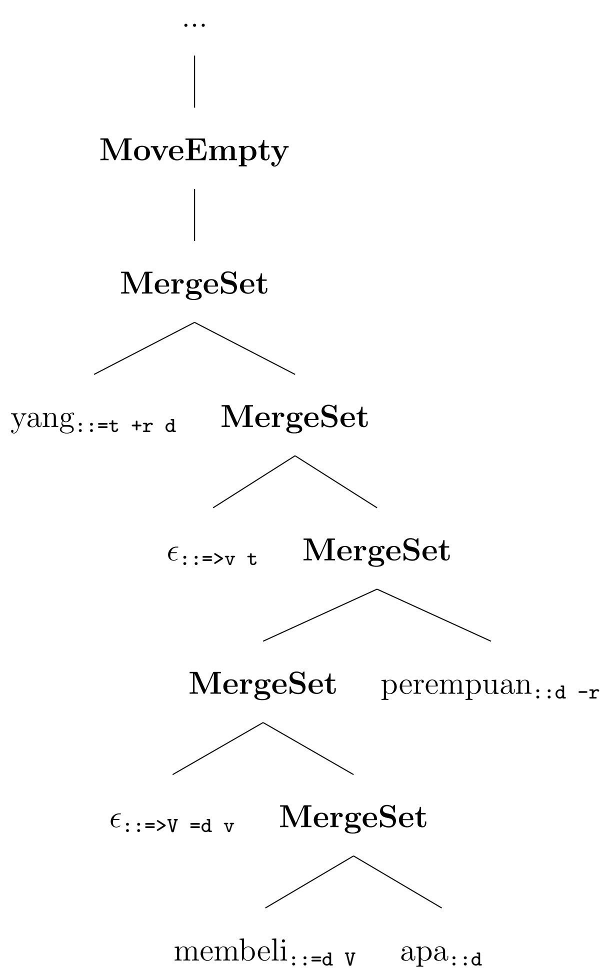
(148) has the structure in (149) if there is no ellipsis and the *wh*-phrase remains in situ. In English, non-movement of the *wh*-phrase would be ungrammatical, but (149) is grammatical in Malay.

(149) Ali bertemu dengan perempuan yang membeli sesuatu, tapi saya tak tahu
Ali met with woman COMP buy something, but I NEG know
Ali bertemu dengan perempuan yang membeli apa
(Ali met with woman COMP buy) what

'Ali met the woman that bought something, but I don't know Ali met the woman who bought what.'

In traditional syntax and also in MG, it is easy to see how a *wh*-phrase in situ (such as in 149) does not violate syntactic islands. With only a `d` feature, it gets combined with the selecting verb straight away as shown in (150). The semantics of a *wh*-phrase in situ is discussed later.

(150)



However, in the MG system created in the previous chapter, the *wh*-remnant is held out of the derivation by the ⊛`F` feature, unlike the *wh*-phrase in situ that combines in immediately. The way the ⊛`F` feature holds out the *wh*-remnant is parallel to the way the a lexical item with a `-wh` feature waits for a `+wh` feature to check it. Both ⊛`F` and `-wh` keep the lexical item with them from fully combining into the derived structure. If island violations are a result of syntactic features holding out the lexical item, both a *wh*-remnant and a moving *wh*-phrase will incur a violation. The goal in the section below is to explain how items with the `wh` feature result in island violations while items with the ⊛`F` feature do not. As we will see in the sections below, *wh*-remnants and *wh*-phrases in situ bypass islands through their set percolation semantics and not their syntactic features. The `wh` feature, which is linked to how a moving *wh*-phrase takes semantic scope is the cause of island violations. In the section below, I implement the Subjacency Condition in MG and show that island violations occur for a moving *wh*-phrase, but not for *wh*-remnants and *wh*-in situ.

6.2.1 Subjacency in MG

Here is a modest attempt to implement the Subjacency Condition (Chomsky 1973) in MG to account for the Complex NP constraints (CNPC). I am not claiming that Subjacency is the right way of contraining movement but it is necessary to show the difference between *wh*-phrases with the `wh` feature (regular *wh*-movement) and *wh*-phrases with the ⊛`F` feature (not *wh*-movement) regarding island violations. Choosing to implement the Subjacency condition is simply a matter of convenience. I will use English instead of Malay in the relevant examples and derivation trees for clarity. The full Malay derivation tree for (148) is given at the end.

The Subjacency Condition is an attempt to explain why certain correlations between a syntactic trace and its antecedent are impossible. The condition is formulated as follows:

(151) In the following structure α and β cannot be related by movmement:

$$...\alpha...[_{BC}...[_{BC}...\beta...]...]$$

where α and β are separated by more than one blocking category

Blocking categories (or Bounding Nodes as they will be referred to henceforth) in English are proposed to be DP and TP. Subjacency gives a kind of measure of syntactic distance in terms of Bounding Nodes (BN), thus it can explain the CNPC as illustrated below.

(152) *[$_{CP2}$ Who did John [$_{TP}$ believe [$_{DP}$ the claim [$_{CP1}$ t_i that [$_{TP}$ Bill saw t_i]]]]]

In (152), the successive cyclic movement of the *wh*-word first moves from base position to CP1 and crosses one BN, namely the TP, which does not violate subjacency. The second movement from CP1 to CP2 crosses two BNs, DP and TP, which violates subjacency.

6.2.2 Successive Cyclic Movement and Bounding Node Counts

As a first step to incorporate subjacency into MG, successive cyclic movement has to be added as part of the derivation. As defined in the previous chapter, *move* in MG is movement in one fell swoop whenever a licensor feature comes into the derivation. A simple way to allow for intermediate positions in MG is to divide the licensor feature for *move* into two polarities, *strong* and *weak* features. *strong* features are the same form as before which is `+f`, while *weak* features will come in the form of `~f`. The idea is that *weak* features are unable to drive *move* as well, however, they will not be able to fully check the `-f` feature. The item with the `-f` feature remains able to move to where `~f` is merged into the derivation but the `-f` feature remains, waiting for a different lexical item with `+f` to fully check it. Thus the features of a MG with subjacency are as follows:

Polarity	Positive		Negative	
Merge	`=x`		`x`	`x` ∈ *sel*
Move	Strong Positive	Weak Positive	Negative	
	`+f`	`~f`	`-f`	`f` ∈ *lic*

Table 6.6: Added features in MG

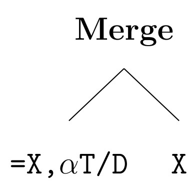
The second step to implementing subjacency is to add a bounding node (BN) count. In this system, any lexical items held out in storage will have a BN count of 0 when first merged into the derivation. The MG that I have defined so far has two storage systems that are closely linked in some aspects. The first is syntactic storage, where lexical items with negative licensing features such as `-wh, -k` are stored and wait to be remerged into a higher syntactic position in the derived tree. The syntactic storage system is not formally defined within *merge* and *move*, but the idea is that items that are waiting to move are in the derivation but kept aside until a final checking feature fully combines them into the derivation tree. The second type of storage is semantic in nature (Cooper storage), where meanings of lexical items may be (but not necessarily) stored through negative licensing features such as `-wh, -q`. The semantics storage is defined within the semantic types. When using semantic storage, the meaning of a lexical item takes scope at the position where it is taken out. These two storage systems are closely linked since both are dependant on negative licensing features given the MG system defined in the chapter before. Thus, a feature like `-wh` uses both syntactic and semantic storage since the `-wh` lexical item moves syntactically to a higher position as well as takes scope in that position. On the other hand, a feature such as one for case `-k` only uses the syntactic storage and has no semantic contribution. In this framework, any lexical item with a negative licensing feature uses syntactic storage but not necessarily semantic storage.

Now the crucial question here is: what should lexical items with the ⊛`F` feature be under, syntactic storage or semantic storage? This would determine the area in which the bounding node count has to come under since lexical items with ⊛`F` cannot be affected by island constraints. The ⊛`F` feature operates like syntactic storage where the lexical item is held out and not merged into the derived tree straight away. At the same time, lexical items with ⊛`F` do not use semantic storage to take scope but use set percolation to take scope instead. Given that the ⊛`F` feature is very much similar to the way a `--wh` is held out in nature for syntactic movement, we want the bounding node counts to be in the semantic storage instead of the syntactic storage. This would differenciate between *wh*-phrases that move and *wh*-phrases that don't move since the bounding node count only affects the usage of the semantics storage. Now, moving *wh*-phrases are not able to cross more than two bounding nodes to take scope while *wh*-phrases that do not move can avoid island constraints through set percolation. This also goes back to the argument that *wh*-phrases that do not move with the ⊛`F` feature are more like indefinites than moved *wh*-phrases in terms of semantics.

6.2.3 Implementation of Subjacency

I will now give an informal description of how subjacency is added into MG. For ease of exposition, the derived structure for the formalism in this section will be expressed in terms of strings instead of trees. To include subjacency, any lexical item that has negative licensing or the ⊛`F` feature, that is merged into the structure will have a bounding node (BN) count of 0 in the semantic storage. These movers are kept aside as the derivation continues and keep track of the number of BNs that are merged into the derivation. Whenever a BN count goes higher than 1, the derivation crashes and is unable to continue. However, if a negative feature is checked by a weak positive feature, the BN count will be reset to 0. The lexical item with a ⊛`F` feature itself never incurs a violation as it does not go into semantic storage. Following Chomsky's (1973) formulation of subjacency, the BNs in this system are the category T or D. Thus, in a derivation tree, BNs are defined by juncture where a selector that is a category T or D merges into the derivation as illustrated in (153).

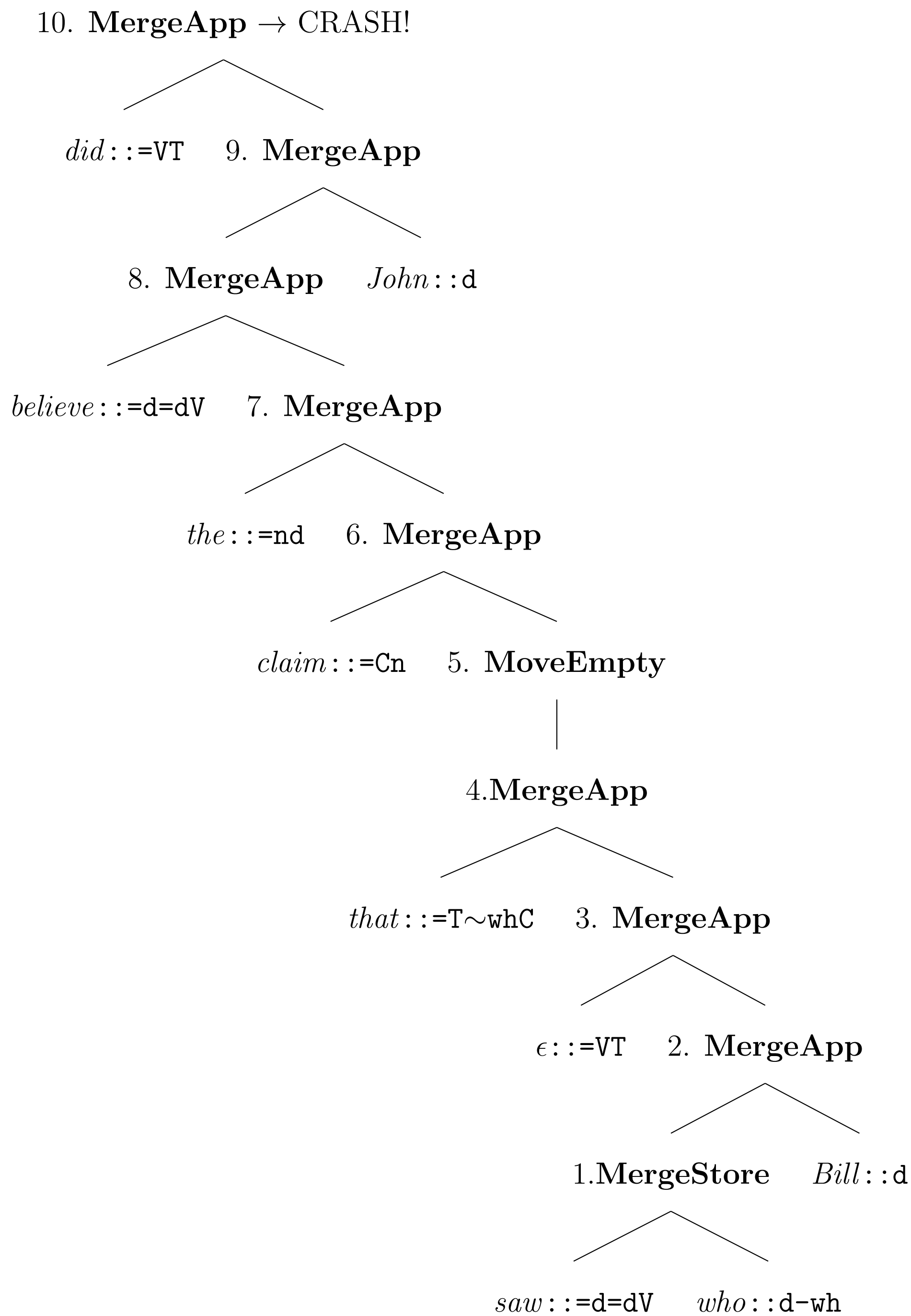
(153)



Now the CNPC can be accounted for in this modified MG system. The derivation tree and the subsequent crash of the derivation for the sentence (152) are given in (154).

(152) *[$_{CP2}$ Who did John [$_{TP}$ believe [$_{DP}$ the claim [$_{CP1}$ t_i that [$_{TP}$ Bill saw t_i]]]]]

(154)



The derived structure and the movers with bounding node counts in the semantic store at each step of the derivation are given in the table below.

Table 6.7: Bounding Nodes count in Semantic Store

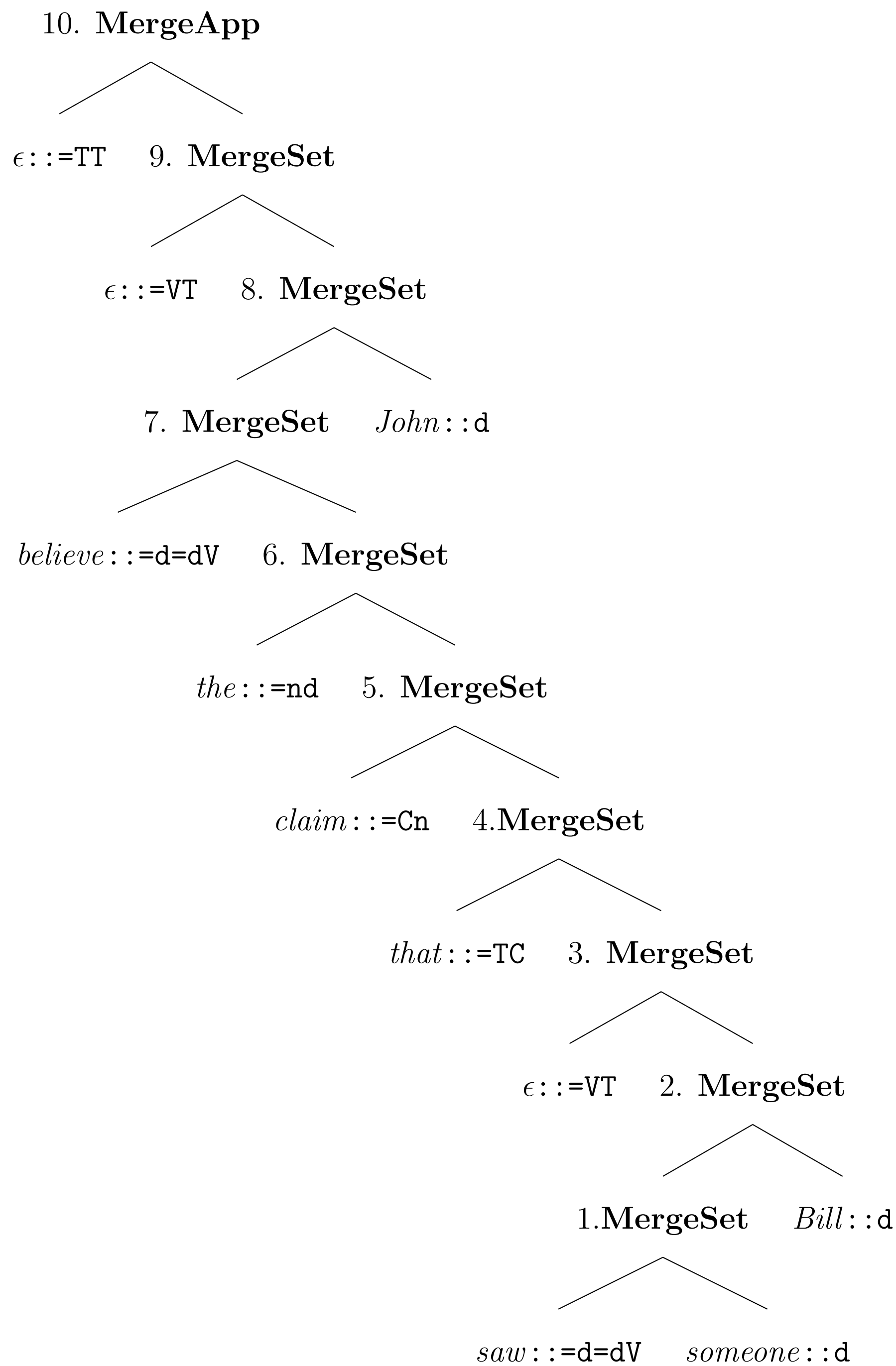
Steps	Derived Structure	Syntactic Store	Semantic Store
1.	saw`::=dV`	who`::-wh`	$\lambda P.WHy[Py]$, BN:0
2.	Bill saw`::V`	who`::-wh`	$\lambda P.WHy[Py]$, BN:0
3.	Bill saw`::T`	who`::-wh`	$\lambda P.WHy[Py]$, BN:1
4.	that Bill saw`::`$\sim$`whC`	who`::-wh`	$\lambda P.WHy[Py]$, BN:1
5.	that Bill saw`::C`	who`::-wh`	$\lambda P.WHy[Py]$, BN:0
6.	claim that Bill saw`::n`	who`::-wh`	$\lambda P.WHy[Py]$, BN:0
7.	the claim that Bill saw`::d`	who`::-wh`	$\lambda P.WHy[Py]$, BN:1
8.	believe the claim that Bill saw`::=dV`	who`::-wh`	$\lambda P.WHy[Py]$, BN:1
9.	John believe the claim that Bill saw`::V`	who`::-wh`	$\lambda P.WHy[Py]$, BN:1
10.	did John believe the claim that Bill saw`::=T`	who`::-wh`	$\lambda P.WHy[Py]$, BN:**2**

In (154), the BN count of *who* first increases by 1 in step 3 when the null element that has a category T merges into the derivation. The count returns to 0 in step 5 when *move* is caused by the ∼`wh` feature. The count increases again when *the claim* with category `d` merges in at step 7. The BN count increases to 2 when *did* with category `T` merges in at step 10, whereby the derivation crashes and can no longer continue.

In constrast to the example in (152), the sentence in (155), which has an indefinite in place of a *wh*-phrase, does not incur any violation. The meaning of the indefinite is not placed within the semantic store. Recall from the previous chapter that indefinites takes scope through set percolation instead, thus the semantic store remains empty. In this case, the syntactic store remains empty as well as there is no overt movement of any lexical items.

(155) $[_{CP2}$ John $[_{TP}$ believe $[_{DP}$ the claim $[_{CP1}$ t_i that $[_{TP}$ Bill saw **someone**]]]]]

(156)



10. MergeApp
ϵ::=TT
9. MergeSet
ϵ::=VT
8. MergeSet
7. MergeSet
John::d
believe::=d=dV
6. MergeSet
the::=nd
5. MergeSet
claim::=Cn
4.MergeSet
that::=TC
3. MergeSet
ϵ::=VT
2. MergeSet
1.MergeSet
Bill::d
saw::=d=dV
someone::d

Table 6.8: Bounding Nodes count for Indefinites

Steps	Derived Structure	Syntactic Store	Semantic Store
1.	saw someone::=dV	ϵ	BN:0
2.	Bill saw someone::V	ϵ	BN:0
3.	Bill saw someone::T	ϵ	BN:0
4.	that Bill saw someone::∼whC	ϵ	BN:0
5.	claim that Bill saw someone::C	ϵ	BN:0
6.	the claim that Bill saw someone::d	ϵ	BN:0
7.	believe the claim that Bill saw someone::=dV	ϵ	BN:0
8.	John believe the claim that Bill saw someone::V	ϵ	BN:0
9.	John believe the claim that Bill saw someone::=T	ϵ	BN:0
10.	John believe the claim that Bill saw someone::=T	ϵ	BN:0

In this implementation, *wh*-phrases with the wh feature uses both semantic (to take scope) and syntactic store (for overt movement) while indefinites use neither as they have a different mechanism to take scope and it do not overtly move. Given that *wh*-phrases with the wh feature uses the semantic store while indefinite do not, I will assume that the semantic store, but not the syntactic store, is island-sensitive. The example of *wh*-phrases with the ⊛F feature below further shows why the semantic store is island sensitive, but not the syntactic store.

A *wh*-phrase in situ, a *wh*-phrase with a d feature and no other licensing feature, has the same semantic system as an indefinite and a *wh*-phrase with the ⊛F feature. That is, it takes scope through set percolation and not semantic storage. Thus, with a *wh*-phrase in situ, the syntactic store is always empty and the BN count is always 0, similar to (6.8).

Table 6.9: Bounding Nodes count for *wh*-in situ

Steps	Derived Structure	Syntactic Store	Semantic Store
1.	saw who::=dV	ϵ	BN:0
2.	Bill saw who::V	ϵ	BN:0
3.	Bill saw who::T	ϵ	BN:0
4.	that Bill saw who::∼whC	ϵ	BN:0
5.	claim that Bill saw who::C	ϵ	BN:0
6.	the claim that Bill saw who::d	ϵ	BN:0
7.	believe the claim that Bill saw who::=dV	ϵ	BN:0
8.	John believe the claim that Bill saw who::V	ϵ	BN:0
9.	John believe the claim that Bill saw who::=T	ϵ	BN:0
10.	John believe the claim that Bill saw who::=T	ϵ	BN:0

A *wh*-phrase with the ⊛F feature does not move like a regular *wh*-phrase with a wh feature. It gets placed within the syntatic store, but its meaning does not get placed within the semantic store, as it takes scope through set percolation. Hence, the BN count is always 0. This is shown in Table 6.10.

Table 6.10: Bounding Nodes count for *wh*-phrases with ⊛F

Steps	Derived Structure	Syntactic Store	Semantic Store
1.	saw::=dV	who::⊛F	BN:0
2.	Bill saw::V	who::⊛F	BN:0
3.	Bill saw::T	who::⊛F	BN:0
4.	that Bill saw::C	who::⊛F	BN:0
5.	claim that Bill saw::n	who::⊛F	BN:0
6.	the claim that Bill saw::d	who::⊛F	BN:0
7.	believe the claim that Bill saw::=dV	who::⊛F	BN:0
8.	John believe the claim that Bill saw::V	who::⊛F	BN:0
9.	John believe the claim that Bill saw::=T	who::⊛F	BN:0

The table above is not complete as there are further interaction between ⊛F and Elide. The semantics of the derivation in Table 6.10 and 6.9 is similar to the one in Table 6.11 where everything is a set until the existential closure operator is merged in.

1.	$\{see(x,a),\ see(x,b)...\},\ \epsilon;\ \epsilon$
2.	$\{see(B,a),\ see(B,b)...\},\ \epsilon;\ \epsilon$
3.	$\{see(B,a),\ see(B,b)...\},\ \epsilon;\ \epsilon$
	...
9.	$\{$John believe the claim that $see(B,a)...\},\ \epsilon;\ \epsilon$
10.	$\bigcup\{$John believe the claim that $see(B,a)...\},\ \epsilon;\ \epsilon \equiv \exists x.[$John believe the claim that $see(B,x)]$

Table 6.11: Semantics of (6.10)

Despite the fact that the derivation in Table 6.10 is incomplete, the crucial difference between Table 6.10 and Table 6.7 is that in Table 6.10, the derivation does not crash during step 9; nothing is added to the semantic store, and the BN count does not increment.

Though the idea of subjacency has largely been disregarded in current syntactic theory, many notions of the current Phase theory still rely on the earlier subjacency condition. In some ways, phases correlate with the idea of bounding nodes where certain phases do not permit *wh*-movement to proceed. Through this implementation of subjacency in MGs, it seems that island violations might be more semantic in nature than syntactic. Nevertheless, this section does not aim to make any strong arguments for subjacency. The point of this section is just to show that it is possible to include constraints on movement in MGs so that movement across islands within the MG framework is ruled out as well. This implementation is needed to highlight the differences in a non-moving *wh*-phrase and a moving *wh*-phrase with regards to islands.

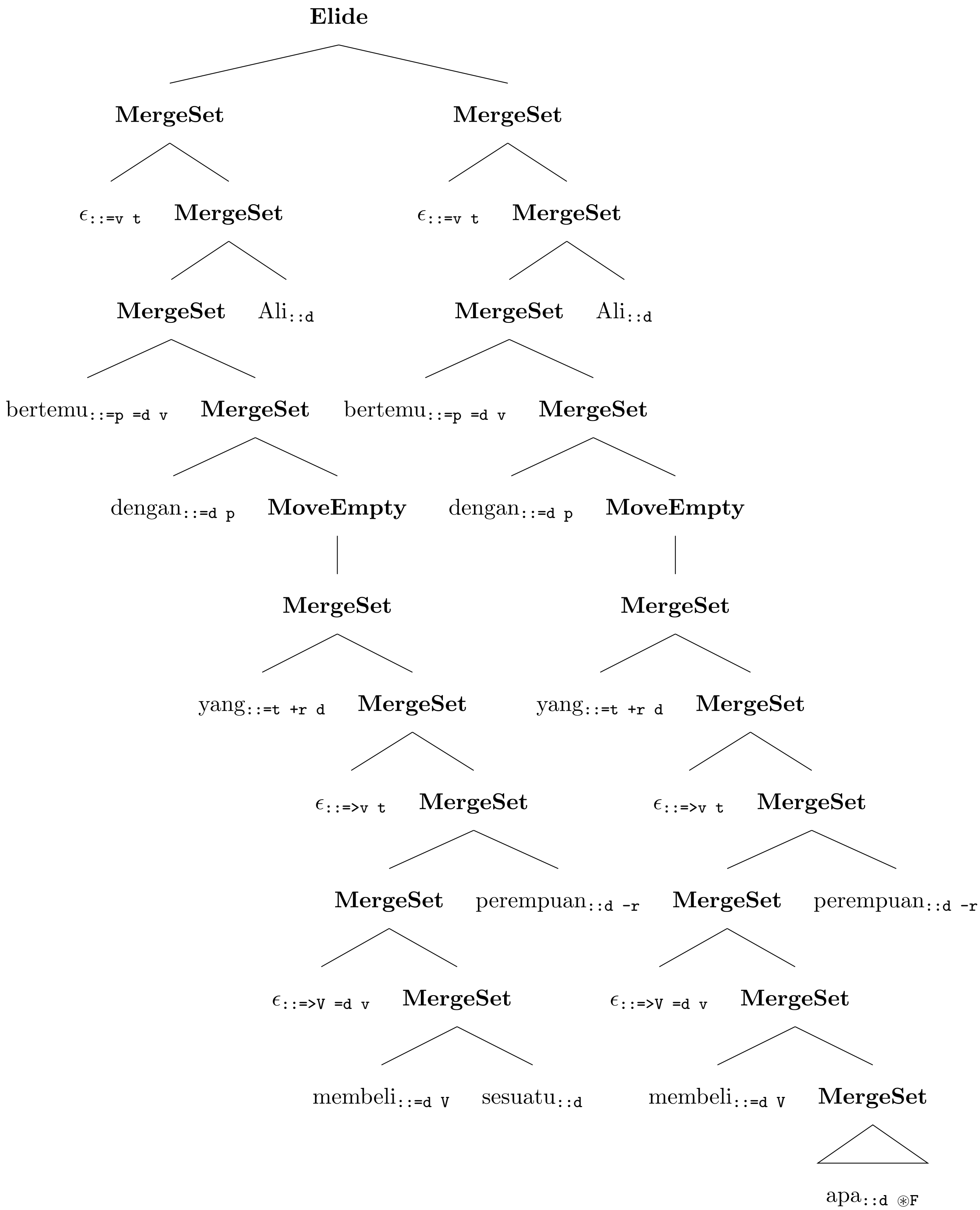
6.2.4 Derivation of Island Insensitivity in Malay

To end this section, I provide the derivation and the derived trees of (148) (repeated below) in conjunction with the Elide operation to show that the ⊛F feature solves the island (in)sensitivity problem.

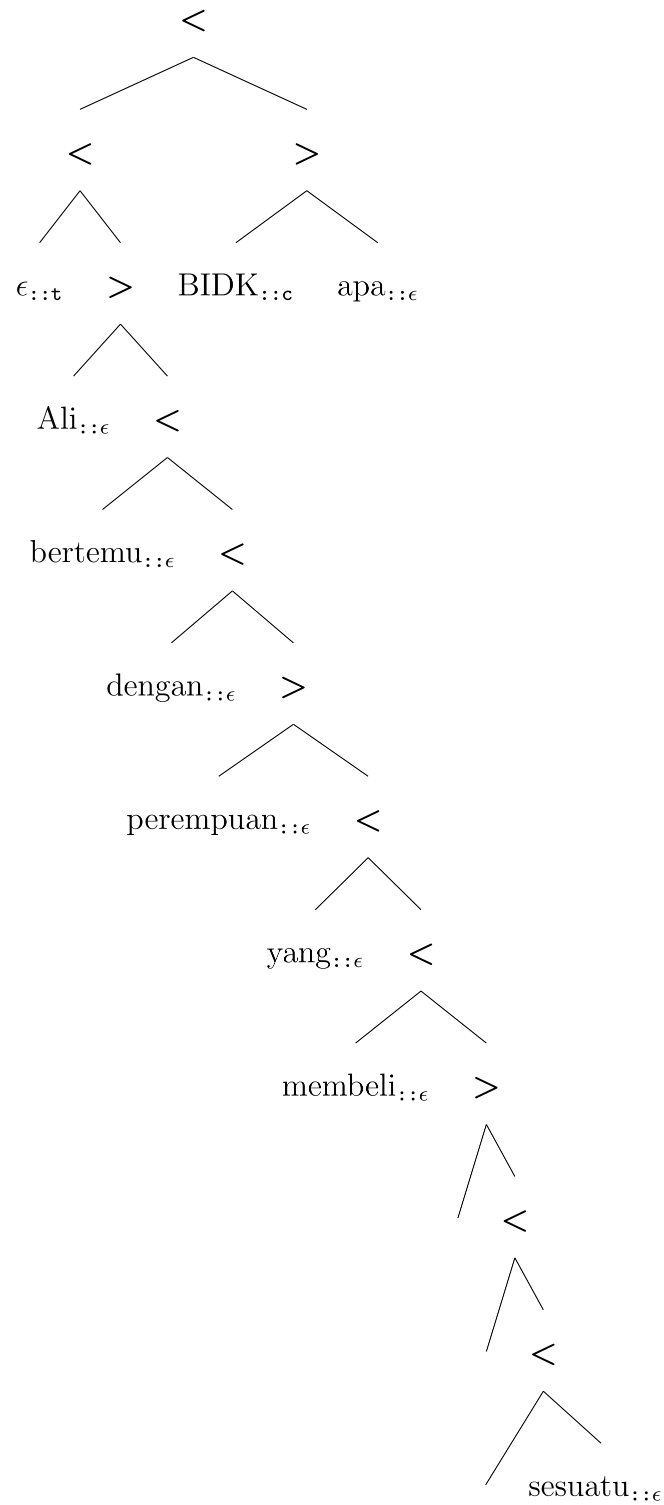
(148) Ali bertemu dengan [perempuan yang membeli sesuatu], tapi saya tak tahu apa
Ali met with woman COMP buy something, but I NEG know what

'Ali met the woman who bought something, but I don't know what.'

(157)


Elide
MergeSet
MergeSet
ϵ::=v t
MergeSet
ϵ::=v t
MergeSet
MergeSet
Ali::d
MergeSet
Ali::d
bertemu::=p =d v
MergeSet
bertemu::=p =d v
MergeSet
dengan::=d p
MoveEmpty
dengan::=d p
MoveEmpty
MergeSet
MergeSet
yang::=t +r d
MergeSet
yang::=t +r d
MergeSet
ϵ::=>v t
MergeSet
ϵ::=>v t
MergeSet
MergeSet
perempuan::d -r
MergeSet
perempuan::d -r
ϵ::=>V =d v
MergeSet
ϵ::=>V =d v
MergeSet
membeli::=d V
sesuatu::d
membeli::=d V
MergeSet
apa::d ⊛F

(158)



The *wh*-phrase is kept aside by the $\circledast$F feature while the rest of the structure is built. Given that a *wh*-phrase with the $\circledast$F denotes a set and takes scope via scope percolation, the

BN count does not increase during any part of the derivation in (157). The Elide operation on the top then destroys the rest of the ellipsis site leaving only the *wh*-phrase. How Elide applies to just the string version of (148) is shown in (159). BIDK is used in (159) and (157) to refer to the Malay version of BIDK, namely 'tapi saya tak tahu'.

(159) Elide

(Ali bertemu dengan perempuan yang membeli sesuatu) [(Ali bertemu dengan perempuan yang membeli $*$)(apa)]

= Ali bertemu dengan perempuan yang membeli sesuatu BIDK apa.

6.3 Preposition-Stranding

Recall from Chapter 2 that Malay does not obey Merchant's (2001) p-stranding generalization, which says languages that do not allow for p-stranding under regular *wh*-movement do not allow it under sluicing either. Malay does not allow for p-stranding under regular *wh*-movement but allows it under sluicing, indicating that a *wh*-movement analysis cannot be applied to sluicing. In (160), the preposition *kepada* is optional.

(160)

Ali	bagi	buku	kepada	seseorang,	tapi	saya	tak	tahu	(kepada)	siapa
Ali	give	book	to	someone,	but	I	NEG	know	(to)	who

'Ali gave a book to someone, but I don't know (to) who.'

Implementing p-stranding for Malay under sluicing is somewhat tricky. With regular *wh*-movement in questions, the preposition is always pied-piped to the front with the *wh*-phrase, while under sluicing, the preposition is optional. Ideally, we want an analysis where the pied-piped preposition under sluicing is derived similarly to the compulsary pied-pipping of prepositions in regular *wh*-movement in Malay. On the other hand, we also want the p-stranding of the *wh*-phrase be exclusive to sluicing. In the MG system here, the *wh*-remnant does not undergo regular *wh*-movement with the `wh` feature, but is kept aside with the ⊛`F` feature. A simple solution is to implement two different p-stranding systems for `wh` and ⊛`F`. However, the result of doing so would be trivial and there would not be any predictions regarding the implementation. A better solution is to use the link established between the `wh` and ⊛`F` features earlier on and implement p-stranding with some similarity between the two features. My implementation involves conversion and percolation of features.

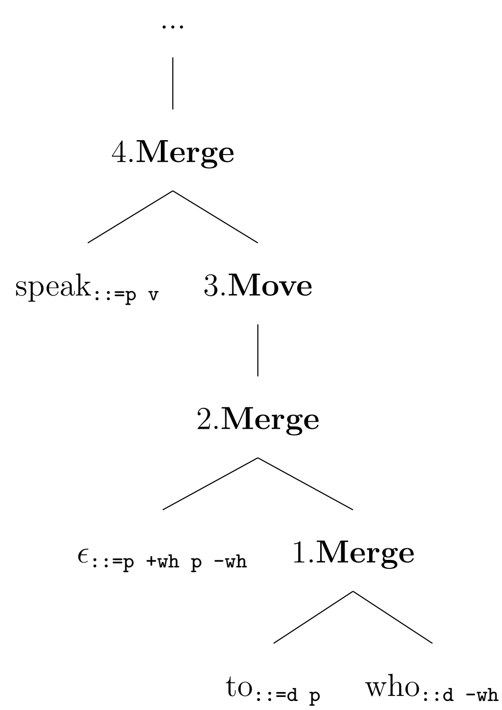
My implementation of p-stranding and pied-piping in Malay is based off Kobele's (2015) implementation of the same phenomena in English. Then, I show how pied-pipping is enfored in regular *wh*-movement in Malay before showing how pied-piping and p-stranding work in Malay sluicing. Finally, I address some ramifications of this particular implementation.

6.3.1 Kobele (2015): P-Stranding and Pied-piping in English

Kobele's implementation of preposition pied-piping in English (following his reading of Cable (2010)) uses feature percolation. In Kobele (2015), feature percolation, that results in pied-piping, is achieved through the presence of a silent lexical item (which I will call the percolator). The percolator has the features `=p +wh p -wh`[3]. The percolator is an implementation of feature percolation, as the `-wh` feature is "carried" further up the tree when the percolator is combined into the structure. The partial derivation tree for pied-piping in English is given in (161).

[3]Using this set of features will give the order 'who to' instead of 'to what' but this could be fixed with an additional rule for *move*. In this case, the *move* operation is covert so the word order does not change. The main idea is that the implementation forces both preposition and *wh*-phrase to move together so word order is ignored here.

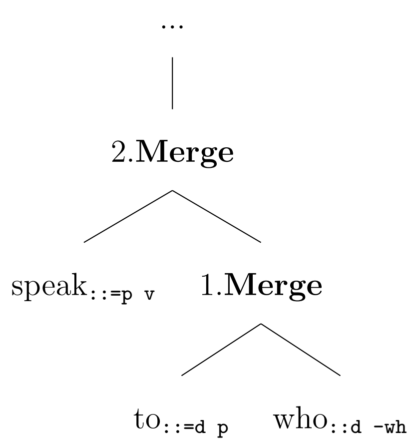
(161)



In step 1, the *wh*-phrase and preposition combine but *wh*-phrase is kept aside because of the `-wh` feature. In step 2, the percolation lexical item combines in and forces a *move* operation in step 3, which combines the *wh*-phrase and the preposition in the derived structure. However, since the lexical item has another `-wh` feature, the derived structure, *to who*, at step 3 has the feature list `p -wh`. In step 4, the verb *speak* selects for the moving preposition phrase. Essentially, Kobele's percolator changes a structure that the `p, -wh` feature to a structure with `p -wh`. A feature list with a comma in the middle refers to a structure with an object in storage. In the case of `p, -wh`, we have a `p` type object with a `-wh` item in storage waiting to move when a suitable licensor appears. This is how the percolator creates the pied-piped structure and allows the derivation to continue with the pied-piped structure ready to move.

When this lexical item is not present, we get a case of p-stranding as show in (162).

(162)



At step 1, the *wh*-phrase is kept aside and has the feature list `p, -wh`. At step 2, the preposition is combined with the verb in the derived structure while the *wh*-phrase is still waiting to move. In this implementation, the percolator is optional and this optionality comes from having the same selector and category features `=p +wh p -wh`. Because the selector and the category of the percolator are identical, the derived structure are the same category after the lexical item is combined in. Thus, the structure created with or without the percolator can still be selected by the same verb as seen in step 4 of (161) and step 2 of (162). In the next section, I show how a slight modification to the percolator can make p-stranding unavailable for *wh*-movement in Malay.

6.3.2 Obligatory Pied-piping in *wh*-movement

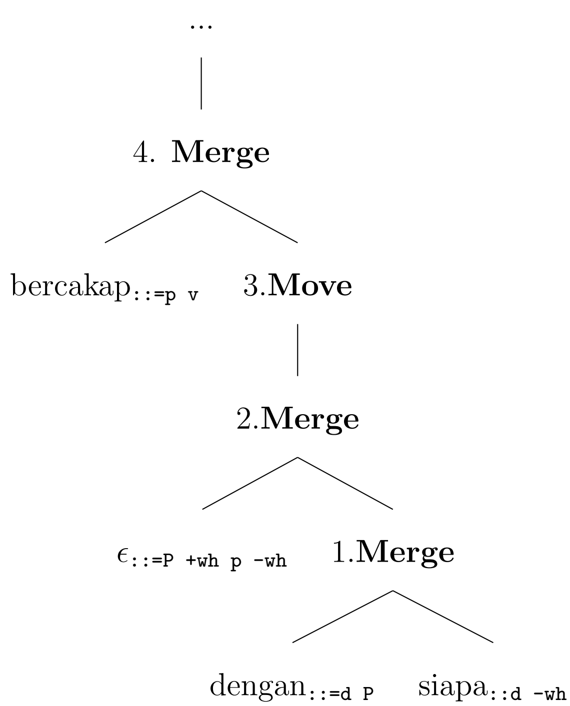
As shown before in Chapter 2, Malay does not allow prepositions to strand under regular *wh*-movement. If there is *wh*-movement, the preposition always get pied-piped along, as shown in (163).

(163) a. *siapa yang Ali bercakap dengan?
who COMP Ali speak with

b. dengan siapa yang Ali bercakap?
with who COMP Ali speak
'With whom does Ali speak?'

For Malay, I modify Kobele's implementation of English preposition pied-piping to make it obligatory in Malay. As discussed in the section before, the percolator in Kobele's implementation is optional because of the same selector and category. Hence, by making the selector and category features different, we can make obligatory the percolator and pied-piping of the preposition in Malay. The feature for the percolator in Malay that I use is `=P +wh p -wh`. This means that Malay prepositions have `P` as their category as oppose to `p`. Verbs in Malay are only able to select for a preposition phrase that has `p` as its category and not `P`. As a result, the percolator is now obligatory and preposition pied-piping will always occur with *wh*-movement. The goal for this section is to create a structure that would be `p -wh`, with the preposition combined in with the *wh*-phrase, and avoid `p, -wh` because that would mean that the preposition is not combined and the *wh*-phrase is still waiting to move. The partial tree using 'bercakap dengan siapa' (speak with who) with preposition pied-piping is shown in (164).

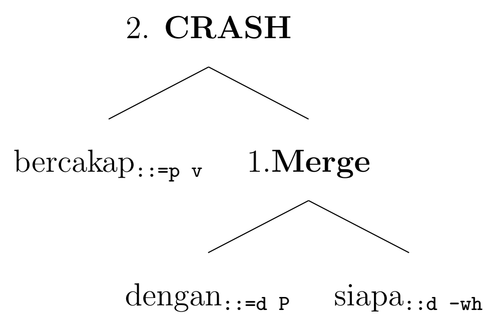
(164)



The percolator in step 2 forces *siapa* (who) to be combined into the derived structure with *dengan* (with). At step 3, the derived structure 'dengan siapa' has the features `p -wh` and will be kept aside. At step 4, the verb *bercakap* (speak) selects for the preposition phrase with `p` category. This slight change to the feature list of the percolator forces it to be obligatory when prepositions and *wh*-phrases are present. This percolator essentially changes structures that have features `P, -wh` to structures that have the `p -wh` features.

(165) shows a derivation crashing when the percolator is absent.

(165)



The derivation crashes at step 2 because *bercakap* (speak) is unable to select for the preposition phrase that has category `P`. By having the distinction between `p` and `P`, preposition pied-piping is always obligatory for *wh*-movement in Malay. The percolator is no longer optional and is required to allow the derivation to continue as verbs that combine in can only select for `p`.

Besides bringing the `-wh` feature further up the tree, the percolator also transforms `P` to `p` in order to allow the verb to select the prepositional phrase. In a declarative sentence with a preposition such as (166), the same rules apply where the preposition has a `P` feature while the verb selects for `p`.

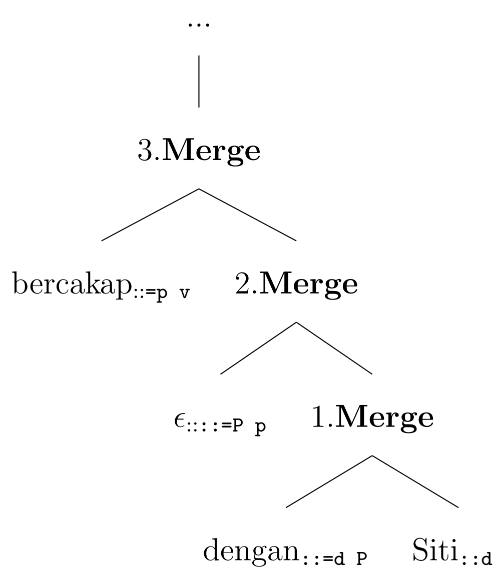
(166) Ali bercakap dengan Siti
Ali speak with Siti
'Ali spoke with Siti.'

However, there is no *wh*-phrase that will undergo movement in a declarative sentence. So the percolator cannot be used in this instance to transform `P` to `p`. In this case, we need a different converter to make the change. For now, I will use a silent lexical item that is `=P p`[4]

[4] `=P p` is not right as we will see in the section further down. For now, we just need something that will get us from `P` to `p`.

to change the category of the prepositional phrase. The derivation tree for (166) is given below:

(167)



At step 2, the lexical item that transforms `P` to `p` combines in, which allows the derivation to continue for the declarative sentence. Having established the converter and percolator for Malay, I move onto the implementation of p-stranding and pied-piping in Malay sluicing.

6.3.3 What happens under Sluicing?

Now that I have established obligatory preposition pied-piping with *wh*-movement in Malay, I will now show the implementation for pied-piping and p-stranding under sluicing. As a reminder, sluicing in Malay allows for p-stranding eventhough normal *wh*-movement does not allow for it as shown in (168).

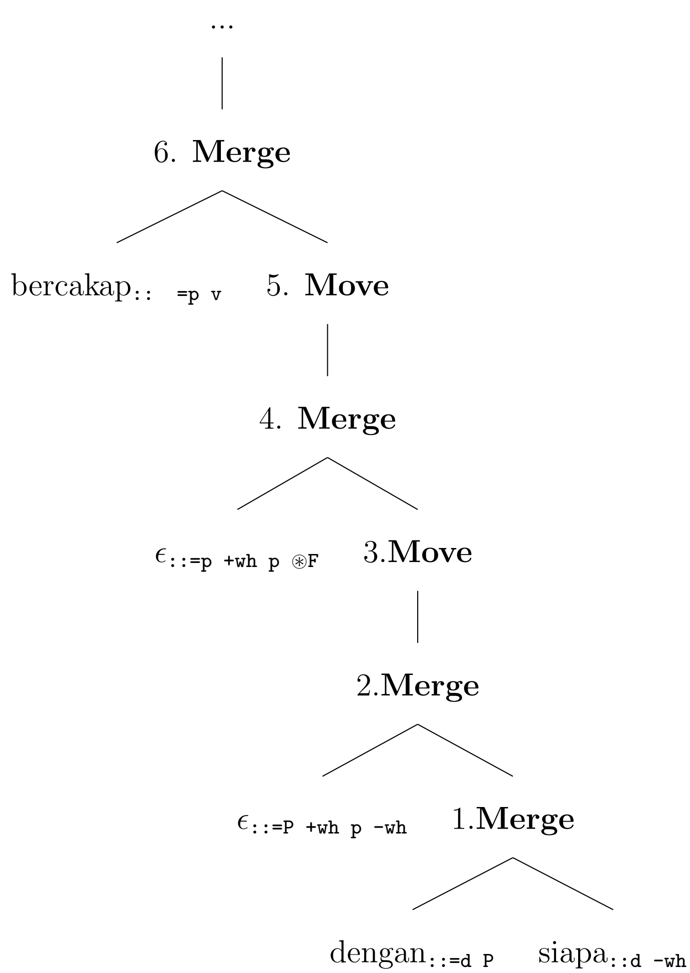
(168) Ali bercakap dengan seseorang, tapi saya tak tahu (dengan) siapa?
Ali speak with someone, but 1.Sg Neg know (with) who
'Ali spoke with someone, but I dont know (towith) who.'

Thus, the goal of this section is to account for how the preposition gets pied-piped or stranded from the *wh*-remnant. Ideally, the pied-piped preposition under sluicing would follow a similar pattern to pied-piped prepositions in *wh*-movement, while also being similar to a *wh*-remnant, in order to allow Elide to operate. On the other hand, a stranded preposition will not have the pied-piping mechanism. The implementation for this pattern will involve the percolator and/or converter that was introduced in the sections above.

6.3.4 Pied-piping under Sluicing

I will first discuss preposition pied-piping with the *wh*-remnant where both percolator and converter are involved. The pied-pipe mechanism follows the same as *wh*-movement using the percolator with `=P +wh p -wh`, thus the beginning of the derivation is similar to (164). This allows the preposition to combine in the same way as *wh*-movement to a would-be *wh*-remnant. For Elide to operate on structures, we need the `⊛F` feature and not `-wh`. Therefore, to keep the same pied-piping idea but allow for sluicing, we need to convert the `-wh` feature into `⊛F`. This is done through the conversion process with the converter that is `=p +wh p ⊛F`. The partial derivation tree for *dengan siapa* (with who), with pied-piping of the preposition under sluicing is shown in (169).

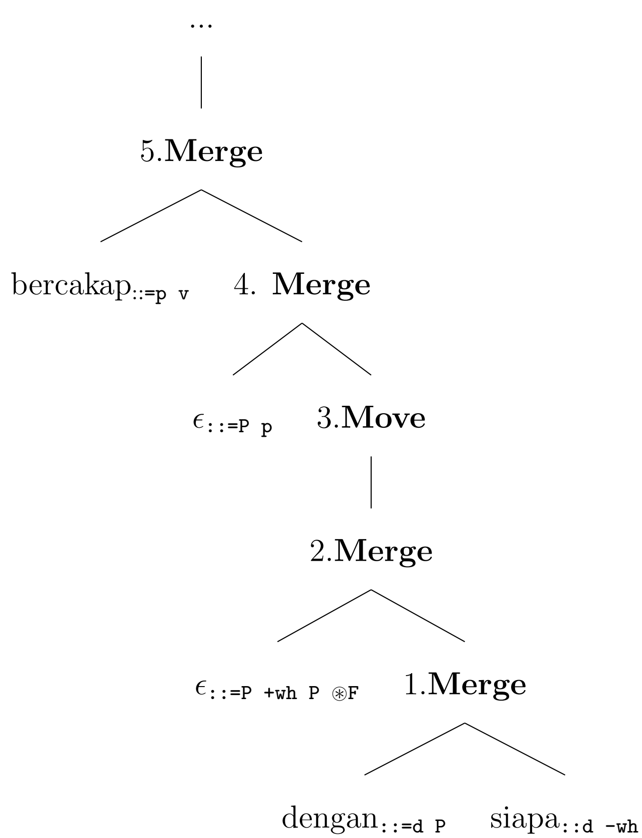
(169)



At step 4, the converter combines in after pied-piping has occured in step 1-3. The `-wh` feature is then "converted" into the ⊛F feature after it is checked during the *move* operation in step 5. At step 5, the derived structure is now the preposition combined with the wh-phrase (*dengan siapa*). The derivation then precedes as normal with the verb selecting for the `p` feature while *dengan siapa* (with who) kept to the side by ⊛`F`. By using the percolator and converter, the changed to structure started out with `P, -wh` feature to `p -wh` (same as obligatory pied-piping in *wh*-movement) to `p` ⊛`F`.

The derivation that is presented above is not the only derivation for pied-piping under sluicing. An alternative way for pied-piping to be derived is by merging the converter that is `=P + wh P` ⊛`F` right after the preposition and using the coverter that changes `P` to `p`. The partial derivation tree for this alternative derivation is given below:

(170)



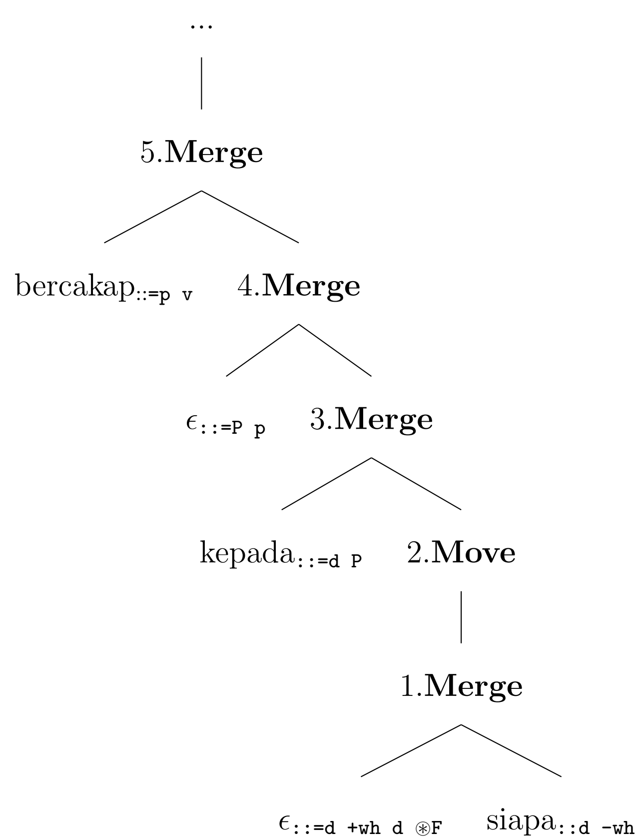
At step 2, the converter that is `=P +wh P ⊛F` combines in which concatenates the preposition with the *wh*-phrase. At step 4, the `=P p` convertor combines in which converts `P` to `p` and allows the derivation to continue. By using these two converters, the change to the structure began with `P, -wh` to `P ⊛F` to `p ⊛F`. The derivation in (170) shows that there are other ways to derive pipe-piping in sluicing that do not involve the `=P +wh p -wh` percolator. This is because percolation comes for free with the converter `=x +wh x ⊛F` as we are allowed to do conversion at different points compared to the percolator. With *wh*-movement, the percolator is the only way to get pied-piping with the *wh*-phrase. Since the converter is allowed to combine at different points in the derivation, it gives rise to the prediction that we get more cases of pied-piping under sluicing compared to *wh*-movement. This prediction is discussed in the section below.

6.3.5 P-stranding under Sluicing

In the first case of pied-piping, the converter is combined after the percolator to ensure that the preposition is attached together with the *wh*-phrase before both structure are kept aside by the ⊛`F` feature. In the alternative case of pied-piping, the converter is combined straight after the preposition, which also allows the preposition to attach with the *wh*-phrase before they are kept aside. With p-stranding under sluicing,it should be obvious that there is no need for the percolator, only the converter[5]. Now the question is when does the converter combine into the derivation? For the case of p-stranding, the converter has to be combined lower in the derivation tree right after the *wh*-phrase combines in. The partial derivation tree is shown in (171).

[5] The current account of sluicing with Elide (without any new lexical items) would not have a problem implementing p-less sluices in Malay. Since the ⊛F feature keeps the *wh*-phrase from merging into the derivation, we technically do not have to change anything to get p-stranding under sluicing in Malay. However, since we established a link between `-wh` and ⊛`F` features, the converstion mechanism is extended to all sluicing cases.

(171)



At step 1, the conversion lexical item combines in and the `-wh` feature on the *wh*-phrase is checked at step 2. The *wh*-phrase is kept aside by the `⊛F` feature eventhough the preposition combines in at step 3. This results in the stranding of the preposition: the preposition does not come in contact with the `-wh` or `⊛F` feature and is combined immediately into the derived structure, whereas the wh-phrase is kept aside. In this case, the structure changed by the converter goes from `d -wh` to `d ⊛F`. To allow the verb to select for the `p` feature, the other converter that changes `P` to `p` needs to be combined at step 4. As mentioned before, the reasoning behind a converter that is `=P p` will be discussed in the section below. The converter cannot be combined in after the preposition as the derivation will end up with pied-piping. This would be the same derivation tree as shown in (170). In the case of pied-piping under sluicing, the conversion happens very early on in the derivation and this stops any pied-piping mechanism from applying.

6.3.6 Prediction and Ramification of the Implementation

Before moving onto a prediction made by the implementation, I first summarise the structures that the percolator and converter have allowed us to have. The main goal of this chapter is to allow p-stranding and pied-piping with *wh*-remnants, feature change from `x -wh` to `x ⊛F` and feature change from `P, -wh` to `p -wh`, while disallowing the feature change from `P, -wh` to `p, -wh`. This is because the feature change from `P, -wh` to `p, -wh` allows for p-stranding under regular *wh*-movement. The feature change for each phenomenon with percolator and/or converter is summarised in the table below:

	Percolator and/or Converter	Feature Change
(a) Obligatory p pied-piping	`=P +wh p -wh`	`P, -wh`→ `p -wh`
(b) p-stranding	`=d +wh d ⊛F`	`d -wh`→ `d ⊛F`
(ci) pied-piping	`=P +wh p -wh`, `=p +wh p ⊛F`	`P, -wh` → `p -wh` → `p ⊛F`
(cii) pied-piping	`=P +wh P ⊛F`, `=P p`	`P, -wh`→ `P ⊛F` → `p, ⊛F`

Table 6.12: Feature change with percolator and/or converter

Given the system created above, there are two issues to be address. The first is a prediction: if we allow for the change `x -wh` → `x ⊛F` with the converter `=x +wh x ⊛F`, we would also allow for `x, -wh` → `x ⊛F`. The change from `x, -wh` to `x ⊛F` is noted with the alternative derivation of pied-piping in (170). As mentioned before, the converter predicts more pied-piping under sluicing compared to *wh*-movement. A converter that results in a structure that is `x, -wh` would predict massive pied-piping with the *wh*-remnant. The second is a ramification of the system: this is seen in the additional derivation for pied-piping and the *wh*-remnant p-stranding section where we need the additional converter `=P p` in certain cases to allow for the right category for the selecting verb. The two issues are further explained and addressed below.

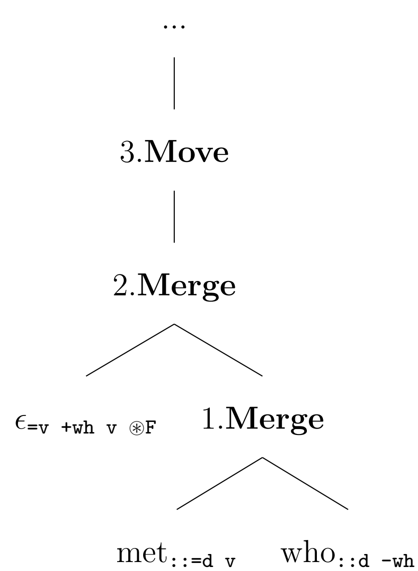
6.3.6.1 Prediction: Massive Pied-piping

In this MG system, the converter, `=x +wh x ⊛F`, is allowed to combine at any maximal projection in the derivation. In the context of prepositions, having the converter combine right after the *wh*-word before the preposition, as seen in (b) in Table 6.12, results in p-stranding. On the other hand, having it combine after the preposition results in pied-piping, (c) in Table 6.12. In the first case, we have a feature change from `x -wh` to `x ⊛F`; in the second case, we have a feature change from `x, -wh` to `x ⊛F`.

The problem with the converter comes from the second case. By allowing `x, -wh` → `x ⊛F`, we predict indiscriminate pied-piping with *wh*-remnants. This is because the *wh*-phrase is kept aside by `-wh` (as noted with the comma), while the derivation continues to build up more structure. When the converter finally combines in, the *wh*-phrase is combined with whatever structure that is built and everything becomes part of the remnant. This can lead to grammatical sentences such as the case of pied-piping in (cii) of Table 6.12, with the change from `P, -wh`→ `P ⊛F`. However, we would also predict the availability of changes such as `v, -wh` → `v ⊛F`. In this case, the verb gets pied-piped with the *wh*-phrase and this results in the ungrammatical sentence in (172). This partial derivation tree is shown in (173).

(172) *John met someone, but I don't know met who.

(173)



The converter is allowed to combine in at Step 2, after the verb and the *wh*-remnant are combined in step 1. This derivation yields the sentence in (172), which is definetely not desired. A possible fix to stop sentences such as (172) is to limit possible converters to just `d` and `p`, `=d +wh d ⊛F` and `=p +wh p ⊛F`. This would still lead to some undesirable sentences such as the pied-piping of a bigger DP in (174).

(174) *John bought pictures of someone, but I don't know [pictures of who].

While sentences such as (174) (or known as massive pied-piping) is still bad under sluicing, there are cases where massive pipe-piping seems to be allowed with sluicing. Ross (1969) observes that while massive pied-piping is not allowed when an indirect question appears in its usual position as in (174), it is allowed when it is fronted (or inverted), as in (175).

(175) He has pictures of somebody, but [pictures of who] I dont know.

In (175), the *wh*-remnant and the pied-piped DP, 'pictures of who', are fronted before 'I don't know' and the sentence is grammatical. The same word order can be applied to verbs pied-piped by *wh*-remnants, as seen in (176).

(176) a. He spent the entire day doing something at the mall, but [doing what] I dont know.

b. *He spent the entire day doing something at the mall, but I dont know [doing what].

In (176b), pied-piping of the verb by the *wh*-remnant is not allowed but once it is fronted, in (176a), the sentence is grammatical. Abe (2015) argues for topicalization under his *wh*-in situ sluicing analysis, while Abels (2019) argues for recursive contrastive left-dislocation accompanied by clausal ellipsis.

While I do not have an answer to the word order and grammaticality of front objects pied-piped by *wh*-remnants under the MG system here, it seems plausible that *wh*-remnants are allowed to build up and pied-pipe bigger structures in certain cases. It becomes slightly clearer when we contrast the grammaticality of massive pied-piping between *wh*-movement, sluicing, and inverted sluicing as shown in (177) and (178).

(177) a. *John bought pictures of someone, but I don't know pictures of who John bought.

b. *John bought pictures of someone, but I don't know pictures of who.

c. John bought pictures of someone, but pictures of who I don't know.

(178) a. *John met someone, but I don't know met who John did.

b. *John met someone, but I don't know met who.

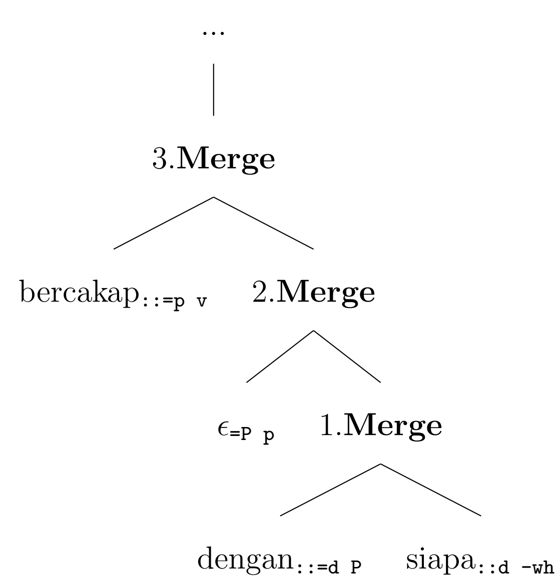
c. John met someone, but met who I don't know.

The sentences in (177) pied-pipe DPs while the sentences in (178) pied-pipe VPs. The (a) sentences of (177) and (178) attemp to massive pied-pipe with *wh*-movement, while the (b) and (c) sentences are of pied-piping under sluicing and inverted sluicing respectively. On a gradient scale, the sentences with *wh*-movement in (a) are a lot worse compared to (b) (eventhough (b) does not sound completely good either) and (c). Thus, it seems that there is more lee way for massive pied-piping with *wh*-remnants in sluicing as compared to *wh*-phrases in *wh*-movement. The rest is left to future research.

6.3.6.2 Ramification: Additional Converters

In the sections before, I introduced another converter `=P p` for pied-piping as well as p-stranding under sluicing. In such cases, we need to be able to change `P` to `p`, but we are unable to use the percolator with the `=P +wh p -wh` feature, because there are no wh features left in the derivation. However, converting using `=P p` results in unwanted derivations. Our goal for prepositions in Malay is to have obligatory pied-piping in regular *wh*-movement, but optional pied-piping under sluicing. Hence, we have to disallow the change from `P, -wh` to `p, -wh` because this leads to p-stranding in regular *wh*-movement as shown by the partial derivation tree in (179).

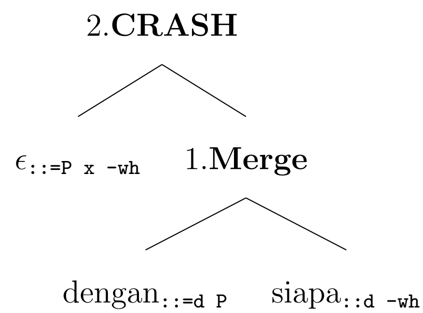
(179)



Since the converter at step 2 does not have any `wh` features, the *wh*-phrase continues to be kept in storage which eventually leads to p-stranding in regular *wh*-movement. Thus, we need another way to get from `P` to `p` without the converter `=P p`.

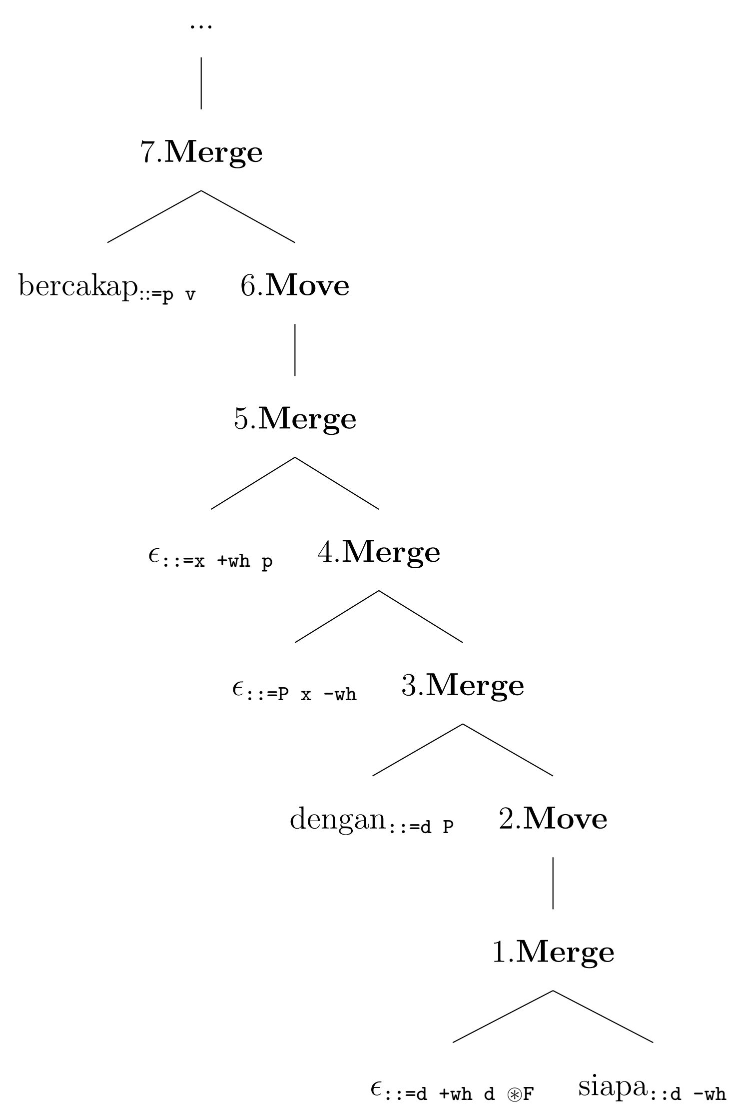
One way to get this conversion without resorting to `=P p` is to break it up into a two-step process with two converters instead of one. The two converters are `=P x -wh` and `=x +wh p`, in that order. This two-step process stops the derivation in (179) because it invokes the Shortest Movement Constraint (SMC), which prohibits two of the same licensee features in storage. This is shown in the derivation tree below:

(180)



The derivation crashes at step 2 because the *wh*-phrase *siapa* with the `-wh` feature is already in storage. The converter at step 2 introduces another `-wh` which leads to an SMC violation. The correct tree for p-stranding under sluicing in Malay with the two additional converters is shown in (181).

(181)



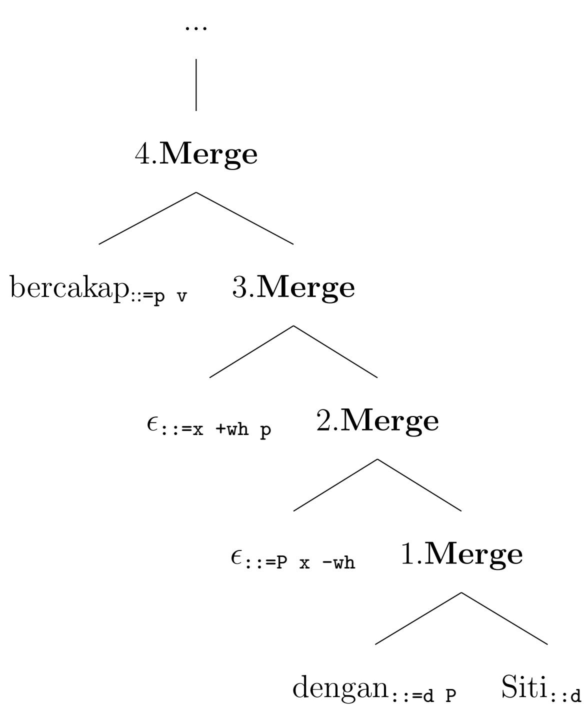
At step 4, the first converter that selects for `P` and has category `x` is combined into the system with a `-wh`. The second converter does the converting as it selects for `x`, checks the `-wh` feature and is of category `p`. When the `-wh` is checked by *move* in step 6, the resulting structure has the right category for the selecting verb that combines in in step 7. The `wh` feature in the two converters does not do anything in this case because the real `wh` feature has already been checked and converted to ⊛`F` in step 2.

A slightly unattractive consequence of implementing the conversion with the two-step converters is that we get the `wh` feature in sentences that does not look like it should have wh feature. An example of such sentence is the declarative sentence seen in (166) (repeated below).

(166) Ali bercakap dengan Siti
Ali speak with Siti
'Ali spoke with Siti.'

In the declarative sentence above, the two converters appear because of the preposition with the P feature as shown in the following derivation tree.

(182)



In order for the verb *bercakap* (speak) to be able to select the prepositional phrase with the p feature, we must use the two-step converters in steps 2 and 3 to change the P to p feature. This two-step conversion is not the most elegant solution for the problem, but it is a result of the MG system built so far and I leave the possibility for a more elegant solution for future work.

6.4 Scope Effects

In Malay, sentences that have universal quantifiers in the subject have two readings. The sentence in (183) is ambiguous, with the indefinite *sesuatu* (something) interpreted with wide or narrow scope with respect to the universally quantified DP *semua pelajar* (every student).

(183) semua pelajar itu membeli sesuatu, tapi saya tak tahu apa
all student the bought something, but 1.Sg NEG know what
'Every student bought something, but I don't know what.'

The readings here are parallel to the readings obtained from *wh*-in situ. The *wh*-in situ question in (184a) has both wide and narrow scope readings with respect to the universal quantifier. In contrast, the moved *wh*-phrase in (184b) only has the wide surface scope reading.

(184) a. semua pelajar itu membeli **apa**? (Readings: $\forall > wh$, $wh > \forall$)
all student the buy what

b. **apa** yang semua pelajar itu beli?(Readings: $\forall > wh^{\star}$, $wh > \forall$)
what COMP all student the buy
'What did every student buy?'

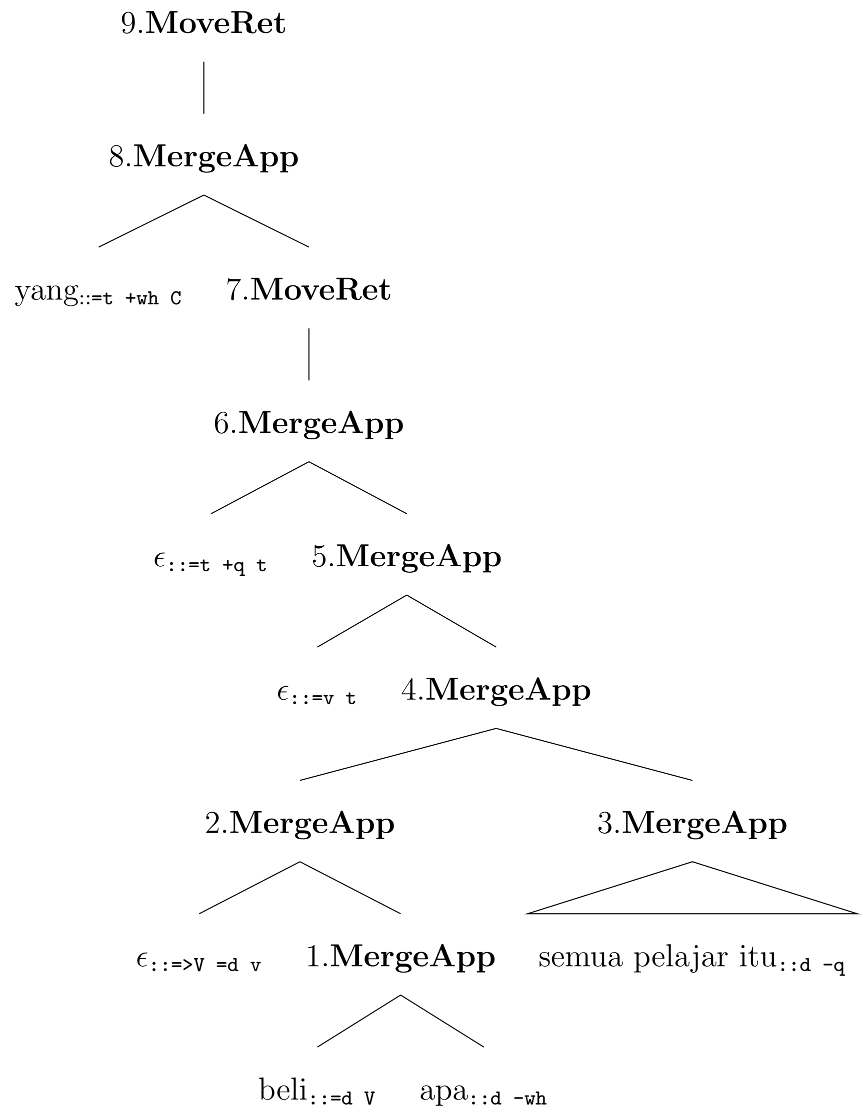
The MG implementation for *wh*-in situ and a moved *wh*-phrase such as seen in (184) can be done using the operations that have been defined. However, from the way that Elide has been implemented, we are unable to get the two readings in (183).

6.4.1 Two Potential Inconsistencies

The different readings between (184a) and (184b) can be attributed to the ways that *wh*-in situ and a moving *wh*-phrase take scope. As mentioned in the subjacency section of this chapter, *wh*-in situ takes scope via alternative set percolation while a moving *wh*-phrase takes scope via semantics storage. Taking scope via set percolation gives more flexibility with scope readings since the existential closure operator is allowed to combine in more than one location. On the other hand, a moving *wh*-phrase is limited to where a lexical item with `+wh` combines, and that is usually at the CP level.

First, I show how a moving *wh*-phrase with the `+wh` feature gets its only reading. The derivation tree and the semantics for the sentence in (184b) is given below.

(185)



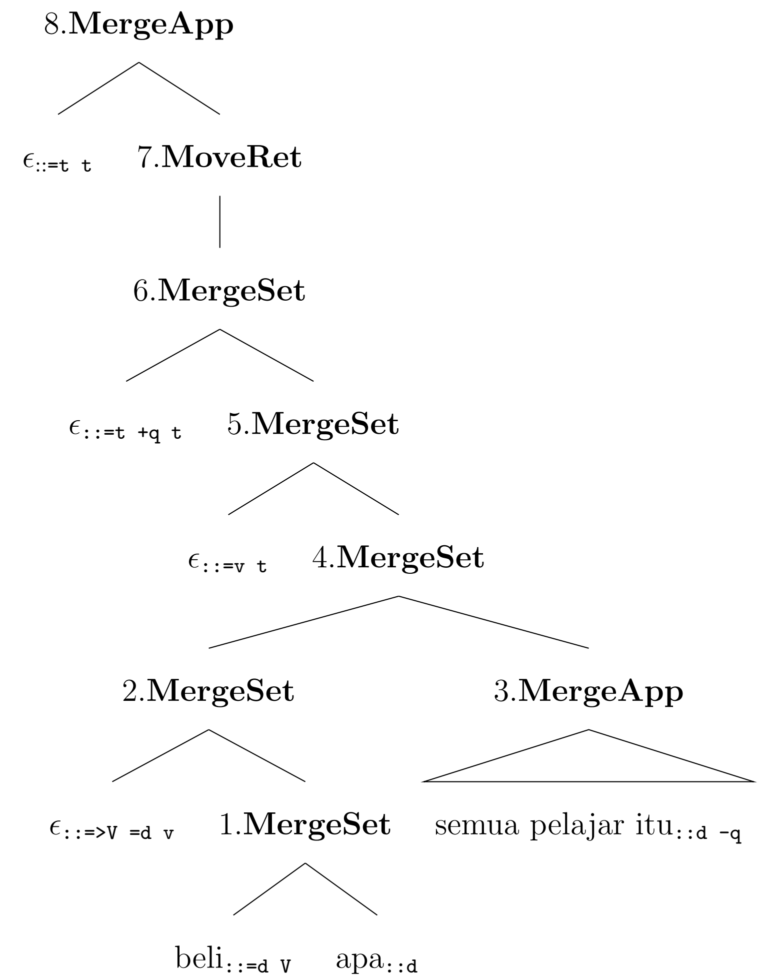
Steps	Main Meaning	Semantic Store
1.	$\lambda x.beli(x,a)$, `-wh`	($\lambda P.WHy[Py]$, a, `-wh`)
2.	$\lambda x.beli(x,a)$, `-wh`	($\lambda P.WHy[Py]$, a, `-wh`)
3.	$\lambda Q.\forall y(pelajar(y)$, `-q`	ϵ
4.	$\lambda x.beli(b,a)$, `-wh -q`	($\lambda P.WHy[Py]$, a, `-wh`), ($\lambda Q.\forall y(pelajar(y)\rightarrow Qy)$, b, `-q`)
5.	$\lambda x.beli(b,a)$, `-wh -q`	($\lambda P.WHy[Py]$, a, `-wh`), ($\lambda Q.\forall y(pelajar(y)\rightarrow Qy)$, b, `-q`)
6.	$\lambda x.beli(b,a)$, `-wh -q`	($\lambda P.WHy[Py]$, a, `-wh`), ($\lambda Q.\forall y(pelajar(y)\rightarrow Qy)$, b, `-q`)
7.	$\forall y(pelajar(y)\rightarrow beli(y,a)$, `-wh`	($\lambda P.WHy[Py]$, a, `-wh`)
8.	$\forall y(pelajar(y)\rightarrow beli(y,a)$, `-wh`	($\lambda P.WHy[Py]$, a, `-wh`)
9.	$WH.x\forall y(pelajar(y)\rightarrow beli(y,x)$	ϵ

Table 6.13: Semantics of (185)

In (185), the moving *wh*-phrase combines in step 1 and its meaning is placed in the semantic store. When the universal quantificational phrase combines in step 4, its meaning is also placed in the semantic store. At step 6, the lexical items that have the `+q` feature combine in, which allows the stored meaning of the universally quantified phrase to be retrieved from the store. At step 8, the lexical item with `+wh` merges in and the meaning of the *wh*-phrase is retrived from the store at step 9. The moved *wh*-phrase always has higher scope than other quantified phrases because the lexical item that has the `+wh` feature is category `C`, the CP phrase. The other quantified phrases that use the semantic store can only be retrived at the TP level which is always below the C.

For the *wh*-phrase in situ, it follows the way indefinites take scope using sets as seen in chapter 3. Since it uses set percolation, the semantic store only has the meaning of the universal quantified phrase. The two derivation trees and semantics for the two readings for the *wh*-phrase are given below.

(186)

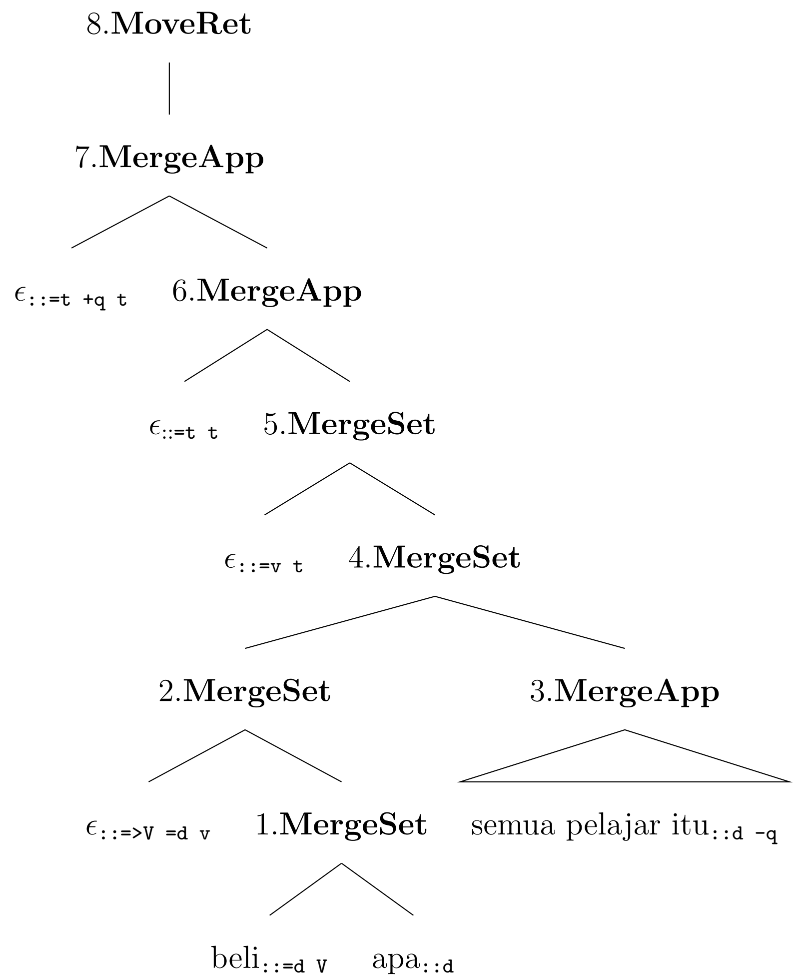


Steps	Main Meaning	Semantic Store
1.	$\{$*beli(x,a)*, *beli(x,b)*...$\}$	ϵ
2.	$\{$*beli(x,a)*, *beli(x,b)*...$\}$	ϵ
3.	$\lambda Q.\forall y($*pelajar(y)*, `-q`	ϵ
4.	$\{$*beli(c,a)*, *beli(c,b)*...$\}$, `-q`	$(\lambda Q.\forall y($*pelajar(y)*$\rightarrow Qy)$, c, `-q`)
5.	$\{$*beli(c,a)*, *beli(c,b)*...$\}$, `-q`	$(\lambda Q.\forall y($*pelajar(y)*$\rightarrow Qy)$, c, `-q`)
6.	$\{$*beli(c,a)*, *beli(c,b)*...$\}$, `-q`	$(\lambda Q.\forall y($*pelajar(y)*$\rightarrow Qy)$, c, `-q`)
7.	$\{\forall y$*(pelajar(y)*$\rightarrow$*beli(y,a)*...$\}$	ϵ
8.	$WH.x\forall y$*(pelajar(y)*$\rightarrow$*beli(y,x)*	ϵ

Table 6.14: Semantics of (186)

In (186), the *wh*-phrase is able to have a wide scope reading over the universally quantified phrase because its set of meaning is closed at step 8 while the meaning of the universally quantified phrase is retrieved at step 7.

(187)



Steps	Main Meaning	Semantic Store
1.	$\{$*beli(x,a)*, *beli(x,b)...*$\}$	ϵ
2.	$\{$*beli(x,a)*, *beli(x,b)...*$\}$	ϵ
3.	$\lambda Q.\forall y($*pelajar(y)*, `-q`	ϵ
4.	$\{$*beli(c,a)*, *beli(c,b)...*$\}$, `-q`	$(\lambda Q.\forall y($*pelajar(y)*$\rightarrow Qy),\ c,$ `-q`$)$
5.	$\{$*beli(c,a)*, *beli(c,b)...*$\}$, `-q`	$(\lambda Q.\forall y($*pelajar(y)*$\rightarrow Qy),\ c,$ `-q`$)$
6.	$WHx.$*beli(c,x)*, `-q`	$(\lambda Q.\forall y($*pelajar(y)*$\rightarrow Qy),\ c,$ `-q`$)$
7.	$WHx.$*beli(c,x)*, `-q`	$(\lambda Q.\forall y($*pelajar(y)*$\rightarrow Qy),\ c,$ `-q`$)$
8.	$\forall yWH.x$*(pelajar(y)*$\rightarrow$*beli(y,x)*	ϵ

Table 6.15: Semantics of (187)

In (187), the set for the *wh*-phrase is closed at step 6, while the meaning of the universally quantified phrase is retrieved at step 8. Hence, the *wh*-phrase has narrow scope relative to the universally quantified phrase.

While the MG system is compatible with both moving *wh*-phrases and *wh*-in situ, it is not able to get the narrow scope reading of the *wh*-remnant in the sluicing case. Eventhough both *wh*-in situ and *wh*-remnants are argued to take scope the same way, through set percolation, the closing of the set is different between the two. For *wh*-remnants, the Elide operation is defined to close the set, while for *wh*-in situ, the set is closed through the lexical item with the features `=t t`. If the set is closed before Elide occurs, we would run into a problem with the identity condition because meaning of the antecedent and ellipsis site would no longer be equivalent. Thus, the system here predicts only the wide scope reading of the *wh*-remnant.

In all fairness, the possibility of pair-list readings with universal quantifiers under sluicing has been contested, at least for English. Chung et al. (1995) imply that these type of readings are generally unavailable under sluicing save for a few cases such as (188) that might involve higher order quantification for the relational interpretation.

(188) Everybody gets on well with a certain relative, but often only his therapist knows which one. (Chung et al. 1995, ex.47)

Barker (2013) has also pointed out the possibility of pair-list readings with universal quantification for sluicing but offers no answer that would be compatible with his continuation analysis. On the other hand, Agüero-Bautista (2007) argues for regular successive-cyclic movement and reconstruction of the *wh*-remnant in intermediate positions to allow for the pair-list readings. While the idea of successive-cyclic movement is not directly implementable in the current MG system, a potential fix to allow for pair-list readings would be to follow Chung et al. (1995) and have certain indefinites be defined as functions from individuals to individuals, instead of a set of individuals.

The second inconsistency is from Malay scope rigidity. In Malay, if one quantifier c-commands another, the first quantifier will outscope the second as shown in (189).

(189) semua pelajar itu membaca sebuah buku
all student the read CL book
'Every student read a book'

Readings:

i $\forall x[\text{student}(x) \rightarrow \exists y[\ \text{book}(y) \wedge \text{ read } (y)(x)]]$

ii $*\exists y[\text{book}(y) \wedge \forall x[\ \text{student}(x) \rightarrow \text{read } (y)(x)]]$

(190) seorang pelajar membaca semua buku
a student read every book
'A student read every book'

Readings:

i $*\forall x[\text{book}(x) \rightarrow \exists y[\ \text{student}(y) \wedge \text{ read } (x)(y)]]$

ii $\exists y[\text{student}(y) \wedge \forall x[\ \text{book}(x) \rightarrow \text{read } (x)(y)]]$

Without sluicing, sentences like (189) and (190) only exhibit one reading. However, once the sluice with *wh*-remnant is added on, they are ambiguous as in (183). I do not have an answer as to why sluicing would force a second scope reading besides satisfying the Scope Parallelism requirement. The scope rigidity on the quantifiers suggest that there is only one possible derivation tree following the combining order of the quantifiers. However, the ambiguous sluicing sentence in (183) suggests that there may be more to the Elide operation in terms of possible derivations. This is left for future research.

6.5 Concluding Remarks

In this dissertation, I argued for a no *wh*-movement analysis for Malay sluicing. The analysis itself is not necessarily *wh*-in situ as I argue that *wh*-remnants are exactly *wh*-phrases in situ. However, the similaries between *wh*-remnants and *wh*-phrases in situ account for the Malay data pattern seen in Chapter 2. This is shown through the implementation of island insensitivity and p-stranding.

CHAPTER 7

Conclusion

7.1 A Brief History of this Dissertation

In a nutshell, this dissertation argued against Merchant's (2001) movement and deletion analysis of sluicing then developed and motivated a no movement analysis of sluicing based on Malay data. The analysis is then implemented within the Minimalist Grammars framework.

To expand in more explicit details, this dissertation began with the claim that the *wh*-remnant does not undergo *wh*-movement and that its scope taking capabilities are similar to those of indefinites and *wh*-phrases in situ. The rough idea of the analysis is illustrated in (191), where the *wh*-remnant does not move while the surrounding material is elided.

(191) John ate something yesterday, but I don't know ~~John ate~~ what ~~yesterday~~.

After presenting the core proposal, the no movement analysis was explored with Malay data in Chapter 2. Malay has both *wh*-movement and *wh*-in situ for question formation and served as a good language to test out this analysis of sluicing. The Malay data presented to argue for no movement in sluicing are summarized in the table below.

• Island Insensitivity • Sluicing Comp Generalization	• Both are well known crosslinguistic generalizations but are a problem for movement and deletion theories • Malay is island sensitive • Sluicing Comp Generalization is not needed if remnant does not move
• Preposition Stranding	• P-stranding generalization is usually used to argue for movement and deletion theories but it does not hold in Malay
• Scope Effects • *meN*-mismatch	• Novel Malay data that patterns with *wh*-in situ instead of *wh*-movement

Table 7.1: Summary of Malay data from Chapter 2

The Minimalist Grammars formalism was introduced in Chapter 3 and expanded for sluicing in Chapter 5. The syntax here followed Stabler's (1996) original operations: *merge* and *move*. Informally, *merge* takes two lexical items and combines them. If a lexical item has a `-x` feature, it will be held out in syntactic storage until a lexical item with `+x` feature combines into the derivation. Then, *move* applies by retrieving the lexical item from storage and combining it into the main structure. A moving *wh*-phrase in a question has a `-wh` feature which will hold the *wh*-phrase out until a lexical item with `+wh` is combined in. In

Chapter 5, I introduced the ⊛F feature for *wh*-remnant. Syntactically, the ⊛F feature also holds out lexical items from combining into the main structure. However, it relies on the Elide operation, and not another feature, to check it.

The semantic operations are based on the operations in Kobele (2006) and Kobele (2012): *mergeApp* (function application), *mergeStore* (meaning storage), *mergeRet* (meaning retrieval). Semantic storage and retrieval are used by Kobele to allow for quantifiers and *wh*-phrases to take scope. However, I argued that indefinites cannot take scope in the same way as other quantifiers and *wh*-phrases as they are shown to have more scoping capabilities. Thus, Hamblin semantics with sets were added into the semantic system. Indefinites, *wh*-phrases in situ and *wh*-remnants were argued to denote a set of alternatives, $\{a,\ b,\ c..\}$, and take scope via set percolation. Moving *wh*-phrases were argued to take scope via semantic storage and retrieval. The syntax and semantics of *wh*-movement, indefinites, *wh*-in situ, and *wh*-remnants are summarized in the table below.

	Syntactic feature	Scope-taking, Semantics
wh-movement	`d -wh`, held out	semantic storage, $\lambda P.WHy[Py]$
Indefinite	`d`, not held out	alternatives, $\{a,\ b,\ c...\}$
wh-in situ	`d`, not held out	alternatives, $\{a,\ b,\ c...\}$
wh-remnant	`d ⊛F`, held out	alternatives , $\{a,\ b,\ c...\}$

Table 7.2: Summary of Syntax and Semantics of *wh*-phrases and Indefinites

Any ellipsis operations require some form of identity condition. A hybrid condition identity, 'The Derivational Isomorphism Condition', was argued for in Chapter 4.

(192) The Derivational Isomorphism Condition:
The elided material and the antecedent must have the same derivation, including all lexical entries, modulo the correlate and *wh*-remnant.

The Elide operation is explained in Chapter 5 and is defined to work on the canonical sluicing case[1]. Here is a brief outline of deriving a sluiced sentence with 'John met someone yesterday, BIDK who' as an example:

1. *who* is held out by the ⊛F feature while the rest of the structure continues to be built up
   (John met someone yesterday) (John met $*$ yesterday, who)

2. At the end of the derivation, Elide checks the identity condition. Structure, feature and meaning of the antecedent (minus correlate) and the ellipsis site (minus *wh*-remnant) must be the same.

3. If the identity condition is met, Elide deletes the derived structure of the ellipsis site, leaving only the held out *wh*-remnant.

4. Elide then reassembles the structure and interprets the final output.

(193) Elide

(John met someone yesterday, t, $\{met(j,x) \mid x \text{ in } \{a,\ b,\ c...\}\}$)

((John met $*$ yesterday, t, $\{met(j,x) \mid x \text{ in } \{a,\ b,\ c...\}\}$), (who, ⊛))

= (John met someone yesterday BIDK who, t, $\exists x.[person(x) \wedge met(x)(j)] \wedge WHx.[person(x) \wedge met(x)(j)]$)

Island insensitivity and preposition stranding data from Chapter 2 were explained in Chapter 6. For island insensitivity, remnants with the ⊛F feature were showed to not trigger island effects because of the way they take scope via alternatives. On the other hand, *wh*-phrases with `-wh` feature were showed to trigger island effects because of their usage of the semantic storage to take scope. The Malay p-stranding data were explained through the usage of a percolator and a set of converters. In Chapter 5, the ⊛F feature was argued to be derived from `-wh` through a converter, `=x +wh x ⊛F`, where `x` can be of any category.

[1]Canonical cases of sluicing are when antecedent and sluice are sisters (John met someone yesterday, BIDK who). Elide as defined in Chapter 5 does not handle other forms of sluices. The hope is that Elide as defined here forms a basis that can eventually be expanded to different kinds of sluicing.

The percolator, `=P +wh p -wh`, is used to bring the `-wh` feature further up the tree in a preposition phrase. The different interactions between converters and the percolator explain the non-p-stranding pattern in normal Malay *wh*-questions and the availability of p-stranding with *wh*-remnants. This implementation of p-stranding with converters and percolator also generated an interesting prediction of massive pied-piping under sluicing that was discussed at the end of Chapter 6.

7.2 Final Remarks

There are, of course, unresolved issues regarding this analysis and implementation of sluicing which I have not adequately addressed in this dissertation. Among them are Malay scope effects (discussed at the end of Chapter 6) as well as cross-linguistic sluicing (briefly discussed at the end of Chapter 2). Furthermore, this dissertation only deals with the classic case of sluicing and does not explore other variants such as sprouting or multiple sluicing. My hope is that my implementation here will form a base that will eventually allow for other types of ellipsis.

A final remark concerning the no movement analysis that this dissertation pursued. At the very beginning of this work, I envisioned an analysis of a fully in situ *wh*-remnant, where there is no holding out of the remnant. As the work progressed, the difficulty of implementing a completely non-moving in situ *wh*-remnant became apparent. The difficulty here arises from deleting a non-constituent. In any analysis of sluicing, the *wh*-remnant has to be saved from deletion. In Merchant's (2001) analysis of sluicing, the saving remnant mechanism comes in the form of *wh*-movement, which also conveniently makes the ellipsis site a constituent. In a no *wh*-movement analysis, the remnant still needs to be saved from deletion. In this case, sluicing has to have restricted non-constituent deletion, where a subconstituent (remnant) gets saved from deletion. To save the remnant from deletion, my implementation of the no movement analysis held out the remnant while the remaining structure is deleted. However, by holding out the remnant, it starts looking like movement (since holding out phrases is one part of movement in MG). Previous analyses of sluicing involving *wh*-in situ remnants by Kimura (2010), Abe & Hornstein (2012), Abe (2015) are not completely without *wh*-movement in some aspects either. In Abe's (2015) in situ analysis of sluicing, assuming the copy theory of movement, the wh-remnant is saved by a *Focus* feature. However the lower copy of the *wh*-remnant is pronounced after deletion due to a ban on string-vacuous application of Move.

(194) BIDK $\langle \text{who}_{Focus} \rangle$ ~~John met~~ $\langle \text{who}_{Focus} \rangle_{PF}$ ~~yesterday~~

So the final question here is: can a *wh*-in situ analysis for sluicing ever be implemented without something that looks like movement? My answer: I don't know.

To end: This dissertation is about something, and hopefully by now, you know what.

REFERENCES

Abe, Jun. 2015. *The in-situ approach to sluicing*, vol. 222. John Benjamins Publishing Company.

Abe, Jun & Norbert Hornstein. 2012. 'lasnik-effects' and string-vacuous atb movement. *Ways of structure building* 169–205.

Abels, Klaus. 2019. On "sluicing" with apparent massive pied-piping. *Natural Language & Linguistic Theory* 37(4). 1205–1271.

Aelbrecht, Lobke. 2009. *You Have the Right to Remain Silent: The syntactic licensing of ellipsis*: Hogeschool-Universiteit Brussel dissertation.

Agüero-Bautista, Calixto. 2007. Diagnosing cyclicity in sluicing. *Linguistic Inquiry* 38(3). 413–443.

Aldridge, Edith. 2008. Phase-based account of extraction in indonesian. *Lingua* 118(10). 1440–1469.

Almeida, Diogo A de A & Masaya Yoshida. 2007. A problem for the preposition stranding generalization. *Linguistic Inquiry* 38(2). 349–362.

AnderBois, Scott. 2010. Sluicing as anaphora to issues. In *Semantics and linguistic theory*, vol. 20, 451–470.

AnderBois, Scott. 2011. *Issues and alternatives*: University of California, Santa Cruz dissertation.

AnderBois, Scott. 2012. Focus and uninformativity in yucatec maya questions. *Natural language semantics* 20(4). 349–390.

Barker, Chris. 2013. Scopability and sluicing. *Linguistics and Philosophy* 36(3). 187–223.

Barros, Matthew. 2014. *Sluicing and identity in ellipsis*: Rutgers University-Graduate School-New Brunswick dissertation.

Barros, Matthew, Patrick Elliott & Gary Thoms. 2014. There is no island repair .

Brasoveanu, Adrian & Donka F Farkas. 2011. How indefinites choose their scope. *Linguistics and philosophy* 34(1). 1–55.

Bruening, Benjamin. 2015. Non-constituent coordination: Prosody, not movement. *University of Pennsylvania working papers in linguistics* 21(1). 5.

Cable, Seth. 2010. *The grammar of Q: Q-particles, wh-movement, and pied-piping*. Oxford University Press.

Charlow, Simon. 2014. On the semantics of exceptional scope. *PhD diss,. New York University* .

Cheng, Lisa Lai-Shen et al. 1994. Wh-words as polarity items .

Chomsky, Noam. 1973. Conditions on transformations. a festschrift for morris halle, ed. by stephen r. anderson and paul kiparsky, 232-86. *New York: Holt* .

Chomsky, Noam. 1986. *Knowledge of language: Its nature, origin, and use*. Greenwood Publishing Group.

Chomsky, Noam. 1995. *The minimalist program*. Cambridge, MA: MIT Press.

Chung, Sandra. 1976. On the subject of two passives in indonesian. *Subject and topic* 57. 99.

Chung, Sandra. 2006. Sluicing and the lexicon: The point of no return. In *Proceedings of the annual meeting of the Berkeley Linguistics Society*, vol. 31, 73–91.

Chung, Sandra. 2013. Syntactic identity in sluicing: How much and why. *Linguistic Inquiry* 44(1). 1–44.

Chung, Sandra, William Ladusaw & James McCloskey. 2011. Sluicing (:) between structure and inference. *Representing language: Essays in honor of Judith Aissen* 31–50.

Chung, Sandra, William A Ladusaw & James McCloskey. 1995. Sluicing and logical form. *Natural Language Semantics* 3(3). 239–282.

Ciardelli, Ivano, Jeroen Groenendijk & Floris Roelofsen. 2012. Inquisitive semantics. *NASSLLI lecture notes* 187.

Cole, Peter & Gabriella Hermon. 1998. The typology of wh-movement, wh-questions in malay. *Syntax* 1(3). 221–258.

Cole, Peter, Gabriella Hermon & Yanti. 2008. Voice in malay/indonesian. *Lingua* 118(10). 1500–1553.

Comon, Hubert, Max Dauchet, Florent Jacquemard, Denis Lugiez, Sophie Tison & Marc Tommasi. 1997. Tree automata techniques and applications.

Cooper, Robin. 1983. Quantification and syntactic theory. *D. Reidel Dordrecht Holland.* .

Dayal, Veneeta. 2012. *Locality in WH quantification: Questions and relative clauses in Hindi*, vol. 62. Springer Science & Business Media.

Eng, Wong Bee. 2008. Wh-questions in malay: An explanation for the restriction of extraction to subject position with yang. *3L: Language, Linguistics, Literature®* 14.

Fiengo, Robert & Robert May. 1994. *Indices and identity*, vol. 24. MIT press.

Fortin, Catherine R. 2006. Reconciling meng-and np movement in indonesian. In *Annual Meeting of the Berkeley Linguistics Society*, vol. 32 2, 47–58.

Fortin, Catherine Rose. 2007. *Indonesian sluicing and verb phrase ellipsis: Description and explanation in a minimalist framework.*: dissertation.

Frey, Werner & Hans-Martin Gärtner. 2002. On the treatment of scrambling and adjunction in minimalist grammars. In *Proceedings of formal grammar*, vol. 2002, 41–52. Citeseer.

Ginzburg, Jonathan & Ivan Sag. 2000. *Interrogative investigations*. Stanford: CSLI publications.

Graf, Thomas. 2010. Locality and the complexity of minimalist derivation tree languages. In *Formal grammar*, 208–227. Springer.

Graf, Thomas. 2013. *Local and transderivational constraints in syntax and semantics*: UCLA dissertation.

Groenendijk, Jeroen & Martin Stokhof. 1985. On the semantics of questions and the pragmatics of answers. *Semantics: Critical Concepts in Linguistics* 288.

Guilfoyle, Eithne, Henrietta Hung & Lisa Travis. 1992. Spec of ip and spec of vp: Two subjects in austronesian languages. *Natural Language & Linguistic Theory* 10(3). 375–414.

Hamblin, Charles L. 1976. Questions in montague english. In *Montague grammar*, 247–259. Elsevier.

Hankamer, Jorge. 1979. *Deletion in coordinate structures*. Garland.

Hornstein, Norbert. 1998. Movement and chains. *Syntax* 1(2). 99–127.

Huang, C-T James. 1998. *Logical relations in Chinese and the theory of grammar*. Taylor & Francis.

Hunter, Tim & Chris Dyer. 2013. Distributions on minimalist grammar derivations. In *Proceedings of the 13th Meeting on the Mathematics of Language (MoL 13)*, 1–11.

Johnson, Kyle. 2001. Sluicing and constraints on quantifier scope. *Glot International* 5(2).

Joshi, Aravind K. 1985. How much context sensitivity is necessary for characterizing structural descriptions: Tree adjoining grammars. *Natural language parsing: Psychological, computational and theoretical perspectives* 206–250.

Karttunen, Lauri. 1977. Syntax and semantics of questions. *Linguistics and philosophy* 1(1). 3–44.

Keenan, Edward L. & Edward P. Stabler. 2003. *Bare Grammar: Lectures on Linguistic Invariants*.

Kimura, Hiroko. 2010. A wh-in-situ strategy for sluicing. *English Linguistics* 27(1). 43–59.

Kobele, Gregory M. 2006. *Generating Copies: An investigation into structural identity in language and grammar*: University of California, Los Angeles dissertation.

Kobele, Gregory M. 2009. Syntax and semantics in minimalist grammars. In *Proceedings of ESSLLI*, vol. 2009, .

Kobele, Gregory M. 2012. Importing montagovian dynamics into minimalism. In *International Conference on Logical Aspects of Computational Linguistics*, 103–118. Springer.

Kobele, Gregory M. 2015. Lf-copying without lf. *Lingua* 166. 236–259.

Kratzer, Angelika & Irene Heim. 1998. *Semantics in generative grammar*, vol. 1185. Blackwell Oxford.

Kratzer, Angelika & Junko Shimoyama. 2002. Indeterminate pronouns: The view from japanese. In *Paper presented at the 3rd Tokyo Conference on Psycholinguistics*, .

Lasnik, Howard. 2006. Multiple sluicing in english? *University of Maryland, College Park.* .

Leung, Tommi. 2014. The preposition stranding generalization and conditions on sluicing: Evidence from emirati arabic. *Linguistic Inquiry* 45(2). 332–340.

Merchant, Jason. 2001. *The syntax of silence: Sluicing, islands, and the theory of ellipsis*. Oxford University Press.

Merchant, Jason. 2005. Fragments and ellipsis. *Linguistics and Philosophy* 27(6). 661–738.

Merchant, Jason. 2013. Voice and ellipsis. *Linguistic Inquiry* 44(1). 77–108.

Michaelis, Jens. 1998. Derivational minimalism is mildly context–sensitive. In *International Conference on Logical Aspects of Computational Linguistics*, 179–198. Springer.

Müller, Gereon. 2017. Structure removal: An argument for feature-driven merge. *Glossa: A Journal of General Linguistics ()* .

Müller, Gereon. 2018. Structure removal in complex prefields. *Natural Language & Linguistic Theory* 36(1). 219–264.

Murphy, Andrew. 2019. *Structure Removal*. Institut für Linguistik, Universität Leipzig.

Nishigauchi, Taisuke. 2012. *Quantification in the Theory of Grammar*, vol. 37. Springer Science & Business Media.

Nomoto, Hiroki. 2008. A unified analysis of funny control. In *12th International Symposium on Malay/Indonesian Linguistics (ISMIL)*, .

Nomoto, Hiroki. 2013. On the optionality of grammatical markers: A case study of voice marking in malay/indonesian. *Voice variation in Austronesian languages of Indonesia* 54. 121–143.

Reinhart, Tanya. 1997. Quantifier scope: How labor is divided between qr and choice functions. *Linguistics and philosophy* 335–397.

Reinhart, Tanya. 1998. Wh-in-situ in the framework of the minimalist program. *Natural language semantics* 6(1). 29–56.

Roberts, Craige. 1996. Information structure in discourse: Towards an integrated formal theory of pragmatics. In Jae-Hak Yoon & Andreas Kathol (eds.), *OSU working papers in linguistics*, vol. 49, 1–69. OSU: The Ohio State University Department of Linguistics.

Rodrigues, Cilene, Andrew Nevins & Luis Vicente. 2009. Cleaving the interactions between sluicing and p-stranding. *Romance languages and linguistic theory 2006* 175–198.

Romero, Maribel. 1998. *Focus and reconstruction effects in wh-phrases*: University of Massachusetts at Amherst dissertation.

Rooth, Mats. 1992. Ellipsis redundancy and reduction redundancy. In *Proceedings of the Stuttgart ellipsis workshop*, vol. 29, Citeseer.

Ross, John Robert. 1967. *Constraints on variables in syntax.*: Massachusetts Institute of Technology dissertation.

Ross, John Robert. 1969. Guess who. In *Fifth regional meeting of the Chicago Linguistic Society*, vol. 5, .

Ruys, Eduard G. 1992. *The scope of indefinites*. LEd.

Saddy, Douglas. 1991. Wh-scope mechanisms in bahasa indonesia. *MIT working papers in linguistics* 15. 183–218.

Sato, Yosuke. 2011. P-stranding under sluicing and repair by ellipsis: why is indonesian (not) special? *Journal of East Asian Linguistics* 20(4). 339–382.

Sato, Yosuke. 2012. Successive cyclicity at the syntax-morphology interface: evidence from standard indonesian and kendal javanese. *Studia Linguistica* 66(1). 32–57.

Sato, Yosuke. 2016. An in-situ syntax of sluicing in indonesian. *AFLA 23: The Proceedings of the 23rd Meeting of the Austronesian Formal Linguistics Association* 243–257.

Schwarzschild, Roger. 1999. Givenness, avoidf and other constraints on the placement of accent. *Natural language semantics* 7(2). 141–177.

Seki, Hiroyuki, Takashi Matsumura, Mamoru Fujii & Tadao Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science* 88(2). 191–229.

Soh, Hooi Ling. 1998. Certain restrictions on a-bar movement in malay. In *Proceedings of the Third and Fourth Meetings of Austronesian Formal Linguistics Association 1996- 1999*, 295–308.

Soh, Hooi Ling & Hiroki Nomoto. 2011. The malay verbal prefix men-and the unergative/unaccusative distinction. *Journal of East Asian Linguistics* 20(1). 77–106.

Stabler, Edward. 1996. Derivational minimalism. In *International Conference on Logical Aspects of Computational Linguistics*, 68–95. Springer.

Stabler, Edward P. 2011. Computational perspectives on minimalism. *Oxford handbook of linguistic minimalism* 617–643.

Stigliano, Laura. 2019. P-stranding in clausal ellipsis in Spanish: an argument for syntactic identity. *Ms. University of Chicago* .

Stjepanović, Sandra. 2008. P-stranding under sluicing in a non-p-stranding language? *Linguistic Inquiry* 39(1). 179–190.

Van Craenenbroeck, Jeroen. 2010. Invisible last resort: A note on clefts as the underlying source for sluicing. *Lingua* 120(7). 1714–1726.

Weir, Andrew. 2014. *Fragments and clausal ellipsis*: University of Massachusetts Amherst dissertation.