The Linearization of V(P)-doubling Constructions

Rong Yin
University of Massachusetts Amherst

THE LINEARIZATION OF V(P)-DOUBLING CONSTRUCTIONS

A Dissertation Presented
by
RONG YIN

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

February 2022

Linguistics

THE LINEARIZATION OF V(P)-DOUBLING CONSTRUCTIONS

A Dissertation Presented
by
RONG YIN

Approved as to style and content by:

Kyle Johnson, Chair

Rajesh Bhatt, Member

Ellen Woolford, Member

Luiz Amaral, Member

Joe Pater, Department Chair
Department of Linguistics

To my husband and my mom,
who I will love forever

To my grandma,
who I will love forever

"The Answer to the Great Question……Of Life, the Universe and Everything……Is……Forty-two."

—Douglas Adams, *The Hitchhiker's Guide to the Galaxy*

ACKNOWLEDGMENTS

I will express my gratitude in words but my gratitude will always be beyond words.

First, I would like to thank my dissertation committee: Kyle Johnson, Rajesh Bhatt, Ellen Woolford, and Luiz Amaral. They were very generous with their time and spared no efforts to help me to improve the dissertation. I received great questions from them, which helped shape both the details and the general framework of this dissertation. I learned a lot from them. In addition, they are all very encouraging, which helped me get through the times when I got stuck. It has been a great pleasure working with them.

Kyle Johnson played the most important role in helping me with my linguistics study and research at UMass. He is both an excellent teacher and advisor. He cares about teaching and cares about helping his students with their research. I enjoyed all his syntax classes/seminars, which were both super insightful and engaging. I really appreciated his handouts, which could easily be part of a book since they are so detailed and well organized. I got the chance to work on my second GP and my dissertation with Kyle. Both were great experiences and both works would never exist if it were not for him. I enjoyed having discussions with him regarding detailed techniques, as well as more abstract topics like how to do research. I learned so much from him and I could not list how many ideas he gave to me. He is so knowledgeable: so as long as a work is about syntax or even remotely related to syntax, he is always be able to make the work better. He was always so patient with me when I got stuck and never hesitated to stay over the scheduled meeting time to help me as long as he did not have other arrangements. I really appreciated that he always spent time telling me the reasons why he thought a change should be made or why things should be written in a certain way—in this way, not only was I able to learn but also felt respected. In addition, I have seen several acknowledgments from other people for Kyle Johnson. They are all so fun to read. This is no coincidence—he is just such a fun person. Here is one of the best memories that I have about him in a class: he said that when he was working on particle shift, an unexpected song came out, and he started to sing "Take on Me." If a linguist were to do stand-up comedy, this would make a perfect punchline. Furthermore, as a great researcher, he is unbelievably modest—you can see so much confidence about linguistics in him but there is no trace of arrogance. I really respect him. I will miss learning from him and chatting with him regularly.

Rajesh Bhatt is a great person to work with. He has the amazing ability to make every meeting very chill but very helpful. The technical details of the dissertation have gone through a lot of changes, but he was always able to understand the idea very quickly, immediately pointed out the problems and gave me very insightful feedback. As a result, he was always able to help me think a lot about the predictions of my work and help me have an idea about what data I should be looking for to verify these predictions.

Ellen Woolford is great at helping people to write. I personally have benefitted a lot from this. She helped me a lot with presenting complex ideas in an understandable way. She asked me very detailed questions about my work, which helped me to see what was missing in the arguments and what seemed to be unclear. Communicating using language can easily cause misunderstandings and confusion but she was always able to make my communication better.

I worked with Tom Roeper on my first GP and he is the most positive researcher I have ever met. He was super encouraging and supportive. He always showed enthusiasm about the ideas that I had and helped me to see how the ideas turned out. I am very inspired by his passion about linguistics.

Jeremy Hartman was another great person to work with. He is great at constructing data to test a hypothesis—without this ability of his, I would never have been able to convince people that the data in my second GP was worth working on. He was also very encouraging and I had a great time working with him on both of my GPs.

Seth Cable's semantics class in the second semester was definitely one of the best classes that I had at UMass. I always found semantics very complex and difficult to understand but he managed to teach those complex ideas in a way such that they could be easily understood. His handouts are very detailed and easy to follow. Anyone would benefit from these handouts.

I had a great time attending Brian Dillon's formal foundation of linguistics class, where I learned a lot about R and statistics, and joining the Bayesian statistics reading group hosted by him. He is a very cheerful person and I really enjoyed the Mandarin (with a Beijing accent) he sometimes suddenly spoke.

It was a great pleasure working as a teaching assistant for Alice Harris, Joe Pater, and Kyle Johnson. Their encouragement and compliments brought me a lot of confidence and inspired me to always to try to be better.

I met Professor Jaklin Kornfilt at Syracuse University before coming to UMass, without whom I would have never been able to achieve what I have. She has been super encouraging and helped me in every way. I always have this thought in the back of my mind, which is that I hope the research that I have done can live up to her expectations and be worth the help she has given to me.

I would also like to thank Professor Omer Preminger who I met at Syracuse University. I had a great time being in his advanced syntax class, which I looked forward to every week during that semester. I enjoyed a lot solving the problem sets that he had for the class, which is where I started to learn how to solve a syntactic problem. He was great at illustrating how to solve complex syntactic problems and I learned a lot from him. I also had a great time learning from Professor Jon Nissenbaum at Syracuse University, who was very encouraging and cheerful. I had a lot of fun time exploring my research ideas with him.

I also want to thank Professor Lichang Su from Nankai University. When I first got interested in linguistics, I was doing my B.A. at Nankai University. There were few linguistics class for undergraduate students, but when Professor Su heard about my interest in linguistics, he allowed me to attend his linguistics classes for M.A. and Ph.D. students to learn more about linguistics. I really appreciated his support.

Turning back to UMass again, the help from Tom Maxfield made my life at UMass much easier. Once in a while, I would have some administrative questions, sometimes quite urgent, and I could always trust him to fix them in no time. I would also like to thank Michelle McBride for helping me getting all the complicated paperwork done, especially when I could not be on campus and do it myself.

I also feel very grateful that I had such wonderful fellow graduate students at UMass—their great questions made the classes even more interesting and helped me improve my work; I enjoyed those questions and learned a lot from them. Among them, I would especially thank Hsin-Lun Huang, Deniz Özyıldız, Petr Kusliy, Katia Vostrikova, Leland Kusmer, David Erschler, Duygu Göksu, Kaden Holladay, Jyoti Iyer and Kimberly Johnson. I am also thankful for having a great cohort—Chris Hammerly, Alex Göbel, Brandon Prickett, Carolyn Anderson, and Jaieun Kim—who I had some great times with. (I reserve a special place for the cohortmate missing here and will talk about him later.) I would like to particularly thank Hsin-Lun Huang, who has been a great friend and feels like a family member. When I first came to UMass, he gave me so many valuable suggestions and always kept an eye out for me. During the years at UMass, it was always very fun to chat with him from time to time about research or just about what was going on in life. Even after he graduated, he has still been helping me with finding a job. I would also like to thank another great friend from Syracuse University—Lily Lewis. I had a lot of fun discussing linguistics, having Friday dinners and game nights, and watching TV shows with her. Also, she would always offer me a ride back to the apartment after class before I even asked. I really appreciated it the time she invited me to her hometown to celebrate July 4th; I had a wonderful time during that week. Although we are not in driving distance anymore, a phone call from time to time still never fails to cheer me up.

Now, I would like to turn to my family, who I can never thank more. My mom has always been super supportive of my interests and given me a lot of trust. When I told her that I would like to go abroad to study linguistics, she had no idea what linguistics was but said she trusted my decision and would do anything to support me. After two years, I finished my M.A. in linguistics but was not able to get an offer from any Ph.D. program. I told her I wanted to try again and once again she supported my decision. Because of her, I could stay at home focusing on the applications without being worried about making a living. I knew how much she would like me to stay near her but she chose to let me fly. One time she sent me an article and said this looked like what I was doing because she saw some syntactic trees. I never really spent time telling her what my research was about but she cared about me so much that she tried to understand what I was doing in her own way.

I grew up with my grandma. She basically spent all her retirement time taking care of me. She always gave me the best. She always told me that she had a lot of confidence in me and told me I was the best. Every time I talked to my family over the phone, I was never able to hold my tears when I talked to her although she was being very cheerful and did not say anything sad at all. I hope that she can visit me in my dreams more. I am not able to talk too much about her despite there being so much to talk about because I am starting to cry already.

I also feel very grateful that I have two cousins who I feel are more like a sister and brother. We had so much fun together when I was a kid and they always took me out to have fancy food using their own pocket money. They have always taken very good care of me. I would always reach out to them first when I felt bad or needed someone to talk to. I still have the picture of the three of us taken in a photo booth in my purse. Despite being an only child, I never feel lonely and I had the best childhood because of them. My aunts and uncles have also been very sweet to me. When I was a kid, they brought me gifts from time to time and always said they were very proud of me. I feel very loved.

I would also like to thank my mother in-law Suzie Wilson and father in-law Robin Wilson for making me feel I am part of their family. They have always been very supportive and when I have difficulties or have a difficult decision to make, I know I can always trust them to help me. My mother in-law Suzie Wilson is a very sweet person and I have been having great Christmases because of her. My sister in-law Sarah Wilson, my brother in-law Stephen Wilson and his wife Miriam Wilson are also very nice to me and I have a lot of fun hanging out with them.

Before ending the acknowledgments, I would like to say that all errors in this dissertation are my own.

Finally, I would like to thank the person who keeps me from falling apart during the hard times of my life and makes me realize that true love does exist—Michael Wilson. He has always been so encouraging, giving me very helpful and valuable comments and suggestions, asking me great questions regarding my research and constantly offering me editorial help no matter how busy he is. He always believes in me even when I lose all my confidence and want to give up. I do not know how I could have survived these years without him. He also has a very pure heart, which is the reason why I feel he is the best person I have ever met. It is my huge luck to have him as my best friend, my cohort, my co-author and my husband. Home is wherever you are. Maimai, I love you forever and ever!

ABSTRACT

THE LINEARIZATION OF V(P)-DOUBLING CONSTRUCTIONS

FEBRUARY 2022

RONG YIN

B.A., NANKAI UNIVERSITY
M.A., SYRACUSE UNIVERSITY
Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Kyle Johnson

When an item moves, it is usually pronounced once but in some cases, it is pronounced multiple times. So, a question is: What determines whether a moved item gets pronounced in only one of its positions or in multiple positions? This dissertation aims at providing an answer to this question by designing a linearization process that yields the correct phonetic realization of a moved item, with a focus on V(P) movement. In particular, this dissertation provides a detailed analysis of how V(P)-doubling cases are linearized and thus show how a V(P) ends up being pronounced multiple times.

Regarding the proposed linearization process in this dissertation, following Kusmer (2019), I assume that the basic linearization process contains Candidates Generator G, which generates a set of precedence relations, and Constraints, which pick the right subset from G. As for the Constraints, I adapt the Totality Constraint and the Asymmetry Constraint from Kayne (1994), the Anti-reflexivity Constraint from Partee, ter Meulen and Wall (1990), and Language Specific Constraints from Wilder (1999) and Kusmer (2019). In addition, I propose an Ordering Deletion rule that gets rid of redundant precedence relations and a Set-to-String algorithm that turns a set of precedence relations into a string. Furthermore, I adopt the idea from Fox and Pesetsky (2005) that linearization of precedence relations is implemented in a cyclic way. Also, I employ the idea behind Nunes (2004)'s morphological reanalysis that a higher level node can be linearized instead of the nodes it contains given certain circumstances.

Finally, I present the predictions made by the proposed linearization process, which can be evaluated against more data for future research.

TABLE OF CONTENTS

List of Tables

List of Abbreviations

$\sqrt{x}$	Root of x
1	1st person
3	3rd person
ACC	Accusative Case
Adj	Adjective
ASP	Aspect
AUX	Auxiliary
CL	Classifier
FEM	Feminine
FOC	Focus
INF	Infinitive
NOM	Nominative Case
NMLZ	Nominalizer
PST	Past Tense
PFV	Perfective
PL	Plural
SG	Singular

CHAPTER 1

Introduction

1.1 The core of the dissertation

This dissertation focuses on the phonetic realization of a moved item. Specifically, this dissertation aims at answering the question in (1) by providing a linearization process that yields the correct phonetic realization of a moved item.

(1) What determines whether a moved item gets pronounced in only one of its positions or in multiple positions?

For instance, in the English example (2), the auxiliary verb *can* undergoes T^0-to-C^0 movement and occupies both the T^0 and C^0 position but it is only pronounced in the C^0 position.

(2) **T-to-C movement**

a. **Can** she run?

b. ***Can** she **can** run?

However, in some cases, a moved item is pronounced multiple times. An example is shown in (3), where the participle *chi-guo* 'eaten' moves but it must be pronounced twice.

(3) **Mandarin verb-doubling**

a. **chi-guo**, Lili dique meiyou **chi-guo** bale.
eat-ASP Lili indeed not **eat**-ASP Guava

'As for having eaten, Lili indeed hasn't eaten Guava (but she has seen it before).'

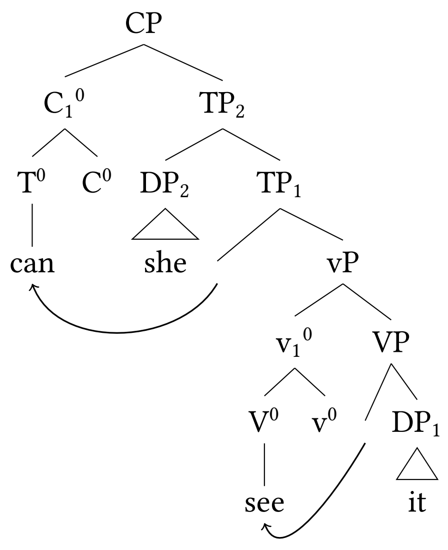
b. ***chi-guo**, Lili dique meiyou bale.
eat-ASP Lili indeed not Guava

Intended: 'As for having eaten, Lili indeed hasn't eaten Guava (but she has seen it before).'

Focusing on verb movement, in addition to the T^0-to-C^0 movement example in (2), there are also other verb movement cases where the verb is only pronounced once. For instance, V^0-to-v^0 movement in English (4)[1], and v^0-to-T^0 movement in Hebrew (5). Note that these examples are normally referred to as "short/local" head movement.[2]

(4) a. *Can she see it?*

b.



[1]Regarding the arguments for V^0-to-v^0 movement in English, please see Beck and Johnson (2004), Bale (2005) and Johnson (2018).

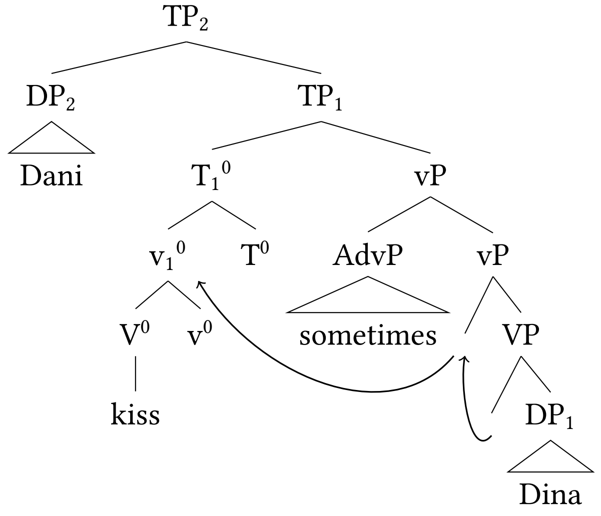
[2]Following Pollock (1989), I assume that all head movements are adjunction.

(5) a. **Hebrew**

Dani	menasek	lif'amim	et	dina
Dani	kisses	sometimes	ACC	Dina

'Dani kisses Dina sometimes.' (Doron 1999: 126, ex.3(a))

b.



For cases where the moved verb is pronounced multiple times, in addition to the Mandarin example in (3), another example is the Asp^0-to-Top^0 movement in Yiddish (6), where the participle *gegessen* 'eaten' is pronounced twice. In fact, these cases are normally referred to as verb-doubling, and are found in Yiddish (DavisPrince 1986, Cable 2004), Hebrew (Landau 2006), Vata (Koopman 1986), Mandarin (Cheng 2013), among other languages. It has been argued in the literature (cf. Koopman (1984), Nunes (2004), Landau (2006) among others) that these verb-doubling cases involve "long/non-local" movement.

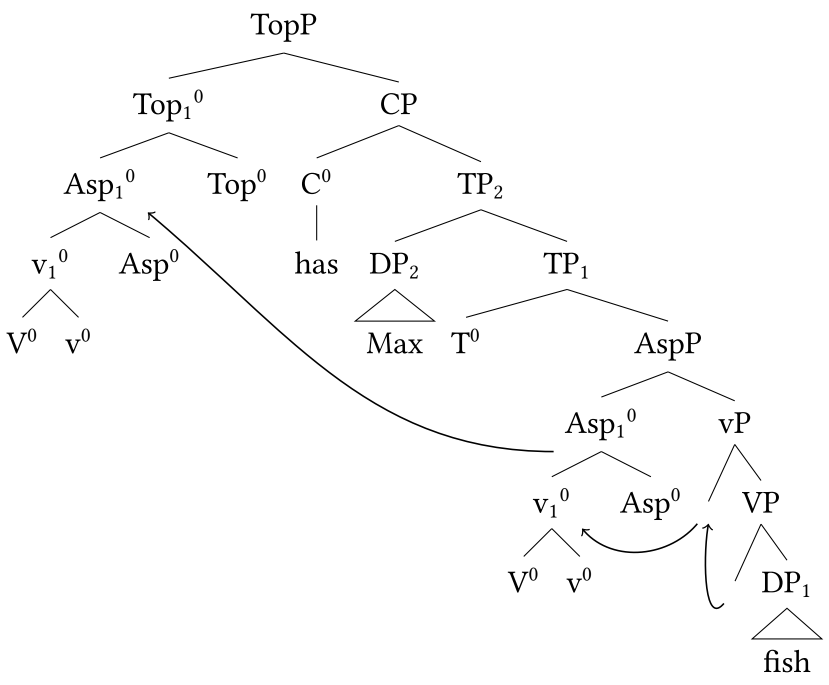
(6) **Yiddish**

a.

Gegessen,	hot	Maks	**gegessen**	fish
eaten	has	Max	**eaten**	fish

'As for having eaten, Max has eaten fish.' (Cable 2004: 2, ex. 2(a))

b.

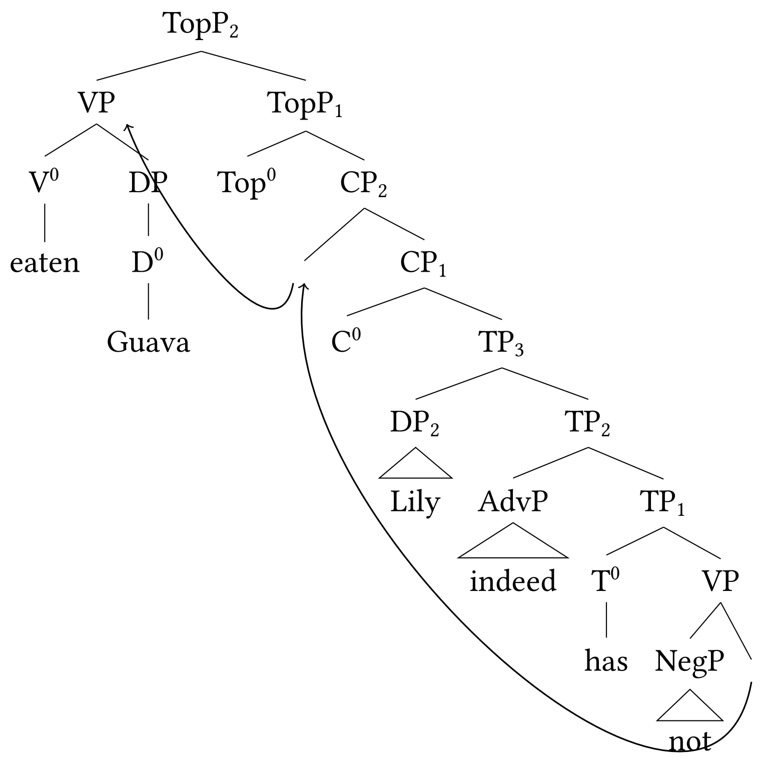


As for VP movement, in some cases, the moved VP is pronounced once. An example is VP-fronting in English (7). I assume in this case that VP moves in a one fell swoop fashion.[3]

[3]For ease of presentation, I ignore the vP layer in this example and also in the next Mandarin example in (8b), which does not affect the analysis.

(7) a. *Eaten Guava, Lily indeed has not.*

b.



However, there are also cases where a VP is pronounced multiple times, which is shown in (8), where *chi-guo bale* 'eaten Guava' is pronounced twice. I propose that in such cases, VP moves in a way that contains multiple steps, the structure of which is shown in (8b).[4]

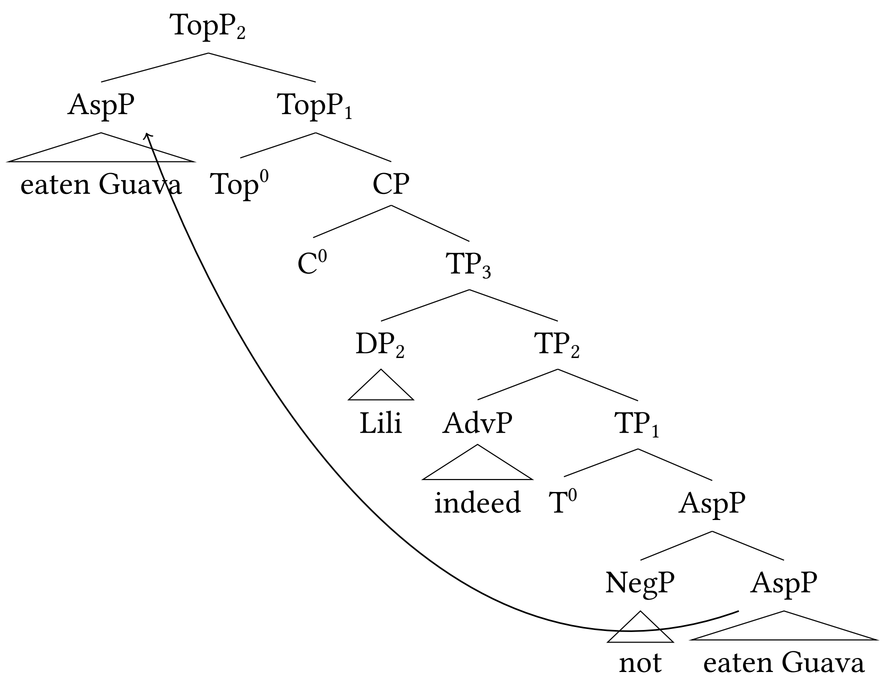
(8) a. **Mandarin**

Chi-guo	**bale**,	Lili	dique	mei	**chi-guo bale**
eat-ASP	**Guava**	Lili	indeed	not	**eat-**ASP **Guava**

'As for having eaten Guava before, Lili indeed hasn't eaten Guava before.'

[4]Reasons for projecting the empty CP layer are discussed in Chapter 6.

b.



Based on my research on the verb movement cases, there seems to be an observation, which is stated in (9).

(9) It seems that multiple pronunciations of a moved verb can occur in a long distance/non-local movement, for instance, topicalization movement, as in (6); and a single pronunciation of a moved verb tends to occur in a short distance/local movement, for instance, T^0-to-C^0 movement, as in (4).

I suggest that the seemingly "long distance/non-local" topicalization verb movement can be a case of movement in (10), while the seemingly "short distance/local" T^0-to-C^0 movement, V^0-to-v^0 movement, etc. are cases of movement in (11). In addition, I suggest that the one fell swoop topicalization VP movement, as in (8), is also a case of movement in (10), while the movement that consists of multiple steps, as in (7), is also a case of movement in (11).

(10) Movement that happens across different spell-out domains via non-initial positions in the spell-out domains

(11) Movement that happens in the same spell-out domain or across different spell-out domains via the initial positions in the spell-out domains

I propose that if a V(P) moves in the way defined in (10), the moved verb will be pronounced multiple times, and the moved VP can be pronounced multiple times under certain circumstances[5] or will lead the linearization to crash; and if a V(P) moves in the way defined in (11), it will be pronounced once. Thus, a brief answer for the question in (1) is presented in (12).

(12) Multiple Spell-out is prevented except when an item moves across a Spell-out domain via a non-initial position (10), where Spell-out domain is defined in (13).

(13) **Spell-out domain**

Spell-out domain refers to the constituents that are mapped by Spell-out, where Spell-out refers to the mapping between syntax and phonology.

Fox and Pesetsky (2005)

Crucially, in this dissertation, I propose a linearization process that aims at deriving (12). To be more specific, following Kusmer (2019), I assume that the basic linearization process contains Candidates Generator G, which generates a set of precedence relations, and Constraints, which pick the right subset from G. I propose an Ordering Deletion rule that gets rid of redundant precedence relations and a Set-to-String algorithm that turns a set of precedence relations into a string. Furthermore, I adopt the idea from Fox and Pesetsky (2005) that linearization of precedence relations is implemented in a cyclic way. Finally, in my analysis, I employ the idea behind Nunes (2004)'s morphological reanalysis that a higher level node can be linearized instead of the nodes it contains in some cases, though morphological reanalysis is not adopted in my analysis.

In addition, the linearization process proposed in this dissertation unifies the explanation for V-doubling and VP-doubling. First, for both V-doubling and VP-doubling,

[5]Details about the circumstances will be discussed in chapter 4.

the linearization process takes a set of ordering statements and yields a string, which eventually becomes the output of PF after lexical insertion. The only difference is that for V-doubling, the ordering statements are between words (i.e., X^0s) but for VP-doubling, the ordering statements are not only between words but also between words and strings (i.e., XPs). In other words, for VP-doubling, the linearization process first forms a string out of a subset of the ordering statements and this string eventually becomes part of the final string; while for V-doubling, the linearization process forms the final string without first forming any sub-strings. A more detailed discussion about the linearization process is presented in chapter 4. In addition, the source of doubling for both V-doubling and VP-doubling comes from a higher node of the moved item(s) participating in the linearization process. To be more specific, for V-doubling, the higher node dominating all the terminal node(s) in a moved verb (i.e., the complex verb node formed by moved verb(s)) participates in the linearization; and for VP-doubling, the higher node dominating all the terminal nodes in a moved verb phrase (i.e., the phrase-level node) participates in the linearization process. A more detailed illustration of the linearization of V(P)-doubling cases is in chapter 6.

Note that there are other movement cases, where the moved items are of different categories from the V(P) movement examples that I have shown so far. For instance, DP movement and PP movement. Similar to V(P) movement, there are cases where a moved DP or PP is pronounced once. An English example for DP movement is shown in (14), where the DP *dumplings* can only be pronounced once. An English example for PP movement is shown in (15), where the PP *to whom* can only be pronounced once. This kind of case is briefly discussed in chapter 5.

(14) a. **Dumplings** she ate.

b. ***Dumplings** she dumplings ate.

(15) a. **To whom** does she give the book ?

b. ***To whom** does she give the book to whom?

However, different from V(P) movement, there do not seem to be cases where the moved DP or PP is **fully** pronounced multiple times. To be more specific, for cases where a DP seems to be pronounced multiple times, it is pronounced as a resumptive pronoun in one of its positions. An example is shown in (16). It is argued in Sichel (2014) that in (16), the DP *ha-iša* 'the woman' and the pronoun *ota* 'her' in the direct object position form a movement chain.

(16)

dani	yimca	et	[ha-iša$_1$	še-hu	mexapes	**ota$_1$**]
Dani	will.find	ACC	the-woman	that-he	searches	her

'Dani will find the woman he is looking for.' (Sichel 2014: 659, ex. 3(b))

Due to the fact that a moved DP/PP does not seem to be fully pronounced multiple times, I assume in this dissertation that the multiple pronunciations of a moved DP/PP (i.e., cases involving resumptive pronouns) should be treated differently from multiple pronunciations of V(P), and I do not present an analysis for the multiple pronunciations of DP/PP in this dissertation.

1.2 The theoretical background of linearization and a preview for the analysis

Before going into the details of the theoretical background of linearization, I start with the question in (17).

(17) Why is linearization needed?

Hornstein et al. (2005) argues that linearization is needed because phrase markers are two-dimensional but after phrase markers are sent to PF, the output of PF needs to be manipulated by the Articulatory-Perceptual (A–P) system, which requires the output of PF to be one-dimensional. To be more specific, phrase markers are two-dimensional in the sense that their lexical items are organized in terms of the sisterhood (i.e., breadth) and the dominance (i.e., depth) relation; however, the output of PF has to be one-dimensional (i.e., the output of PF has to be a string of sounds or signs) because the output of PF will be manipulated by the Articulatory-Perceptual (A–P) system, where the vocal apparatus can only produce words in a linear, one-dimensional order. Thus, a process is needed to reduce the dimension of the two-dimensional phrase markers to be a one-dimensional string so that the output of PF can eventually be manipulated by the A–P system.

Another claim can be found in Fukui and Takano (1998) that syntactic structures do not provide any information about the linear order of the nodes. This claim follows Chomsky (1995)'s discussion about bare phrase structure, which states that syntactic derivations only need Merge and Move ("remerge") and the syntactic trees do not have any information about linearization. To be more specific, Chomsky (1995) argues that linearization is a PF phenomenon given that there is no clear evidence showing that LF needs information about order.

As a quick summary, either from the view that phrase markers are two dimensional and their dimension needs reducing for them to be uttered, or from the view that syntactic structures do not provide any information about linear order, linearization is needed to order syntactic structures. Now, the question is:

(18) What should be included in linearization?

Under the traditional Government and Binding approach, linearization contains directionality parameters, which specify the order between heads and complements, as well as the order between specifiers and the rest of the phrase for a given structure (cf. Chomsky (1981), Stowell (1981), Koopman (1984), and Travis (1984)). In this way, from the perspective of Hornstein et al. (2005), linearization adds precedence relations to a two-dimensional structure, where precedence relations are one-dimensional (i.e., if α precedes β, α is pronounced before β). From the perspective of Fukui and Takano (1998), directionality parameters specify the precedence relations for the unordered structures.

For instance, to explain the difference between (19a) and (19b) that the verb precedes the object in English but the verb follows the object in Japanese, the directionality parameters approach states that English is set to have a VO order following the option in (20a) and Japanese is set to have a OV order following the option in (20b), where the verb is X and the object is Compl. Thus, the cross-linguistic variation of word order is captured by the directionality parameters in (20) and the ones in (21), where the order of head and complement is specified either as (20a) or (20b), and the order of specifier and the rest of the phrase is specified either as (21a) or (21b) for a given structure established by the X'-Theory.

(19) a. Norbert [$_{\text{VP}}$ ate bagels].

b. **Japanese**

Jiro-ga	[$_{\text{VP}}$	sushi-o	tabeta].
Jiro-NOM		sushi-ACC	ate

'Jiro ate sushi.' (Hornstein et al 2005: 218, ex. 1 & 2)

(20) a. X' → X Compl

b. X' → Compl X

(Hornstein et al 2005: 220, ex. 3)

(21) a. XP → Spec X'

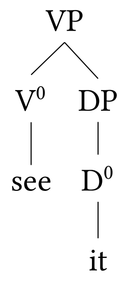
b. XP → X' Spec

(Hornstein et al 2005: 220, ex. 4)

In a later approach by Kayne (1994), linearization still involves generating precedence relations; however, the precedence relations are not added to a given structure by specifying the parameters but are established by the syntactic relations (i.e., asymmetric c-command relations) in a syntactic structure under the X'-Theory. For instance, in (22), V^0 asymmetrically c-commands D^0 (i.e., V^0 c-commands D^0 but D^0 does not c-command V^0), so according to Kayne (1994)'s LCA, *see* precedes *it.* A more detailed review of Kayne

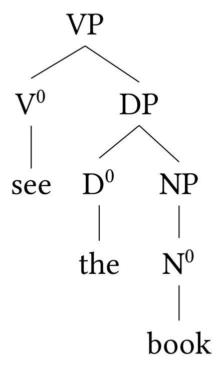
(1994)'s Linear Correspondence Axiom (LCA) is in chapter 3 and it is discussed under the bare phrase structure theory.

(22)



Regardless of what is encoded in linearization, being either directionality parameters or LCA, both approaches involve adding or generating precedence relations, which for Hornstein et al. (2005) enable the two-dimensional phrase markers to be manipulated by the A–P system as a one-dimensional object, or for Fukui and Takano (1998) provide linearization information at PF.

In this dissertation, I also adopt the idea that precedence relations are needed in linearization. In fact, later on, when I introduce cyclic linearization of Fox and Pesetsky (2005) in chapter 5, it can be seen that cyclic linearization also depends heavily on information about precedence relations. Note that despite both the directionality parameters approach and LCA involving precedence relation, they do differ in the specific content of precedence relations. In the earlier approaches, (20) and (21) are used to specify the precedence relations; however, precedence relations do not seem to be defined very clearly. For instance, since English is set to have the parameter in (20a), in structure (23), *see* is the head and *the book* is the complement and head should precede its complement, but what does it mean by *see* preceding *the book*? One can imagine two situations in (24).

(23)


(24) a. *see* precedes *the*, *see* follows *book*

b. *see* precedes *the*, *see* precedes *book*

When one says head precedes complement, one means (24b), but the head-complement parameter statement in (20a) does not explicitly state (24b). Thus, in order to be explicit and specific about the ordering between lexical items, one needs a set of precedence relations that specify the ordering relations between lexical items. For instance, there should be a set of precedence relations that states *see* precedes *the*, *see* precedes *book* and *the* precedes *book*. In fact, this is what is implemented in Kayne (1994) and in this dissertation, I follow Kayne (1994) and include such sets of explicit and specific precedence relations in linearization.

Now that I have presented that linearization needs not only precedence relations, but a set of precedence relations that specify the ordering relations between items, the question is:

(25) What properties should a set of precedence relations have?

Kayne (1994) proposes that the linearization of lexical items should subject to the following conditions:

(26) a. All vocabulary items in the phrase marker p must be in the linearization of p. (Totality)

b. For all vocabulary items, a and b in p, the linearization of p cannot include both $a < b$ and $b < a$. (Antisymmetry)

c. For all vocabulary items, a, b, c in p, if the linearization of p includes $a < b$ and $b < c$ then it must include $a < c$. (Transitivity)

In this dissertation, following Kayne (1994), I adopt the view that a linearization of items should have the Totality and Antisymmetry property, and in addition, I follow Partee, ter Meulen and Wall (1990), and adopt the view that a linearization of items should also have the Anti-reflexivity property — to put it simply, an item cannot be ordered relative to itself. Now that the properties of a set of precedence relations are determined, the question is:

(27) How is a set of precedence relations generated?

Although linearization in this dissertation has the same goal of restricting the precedence relations as Kayne (1994)'s LCA, different from Kayne (1994)'s LCA, I follow Kusmer (2019) and adopt the view that linearization process contains the Candidates Generator G and the Constraints, which generate all possible precedence relations but rule out the illegitimate ones. Regarding the Constraints, I adapt the Totality Constraint and the Asymmetry Constraint, from Kayne (1994), the Anti-reflexivity Constraint from Partee, ter Meulen and Wall (1990), and Language Specific Constraints from Wilder (1999) and Kusmer (2019). Note that the Totality Constraint, the Asymmetry Constraint and the Anti-reflexivity Constraint makes sure that the linearization is consistent and the Language Specific Constraints make sure that linearization yields the correct ordering of the items. Details of the constraints are presented in chapter 4. Another difference between Kayne (1994)'s LCA and the linearization in this dissertation is that I follow Halle and Marantz (1993)'s work about distributed morphology, and assume that vocabulary items are inserted after syntax. Thus, in this dissertation, the terminal nodes in syntax are X^0 and not vocabulary items. Instead, they are occupied by the morphemes and features that vocabulary items will appear. Correspondingly, the Candidate Generator and the Constraints work on X^0 nodes. Specifically, I propose that only the highest X^0 nodes can serve as the input for the Candidate Generator, details of which are discussed in chapter 4.

As a quick summary, so far, I have discussed that a set of precedence relations is needed in linearization, what properties a set of precedence relations should have, and how the linearization in this dissertation yields such sets of precedence relations. However, I propose that only having precedence relations is not enough; a linguistic representation of a string is also needed in linearization. To be more specific, precedence relations (e.g., $A < B$, $A < C$, $B < C$) should be further mapped to a string of nodes (e.g., $<_S$ ABC $>$), and I propose a detailed Set-to-String algorithm to implement this mapping process.

Evidence motivating the linguistic representation of a string comes from the fact that sometimes, prosodic operations need to look at the beginning and ending of a string. For instance, according to Selkirk (1996), in English, when functional words appear at the end of a phrase, it cannot have the reduced form, examples for which are shown in (28). Since the precedence relations do not, at least directly, provide the information about the boundary of a string, having a linguistic representation of a string at PF in addition to precedence relations is useful.

(28) a. I don't know where Ray **is**. [ɪz] *[əz] *[z]

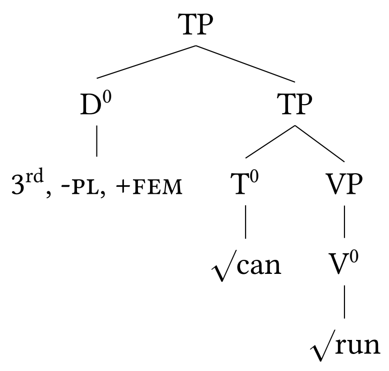
b. I can eat more than Ray **can**. [kæn] *[kən] *[kn̩]

(Selkirk 1996: 200, ex. 27)

Once a string of nodes is formed, lexical items can be inserted and the string of lexical items becomes the output of PF.

Before going to the next section, I use a simple example in (29) to illustrate how the basic linearization process works in this dissertation under the bare phrase structure theory (cf. Chomsky (1995, 2000, 2001)).

(29)



In the first part of the linearization process, following Kusmer (2019), there are candidates Generator G and Constraints, where I propose that only maximal X^0 nodes serve as the input for Candidates Generator G. After imposing Constraints on the sets generated by Candidates Generator G, linearization yields a set of ordered pairs (30), which shows the precedence relations between nodes in that structure (the symbol "<" represents "precedes").

(30)
$$\begin{Bmatrix} D^0 < T^0 & T^0 < V^0 \\ D^0 < V^0 & \end{Bmatrix}$$

Next, the precedence relations (30) are mapped to a string of nodes (31). This mapping is implemented in by the Set-to-String algorithm that I propose.

(31) $<_S D^0T^0V^0 ... >$

Finally, following Halle and Marantz (1993)'s work on distributed morphology, I assume that vocabulary items are inserted after syntax. In the proposed linearization process, the last step is to form lexical insertion sites (32) for each node in the string, and vocabulary items are inserted to these sites based on the features encoded in each node, an example of which is shown in (33). In chapter 5, I explain how this basic linearization process work in corporation with Fox and Pesetsky (2005)'s cyclic linearization.[6] Note

[6]One question about the architecture of this linearization process is how it deals with post-syntactic operations that rearrange the word order (cf. Embick and Noyer (2001)). One example is Latin conjunction, shown in (i). Embick and Noyer (2001) argues that (i) involves local dislocation, which is a post-syntactic operation, that cliticizes the conjunction *=que* to the first syntactic element of the right conjunct (in syntax,

that for ease of presentation, I do not show the features of each node in the examples in this dissertation but assume that the features are there and vocabulary items are inserted based on the corresponding features.

(32) $<_{S}$ #D^0##T^0##V^0#...>

(33) *she can run*

1.3 The phonetic realization of a moved item

Movement is a syntactic operation that causes one item to occupy more than one syntactic position, and normally, the moved item is only pronounced once. For instance, in the English example (34) (repeat of example (2)), the auxiliary verb *can* undergoes T^0-to-C^0 movement and occupies both the T^0 and C^0 position but it is only pronounced in the C^0 position. Another example from English is in (35), where the verb phrase *eating apples* is topicalized and occupies both the VP position and the topic position but it is only pronounced in its topic position.

(34) **T-to-C movement**

a. **Can** she run?

b. ***Can** she **can** run?

the conjunct is in between the two constituents but at PF it is cliticized to the first syntactic element of the right conjunct by local dislocation). I will not go into the details of how to solve this problem but I suggest that following Kusmer (2019), it might be possible that local dislocation could be dealt with by adding more constraints on the set of precedence relations in addition to the constraints proposed in this dissertation. For instance, for the Latin conjunction case, there can be a constraint stating that the conjunction has to follow the first syntactic element but precede the rest of the elements of the right conjunct.

(i) a. boni pueri bonae=que puellae
good boys good=and girls
'good boys and good girls'

b. Syntax: [[boni pueri] que [bonae puellae]]

c. PF: [[boni pueri] [bonae=que puellae]]

(35) **Topicalization**

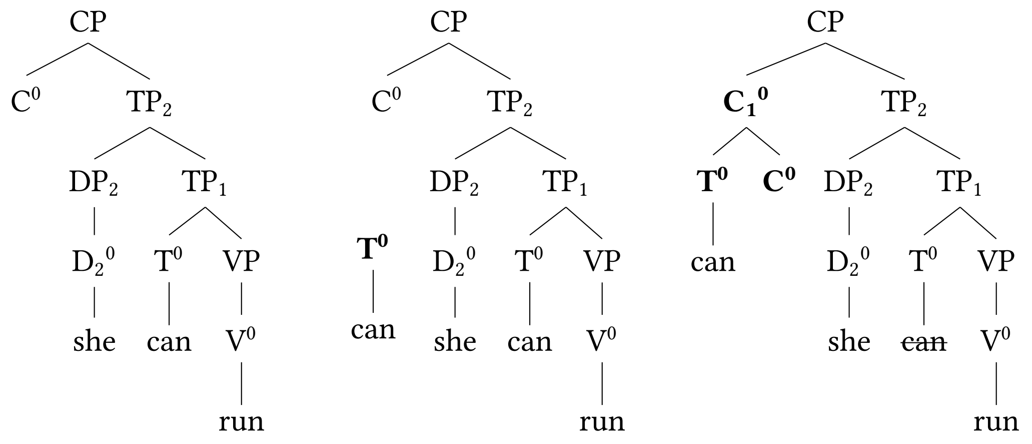
a. **Eating apples**, I like ___ .

b. ***Eating apples**, I like **eating apples**.

Take the English sentence in (2) as an example. Before movement, the structure of the example is shown in 1[7]. Under the copy theory of movement (cf. Chomsky (1993), Nunes (2004) among othters), a copy of T^0 is made 2; and then T^0 and C^0 are merged 3. Finally, the lower copy of T^0 is deleted.

(36)

a. **Build the structure** b. **Make a copy** c. **Merge T^0 and C^0**



However, in some cases, a moved item is pronounced multiple times. One of these cases is verb-doubling, which has been argued to be derived by movement, in Yiddish (DavisPrince 1986, Cable 2004), Hebrew (Landau 2006), and Mandarin (Cheng 2013), among other languages. An example of V-doubling is shown in (37), where the participle *chi-guo* 'eaten' moves but it must be pronounced twice.[8]

[7]For ease of presentation, I ignore vP and the base position of the external subject.

[8]In this dissertation, I use V-doubling as an umbrella term to refer to $V^0/v^0/Asp^0$-doubling.

(37) **Mandarin verb-doubling**

a. **chi-guo**, Lili dique meiyou **chi-guo** bale.
eat-ASP Lili indeed not **eat**-ASP Guava

'As for having eaten, Lili indeed hasn't eaten Guava (but she has seen it before).'

b. ***chi-guo**, Lili dique meiyou bale.
eat-ASP Lili indeed not Guava

Intended: 'As for having eaten, Lili indeed hasn't eaten Guava (but she has seen it before).'

Another case, where a moved item is pronounced more than one time, is VP-doubling, which has been argued to be derived by movement, in Yoruba (Manfredi 1993).[9] Later in this dissertation, I provide new data showing that VP-doubling also exists in Mandarin, which is also derived by movement. An example of VP-doubling in Mandarin is shown in (38), where the VP can be pronounced once or multiple times.[10]

(38) **Mandarin VP-doubling**

a. **Chi-guo bale**, Lili dique meiyou **chi-guo bale**
eat-ASP **Guava** Lili indeed not **eat**-ASP **Guava**

'As for having eaten Guava before, Lili indeed hasn't eaten Guava before.'

b. ??**Chi-guo bale**, Lili dique meiyou .
eat-ASP **Guava** Lili indeed not

'As for having eaten Guava before, Lili indeed hasn't eaten Guava before.'

Problems arise if one applies the same operations for deriving the sentence in (2) to the V(P)-doubling cases in (37a) and (38a): it is unexplained why the lower copy of the

[9]Note that I use VP-doubling as an umbrella term to refer to VP/vP/AspP-doubling.

[10]The judgement reported for (38b) might vary; one of my consultants can accept it to some extent, though he notes other alternatives are preferable—for example, he finds fronting only the object improved in comparison. I and my other consultant agree on the judgement shown.

V(P) in V(P)-doubling cases is also pronounced rather than deleted. This dissertation aims at addressing this problem and provides a linearization process that correctly yields the phonetic realization of the moved items for cases like (2) (chapter 4) and (35) (chapter 5) and also for the verb-doubling cases in (37) and (38) (chapter 6).

1.4 PIC, cyclic linearization and my analysis

In this section, I provide a comparison between Chomsky (2000, 2001), with a focus on the Phase Impenetrability Condition (PIC), and Fox and Pesetsky (2005)'s cyclic linearization, which I adopt in my analysis, and then provide a comparison between Fox and Pesetsky (2005)'s cyclic linearization and my analysis.

1.4.1 PIC vs. cyclic linearization

Both Chomsky (2000, 2001) and Fox and Pesetsky (2005) assume that the mapping between syntax and phonology occurs at various points in the course of the derivation instead of mapping the whole structure to phonology all at once. However, they differ in which syntactic units are mapped, and crucially they have different restrictions on the mapped syntactic units, which results in different predictions about how movement can happen. In the following, I start with their differences in which syntactic units are mapped, and then discuss their different restrictions on those syntactic units, and finally, focus on the different predictions they make for movement.

Regarding the mapped syntactic units, Chomsky (2000, 2001) proposes that it is the complements of the phase heads (i.e., v, C) that should be mapped to phonology in the course of derivation, and refers to the complements of the phase heads as "Spell-out domains". Here, phase is defined as a syntactic object that is relatively independent in terms of interface properties in Chomsky (2000).[11] However, Fox and Pesetsky (2005) has a

[11]Roughly speaking, based on Chomsky (2000), for LF, a phase should be propositional and based on Chomsky (2001), for PF, a phase should have a degree of phonetic independence.

different idea about which syntactic units are mapped — for Fox and Pesetsky (2005), it is all the material in CP, VP and DP that should be mapped during the derivation and Fox and Pesetsky (2005) refer to the entire CP, VP and DP as "Spell-out domains."[12] So, despite both approaches using the same term "Spell-out domain," they refer to different syntactic units.

Take (39) as an example. For Chomsky (2000, 2001), during the derivation, once vP is built, the complement of v is sent to phonology to be linearized, and then later on once the CP layer is built, the complement of C is sent to phonology to be linearized, and this process continues until it reaches the highest CP.[13] For Fox and Pesetsky (2005), once the lowest DP is built, it is linearized and the ordering statements are stored at PF, and later on, once the VP is built, it is linearized and the ordering statements are also stored at PF along with the ordering statements collected from the previous DP Spell-out domain. So, one can imagine that there is a "pool" of linearization statements at PF, and every time a Spell-out domain is linearized, ordering statements are added to the pool, and eventually the pool contains the ordering statements from all the Spell-out domains. This process continues until it reaches the highest CP.

(39) $[_{\text{CP}}$ I $[_{\text{vP}}$ $[_{\text{VP}}$ wonder $[_{\text{CP}}$ which person Mary $[_{\text{vP}}$ $[_{\text{VP}}$ saw $t_{\text{which person}}]]]]]]$

Now that I have presented the syntactic units that are mapped in the two approaches, next I show the different restrictions on the mapped syntactic units between these two approaches. Chomsky (2000, 2001) proposes the Phase Impenetrability Condition (40), which restricts the accessibility of the complement of the head in a phase to operations outside the phase.

[12]Note that Sabbagh (2003) argues that PP is also a Spell-out domain and Ko (2005) argues that vP is also a Spell-out domain.

[13]For ease of discussion, I ignore the base position of the subject, which does not affect the analysis.

(40) In a phase α with head H, the domain of H is not accessible to operations outside α, only H and its edge are accessible to such operations.

Chomsky (2000, 2001)

To be more specific, once a phase is constructed in syntax, the domain (i.e., the complement of the head) in the phase is sent to PF and LF, and it is not accessible to operations (e.g., movement) outside the phase. As a result, if the material in the complement of the head in the current phase needs to move out, it has to move to the edge of the phase first in order to be accessible to this movement operation, assuming that the head that triggers this movement is out of the current phase (if the complement stays in situ, by the time that the higher head that triggers the movement of the complement is constructed, the head can only get access to the edge of the lower phase, but the in situ complement is not at the edge of the lower phase). Here, the edge of a phase includes the phase head and its specifiers and is referred to as "escape hatch" in Chomsky (2000, 2001).[14] An example of such movement is shown in (41), where "who" in the vP phase needs to move to the edge of vP in order to move to Spec, CP.

(41) I wonder [$_{\text{CP}}$ who Mary [$_{\text{vP}}$ ___ saw ___]]

Different from the approach in Chomsky (2000, 2001), Fox and Pesetsky (2005) do not use the notion of "phase" nor do they have the restrictions stated in PIC. Instead, they propose Order Preservation (42), which states that the linearization information of each Spell-out domain must be preserved, and they further propose the restriction that linearization information collected from all the Spell-out domains cannot be contradictory.

[14]Evidence supporting successive cyclic movement through Spec, CP and Spec, VP can be found in literature, such as Brass (1986), Lebeaux (1991) (for CP) and Fox (1999) (for VP).

(42) **Order Preservation**

Information about linearization, once established at the end of a given Spell-out domain, is never deleted in the course of a derivation. The sole function of Spell-out is to add information.

Fox and Pesetsky (2005)

Recall that once a CP/VP/DP is linearized, the ordering statements are added to a pool at PF. So, for instance, if an item moves successive-cyclicly (43a), there will be no contradictory ordering statements in the pool; however, if it does not move successive-cyclicly (43b), the ordering statement "saw $<$ who" is collected from the VP Spell-out domain and "who $<$ saw" is collected in the CP Spell-out domain. Due to Order Preservation, linearization information presented by both ordering statements should be preserved, and thus, both of the statements are kept in the pool at PF. However, since these two ordering statements contradict each other, the linearization process crashes.

(43) a. I wonder [$_{CP}$ who Mary [$_{VP}$ saw]]

b. *I wonder [$_{CP}$ who Mary [$_{VP}$ saw]]

As a quick summary, both approaches have restrictions on the mapped syntactic units/Spell-out domains, but the difference is that Chomsky (2000, 2001) proposes PIC, which restricts the accessibility of the Spell-out domains (i.e., the complement of the head in a phase) to operations outside the current phase; while Fox and Pesetsky (2005) restricts that linearization information collected from each Spell-out domains should be preserved and cannot be contradictory.

The important thing is that their difference in the restrictions of the Spell-out domains lead to different predictions for how movement can happen. For Chomsky (2000, 2001), since the complement of the phase must first move to the escape hatch (i..e, the edge of the current phase) in order to move out, this forces movement to happen in a successive-cyclic fashion instead of moving in one fell swoop. In contrast, Fox and Pesetsky

(2005) does not have such escape hatch since they do not have any syntactic restrictions on the accessibility of the complement of a head, and thus there is no requirement for an item to move in a successive-cyclic fashion. For Fox and Pesetsky (2005), as long as the linearization information collected from each Spell-out domain does not cause the linearization process to crash, movement can happen in a successive-cyclic fashion or a non successive-cyclic fashion. For instance, the previous case in (43) is an example of how successive-cyclic movement satisfies the linearization requirement (i.e., no contradictory linearization information). However, Fox and Pesetsky (2005) also predict that it should be possible for an item X not to move successive-cyclicly (e.g., a phrase does not have to move to the edge of a Spell-out domain), as long as all the material preceding the item X in the Spell-out domain also moves and lands at a higher position than X. Take (44) as an abstract example. Assuming both βP and αP are Spell-out domains, Y moves directly from its base position in βP to a higher position in αP without moving to the edge of βP first. However, this does not cause an issue because the item X that precedes Y also moves to αP and it lands at a position still higher than Y. To be more specific, the ordering statement "X $<$ Y" is collected in the βP Spell-out domain and in αP, it is still "X $<$ Y" that is collected as an ordering statement. Thus, the ordering statements in the pool at PF are consistent (and in this case they are the same). In this case, Y does not move successive-cyclicly, but this is predicted to be allowed by Fox and Pesetsky (2005) because the requirement that there cannot be contradictory linearization information is satisfied.

(44) $[_{\alpha P}$ X Y $[_{\beta P}$ t_X $t_Y]]$

Of course, suppose instead that only Y moved but X did not, as in (45).

(45) $[_{\alpha P}$ Y $[_{\beta P}$ X $t_Y]]$

"X $<$ Y" would be collected in the βP Spell-out domain but "Y $<$ X" would be collected in the αP Spell-out domain. As a result, when both statements are in the pool, they contradict each other and cause the linearization process to crash. Thus, the key is that an item can move in a non successive-cyclic way as long as the material preceding the item also moves. A concrete example of this is object shift, discussed in chapter 5.

To summarize, in Fox and Pesetsky (2005)'s system, there are two ways that movement across a Spell-out domain can occur. One way is for an item X to move successive-cyclicly, and the other is for all the material preceding a moved item X in its initial Spell-out domain to move if X does not move successive-cyclicly. A more detailed discussion of Fox and Pesetsky (2005) is presented in chapter 5.

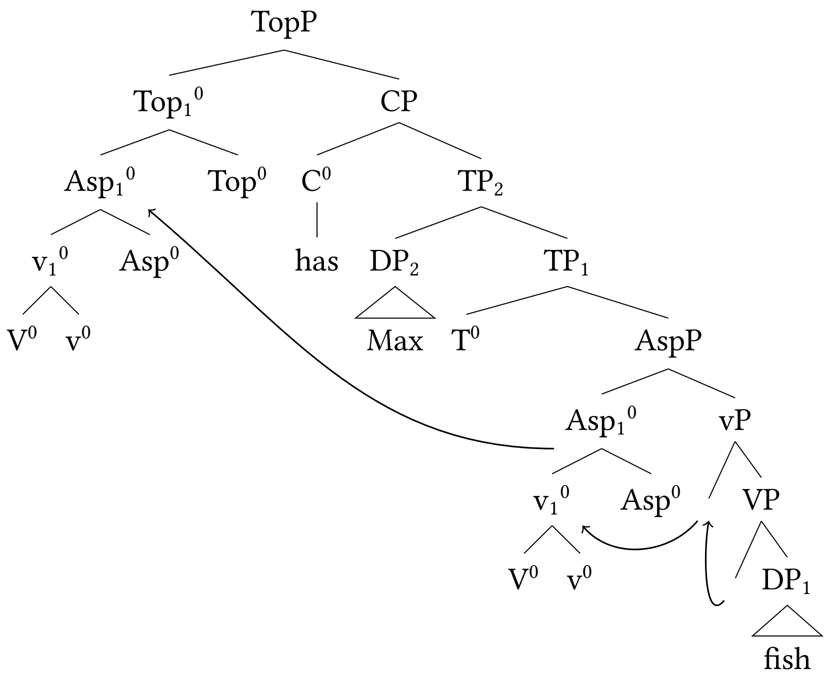
In short, Chomsky (2000, 2001) proposes a notion of phasehood and an accompanying Phase Impenetrability Condition, which forces a moved item to move to an escape hatch in order to move out of a phase, which results in successive-cyclic movement. In contrast, Fox and Pesetsky (2005) do not make use of the notion of phasehood but propose that DP, VP and CP are domains that are mapped from syntax to phonology, and the ordering statements that are cyclicly collected from them are stored at PF and there cannot be contradictory linearization information. Two ways of movement that crosses a Spell-out domain are made possible: one way is for an item to move successive-cyclicly, and the other is to move both an item X and the material preceding X in its initial Spell-out domain to a higher Spell-out domain. Both ways ensure the ordering statements that are collected from the domains are consistent.

1.4.2 cyclic linearization vs. my analysis

In my analysis, following Fox and Pesetsky (2005) instead of Chomsky (2000, 2001), I also take the perspective that an item does not have to move successive-cyclicly as long as the linearization requirement is satisfied. However, my system provides one more way of deriving movement that crosses a Spell-out domain. This new possibility occurs when an item moves non-successive-cyclicly without the material preceding it in its initial Spell-out domain moving, which leads to the moved item being pronounced multiple times. For instance, in an earlier Yiddish example, repeated below as (46), I argue that the verb moves non-successive-cyclically and the material preceding it in the CP Spell-out domain does not move. This movement gives rise to double pronunciation.

(46) **Yiddish**

a. **Gegessen**, hot Maks **gegessen** fish
eaten has Max **eaten** fish
'As for having eaten, Max has eaten fish.' (Cable 2004: 2, ex. 2(a))

b.


My analysis deals with both ways of deriving movement that crosses a Spell-out domain discussed in Fox and Pesetsky (2005) and also with long distance movement, such as the one in (46). It thus provides a more general picture of the mechanism of cyclic spell-out, with a particular focus on what ordering statements should be kept or deleted when cyclic spell-out is applied. The technique to implement this is a rule of Ordering Deletion, which I discuss in more detail in chapter 5.[15]

My analysis also provides a detailed view of the linearization process that differs in some respects from the implementation of Fox and Pesetsky (2005)'s cyclic linearization:

[15]Note that part of the technique deals with deleting ordering statements, not deleting linearization information, which still obeys Fox and Pesetsky (2005)'s Ordering Preservation. More discussion about this is in chapter 5.

(47) a. My analysis uses a different linearization algorithm that generates sets of ordered pairs, which includes the Candidates Generator G and the Constraints following Kusmer (2019). Detailed discussion can be found in chapter 4.

b. Cyclic linearization is applied after the whole syntactic structure is built instead of at different points in the derivation. Concrete examples of how this works are discussed in chapter 5.

c. In addition to DP, VP, and CP, the root node is always a Spell-out domain.

d. All the material inside the current Spell-out domain is linearized (in Fox and Pesetsky (2005), material that is linearized in the previous Spell-out domain is not linearized again in the current Spell-out domain), and the ordering statements from all the Spell-out domains are put together by the union operation, which is discussed in chapter 5.

e. In addition to precedence relations, my analysis also has a string (of nodes) representation, which is discussed in chapter 4.

To summarize, my analysis details another way that movement crossing a Spell-out domain can occur, in addition to the two ways proposed in Fox and Pesetsky (2005), and thus provides a more general way of deciding which ordering statements are kept and which are deleted when cyclic linearization is applied. In addition, I also provide a detailed account of the linearization process that differs in some ways from Fox and Pesetsky (2005)'s cyclic linearization.

1.5 Theoretical assumption: HMC

Based on the Head Movement Constraint (cf. Travis 1984), head movement that skips head(s) (49) is an illegitimate operation. This is true of Germanic languages, however, it has been shown in the literature (cf. Rivero (1994) Koopman (1984), Roberts (2010) among others) that there are cases where head movement skips an intermediate head. For instance, Rivero (1994) argues that in Bulgarian, verbs move across auxiliaries in the present perfect context in main clauses. An example is shown in (48). In this dissertation, I propose that something similar happens in Yiddish — in the Yiddish verb-doubling constructions, v^0 moves to Top^0 skipping intermediate heads.

(48) **Bulgarian**

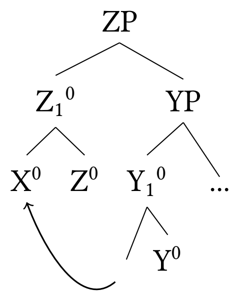
a. **Pročel** sŭm knigata
read I.have book.the

'I have read the book.'

b. *Sŭm **Pročel** knigata (Rivero 1994: 87, ex. 34)

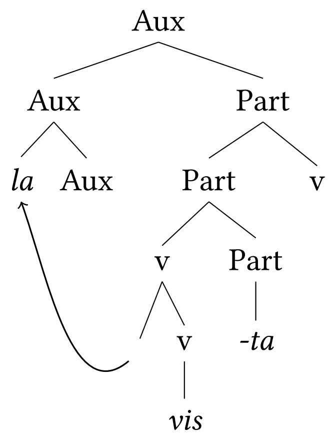
In addition, Head Movement Constraint states that excorporation (49) is also illegitimate.

(49)



However, there is also literature (cf. Roberts (2010)) that argues that in some cases, excorporation is possible. For instance, Roberts (2010) argues that in the Italian sentence *l'ha vista* 'she/he saw her', the clitic *la* excorporates to Aux, which is shown in (50).

(50) **Italian**



(Roberts 2010: 207, ex. 33)

In this dissertation, I assume that excorporation is possible in Yiddish, Hebrew and Mandarin.

1.6 Theoretical background: Multidominance

In this dissertation, I use the multidominance representation (cf. Engdahl (1980), Gärtner (1997), Nunes (2001), Starke (2001), Frampton (2004), Citko (2005), Fitzpatrick and Groat (2005), Johnson (2012), and many others) to model movement, rather than the copy theory of movement (cf. Chomsky (1993), Nunes (2004)).

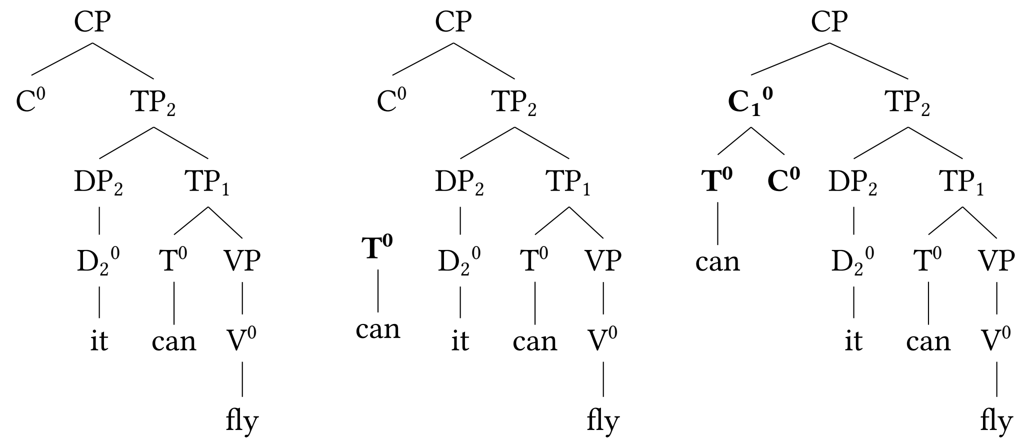
Take the English sentence in (51) as an example. Traditionally, to model movement, the copy theory of movement follows these steps: (i) build up the structure before copying (52a),[16] (ii) make a copy of T^0 (52b), (iii) merge T^0 and C^0 (52c).

[16]For ease of presentation, I ignore vP and the base position of the external subject.

(51) *Can it fly*?

(52)

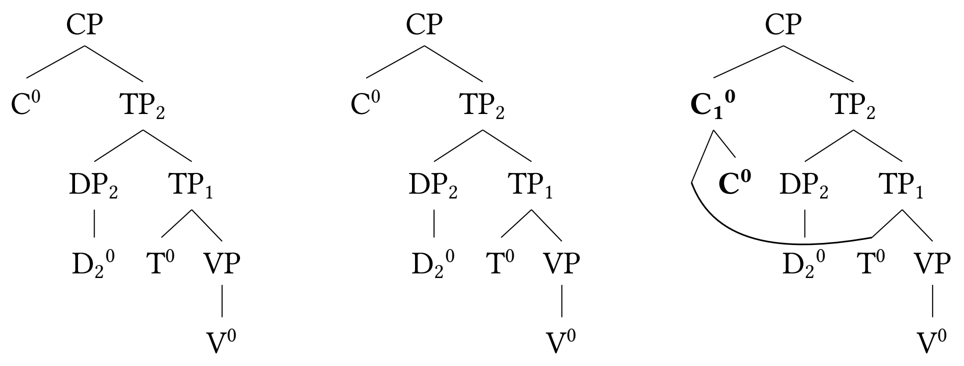
a. **Build the structure** b. **Make a copy** c. **Merge T^0 and C^0**



However, in the multidominance representation, there is no operation of copying; it is just one item but merged twice. In both (52) and (53), one first builds the structure. However, unlike the copy theory of movement (52), there is no copy made in multidominance (53b). Instead, T^0 is remerged with C^0 (53c). So, under multidominance, T^0 is multidominated — T^0 is immediately dominated by both TP_1 and ${C_1}^0$, and has two sisters (i.e., VP and C^0), whereas under the copy theory of movement, the lower copy is immediately dominated by TP_1 with only one sister (i.e, VP) and the higher copy is immediately dominated by ${C_1}^0$ with only one sister (i.e., C^0).

(53)

a. **Build the structure** b. **No copy is made** c. **Remerge T^0 with C^0**



Later in chapter 3, I discuss Nunes (2004)'s analysis, which uses the copy theory of movement, and I will show why multidominance is a better way to model movement than the copy theory of movement, especially with respect to the phonetic realization of a moved item.[17]

1.7 The organization of the dissertation

In chapter 2, I introduce V(P)-doubling constructions, showing evidence for the movement approach, the morphology of the fronted verb and provides a brief introduction of their pragmatics. In chapter 3, I review Nunes (2004)'s and Landau (2006)'s analysis for the phonetic realization of a moved item and discuss their problems, where I will also compare multidominance and the copy theory of movement and show why multidominance is better at modeling movement. In chapter 4, I present a basic linearization process and discuss how it works for cases where the moved item is pronounced in only one of its position. In chapter 5, I introduce Fox and Pesetsky (2005)'s cyclic linearization and update the basic linearization process by incorporating Fox and Pesetsky (2005)'s cyclic linearization, and show how the updated linearization process works for cases where the moved item is

[17]Note that multidominance is a representation of movement and as such is subject to constraints on movement dependencies. For instance, remerge a DP to a CP is not possible if the DP is inside a complex subject.

pronounced in only one of its positions. In chapter 6, I show how the cyclic linearization process works for the verb-doubling cases from Hebrew, Yiddish and Mandarin.

CHAPTER 2

V(P)-doubling constructions

In this dissertation, I focus on verb-doubling constructions with verb fronting and VP-fronting; and VP-doubling constructions with VP fronting. Examples for verb-doubling are shown in (1): (1a) shows verb-doubling with verb fronting, where the root verb [k, n, t] is fronted and doubled; and (1b) shows verb-doubling with VP fronting, where the root verb [sh, t, f] 'buy' and the object *et ha-praxim* 'the flowers' are fronted but only the verb root [sh, t, f] 'buy' is doubled. (2) shows VP-doubling with VP fronting where the verb phrase *jian-guo bale* 'having seen Guava' is fronted and doubled.

(1) **Verb-doubling** (Hebrew)

a. **Verb-doubling (with verb-fronting)**

Li**knot**, hi **k**a**nt**a et ha-praxim
buy.INF she **buy**.PST ACC the-flowers
'As for buying, she bought the flowers.'

b. **Verb-doubling (with VP-fronting)**

Li**knot** et ha-praxim, hi **k**a**nt**a
buy.INF ACC the-flowers she **buy**.PST
'As for buying the flowers, she bought.' (Landau 2006: 6, ex. 8)

(2) **VP-doubling** (Mandarin)

Jian-guo bale, Lili dique mei **jian-guo bale**
see-ASP **Guava** Lili indeed not **see**-ASP **Guava**
'As for having seen Guava before, Lili indeed hasn't seen Guava before.'

In the following sections, I present properties of V(P)-doubling constructions, including evidence that support a movement account of V(P)-doubling, the inflections of the fronted verb and a brief introduction of their pragmatics.

2.1 Evidence for movement

In this section, I discuss evidence supporting the analysis that the fronted verbal material is derived by movement in V(P)-doubling constructions, which include island effects and obligatory verb-matching. I first focus on verb-doubling constructions. Supporting evidence for movement comes from the facts that verb-doubling constructions show island effects (i.e., if an item moves, the two positions that the moved item is related to are subject to locality conditions). Take Hebrew as an example, a sentence is marginally degraded when the (V)P upstairs is across a wh-island (4); a complex NP (5), a subject-island (6) or an adjunct clause (7); however, V(P) fronting is permitted across finite clause boundaries(3), which is typical for A-bar movement.[1]

(3) **Finite clause**

a. V fronting

Lenakot,	nidme	li	[še-Rina	amra	[še-Gil	kvar	nika	et	ha-xacer]].
clean.INFL	seems	to.me	that-Rina	said	that-Gil	already	cleaned	ACC	the-yard

'As for cleaning, it seems to me that Rina said that Gil had already cleaned the yard.'

[1]Note that the English translations for V(P)-doubling cases are sentences that start with "as for V(P)", and "as for V(P)" is base-generated, not derived from moving from downstairs. Thus, these English sentences with "as for V(P)" are not sensitive to islands.

b. VP fronting

Lenakot, et ha-xacer, nidme li [še-Rina amra [še-Gil kvar nika]].
clean.INFL ACC **the-yard** seems to.me that-Rina said that-Gil already cleaned

'As for cleaning the yard, it seems to me that Rina said that Gil had already cleaned.'

(4) **Wh-island**

a. V fronting

??**Likro**, ša'alti [matay Gil kvar kara et ha-sefer].
read.INFL asked.1.SG when Gil already read ACC the-book
'As for reading, I asked when Gil had already read the book.'

b. VP fronting

??**Likro et ha-sefer**, ša'alti [matay Gil kvar kara].
read.INFL ACC **the-book** asked.1.SG when Gil already read
'As for reading the book, I asked when Gil had already read.'

(5) **Complex NP island**

a. V fronting

***Likro**, Gil daxa et ha-te'ana [še-hu kvar kara et ha-sefer].
read.INFL Gil rejected ACC the-claim that-he already read ACC the-book
'As for reading, Gil rejected the claim that he already read the book.'

b. VP fronting

***Likro et ha-sefer**, Gil daxa et ha-te'ana [še-hu kvar kara].
read.INFL ACC **the-book** Gil rejected ACC the-claim that-he already read
'As for reading the book, Gil rejected the claim that he already read.'

(6) **Subject island**

a. V fronting

***Likro**, [še-yevakšu me-Gil še-yikra et ha-sefer] za ma'aliv.
read.INFL that-will.ask.3.PL from-Gil that-will.read.3.SG ACC the-book it insulting
'As for reading, that they would ask Gil to read the book is insulting.'

b. VP fronting

***Likro**	**et**	**ha-sefer**,	[še-yevakšu	me-Gil	še-yikra]	za	ma'aliv.
read.INFL	**ACC**	**the-book**	that-will.ask.3.PL	from-Gil	that-will.read.3.SG	it	insulting

'As for reading the book, that they would ask Gil to is insulting.'

(7) **Adjunct island**

a. V fronting

***Likro**,	nifgašnu	[axarey	še-kulam	kar'u	et	ha-sefer].
read.INFL	met.1.PL	after	that-everybody	read.3.PL	ACC	the-book

'As for reading, we have met after everybody read the book.'

b. VP fronting

***Likro**	et	ha-sefer,	nifgašnu	[axarey	še-kulam	kar'u].
read.INFL	**ACC**	**the-book**	met.1.PL	after	that-everybody	read.3.PL

'As for reading the book, we have met after everybody read.'

(Landau 2006: 43, ex. 23-26)

Additionally, the movement analysis is compatible with the facts that mismatch between the verb roots is not allowed. This phenomenon is also referred to as the absence of any genus-species effects (Cable, 2004). To be more specific, under the movement analysis, it is the same item that occupies multiple positions, so there should not be a mismatch between the verbs. However, under the base-generation analysis, it should be possible for a mismatch between the V(P)s upstairs and downstairs, where the two V(P)s have a genus-species relation. For instance, in (8), the verb root upstairs is 'travel' and the verb root downstairs is 'fly'. The mismatch of the two verb roots in (8) makes the sentence ungrammatical. If the V(P) fronting is derived by base-generation, (8) should have been possible, given that 'travel' and 'fly bears a genus-species relation and thus, semantically the sentence is possible (i.e., it is semantically possible to say the English counterpart of the sentence).

(8) **Hebrew**

***Letayel** le-amerika, **tasti** le-nyu-york
travel.INF to-America **fly**.1.SG.PST to-New-York
'As for traveling to America, I flew to New York.' (Bleaman 2021: 6, ex. 11(a))

In contrast, consider the previous Hebrew examples in (1), the verb roots upstairs and downstairs match and the sentences are grammatical. Note that the verb upstairs and downstairs can have different inflections. For instance, in (1), the verb upstairs is in the infinitival form but the verb downstairs is in the finite form. More discussions about the inflectional forms of the verb upstairs are in the next section. The important point is that whatever is moved should match in the higher and lower position. In the case of (1), it is the verb root that moves, so the verb roots upstairs and downstairs should match.

Having shown evidence supporting a movement analysis for verb-doubling constructions, in the following, I present evidence supporting a movement analysis for VP-doubling constructions using data from Mandarin. VP-doubling in Mandarin also shows island effects. (9) shows that VP fronting is not blocked by a finite clause boundary, which is expected for A-bar movement. In contrast, examples in (10) show that the VP-doubling constructions are sensitive to different kinds of islands.

(9) **Finite clause boundary**

Jian-guo bale, wo zhidao [Lili kending mei **jian-guo bale**].
see-ASP **Guava** 1.SG know Lili definitely not **see**-ASP **Guava**
'As for having seen Guava before, I know Lili definitely hasn't seen Guava.'

(10) a. **Wh-island**

*Xihuan bale, Lili wen [weishenme Moli bu xihuan bale].
like bale Lili ask why Moli not like bale
'As for liking Guava, Lili asks why Moli does not like Guava.'

b. **Complex NP island**

*Xihuan bale, Lili fouding-le [Moli bu xihuan bale de chuanyan].
like bale Lili deny-ASP Moli not like bale DE rumor
'As for liking Guava, Lili denies the rumor that Moli does not like Guava.'

c. **Adjunct island**

*Jian-guo bale, [zai Lili mei jian-guo bale zhiqian], ta zui xihuan
see-ASP Guava PREP Lili not see-ASP Guava before 3.SG most like
xigua
watermelon
'As for having seen Guava before, before Lili has not seen Guava, she likes watermelon the most.'

d. **Coordination island**

*Jian-guo bale, Lili [mei chi-guo bale] bingqie [mei jian-guo bale].
see-ASP Guava Lili not eat-ASP Guava and not see-ASP Guava
'As for having seen Guava before, Lili has not eaten Guava before and has not seen Guava before.'

e. **Relative clause island**

*Xihuan bale, wo jian-guo [bu xihuan bale de ren]
like Guava 1.SG see-ASP not like Guava DE people
'As for liking Guava, I have seen people who do not like Guava.'

In addition, in Mandarin, VP-doubling also requires lexical identity. Both *zou* and *chui* have very similar meaning 'to bash/punch', and both *yonlide* and *shijinde* have roughly the same meaning "firmly/heavily". However, both the verbs and adverbs have to be identical upstairs and downstairs to make the sentence grammatical.[2]

(11) a. * Yonglide zou, wo dique mei shijinde chui
firmly bash 1.sg indeed not heavily punch
'As for bashing firmly, I indeed did not punch heavily.'

b. * Yonglide zou, wo dique mei yonglide chui
firmly bash 1.sg indeed not firmly punch
'As for bashing firmly, I indeed did not punch firmly.'

[2]Note that in (11), the VP that is used for testing the identity condition consists of an adverb and a verb. It is worth pointing out that the lexical identity test is not applicable to the Mandarin data, where the VPs contain a verb and an object. The reason is that Mandarin has the so-called "dangling topic" constructions, where the topic is base-generated (cf. Chafe (1976)), which makes it possible for the VPs upstairs and downstairs to be different. For instance, in 2, the verbs upstairs and downstairs are different and so are the objects. Thus, for VPs that have a verb and an object, I only resort to island tests (10a)-(10e). As for why "dangling topic" is not an option for the VPs in (11), I leave it as an open question.

(ii) Chao fangbianmian, wo dique jingchang yong henduo you
Fry instant.noodles 1sg indeed often use a.lot.of oil
'As for frying instant noodles, I indeed often use a lot of oil.'

c. * Yonglide zou, wo dique mei shijinde zou
firmly bash 1.sg indeed not heavily bash
'As for bashing firmly, I indeed did not bash heavily.'

d. Yonglide zou, wo dique mei yonglide zou
firmly bash 1.sg indeed not firmly bash
'As for bashing firmly, I indeed did not bash firmly.'

So far, I have shown evidence supporting the analysis that the fronted verb or VP in V(P)-doubling constructions is derived by movement. In the next section, I discuss the possible forms of inflections of the verb upstairs.

2.2 The morphology of the fronted verb

In V(P)-doubling constructions, the fronted verb can be in the infinitive form (12), in the nominalized form (13) or with aspectual marking (14). Based on the typological study of V(P)-doubling in Hein (2018), the fronted material can be V(P), v(P), or Asp(P) but material higher than Asp(P) is not found. For instance, tense marking is not found on the fronted verb.[3]. In Chapter 6, I will show how the morphology of the fronted verb is derived.

(12) **Infinitive** (Hebrew)

a. Likro, hu kara et ha-sefer.
read.INF 3.SG read ACC the-book
'As for reading, he read the book.'

b. Likro et ha-sefer, Gil kara.
read.INF ACC the-book Gil read
'As for reading the book, Gil read.' (Landau 2006: 50, ex. 37(a)-(b))

(13) **Nominalized** (Yoruba)

a. Rí-rà ni, Ajé ra ìwé.
NMLZ-buy FOC Aje buy paper
'It is a buying that Aje {is doing/did} to {a book/books} [i.e., he didn't steal it/them].'

[3] I leave it as an open question regarding why material higher than Asp(P) (e.g., T(P)) is banned from moving to the topic/focus position in V(P)-doubling constructions.

b. Rí-rà-ìwé ni, Ajé ra ìwé.
NMLZ-buy-paper FOC Aje buy paper
'It is book-buying that Aje {is doing/did} [i.e., he didn't go yam-selling].'

(Manfredi 1993: 20, ex. 46(a)-(b))

(14) **Aspect** (Yiddish)

a. Gegessen, hot Maks gegessen fish.
eat.PFTV has Max eat.PFTV fish
'As for having eaten, Max has eaten fish.'

b. Gegessen fish, hot Maks gegessen.
eat.PFTV fish has Max eat.PFTV
'As for having eaten fish, Max has eaten (them).'

(Cable 2004: 2, ex. 2(a)-(b))

2.3 The pragmatics of V(P)-doubling

In this dissertation, I do not provide a detailed analysis for the pragmatics of V(P)-doubling, but provide a brief introduction about the pragmatic contexts where V(P)-doubling is used. Crosslinguistically, V(P)-doubling constructions have either a topic (e.g., Hebrew, Yiddish and Portuguese, etc.) or a focus (e.g., African and Caribbean Creole languages, etc.) interpretation. An example for the topic interpretation is shown in (15a); and an example for the focus interpretation is shown in (15b), where *axerim* 'others' is focused.

(15) **Hebrew**

a. lihyot zamin, Gil lo tamid haya.
to.be available Gil not always was
'As for being available, Gil wasn't always.'

(Landau 2006: 10, ex. 18(a))

b. le-Rina yeš xuš humor, aval licxok hi coxeket rak al axerim.
to-Rina there.is sense humor but to.laugh she laughs only on others
'Rina has a sense of humor, but as for laughing, she will only laugh on others.'

(Landau 2006: 10, ex. 16(a))

In this chapter, I provide a detailed discussion of the verb doubling constructions. In the next chapter, I discuss the previous analyses for the phonetic realization of a moved item and provide an introduction to multidominance, which I use in the proposed analysis.

CHAPTER 3

Previous analyses

In this chapter, I provide background information for the V(P)-doubling cases. Specifically, I discuss two previous analyses for verb-doubling: one from Nunes (2004) in section 3.1 and the other from Landau (2006) in section 3.2.

3.1 Nunes (2004)

Nunes (2004) provides both an analysis for cases where the moved item is only pronounced in one of its positions, and an analysis for the verb-doubling constructions under the copy theory of movement. Note that Nunes (2004) does not provide an analysis for VP-doubling constructions. In section 3.1.1, I provide a brief introduction to Kayne (1994)'s Linear Correspondence Axiom (LCA), which is adopted in Nunes (2004). In section 3.1.2, I present Nunes (2004)'s analysis for cases where the moved item is only pronounced once. In section 3.1.3, I discuss Nunes (2004)'s analysis for cases where the moved item is pronounced multiple times. In section 3.1.4, I pose some problems for Nunes (2004)'s analysis and provide a brief preview of how these problems are dealt with in my analysis. Finally, in section 3.1.5, I compare the copy theory of movement and the multidominance representation of movement and show why multidominance is better at modeling movement given the technics used in Nunes (2004)'s analysis.

3.1.1 Kayne (1994)'s Linear Correspondence Axiom

Kayne (1994) proposes Linear Correspondence Axiom that maps a syntactic structure to precedence relations (i.e., a set of ordered pairs); specifically, he proposes that asymmetric c-command in the syntactic structure maps into linear precedence. LCA is defined in (1).[1] Kayne (1994) defines c-command in (2).

(1) **Linear Correspondence Axiom**

Let X, Y be nonterminals and x, y terminals such that X dominates x and Y dominates y. Then, if X asymmetrically c-commands Y, x precedes y.

(2) α c-commands β iff α and β are categories and every category that dominates α dominates β, and α excludes β.

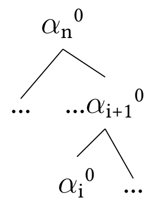
(3) A category α excludes β iff no segment of α dominates β.

(4) α dominates β if every segment of α contains β.

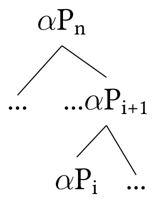
The definition of category and segment are shown in (5) (cf. May (1985) and Chomsky (1986)).

(5) A category is the set of α^0 in (6) such that each $\alpha_{i+1}{}^0$ is a projection of $\alpha_i{}^0$, or the set of αP in (7) such that each αP_{i+1} is a projection of αP_i. Each $\alpha_i{}^0$ or αP_i is a segment of that category.

(6)



(7)

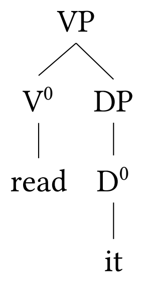


Take (8) as a concrete example. In this case, There are four categories, which are shown in Table 3.1.

[1]Note that this section is largely based on Johnson (2004).

(8) *read it*

(9)



Category	**Segments in the category**
{V^0}	V^0
{VP}	VP
{DP}	DP
{D^0}	D^0

Table 3.1: Categories and Segments

V^0 c-commands D^0 because (i) they are both categories, (ii) every category that dominates V^0 also dominates D^0 (i.e., VP dominates V^0 and D^0), and (iii) V^0 excludes D^0 (i.e., all the segments of category V^0 (i.e., V^0) do not dominate D^0). However, D^0 does not c-command V^0 because despite the fact that (i) they are both categories and (ii) D^0 excludes V^0, there exists the category DP that dominates D^0 but does not dominate V^0. In this sense, V^0 asymmetrically c-command D^0. The relevant dominance and exclusion relations are shown in (10) and (11).

(10) a. The category {VP} dominates V^0 and D^0.
b. The category {DP} dominates D^0.

(11) a. The category {V^0} excludes VP, DP and D^0.
b. The category {D^0} excludes DP, VP and V^0.

Thus, based on LCA, the terminal node *read*, which is dominated by the non-terminal node

V^0, precedes the terminal node *it*, which is dominated by the non-terminal node D^0,which is shown in (12). "A < B" means A precedes B.

(12) {*read* < *it*}

In addition, Kayne (1994) proposes that the linear ordering that LCA yields should have three defining properties, which are formulated by Johnson (2016) in (13). These conditions require that all the vocabulary items in the sentence should be linearized and the linearization should be consistent. In Kayne (1994) and Nunes (2004), vocabulary items are the same as terminal nodes.

(13) a. All vocabulary items in the phrase marker *p* must be in the linearization of *p*. (Totality)

b. For all vocabulary items, *a* and *b* in *p*, the linearization of *p* cannot include both $a < b$ and $b < a$. (Antisymmetry)

c. For all vocabulary items, *a*, *b*, *c* in *p*, if the linearization of *p* includes $a < b$ and $b < c$ then it must include $a < c$. (Transitivity)

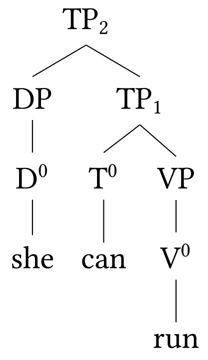
In the case of (8), it satisfies the Totality Condition because all the vocabulary items (i.e., *read* and *it*) are ordered. It satisfies the Antisymmetry Condition because there is no contradictory orderings. It trivially satisfies the Transitivity Condition because there is only one ordering statement in (8).

Another concrete example, which involves specifiers, is shown in (14). The structure of it is shown in (15). For ease of presentation, I ignore vP and the base-generated position of the agent. A summary of the a-symmetric c-command relations are shown in Table 3.2.[2] The categories and segments are summarized in Table 3.3.

[2]Note that under LCA, all the non-terminal nodes (i.e., TP_2, TP_1, DP, D^0, T^0, VP, V^0) are linearized, so Table 3.2 is not exhaustive. However, this table contains all the necessary asymmetric c-command relations, and the asymmetric c-command relations that are left out do not add any information to this table.

(14) *She can run.*

(15)



Non-terminal nodes	Asymmetrically command	Non-terminal nodes
DP	Asymmetrically command	{TP^1, TP^2}
T^0	Asymmetrically command	V^0

Table 3.2: Asymmetric c-command

Category	Segments in the category
{TP$_1$, TP$_2$}	TP$_1$, TP$_2$
{DP}	DP
{D^0}	D^0
{VP}	VP
{T^0}	T^0
{VP}	VP
{V^0}	V^0

Table 3.3: Categories and Segments

The relevant dominance and relations shown in (16) and (17).

(16) a. The category {DP} dominates D^0, *she*

b. The category {TP_1, TP_2} dominates T^0, VP, V^0, *run*, *can*

c. The category {VP} dominates V^0, *run*

(17) a. The category {DP} excludes {TP_1, TP_2} (i.e., no segment of the category {DP} dominates {TP_1, TP_2}).

b. The category T^0 excludes V^0.

Note that DP c-commands {TP_1, TP_2} because (i) they are both categories, (ii) every category that dominates DP also dominates {TP_1, TP_2} (i.e., this is trivially satisfied because there is no category that dominates DP — the category {TP_1, TP_2} does not dominate DP because there exists a segment TP_1 that does not dominate DP), and (iii) DP excludes {TP_1, TP_2}. However, {TP_1, TP_2} does not c-command DP because despite the fact that (i) they are both categories, (ii) every category that dominates {TP_1, TP_2} also dominates DP, {TP_1, TP_2} does not exclude DP (i.e., there is a segment of the category {TP_1, TP_2} (i.e., TP_2) that dominates DP). Thus, DP asymmetrically c-commands {TP_1, TP_2}. The precedence relations are shown in (18).

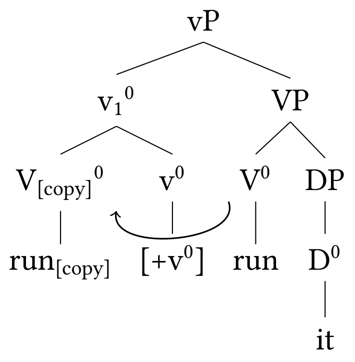
(18) $$\left\{\begin{matrix} she<can & can<run \\ she<run & \end{matrix}\right\}$$

The set of precedence relations in (18) satisfies the Totality Condition because all the vocabulary items (i.e., *she*, *can*, *run*) are in the set; it satisfies the Antisymmetry Condition because there is no contradictory ordering statements; and it satisfies the Transitivity Condition because the set does have the ordering statement *she* < *run* when the set has *she* < *can* and *can* < *run*.

Finally, I show how LCA works for head movement cases (19). A summary of the a-symmetric c-command relations are shown in Table 3.4. The categories and segments are summarized in Table 3.4.

(19) a. *run it*

b.



Non-terminal nodes	Asymmetrically command	Non-terminal nodes
$\{v^0, v_1{}^0\}$	Asymmetrically command	V^0, DP, D^0
$\{V_{[copy]}{}^0\}$	Asymmetrically command	$\{v^0, v_1{}^0\}$, V^0, D^0, DP
$\{V^0\}$	Asymmetrically command	D^0

Table 3.4: Asymmetric c-command

Category	Segments in the category
{vP}	vP
$\{v^0, v_1{}^0\}$	v^0, $v_1{}^0$
$\{V_{[copy]}{}^0\}$	$V_{[copy]}{}^0$
{VP}	VP
$\{V^0\}$	V^0
{DP}	DP
$\{D^0\}$	D^0

Table 3.5: Categories and Segments

The relevant dominance and relations shown in (20) and (21).

(20) a. The category {vP} dominates v^0, $v_1{}^0$, $V_{[copy]}{}^0$, VP, V^0, DP, D^0, *run$_{[copy]}$*, [+v^0], *run*, *it*

b. The category {VP} dominates V^0, DP, D^0, *run*, *it*

c. {v^0, $v_1{}^0$} dominates [+v^0].

d. The category {DP} dominates D^0, *it*

(21) a. The category {$V_{[copy]}{}^0$} excludes {v^0, $v_1{}^0$} (i.e., no segment of the category {$V_{[copy]}{}^0$} dominates {v^0, $v_1{}^0$}).

b. The category {v^0, $v_1{}^0$} excludes VP.

c. The category {V^0} excludes D^0.

Note that $V_{[copy]}{}^0$ asymmetrically c-commands {v^0, $v_1{}^0$}. $V_{[copy]}{}^0$ c-commands {v^0, $v_1{}^0$} because (i) they are both categories, (ii) every category that dominates $V_{[copy]}{}^0$, namely vP, dominates {v^0, $v_1{}^0$} (note that {v^0, $v_1{}^0$} does not dominate $V_{[copy]}{}^0$ because there is the segment $v_1{}^0$ that dominates $V_{[copy]}{}^0$), and (iii) $V_{[copy]}{}^0$ excludes {v^0, $v_1{}^0$}. However, {v^0, $v_1{}^0$} does not c-command $V_{[copy]}{}^0$ because despite the fact that (i) they are both categories, (ii) everything that dominates {v^0, $v_1{}^0$} (i.e., vP) dominates $V_{[copy]}{}^0$, {v^0, $v_1{}^0$} does not exclude {$V_{[copy]}{}^0$} (i.e., there exists the segment $v_1{}^0$ that dominates $V_{[copy]}{}^0$).

Based on the asymmetric c-command relations, the precedence relations are shown in (22). A string representation of the precedence relations is *run$_{[copy]}$ [+v^0] run it.* Note that under LCA, both of the copies (i.e., *run* and *run$_{[copy]}$*) are linearized. In the next section, I present Nunes (2004)'s analysis for how only one of the copies ends up being pronounced.

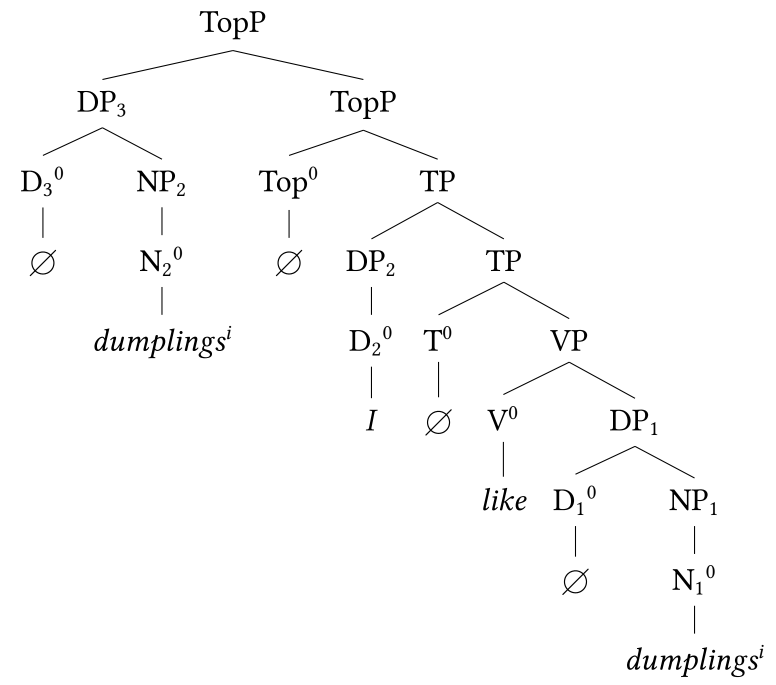
(22)
$$\left\{\begin{array}{lll} \textit{run}_{[copy]}<[+v^0] & [+v^0]<\textit{run} & \textit{run}<\textit{it} \\ \textit{run}_{[copy]}<\textit{run} & [+v^0]<\textit{it} & \\ \textit{run}_{[copy]}<\textit{it} & & \end{array}\right\}$$

3.1.2 A moved item being pronounced once

In this section, I review Nunes (2004)'s analyses for the phonetic realization of a moved item. Take (23) as an example, where the moved item "dumplings" occupies two positions but is pronounced in only one of its positions. The structure for (23) is shown in (24). For ease of presentation, I ignore vP, the base-generation position of the subject, and the morphology under T^0; in addition, only terminal nodes that do not correspond to empty morphology are ordered in this example. Note that this simplification does not affect the analysis.

(23) Dumplings, I like

(24)



LCA takes the structure in (24) as the input and yields a set of ordered pairs in (25).

(25)
$$\left\{ \begin{array}{lll} \textit{dumplings}^i < I & I < \textit{like} & \textit{like} < \textit{dumplings}^i \\ \textit{dumplings}^i < \textit{like} & I < \textit{dumplings}^i & \\ \textit{dumplings}^i < \textit{dumplings}^i & & \end{array} \right\}$$

Under the copy theory of movement, there are two copies of the moved item: *dumplings*^{i} under the node $N_1{}^0$ and *dumplings*^{i} under the node $N_2{}^0$. Nunes (2004) proposes that for the Antisymmetry Condition, the two *dumplings*^{i} are treated the same in the sense that if a set of ordered pairs contains both *dumplings*^{i} $<$ *a* and *a* $<$ *dumplings*^{i}, where *dumplings*^{i} are copies, the set is considered to violate the Antisymmetry Condition. Thus, since the set in (25) indeed contains *dumplings*^{i} $<$ *I* and *I* $<$ *dumplings*^{i}, the set violates the Antisymmetry Condition. Note that the two *dumplings*^{i} are treated differently by the Totality Condition because Nunes (2004) proposes that both the *dumplings*^{i} under $N_1{}^0$ and the *dumplings*^{i} under $N_2{}^0$ should be linearized. In addition to the three conditions in (13), Nunes (2004) also assumes that a linear ordering should obey the Irreflexivity Condition, which is defined in (26). Given that Nunes (2004) assumes that the two copies of *dumplings*^{i} are treated the same by the Irreflexivity Condition, the set (25) also violates the Irreflexivity Condition.

(26) If xRy, then x is different from y.

(Partee, ter Meulen and Wall 1990)

However, according to Nunes (2004), despite the fact that the set violates the Antisymmetry Condition and Irreflexivity Condition, the linearization process does not crash. Instead, a deletion process is trigged at PF, and this deletion process removes one of the copies in the linearization process, which will change the set of ordered pairs to (27) or (28). Both (27) and (28) satisfy the conditions in (13) and (26) — both get rid of paradoxical ordering statements (i.e., *dumplings*^{i} $<$ *I* and *I* $<$ *dumplings*^{i}) and remove the statement where the two copies are ordered to each other (i.e., *dumplings*^{i} < *dumplings*^{i}).

(27) $$\begin{Bmatrix} \textit{dumplings}^i < I & I < \textit{like} \\ \textit{dumplings}^i < \textit{like} & \end{Bmatrix}$$

(28) $$\begin{Bmatrix} I < \textit{like} & \textit{like} < \textit{dumplings}^i \\ I < \textit{dumplings}^i & \end{Bmatrix}$$

So, there are two possible outcomes: (i) Dumplings, I like, where *dumplings*^{i} under $N_2{}^0$ is spelled out and *dumplings*^{i} under $N_1{}^0$ is deleted; and (ii) I like dumplings, where *dumplings*^{i} under $N_1{}^0$ is spelled out and *dumplings*^{i} under $N_2{}^0$ is deleted. Nunes (2004) further proposes a mechanism that determines which copy is to be deleted. According to Nunes (2004), the structure in (24) has a topic force and this topic force has to be realized by pronouncing the DP that is associated with it — in this case, DP_3; without realizing the topic force the derivation fails. In this sense, since the topic force is only realized in (i) *Dumplings, I like*, where *dumplings*^{i} is linearized high not (ii) *I like dumplings*, where *dumplings*^{i} is linearized low, (i) should be the outcome, where "dumplings" is pronounced in the topic position. So, (27) is the final precedence relations for the structure in (24).

Remember that in the previous section, in example (19) *run it*, the deletion process is triggered because the precedence relations violate the Anti-symmetry Condition. To be more specific, the set of precedence relations (repeated below in (29)) contain both $run_{[copy]}$ $<$ [+v^0] and [+v^0] $<$ *run*. After deleting the lower copy of *run*, the sentence is linearized as *run [+v^0] it* (30), instead of *run [+v^0] run it*.

(29)
$$\left\{\begin{array}{lll} run_{[copy]} < [+v^0] & [+v^0] < run & run < it \\ run_{[copy]} < run & [+v^0] < it & \\ run_{[copy]} < it & & \end{array}\right\}$$

(30)
$$\left\{\begin{array}{ll} run_{[copy]} < [+v^0] & [+v^0] < it \\ run_{[copy]} < it & \end{array}\right\}$$

As a short summary, pronouncing a moved item in all its positions fails the linearization process due to the violation of the Antisymmetry Condition and the Irreflexivity Condition. However, Nunes (2004) proposes that the violations can be resolved by a deletion process that can remove a moved item in all but only one of its syntactic positions in the linearization process. As for in which position a moved item should get pronounced depends on the particular formal feature of the copies; in the case of example (23), the topic force is the particular formal feature that prevents the higher copy to be deleted. In general, Nunes (2004) proposes that it is always possible for a moved item to get pronounced in all its syntactic positions, but the violation of the Antisymmetry Condition and the Irreflexivity Condition forces a deletion process at PF, which results in the moved item being pronounced in only one of its positions.[3]

3.1.3 A moved item being pronounced multiple times

Regarding cases where multiple copies are spelled out in head movement, Nunes (2004) proposes the mechanism of morphological reanalysis, where the higher copy is fused to be part of a larger head. To be more specific, in Nunes (2004), it is assumed that all the terminal nodes get linearized, unless morphological reanalysis is applied to two terminal nodes they become one single node and enter the linearization process as a whole. In the following, I use the Vata data in (31) as an example to illustrate how morphological reanalysis works. The structure of the Vata data given in Nunes (2004) is shown in (32). Following Nunes (2004), I also ignore the subject and the object in this case and only focus on the V^0-to-Foc^0 movement; and in addition, I assume that the verb moves directly to Foc^0 due to the fact that T^0 is already occupied by the auxiliary verb following Koopman (1984) and Nunes (2004).

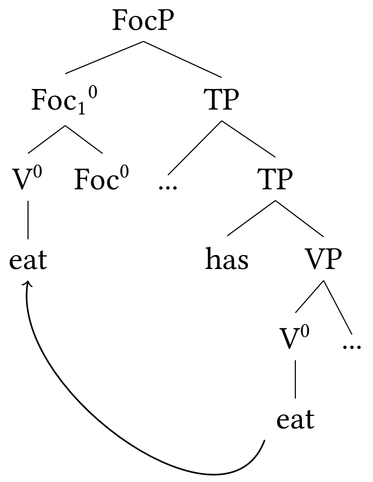
(31) **Vata**

li	O	da	saka	li
eat	s/he	PFV-AUX	rice	eat

‘S/he has EATEN rice.’ (Koopman 1984: 38, ex. 50(b))

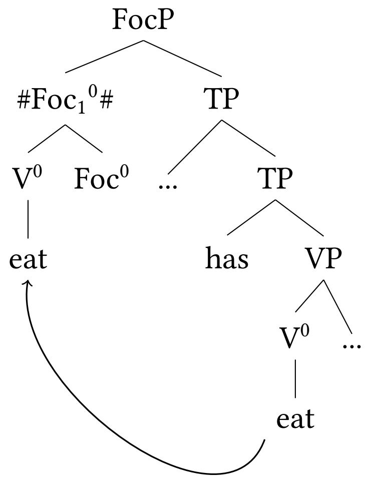
[3]According to Nunes (2004), a derivation where a moved item is spelled out in all its positions is more economical than a derivation where a moved item is spelled out in some but not all of its positions because everything else being equal, an operation of deleting a moved item in a position is needed when a moved item is not spelled out in a position; this increases the overall number of operations and thus makes the derivation less economical.

(32)



In (32), there are two copies of V^0: one in VP and the other in $Foc_1{}^0$. Nunes (2004) claims that $Foc_1{}^0$ obligatorily undergoes morphological reanalysis, where the two terminal nodes (i.e., V^0 and Foc^0) under $Foc_1{}^0$ is reanalyzed as one terminal node, which is marked as #$Foc_1{}^0$#, which is shown in (33).

(33)



Nunes (2004) proposes that if two terminal nodes undergo morphological reanalysis, they can no longer be linearized (i.e., they cannot be seen by the linearization process); instead, the complex head composed by the two terminal nodes will be linearized. In this sense, in (33), only the lower copy of V^0 in VP will be linearized. The higher copy of V^0 in #$Foc_1{}^0$# and Foc^0 will not be linearized but #$Foc_1{}^0$# will be linearized, despite the fact that V^0 and

Foc^0 are terminal nodes. And so this avoids a violation of Antisymmetry and Irreflexivity.

To summarize, morphological reanalysis keeps one of the copies invisible from linearization and allows only one of the copies and the complex head that contains the other copy to enter linearization. Unlike the two copies that are treated as the same item/node (i.e., V^0) in the linearization, the complex head and the lower copy are distinct nodes (i.e., $\#\text{Foc}_1{}^0\#$ and V^0) and thus, will not result in paradoxical orderings. For instance, after morphological reanalysis, there will be ordering statements like *li 'eat' ($\#Foc_1{}^0\#$) < da 'has'* and *da 'has' < li 'eat' (V^0)* but they are not contradictory. If morphological reanalysis is not applied, there will be ordering statements like *li 'eat' (V^0) < da 'has'* and *da 'has' < li 'eat' (V^0)*, which are contradictory.

3.1.4 Problems

So far, I have presented the analysis in Nunes (2004) that explains cases where the moved item is pronounced in one of its positions and cases where the moved item is pronounced in multiple positions. However, there is one problem with Nunes (2004), especially regarding morphological reanalysis, which is it is not predictable which complex verb can undergo morphological reanalysis and which cannot. For instance, morphological reanalysis cannot apply to V^0-to-T^0 movement since if it can, it should predict that verb-doubling happens for V^0-to-T^0 movement, which is not true. Note that this problem is also pointed out in Nunes (2004), but the solution is to stipulate that for some complex verbs (e.g., Foc^0), morphological reanalysis must apply but for some verbs (e.g., T^0), either morphological reanalysis does not apply or it applies but the terminal nodes inside the complex head should still not be linearized. Thus, a more explanatory solution is needed for under what conditions verb-doubling occurs.

In addition, Nunes (2004) proposes that morphological reanalysis should only be applied to heads but not maximal projections, which predicts that VP-doubling is not possible. However, VP-doubling is indeed found, for instance, in Yoruba (Manfredi (1993)) and Mandarin. So, an analysis for VP-doubling is needed.

In my analysis, regarding the verb-doubling constructions, different from Kayne (1994) and Nunes (2004) where the terminal nodes are linearized, I propose that only the highest X^0 nodes are linearized and I also propose an Ordering Deletion rule that gets rid of redundant ordering statements. Regarding the VP-doubling constructions, I employ the idea behind morphological reanalysis that a higher level of node is linearized instead of the lower ones under certain circumstances, but with a different technique.

Finally, in Nunes (2004), the goal of the deletion process is to get rid of the ordering statements that contain the copy which does not end being pronounced. However, in Nunes (2004), the deletion process does not delete ordering statements but deletes copy(s) in a chain. The problem is it is unclear what "deletes copy(s)" means here. To be more specific, it cannot be the case that the copy, as part of the structure, is deleted in syntax since the complete structure is needed to get interpreted at LF. In this sense, by saying "deletes copy(s)", it actually means ignoring the copy/part of the structure when linearization happens. In my analysis, I propose a linearization process, where only linearizing one of the copies is a structural decision, which is forced by the Language Specific Constraints. The details of the Language Specific Constraints are discussed in chapter 4.[4]

3.1.5 Copy theory vs. Multidominance

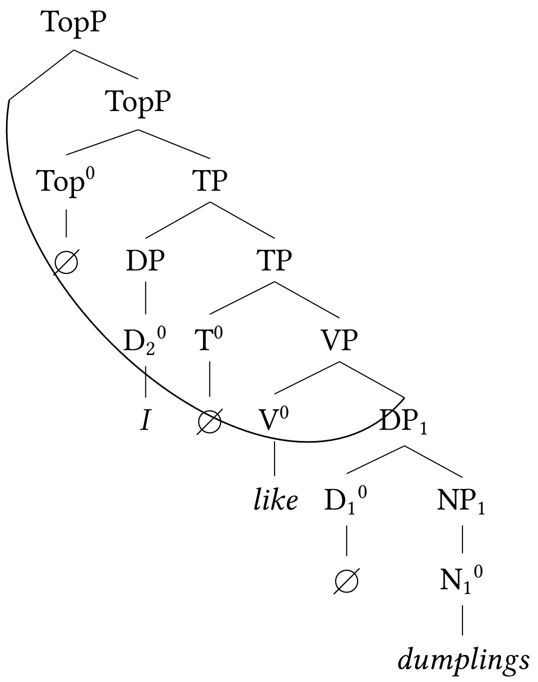
In Nunes (2004), two stipulations are made in the analysis: (i) the two copies are treated the same by the Antisymmetry Condition and the Irreflexivity Condition but treated as different terms by the Totality Constraint[5]; and (ii) when one of the copy precedes but the other follows the same item, there is a violation of the Antisymmetry Condition, and when two copies are ordered relative to each other, there is a violation of the Irreflexivity

[4]My analysis does not predict the "scattered deletion" cases in Nunes (2004), where part of the head and part of the trace get deleted, instead of the whole head or the whole trace being deleted. However, it is possible that these cases are not derived by deletion, but by items moving independently. In this dissertation, I do not go into the details of these cases, but leave it for future research.

[5]In Nunes (2004), the two copies are treated as different terms by the Totality Constraint, which forces both of the copies to be linearized.

Condition; so a deletion process is triggered to delete one of the copies to avoid the violations. Both of these stipulations can be removed if one adopts the multidominance representation of movement, which uses the remerge operation (cf. Engdahl (1980), Gärtner (1997), Nunes (2001), Starke (2001), Frampton (2004), Citko (2005), Fitzpatrick and Groat (2005), Johnson (2012), and many others). Take (23) *Dumplings, I like* as an example. A multidominance representation of it is shown in (34).

(34)



In (34), instead of making a copy of DP_1, DP_1 is remerged with TopP, so there is only one phrase DP_1 but occupies two different positions: the sister of V^0 and the sister of TopP. So, the first stipulation that there are two indistinguishable copies is removed; instead there is only one term (i.e., DP_1) in the structure. In (34), if the linearization process evaluates the moved item DP_1 in both of its positions, Antisymmetry arises because *dumplings* will be ordered by LCA to precede *like*, (DP_1 being the sister of TopP) but also follow *like* (DP_1 being the sister of V^0). This creates a violation of the Antisymmetry Condition. The full precedence relations are shown in (35). Note that since there is only one item (i.e., DP_1), the linearization algorithm will not produce *dumplings*^{i} $<$ *dumplings*^{i} so that this set does not violate the Irreflexivity Condition.

(35) $$\begin{Bmatrix} \textit{dumplings}^i < I & I < \textit{like} & \textit{like} < \textit{dumplings}^i \\ \textit{dumplings}^i < \textit{like} & I < \textit{dumplings}^i & \end{Bmatrix}$$

However, unlike the copy theory of movement, since the moved item is multidominated — there is only one term of the moved item in the structure — only linearizing the moved item in one of its positions will satisfy all the conditions (i.e., the Antisymmetry Condition, the Irreflexivity Condition and the Totality Condition). Specifically, without resorting to the deletion process proposed in Nunes (2004), the linearization algorithm can yield the precedence relations in either (36) or (37), depending on which position the moved item is linearized under. In other words, in the multidominance representation, a moved item can be linearized in one of its positions or in multiple positions, but in order to satisfy all the conditions, the moved item can only be linearized in one of its positions.

(36) $$\begin{Bmatrix} \textit{dumplings}^i < I & I < \textit{like} \\ \textit{dumplings}^i < \textit{like} & \end{Bmatrix}$$

(37) $$\begin{Bmatrix} I < \textit{like} & \textit{like} < \textit{dumplings}^i \\ I < \textit{dumplings}^i & \end{Bmatrix}$$

As for how to get the moved item pronounced in its higher position, solutions can be found in Johnson (2012) and Wilder (1999), among others. I do not go into the details of the solutions here but I have a detailed discussion about a solution in chapter 4, where I present the proposed basic linearization algorithm.

As a quick summary, it can be seen that using the multidominance representation can get rid of the stipulations made in Nunes (2004) but still keeps the crucial idea that linearizing a moved item in all of its positions can cause Antisymmetry and Irreflexivity problems, so the moved item can only be linearized in one of its positions. As for which position should the moved item be linearized, this is a separate question but can be solved by putting some constraints in the linearization process, which I will show how this can be done using the multidominance representation in chapter 4.

3.2 Landau (2006)

Landau (2006) provides a different view from Nunes (2004)'s regarding why movement normally results in a term being pronounced in only one of its syntactic positions. Landau (2006) claims that it is always preferred not to pronounce a moved item in any of its positions unless there is some force that requires a moved item to be pronounced in one or more of its positions. Notice that the preference of always not pronouncing a moved item in any of its positions in Landau (2006) is opposite to the preference for always pronouncing a moved item in all of its positions in Nunes (2004).

According to Landau (2006), there are two principles that determine the (non-)pronunciation of a moved item, which are listed in (38a) and (38b). The "up to P-recoverability" in (38a) indicates that P-recoverability always overrides economy (i.e., deleting all chain copies).

(38) a. *Economy of Pronunciation*
Delete all chain copies at PF up to P-recoverability.

b. *P-Recoverability*
In a chain <X_1...X_2...X_n>, where some X_i is associated with phonetic content, X_i must be pronounced.

According to Landau (2006), X_i is "associated with phonetic content" if X_i has a high pitch accent or X_i supports an affix that cannot stand alone.

Take (23) as an example: "dumplings" in the topic position is associated with a high pitch accent which is imposed by the topic head Top^0, and thus economy is overridden and "dumplings" in the topic position gets pronounced; however, "dumplings" in the logical object position is not associated with any high pitch accent, nor does it host an affix. Thus, “dumplings” in the logical object position gets deleted due to economy (i.e., pronouncing as few copies as possible). Notice that though Landau (2006) also uses the word “economy”, it is different from Nunes (2004)’s notion of “economy”. In Landau (2006), “economy” means “say as little as possible” which corresponds to the assumption that all chain copies tend to be not pronounced; while in Nunes (2004), “economy” means “use as few operations as possible” which corresponds to the assumption that all chain copies tend to be pronounced (i.e., not pronouncing a chain copy involves a deletion process and thus a derivation is less economical when more copies are not pronounced/deleted).

However, Landau (2006)’s economy principle seems to be problematic when considering ellipsis. To be more specific, if the economy principle (i.e., say as little as possible) is true, it should be predicted that whenever ellipsis is possible, it should be forced. However, ellipsis is an optional process, and thus the economy principle does not seem to be supported.

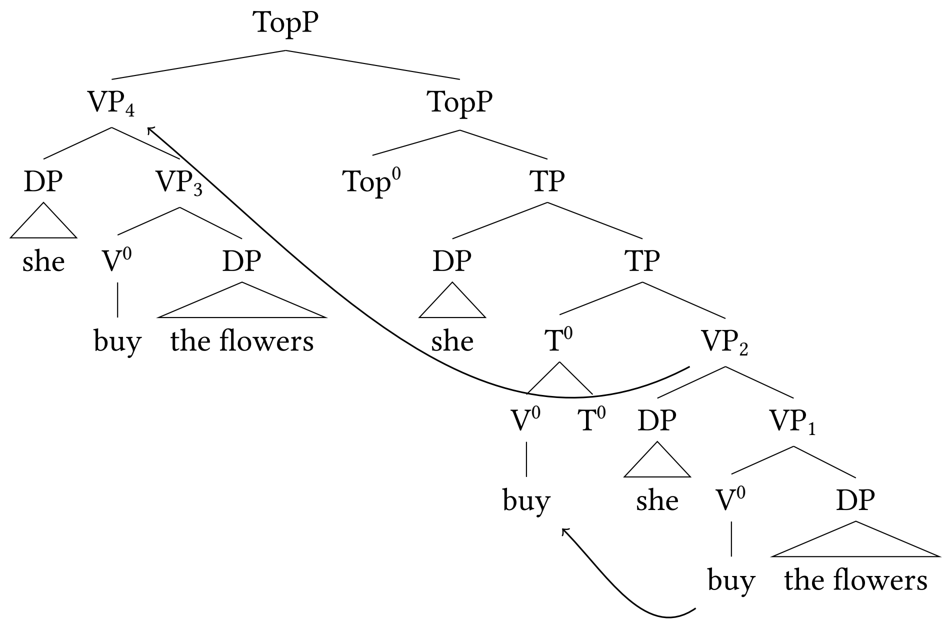
In addition, for the Hebrew example in (39), according to Landau (2006), the highest copy “liknot” has a high pitch accent imposed by Top^0 and the lower copy “kanta” needs to support the tense affix, so they are both associated with intrinsic phonetic content and get pronounced. The structure of Hebrew V-fronting is shown in (40). However, the Mandarin data in (41) poses problems for Landau (2006)’s analysis: though it can be explained why the verb “xihuan” in the topic position is pronounced, it is unclear why the verb “xihuan” downstairs is also pronounced. To be more specific, in the Mandarin example (41), regarding the higher copy, it should be pronounced since it has a high pitch accent imposed by Top^0, which is similar to the Hebrew data in (39); however, according to Landau (2006), the copy in VP should not have been pronounced since it does not have high pitch accent nor need to support an affix.

(39) **Hebrew V-fronting**

liknot,	hi	**kanta**	et	ha-praxim
to-buy,	she	bought	ACC.	the-flowers

'As for buying, she bought the flowers.'

(40)



(41) **Mandarin verb doubling**

xihuan,	ta	dique	hen	**xihuan**	jiaozi
like	3sg.	indeed	very	like	dumplings

'As for liking, she indeed likes dumplings a lot.'

For the same logic, the Mandarin data in (42) cannot be explained either if the repeated VPs are derived by movement. To be more specific, assuming that VP moves to spec-TopP, the whole VP downstairs should not have been pronounced since none of the material in the VP downstairs in (42) has a high pitch accent or needs to support an affix.

(42) **Mandarin repeated Object**

xihuan	**jiaozi**,	ta	shi	hen	**xihuan**	**jiaozi**
like	dumplings	3sg.	shi	very	like	dumplings

'As for very liking, she indeed likes a lot.'

As a quick summary, Landau (2006)'s analysis for the (non-)pronunciation of the copies mainly consists of two principles: Economy of Pronunciation and P-Recoverability.

Regarding the principle of Economy of Pronunciation, Thoms (2010) has argument against it; and for the principle of P-Recoverability, it does not seem to extend to the Mandarin data very easily without further stipulations. In my analysis, I adopt the structures proposed in Landau (2006) but will not employ the analysis.

CHAPTER 4

The basic linearization process

I assume that the basic linearization process contains the Candidates Generator G and the Constraints following Kusmer (2019), in addition to which, I propose the algorithm Set-to-String that turns a set into a string. So, a linearization of p, where p represents a phrase marker, is a set of ordered pairs and a conversion of that set to a string by the Set-to-String algorithm.

4.1 Candidates Generator G and Constraints

In this section, I present the Candidates Generator G and the Constraints, and illustrate how they work by applying them on concrete examples. I start with the Candidates Generator G, which is defined in (1).

(1) **Candidates Generator G**

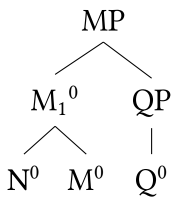
The Candidates Generator G is a function that maps all Maximal X^0 Nodes of a phrase p to all possible sets of ordered pairs, where Maximal X^0 Nodes are X^0s that are not dominated by another Y^0 node.[1]

Take (2) as an abstract example, there are four X^0 nodes in (2): N^0, M^0, ${M_1}^0$ and Q^0. Only ${M_1}^0$ and Q^0 are Maximal X^0 Nodes: both ${M_1}^0$ and Q^0 are X^0 nodes and are not dominated

[1] G(p) does not generate the empty set.

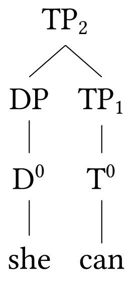
by another β^0 node; while despite that both M^0 and N^0 are X^0 nodes, M^0 and N^0 are dominated by another β^0 node, namely ${M_1}^0$. Thus, only ${M_1}^0$ and Q^0 serve as the input for the Candidates Generator G.

(2)



In the following, I use some concrete examples to show how the Candidates Generator G works. I start with the structure in (3).

(3)



The Maximal X^0 Nodes in (3) are D^0 and T^0. The Candidates Generator G takes D^0 and T^0 as the input and yields all possible sets of ordered pairs, which are shown in (4).[2]

(4) G(n) =

$$\left\{\begin{array}{llll}
\{D^0 < D^0\} & \{D^0 < D^0, T^0 < T^0\} & \{D^0 < D^0, T^0 < T^0, D^0 < T^0\} & \{D^0 < D^0, T^0 < T^0, D^0 < T^0, T^0 < D^0\} \\
\{T^0 < T^0\} & \{D^0 < D^0, D^0 < T^0\} & \{D^0 < D^0, T^0 < T^0, T^0 < D^0\} & \\
\{D^0 < T^0\} & \{D^0 < D^0, T^0 < D^0\} & \{D^0 < D^0, D^0 < T^0, T^0 < D^0\} & \\
\{T^0 < D^0\} & \{T^0 < T^0, D^0 < T^0\} & \{T^0 < T^0, D^0 < T^0, T^0 < D^0\} & \\
 & \{T^0 < T^0, T^0 < D^0\} & & \\
 & \{D^0 < T^0, T^0 < D^0\}, & &
\end{array}\right\}$$

The four columns in (4) include sets that have one ordered pair (e.g., $\{D^0 < D^0\}$), two ordered pairs (e.g., $\{D^0 < D^0, T^0 < T^0\}$), three ordered pairs (e.g., $\{D^0 < D^0, T^0 < T^0,$

[2]In (4), there are 4 distinct singleton sets (i.e., $\{D^0 < D^0\}, \{T^0 < T^0\}, \{D^0 < T^0\}, \{T^0 < D^0\}$), and the rest of the sets must contain at least two ordered pairs from the singleton sets. So, the number of all possible sets of ordered pairs is calculated as: C(4, 1) + C(4, 2) + C(4, 3) + C(4, 4), which is 2^4-1 (i.e., the sum of all possible combinations of n distinct things is 2^n, and 2^n-1 is the sum that excludes the empty set, where n distinct things here correspond to 4 distinct ordered pairs)

$D^0 < T^0$}) and four ordered pairs (e.g., {$D^0 < D^0$, $T^0 < T^0$, $D^0 < T^0$, $T^0 < D^0$}), respectively. Notice that among the sets that are generated by G in (4), only {$D^0 < T^0$} gives the right linearization (i.e., *she can*). The other sets are illegitimate.

Next, I introduce the Constraints, which will eliminate the illegitimate sets of ordered pairs generated by G. Before going into the details of the Constraints, I would like to discuss different types of illegitimate sets generated by G, which should be eliminated by the Constraints. The representative examples for each illegitimate type of sets are shown in (5a) - (5d).

(5) a. {$D^0 < D^0$}

b. {$D^0 < D^0$, $D^0 < T^0$}

c. {$D^0 < T^0$, $T^0 < D^0$}

d. {$T^0 < D^0$}

1. Missing orders. For instance, in (5a), D^0 and T^0 are not ordered.

2. Nodes being ordered relative to themselves. For instance, in (5a) and (5b), D^0 is ordered relative to itself. Since in an utterance, a vocabulary item cannot precede/follow itself, a linearization like the one in (5a) and (5b) will fail to generate an utterance.

3. Paradoxical orders. For instance, in (5c), the orders $T^0 < D^0$ and $D^0 < T^0$ are contradictory. Since in an utterance, a vocabulary item cannot precede/follow another vocabulary item at the same time, a linearization like the one in (5c) will fail to generate an utterance.

4. Wrong orders. For instance, in (5d), there is an ordering between T^0 and D^0, but it is the wrong one. D^0 should have preceded T^0. As a result, the wrong utterance that has T^0 preceding D^0 is generated (i.e., *can she*).

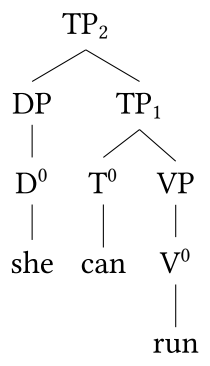
As a quick summary, (5a) has the problems of missing nodes and a node being ordered relative to itself; (5b) has the problem of a node being ordered relative to itself;

(5c) has the problems of bearing contradictory ordering statements and wrong ordering statements; and (5d) has the problem of containing wrong ordering statements.

Regarding the above problems, I propose four corresponding constraints that aim at ruling out the illegitimate sets of ordered pairs generated by the Candidates Generator G before passing the set[3] to the Set-to-String Algorithm.[4] The four constraints are the Totality Constraint, the Anti-reflexivity Constraint, the Asymmetry Constraint, and the Language Specific Constraints. The Totality Constraint and the Asymmetry Constraint are adapted from Kayne (1994), the Anti-reflexivity Constraint is adapted from Partee, ter Meulen and Wall (1990), and the Language Specific Constraints are adapted from Wilder (1999) and Kusmer (2019).[5]

Since some of the illegitimate ordered pairs in (5) have more than one problem, in the following, I switch to the structure in (6), where the problems of missing orders, nodes being ordered to themselves, paradoxical orders and wrong orders can be independently presented.

(6)



In (7), I list four sets generated by G that correspond to the four representative problems. (7a) has the problem of missing orders; (7b) has the problem of having a node ordered

[3]Here, I assume that if there exist sets that satisfy all the constraints there should only be one such set. However, potentially, there could be more than one set that satisfies all the constraints.

[4]Note that later when implementing the basic linearization process in a cyclic way, there will be situations where despite all the illegitimate sets generated by the Candidates Generator G can be ruled out, the union of the legitimate sets violates the Asymmetry Constraint. I will discuss how to deal with such situations when presenting the Set-to-String Algorithm.

[5]Different from Kayne (1994), I do not include "Transitivity" as a condition/constraint. The reason is that at least for cases discussed in this dissertation, having "Transitivity" as a constraint does not add any new information for forming a string. I will discuss this in more details with concrete examples in chapter 6.

to itself; (7c) has the problem of having paradoxical orders; and (7d) has the problem of having wrong orders.

(7) a. $\{D^0 < T^0\}$

b. $\{\mathbf{D^0} < \mathbf{D^0}, D^0 < T^0, D^0 < V^0, T^0 < V^0\}$

c. $\{\mathbf{T^0} < \mathbf{D^0}, \mathbf{D^0} < \mathbf{T^0}, D^0 < V^0, T^0 < V^0\}$

d. $\{\mathbf{T^0} < \mathbf{D^0}, D^0 < V^0, T^0 < V^0\}$

I propose the Totality Constraint to solve the problem in (7a), the Anti-reflexivity Constraint to solve the problem in (7b), the Asymmetry Constraint to solve the problem in (7c), and the Language Specific Constraints to solve the problem in (7d). I first discuss the Totality Constraint, which is defined in (8).

(8) **Totality Constraint**

For a given set of ordering statements x, x satisfies the Totality Constraint iff. for all Maximal X^0 Nodes α and β of phrase p, x contains an ordering of α and β.

In (6), the Maximal X^0 Nodes of the phrase TP_2 are shown in (9). These Maximal X^0 Nodes will serve as the input for the Candidates Generator G, and the Totality Constraint will be evaluated against them.

(9) Maximal X^0 Nodes of TP_2 = $\{D^0, T^0, V^0\}$

The Maximal X^0 Nodes in (9) can form three pairs, which are shown in (10). Based on the Totality Constraint, the legitimate set of ordered pairs should contain an ordering statement for each pair in (10).

(10) a. D^0 and T^0

b. D^0 and V^0

c. T^0 and V^0

Notice that (7a) only has the ordering statement of D^0 and T^0 and misses the ordering statements of D^0 and V^0 as well as T^0 and V^0. Thus, (7a) violates the Totality Constraint and should be ruled out. In contrast, (7b) - (7d) contain ordering statements for each pair in (10), and thus, they all satisfy the Totality Constraint.

Next, I present the Anti-reflexivity Constraint in (11).

(11) **Anti-reflexivity Constraint**

For a given set of ordering statements x, x satisfies the Anti-reflexivity Constraint iff. for all orderings of α and β in x, α and β are distinct nodes. Distinctiveness is defined in (11a).

a. **Distinctiveness**

α and β are distinct nodes iff. α does not self-dominate β.

Note that (7b) contains the ordering statement $D^0 < D^0$, where D^0 and D^0 are not distinct nodes (D^0 self-dominates D^0). Thus, (7b) violates the Anti-reflexivity Constraint and should be ruled out. In contrast, in (7a), (7c) and (7d), all the ordering statements have a pair of distinct nodes, so they all satisfy the Anti-reflexivity Constraint.

Now, let's look at the Asymmetry Constraint, which is defined in (12).

(12) **Asymmetry Constraint**

For a given set of ordering statements x, x satisfies the Asymmetry Constraint iff x does not include both $\alpha < \beta$ and $\beta < \alpha$.

(7a), (7b) and (7d) all satisfy the Asymmetry Constraint since none of them contains paradoxical orders. However, (7c) violates the Asymmetry Constraint since it has both $T^0 < D^0$ and $D^0 < T^0$.

Lastly, I discuss the Language Specific Constraints, which are stated in (13).[6]

[6]The constraints in (13a) and (13b) are for languages that are discussed in this dissertation (i.e., Hebrew, English, Mandarin and Yiddish — they are all head-initial languages). In addition, in this dissertation, I only consider cases where adjuncts are left adjoined, so the adjunct constraint in (13c) is only for cases with left-adjoined adjuncts. For other languages, for instance, Japanese (a head-final language), in order for the Language Specific Constraints to work, details need to be changed. In other words, the Language Specific Constraints in this dissertation are not designed for typology, they are a particular instantiation of some

(13) **Language Specific Constraints**

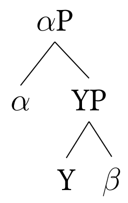
For a given set of ordering statements x, x satisfies the Language Specific Constraints, iff x satisfies the constraints in a-c.

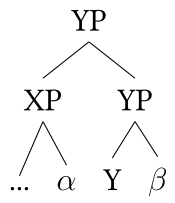
a. If x contains $\alpha < \beta$ or $\beta < \alpha$, and α is a head, there must be an ordering statement $\alpha < \beta$ if β is fully dominated by YP, where YP is the sister of α (see 1).

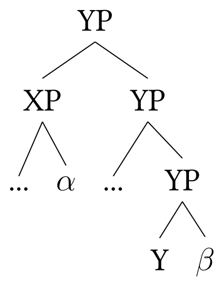
b. If x contains $\alpha < \beta$ or $\beta < \alpha$, there must be an ordering statement $\alpha < \beta$ if α is dominated by a specifier XP and β is fully dominated by YP, where YP is XP's sister (see 2).

c. If x contains $\alpha < \beta$ or $\beta < \alpha$, there must be an ordering statement $\alpha < \beta$, if α is dominated by an adjunct XP and β is fully dominated by YP, where YP is XP's sister (see 3).

(14)

a. 

b. 

c. 

Here, "*full dominance*", "*path*" and "*sister*" are defined in (15a), (15a-i) and (15b), respectively.

(15) a. **Full dominance**

α fully dominates β if every path for α includes β.

specific languages.

(i) ***Path***

A path for α is a series of nodes such that the first node immediate dominates α, the other nodes immediately dominate the preceding nodes, and the last node is the root node.

b. **Sisterhood**

α is a sister of β if the immediately dominating mother of α and β fully dominates both of them.

Before I use a concrete example to show the details about how Language Specific Constraints work, recall that Language Specific Constraints are designed to rule out illegitimate sets of precedence relations that do not contain the right orders (cf. (5d) *She can.* *{$T^0 < D^0$}. To be more specific, Language Specific Constraints make sure that if two nodes X^0 and Y^0 are ordered, the language particular order about them must be contained in the set of precedence relations. Note that the condition "if two nodes X^0 and Y^0 are ordered" refers to three possible scenarios, one where the set contains $X^0 < Y^0$ (e.g., $D^0 < T^0$ in (17a)), one where the set contains $Y^0 < X^0$ (e.g., $T^0 < D^0$ in (17b)), and one where the set contains both $X^0 < Y^0$ and $Y^0 < X^0$ (e.g., {$T^0 < D^0$, $D^0 < T^0$} in (17c)) (in all three cases (17a) - (17c), they have ordering statements regarding D^0 andT^0). Then, Language Specific Constraints, (13b) in particular, require that the right order between D^0 and T^0 (i.e., $D^0 < T^0$) must be in the set. In this sense, both (17a) and (17c) satisfy the Language Specific Constraints because both of them contain the right order $D^0 < T^0$ (D^0 — the node in the specifier DP, precedes T^0 — the node fully dominated by the sister of the DP specifier). It can be seen that Language Specific Constraints require that the right order be in the set, and whether the wrong one is also there is not a concern (it is a concern for the Asymmetry Constraint, which means that (17c) will be ruled out by the Asymmetry Constraint).

(16) *she can*

(17) Some possible candidates generated by G

a. $\{D^0 < T^0\}$ — satisfy the Language Specific Constraints

b. $\{T^0 < D^0\}$ — violate the Language Specific Constraints

c. $\{T^0 < D^0, D^0 < T^0\}$ — satisfy the Language Specific Constraints

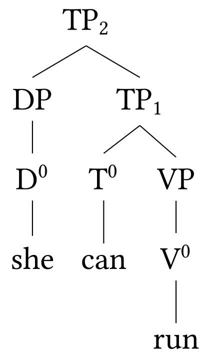
Now, think about how all the constrains work together. Considering the possible candidates in (17) as well as the one in (18), the Totality Constraint makes sure that all the Maximal X^0 nodes are ordered, but it does not care whether the set has an ordering statement(s) that orders the same node (e.g., (18) satisfies the Totality Constraint but it orders D^0 relative to itself), nor does the Totality Constraint care whether every statement orders the two nodes right (e.g., (17a)-(17c) all satisfy the Totality Constraint despite that (17b) and (17c) both have wrong orders $T^0 < D^0$). So, the Anti-reflexivity Constraint comes in to make sure that each statement in the set orders different nodes (e.g., (18) is ruled out and only (17a)-(17c) satisfy the Anti-reflexivity Constraint). Meanwhile, the Language Specific Constraints and the Asymmetry Constraint work together to make sure only the right orders are in the set — the Language Specific Constraints make sure that for the two nodes in each ordered pair, the set must have the right order between them (e.g., (17b) is ruled out and only (17a) and (17c) satisfy the Language Specific Constraints) and the Asymmetry Constraint rules out the set that has both the right order and the wrong order (e.g., (17c) is then ruled out and only (17a) satisfies the Asymmetry Constraint).

(18) $\{D^0 < T^0, D^0 < D^0\}$

Having discussed some intuition behind how Language Specific Constraints work and how all the constraints work together, in the following, I will go into more technical details of the Language Specific Constraints — I will first illustrate the key definitions in (15) using the concrete example in (6), which is repeated below as (19), and then show how

Language Specific Constraints work for this example. Note that in this example, all the dominance relationships are also full-dominance relationships. In the next chapter, I will discuss an example, where dominance and full-dominance can be distinguished and this distinction is crucial.

(19)



(20) summarizes the full-dominance relationship in (19).

(20) a. TP_2 fully dominates DP, D^0, TP_1, T^0, VP, V^0

b. DP fully dominates D^0

c. TP_1 fully dominates T^0, VP, V^0

d. VP fully dominates V^0

To determine the full-dominance relationship between α and β, one needs to check whether every path for α includes β. Take TP_1 and V^0 as an example. The series of nodes that form the path for V^0 is shown in (21).

(21) a. $p(V^0) = (VP, TP_1, TP_2)$

In this path, VP is the node that immediately dominates V^0, and TP_1 immediately dominates VP, and the last node TP_2 immediately dominates TP_1, and TP_2 is the root node. Since this is the only path for V^0 and this path includes TP_1, TP_1 fully dominates V^0 (i.e., every path for V^0 includes TP_1).[7] The paths for each node in (19) is shown in (22).

[7]Similarly, VP and TP_2 also fully dominates V^0.

(22) a. $p(D^0) = (DP, TP_2)$

b. $p(T^0) = (TP_1, TP_2)$

c. $p(V^0) = (VP, TP_1, TP_2)$

d. $p(DP) = (TP_2)$

e. $p(TP_1) = (TP_2)$

f. $p(VP) = (TP_1, TP_2)$

Now, I turn to sisterhood. Below, Table 4.1 is a list of sisterhood relationship in (19). "IDM" is short for immediate dominating mother.

		IDM	**relationship**
(DP	TP_1)	TP_2	sisters
(T^0	VP)	TP_1	sisters

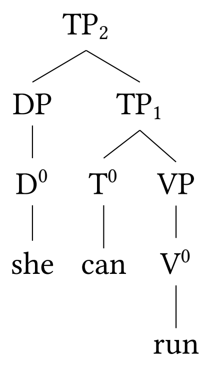
Table 4.1: TP_2 relation

To determine the sisterhood relationship between α and β, one needs to see whether the immediate dominating mother of α and β fully dominates both α and β. Take DP and TP_1 as an example. They are sisters because their immediate dominating mother TP_2 fully dominates both DP and TP_1 (see (20) for the list of full-dominance relationship in (19)). The reason TP_2 fully dominates DP and TP_1 is that every path for DP and TP_1 includes TP_2 (see (22) for the list of paths).

Next, I illustrate how the Language Specific Constraints work for (19) (since there is no adjunct in this example, I will only show how the constraints regarding heads and specifiers work). I start with the Language Specific Constraints for heads. Take the set in (7d) that violates the Language Specific Constraints as an example, which is shown in (23). The structure of this case is repeated below in (24).

(23) $\{T^0 < D^0, D^0 < V^0, T^0 < V^0\}$

(24)



I start with the third ordering statement $T^0 < V^0$. Based on the first constraint regarding head in (13a) in Language Specific Constraints, if a set contains $\alpha < \beta$ or $\beta < \alpha$, and α is a head, there must be an ordering statement $\alpha < \beta$ if β is fully dominated by YP, where YP is the sister of α. In this case, (i) the set contains $T^0 < V^0$, (ii) T^0 is a head, (iii) T^0 and VP are sisters, and (iv) VP fully dominates V^0. Thus, there must be an ordering statement $T^0 < V^0$ in the set. So, so far, nothing in this set has violated the Language Specific Constraints.

Now, I examine the second ordering statement $D^0 < V^0$. Based on the second constraint regarding specifiers in (13b) in Language Specific Constraints, if a set contains $\alpha < \beta$ or $\beta < \alpha$, there must be an ordering statement $\alpha < \beta$ if α is dominated by a specifier XP and β is fully dominated by YP, where YP is XP's sister. In this case, (i) the set contains $D^0 < V^0$, (ii) DP is a specifier, (iii) D^0 is dominated by DP[8], (iv) DP and TP_1 are sisters, and (v) TP_1 fully dominates V^0. Thus, there must be an ordering statement $D^0 < V^0$ in the set. So, still, nothing in this set has violated the Language Specific Constraints yet.

Lastly, I examine the first ordering statement $T^0 < D^0$. Similar to the second ordering statement, based on the second constraint regarding specifiers in (13b) in Language Specific Constraints, (i) the set contains $T^0 < D^0$, (ii) DP is a specifier, (iii) D^0 is dominated by DP, (iv) DP and TP_1 are sisters, and (v) TP_1 fully dominates T^0. Thus, there must be an ordering statement $D^0 < T^0$ in the set. However, there is no such ordering statement. Thus, the set in (23) violates the Language Specific Constraints. Note that having the ordering statement $T^0 < D^0$ in the set does not violate the Language Specific Constraints. It is the

[8]Note that DP is only required to dominate D^0, not fully dominate - α can dominate β without fully dominating β, but if α fully dominates β, α must dominate β; fully dominate is a special case of dominate.

lack of $D^0 < T^0$ due to the existence of $T^0 < D^0$ that makes the set violate the Language Specific Constraints. In other words, the set in (7c), repeated below in (25), satisfies the Language Specific Constraints (though it violates the Asymmetry Constraint). Note that the set in (25) is the same as the one in (23) except that (25) has an extra ordering statement $D^0 < T^0$. (25) satisfies the Language Specific Constraints because the existence of $T^0 < D^0$ requires the existence of the ordering statement $D^0 < T^0$, which the set has; the existence of $D^0 < T^0$, $D^0 < V^0$ and $T^0 < V^0$ requires the existence of the ordering statement $D^0 < T^0$, $D^0 < V^0$ and $T^0 < V^0$, which the set has.

(25) $\{T^0 < D^0, \underline{D^0 < T^0}, D^0 < V^0, T^0 < V^0\}$

Table 4.2 summarizes all the ordering statements required by the Language Specific Constraints for all Maximal X^0 Nodes α and β of TP_2 if α and β are ordered as a pair in a given set.

		IDM	**relationship**	**orders**
(T^0	VP)	TP_1	sisters	$T^0 < V^0$
(DP	TP_1)	TP_2	sisters	$D^0 < T^0$, $D^0 < V^0$

Table 4.2: TP_2 orders

So far, I have illustrated how all four constraints work. I repeat the illegitimate sets below in (26) and here is a quick summary. Though (26a) satisfies the Anti-reflexivity Constraint, the Asymmetry Constraint, and the Language Specific Constraints, it violates the Totality Constraint since the ordering statements of D^0 and V^0, as well as T^0 and V^0 are missing. Though (26b) satisfies the Totality Constraint, the Asymmetry Constraint, and the Language Specific Constraints, it violates the Anti-reflexivity Constraint by having D^0 ordered to itself. Though (26c) satisfies the Totality Constraint, Anti-reflexivity Constraint and the Language Specific Constraints, it violates the Asymmetry Constraint by having paradoxical ordered pairs $T^0 < D^0$ and $D^0 < T^0$. Though (26d) satisfies the Totality

Constraint, the Anti-reflexivity Constraint and the Asymmetry Constraint, it violates the Language Specific Constraints by not having $D^0 < T^0$. Thus, (26a) - (26d) should all be ruled out. The only set that satisfies all the constraints is $\{D^0 < T^0, D^0 < V^0, T^0 < V^0\}$.

(26) a. $\{D^0 < T^0\}$

b. $\{\mathbf{D^0} < \mathbf{D^0}, D^0 < T^0, D^0 < V^0, T^0 < V^0\}$

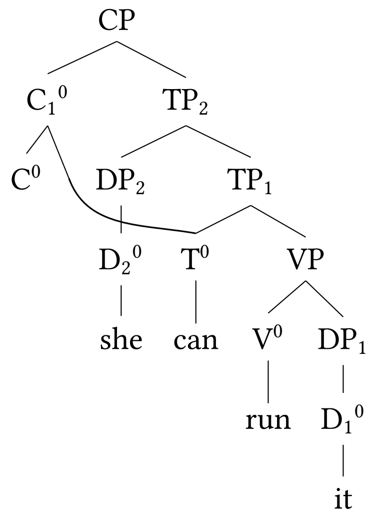
c. $\{\mathbf{T^0} < \mathbf{D^0}, \mathbf{D^0} < \mathbf{T^0}, D^0 < V^0, T^0 < V^0\}$

d. $\{\mathbf{T^0} < \mathbf{D^0}, D^0 < V^0, T^0 < V^0\}$

In the following, I illustrate how the Candidates Generator G and the Constraints work for cases that involve movement by using the example in (27), the structure of which is shown in (28).

(27) *Can she run it?*

(28)



In (28), the Maximal X^0 Nodes are shown in (29). Note that T^0 is not a Maximal X^0 Node because T^0 is dominated by another node, namely $C_1{}^0$.

(29) Maximal Nodes of VP = $\{C_1{}^0, D_2{}^0, V^0, D_1{}^0\}$

The Candidates Generator G takes the nodes in (29) and generates all possible sets of ordered pairs. There is only one set that satisfies all the constraints, which is shown in (30).

(30) $$\begin{Bmatrix} {C_1}^0 < {D_2}^0 & {D_2}^0 < V^0 & V^0 < {D_1}^0 \\ {C_1}^0 < V^0 & {D_2}^0 < {D_1}^0 & \\ {C_1}^0 < {D_1}^0 & & \end{Bmatrix}$$

This set satisfies Totality Constraint because all the Maximal X^0 Nodes in (29) are ordered relative to each other. It satisfies the Anti-reflexivity Constraint because no node is ordered relative to itself. It satisfies the Asymmetry Constraint because there are no contradictory orderings. To see why it satisfies the Language Specific Constraints, I examine each ordering statement in this set.

Before going into the details of how Language Specific Constraints are evaluated, I would like to point out that similar to the previous example, all the full-dominance relationships discussed in this example, which are listed later in (31), are also dominance relationships. Note that in the next chapter, I will discuss an example where the distinction between dominance and full-dominance matters.

Since the set has the ordering statement ${C_1}^0 < {D_2}^0$, it requires ${C_1}^0 < {D_2}^0$ to be in the set: based on the Language Specific Constraints regarding heads, since (i) the set has ${C_1}^0 < {D_2}^0$, (ii) ${C_1}^0$ is a head, (iii) TP_2 is the sister of ${C_1}^0$, (iv) TP_2 fully dominates ${D_2}^0$, ${C_1}^0 < {D_2}^0$ is required. Similarly, ${C_1}^0 < V^0$, ${C_1}^0 < {D_1}^0$ and $V^0 < {D_1}^0$ are also required. Note that in this case, since T^0 is not a Maximal X^0 Node, there is no need to determine the relationship between T^0 and the other Maximal X^0 Nodes.[9] In addition, since the set has the ordering statement ${D_2}^0 < V^0$, it requires ${D_2}^0 < V^0$ to be in the set: based on the Language Specific Constraints regarding specifiers, since (i) the set has ${D_2}^0 < V^0$, (ii) DP_2 is a specifier, (iii) ${D_2}^0$ is dominated by DP_2, (iv) TP_1 is the sister of DP_2, and also (v) TP_1 fully dominates V^0, ${D_2}^0 < V^0$ is required. Similarly, ${D_2}^0 < {D_1}^0$ is also required.

Table 4.3 summarizes the relevant structural relationship in CP, and the required

[9]If T^0 were a Maximal X^0 Node, TP_2 dominates but does not fully dominate it because there is a path for T^0 that does not go through TP_2.

orders by the Language Specific Constraints for all Maximal X^0 Nodes α and β of CP if α and β are ordered as a pair in a given set.

		IDM	**relationship**	**ordered pairs**
($C_1{}^0$	TP_2)	CP	sisters	$C_1{}^0 < D_2{}^0$, $C_1{}^0 < V^0$, $C_1{}^0 < D_1{}^0$
(DP_2	TP_1)	TP_2	sisters	$D_2{}^0 < V^0$, $D_2{}^0 < D_1{}^0$
(V^0	DP_1)	VP	sisters	$V^0 < D_1{}^0$

Table 4.3: CP relation

The relevant path and full-dominance relationship are summarized in (31) and (32). For instance, $C_1{}^0$ and TP_2 are sisters because their immediate dominating mother CP fully dominates $C_1{}^0$ and TP_2. The reason CP fully dominates $C_1{}^0$ and TP_2 is that every path for $C_1{}^0$ and every path for TP_2 includes CP, which is shown in (32a) and (32b) respectively.

(31) a. CP fully dominates $C_1{}^0$ and TP_2

b. TP_2 fully dominates DP_2, TP_1, $D_2{}^0$, V^0, $D_1{}^0$

c. DP_2 fully dominates $D_2{}^0$

d. TP_1 fully dominates V^0, $D_1{}^0$

e. VP fully dominates V^0, DP_1

f. DP_1 fully dominates $D_1{}^0$

(32) a. $p(C_1{}^0) = (CP)$

b. $p(TP_2) = (CP)$

c. $p(DP_2) = (TP_2, CP)$

d. $p(TP_1) = (TP_2, CP)$

e. $p(V^0) = (VP, TP_1, TP_2, CP)$

f. $p(DP_1) = (VP, TP_1, TP_2, CP)$

g. $p(D_1{}^0) = (DP_1, VP, TP_1, TP_2, CP)$

Note that so far, for both the non-movement case (i.e., *She can run*) in (24) and the movement case (i.e., *Can she run it*) in (28), all the relevant nodes only have one path (see (22) and (32))[10]. In the next chapter, I will discuss cases where for some nodes, there is more than one path.

In this section, I introduced the Candidates Generator G and the Constraints, and showed how they work on concrete examples. In the next section, I show how to turn the surviving set of ordered pairs into a string by implementing the Set-to-String algorithm.

4.2 The Set-to-String Algorithm

In the previous section, for the non-movement case (i.e., *She can run*) in (24), after imposing the Constraints on the sets generated by the Candidates Generator G, the illegitimate sets of ordered pairs are ruled out and one gets the set of ordered pairs that satisfies all the constraints, which is shown in (33). In this section, I show how this set becomes a string via the Set-to-String algorithm.

(33) a. $\begin{Bmatrix} D^0 < T^0 & T^0 < V^0 \\ D^0 < V^0 & \end{Bmatrix}$

The Set-to-String algorithm contains four stages, and I state and discuss each stage in (34). Note that in the case of (33), the Set-to-String algorithm will only proceed to Stage 1 because this set does not violate the Asymmetry Constraint. Later, in chapter 5, I discuss a case that has non successive-cyclic movement, of which the final union set violates the Asymmetry Constraint and thus proceeds to Stage 2, but since it fails to form a string, it causes the linearization process to crash and does not proceed to Stage 3 and 4; and in chapter 6, I discuss the VP-doubling case in Mandarin, of which the final union set also violates the Asymmetry Constraint and proceeds to Stage 2 but succeeds in forming a

[10]If T^0 were to be considered in the movement case, it has two paths: (TP_1, TP_2, CP) and (${C_1}^0$, CP)

string, and thus proceeds to Stage 3 and 4.[11]

(34) **Set-to-String**

The algorithm first examines whether a set satisfies the Asymmetry Constraint by checking whether the set has ordering contradiction. In other words, the algorithm needs to check each node and see whether there exists a node that both precedes and follows the same node. One way to implement this is stated in Stage 1, where for each node n, form a set X that has all the nodes that precede n and a set that has all the nodes that follow n. Then, take the intersection set of X and Y. If for instance, there exists an ordering contradiction, where a node precedes and follows the same node n_1, n_1 will be in both set X and set Y, so the intersection set of X and Y will include n_1 and not be empty. If all the intersection sets are empty, it means that there is no contradiction (i.e., there does not exist a node that precedes and follows the same node(s)) and the set can be linearized directly by the String Forming Mechanism. If there exists an ordering contradiction, proceed to the next stage.
Stage 1:

1. **Find the intersection sets:** For each distinct node n_1, n_2, n_3, ... in the set of ordered pairs O = $\{n_1 < n_3, n_3 < n_2, ...\}$, get the set X = $\{..., n_1, ...\}$ that contains all the nodes that precedes n and the set Y = $\{..., n_2, ...\}$ that contains all the nodes that follows n, and get the set $I = X \cap Y$, which is the intersection of set X and set Y.

2. **Check the intersection sets:** If all the intersection sets are empty, run the String Forming Mechanism, which is defined in (35) below, on set O and the set will be turned into a string. **Otherwise, proceed to Stage 2.**

I propose that if a set only violates the Asymmetry Constraint (i.e., having non-empty intersection sets), the linearization does not crash immediately.

[11] Note that a final union set either satisfies all the constraints or only violates the Asymmetry Constraint. One situation where a final union set only violates the Asymmetry Constraint is that the precedence relations between a moved item and (some of) the other nodes in the Spell-out domain XP are changed in the next Spell-out domain YP, which causes ordering contradiction after applying the union operation on XP and YP.

Instead, check whether it is possible to linearize the moved item in its higher position as a string but linearize the nodes in the moved item in its lower position.[12] I further propose that a moved item can only be linearized as a string if the nodes in a moved item can form a qualified constituent. I propose that whether a constituent is qualified depends on the particular language. For instance, in English, no constituent is qualified; and in Mandarin, [$_{VP}$ V bare-NP] is qualified (regarding whether the other kinds of VP structures are qualified, it varies among speakers).[13] Thus, there are two tasks the next stage of the Set-to-String algorithm should accomplish: one is to find all the nodes in a moved item, and the other is to check whether those nodes form a qualified constituent. A more concrete example will be discussed in chapter 5.

Before going into the details of Stage 2, I provide a brief discussion about how the first task of finding all the nodes in a moved item is done in a formal way in Stage 2. Imagine the following situation: x is a moved item, and y < x in the XP Spell-out domain but x < y in the next Spell-out domain YP. Assuming that the final union set includes both y < x and x < y, the intersection set for x will include y (i.e., x both precedes and follows y), and relatively the intersection set for y will include x. In this sense, the intersection set of y includes the moved item x and the intersection set of x includes the node y that is ordered differently to x in XP and YP. Since the nodes in the moved item normally form a constituent, but the nodes that are ordered differently regarding the moved item do not[14], I use this as a way for the algorithm to locate the intersection set that contains the nodes in a moved item.[15] Now that the nodes in a moved item are found, the next step is to see whether the nodes form a qualified constituent. If they do, proceed to the next stage (i.e.,

[12]Here, I stipulate that the nodes in the moved item in its higher position should be the candidate for forming a string. There could possibly be a more general way to choose which nodes to form a string but for now, I stick to this less general approach and discuss some alternative approaches later.

[13]I will revisit qualified constituents in Mandarin VP-doubling cases in chapter 6.

[14]At least in all the examples in this dissertation, the nodes that are ordered differently regarding the moved item do not form a constituent.

[15]The property of being able to form a constituent is just a general observation about the nodes in a moved item, which differs from the nodes that are ordered differently from the moved item. This property does not provide any explanation for why the nodes in the moved item should be chosen but only serves as a way for the algorithm to find all the nodes in the moved item in a formal way.

Stage 3); otherwise, the linearization process crashes.

Stage 2:

1. **Collect all the non-empty intersection sets:** Form a set M = {I_1, I_2,...} that contains all the non-empty intersection sets.

2. **Try forming a string:**

   (a) **No qualified constituent:** For all given intersection sets I_1 = {n_1, n_2, ...}, I_1 = {n_3, ...}, ... in set M, if none of them forms a qualified constituent, the linearization process crashes.

   (b) **Qualified constituent:** If there exists intersection set I^* that forms a qualified constituent, **proceed to Stage 3.**

If the nodes form a constituent, the next step is to form a string out of the nodes. The formal way to implement this is to form a set S that contains all the ordered pairs that order the nodes that form a qualified constituent in the intersection set I^* (i.e., the nodes that are in the moved item), and run the String Forming Mechanism on set S. This is stated in Stage 3.

Stage 3:

1. **Form set S:** Form the subset S = {$n_1 < n_2$, ...} of set O that contains all the ordered pairs $\alpha < \beta$, where both α and β are in the intersection set I^*.

2. **Forming a string:** Run the String Forming Mechanism algorithm, which is stated in (35), on set S and get the string str = $<_S n_1 n_2 ... >$.

After forming the string, the next step is to replace the nodes in the moved item in its higher position with the string. Note that there are ordering statements that order the nodes within the moved item and they should not be replaced in those statements, or the string will be ordered relative to itself. For instance, x and y are the nodes in a moved item and the ordering statements include x $<$ y. If x and y are replaced by the string $<_S$ xy $>$, there will be an ordering statement of $<_S$ xy $>$ preceding $<_S$ xy $>$, which violates the Anti-reflexivity Constraint. In other words, replacement should only be applied to statements that are not the ones like x $<$ y, where both x and y are nodes in the moved item. The way to implement this idea is to find a set D, which excludes ordering statements like x $<$ y from the original set O, and only apply replacement to the ordering statements in the original set O if they are in set D. Note that previously, the algorithm has formed set S, which contains all the ordering statements that order the nodes in the moved item, so the algorithm can get set D by subtracting set S from the original set O. To implement the idea of only forming a string in the higher position of the moved item, replacement only happens to the α position of $\alpha < \beta$.

Before showing the formal details in Stage 4, I provide a brief illustration of how replacement can solve the asymmetric problem. For instance, assuming that the original set is {**A** $<$ C, **B** $<$ C, C $<$ A, C $<$ B, A $<$ B}, A and B are the nodes in a moved item, and A and B form a qualified constituent. So, A and B form the string $<_S$ AB$>$. Then, replace A and B with the string when A and B are in their higher position (i.e., A and B are the preceding nodes in the ordering statements). As a result, the original set becomes {$\underline{<_S \text{AB} >}$ $<$ C, C $<$ A, C $<$ B, A $<$ B} and there is no contradictory ordering. A more concrete example will be discussed in chapter 6.

Stage 4:

1. **Replace nodes with string:**

   (a) **Get set D = O - S** Form set D = O - S (i.e., D = {$n_1 < n_3$, $n_3 < n_2$ ...}, O = {$n_1 < n_3$, $n_3 < n_2$, $n_1 < n_2$, ...} and S = {$n_1 < n_2$})

   (b) **Replace** For every ordering statement $\alpha < \beta$ in the original set O, if the ordering statement is in set D, replace the node with the string str = $<_S n_1 n_2 ...>$ if the node is in intersection set I^* and in the α position.

The updated set O = {$\underline{<_S n_1 n_2 ... >}$ $< n_3$, $n_3 < n_2$, $n_1 < n_2$, ...}.

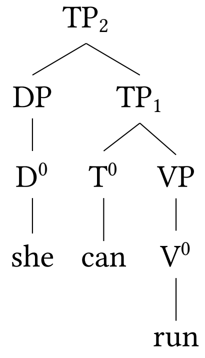
2. **Linearizing O:** Run the String Forming Mechanism algorithm in (35) on the updated set O.

(35) **String Forming Mechanism**

For a given set of ordered pairs x, it is associated with a string str that is by default empty. Concatenate node/string α to string str if the node/string α is not preceded by any other node, and then delete all the ordered pairs that contain α. Repeat this procedure unless there is only one ordered pair $\gamma < \delta$ left in set x, then, first concatenate γ and then concatenate δ to string str.

In the following, I illustrate how the Set-to-String algorithm works using the set in (33), which is repeated below in (36). The structure in question is repeated below in (37).

(36)
$$\begin{Bmatrix} D^0 < T^0 & T^0 < V^0 \\ D^0 < V^0 & \end{Bmatrix}$$

(37)


First, implement Stage 1 of the Set-to-String algorithm, where the algorithm determines whether the final union set (36), which I will refer to as Set O, violates the Asymmetry Constraint. Specifically, find the intersection sets for all the nodes (i.e., D^0, T^0 and V^0) in O. For each node, take the intersection of the set that has the preceding nodes and the set that contains the following nodes. The intersection sets are shown in (38).

(38) a. D^0: $\varnothing = \varnothing \cap \{T^0, V^0\}$

b. T^0: $\varnothing = \{D^0\} \cap \{V^0\}$

c. V^0: $\varnothing = \{D^0, T^0\} \cap \varnothing$

Note that since all the intersection sets are empty, the algorithm will not proceed to Stage 2, but run the String Forming Mechanism algorithm on set O, which yields the string $<_S D^0T^0V^0>$. Here is how the string is formed by the String Forming Mechanism algorithm. First, find the node that is not preceded by any other nodes (i.e., the node is D^0) and concatenate this node (i.e., D^0) to the default empty string str. Then, delete all the ordering statements that contain D^0. The next step is supposed to repeat this procedure (i.e., find the next node that is not preceded by any other nodes, concatenate this node to the string (i.e., $<_S D^0 >$), and delete all the statements that contain the node D^0). However, since there is only one ordering statement (i.e., $T^0 < V^0$) left, the procedure will not be repeated but T^0 will be concatenated to string str $<_S D^0 >$ and then V^0 will be concatenated to string str $<_S D^0T^0 >$. As a result, the final string is $<_S D^0T^0V^0 >$.

Now, it comes to the last step of the basic linearization algorithm: insert vocabulary items. First, for each node, form the lexical insertion site, which is shown in (39). A $\sim$ B represents that The lexical insertion site for A is B. Then, the string becomes $<_S \#D^0\#\ \#T^0\#\ \#V^0\#>$. The lexical insertion sites might seem to be redundant for now but it will become more obvious why the lexical insertion sites are necessary later for cases that involve complex verbal morphology.

(39) a. $D^0 \sim \#D^0\#$

b. $T^0 \sim \#T^0\#$

c. $V^0 \sim \#V^0\#$

Then, insert lexical items in the lexical insertion site, which is shown in (40). A $=$ B represents that the lexical insertion for A is B. As a result, the string $<_S \#D^0\#\ \#T^0\#\ \#V^0\#>$ is updated as $<_S$ she can run$>$. Finally, we get *she can run* as the utterance.

(40) a. $\#D^0\# = $ *she*

b. $\#T^0\# = $ *can*

c. $\#V^0\# = $ *run*

4.3 The different stages of the Set-to-String algorithm

In the previous section, I presented the four stages of the Set-to-String algorithm. As a quick summary, Stage 1 is for checking whether a certain final union set of the ordering statements violate the Asymmetry Constraint, and it only proceeds to Stage 2 if the final union set violates the Asymmetry Constraint. For instance, like the previous case in (36), the final union set does not have contradictory orderings, and thus satisfies the Asymmetry Constraint, so the linearization process only implements Stage 1 and does not proceed to Stage 2. Later on, in chapter 6, when verb-doubling is discussed, I will show that the linearization of the V-doubling cases also stops at Stage 1 since the final union set satisfies all the constraints, including the Asymmetry Constraint. A preview for the V-doubling example is shown in (41), where the final union set in (41b) satisfies all the constraints. So, for both simple cases like (36) or the V-doubling cases like (41), the linearization process only proceeds to Stage 1 since the final union set satisfies all the constraints, and the sets are mapped to a string by the String Forming Mechanism, which eventually becomes the output of PF after lexical insertion. In addition, Stage 1 only deals with “words”, namely, X^0-level nodes. This can be seen in both the final union set of the previous case in (36) and the final union set of the V-doubling case in (41b), where the ordering statements only contain X^0 nodes.

(41) a. **Hebrew**

Li**kn**o**t**,	hi	**k**a**nt**a	et	ha-praxim
buy.INF	she	**buy**.PST	ACC	the-flowers

'As for buying, she bought the flowers.'

b. **The final union set**

$$\left\{\begin{array}{llllll} Top_1{}^0 < C^0 & C^0 < D_1{}^0 & D_2{}^0 < T_1{}^0 & T_1{}^0 < D_1{}^0 & D_1{}^0 < N^0 & \mathbf{V^0 < D_2{}^0} \\ Top_1{}^0 < D_2{}^0 & C^0 < T_1{}^0 & D_2{}^0 < D_1{}^0 & T_1{}^0 < N^0 & & \mathbf{V^0 < N^0} \\ Top_1{}^0 < T_1{}^0 & C^0 < D_2{}^0 & D_2{}^0 < N^0 & & & \\ Top_1{}^0 < D_1{}^0 & C^0 < N^0 & & & & \\ Top_1{}^0 < N^0 & & & & & \end{array}\right\}$$

However, once a final union set violates the Asymmetry Constraint, the linearization process proceeds to Stage 2, which is the stage that starts involving strings in the linearization. To be more specific, Stage 2 aims at checking whether there is a qualified constituent in the structure that can enter the linearization process as a string, and it only proceeds to Stage 3 and 4 if there is a qualified constituent. For instance, in chapter 5, I will discuss the non successive-cyclic movement of the wh-phrase in English. A preview for this example is shown in (42). In this example, the final union set (42c) violates the Asymmetry Constraint since the final union set contains ordering statements like $P^0 < D_2{}^0$ and $D_2{}^0 < P^0$, and thus, it proceeds to Stage 2.

(42) a. *To whom will she give it?*

b. **Non successive-cyclic movement**

*[$_{CP}$ **To whom** will she [$_{VP}$ ______ give it **to whom**]]?

c. **Final union set**

$$\left\{\begin{array}{lllllll}
P^0 < {D_1}^0 & {D_1}^0 < C^0 & C^0 < {D_3}^0 & {D_3}^0 < T^0 & T^0 < V^0 & V^0 < {D_2}^0 & \mathbf{{D_2}^0 < P^0} \\
P^0 < C^0 & {D_1}^0 < {D_3}^0 & C^0 < T^0 & {D_3}^0 < V^0 & T^0 < {D_2}^0 & \mathbf{V^0 < P^0} & \mathbf{{D_2}^0 < {D_1}^0} \\
P^0 < {D_3}^0 & {D_1}^0 < T^0 & C^0 < V^0 & {D_3}^0 < {D_2}^0 & & \mathbf{V^0 < {D_1}^0} & \\
P^0 < T^0 & \mathbf{{D_1}^0 < V^0} & C^0 < {D_2}^0 & & & & \\
\mathbf{P^0 < V^0} & \mathbf{{D_1}^0 < {D_2}^0} & & & & & \\
\mathbf{P^0 < {D_2}^0} & & & & & &
\end{array}\right\}$$

Later, in chapter 6, I will show that for the VP-doubling case in Mandarin, its final union set also violates the Asymmetry Constraint. A preview for this case is shown in (43).

(43) a. **Mandarin (aspectual form)**

Chi-guo bale, Lili dique mei **chi-guo bale**
eat-ASP Guava Lili indeed not **eat-ASP Guava**
'As for having eaten Guava before, Lili indeed hasn't eaten Guava before.'

b. **The final union set**

$$\left\{\begin{array}{llllll}
{Asp_1}^0 < {D_1}^0 & {D_1}^0 < Top^0 & Top^0 < C^0 & C^0 < {D_2}^0 & {D_2}^0 < Adv^0 & Adv^0 < Neg^0 \\
{Asp_1}^0 < Top^0 & {D_1}^0 < C^0 & Top^0 < {D_2}^0 & C^0 < Adv^0 & {D_2}^0 < Neg^0 & Adv^0 < T^0 \\
{Asp_1}^0 < C^0 & {D_1}^0 < {D_2}^0 & Top^0 < Adv^0 & C^0 < Neg^0 & {D_2}^0 < T^0 & \mathbf{Adv^0 < {Asp_1}^0} \\
{Asp_1}^0 < {D_2}^0 & {D_1}^0 < Adv^0 & Top^0 < Neg^0 & C^0 < T^0 & \mathbf{{D_2}^0 < {Asp_1}^0} & \mathbf{Adv^0 < {D_1}^0} \\
{Asp_1}^0 < Adv^0 & {D_1}^0 < Neg^0 & Top^0 < T^0 & \mathbf{C^0 < {Asp_1}^0} & \mathbf{{D_2}^0 < {D_1}^0} & \\
{Asp_1}^0 < Neg^0 & {D_1}^0 < T^0 & & \mathbf{C^0 < {D_1}^0} & & \\
{Asp_1}^0 < T^0 & & & & & \\
 & & & & & \\
Neg^0 < T^0 & \mathbf{T^0 < {Asp_1}^0} & \mathbf{V^0 < {D_1}^0} & & & \\
\mathbf{Neg^0 < {Asp_1}^0} & \mathbf{T^0 < {D_1}^0} & & & & \\
\mathbf{Neg^0 < {D_1}^0} & & & & &
\end{array}\right\}$$

So, for both the non successive-cyclic movement case (42) and the VP-doubling case (43), they violate the Asymmetry Constraint, and thus the linearization process proceeds to Stage 2. However, only the VP-doubling case continues to Stage 3 and 4 because it has a qualified constituent in the structure; while the non successive-cyclic movement case stops at Stage 2 and causes the linearization process to crash since there is no qualified constituent. For the VP-doubling case, during Stage 3 and 4, the qualified constituent, which is an XP-level syntactic object, is turned into a string (44a), and this string is put into the linearization process (44b), where the set not only contains ordering statements that order words but also contains ordering statements that order words and strings. Then, the set in (44b), which contains both words (i.e., X^0s) and strings (i.e., XPs), is finally turned into a string (45) by the String Forming Mechanism, which is eventually the output of PF after lexical insertion.

(44) a. $<_S \mathrm{Asp}_1{}^0\mathrm{D}_1{}^0>$

b.

$$\left\{\begin{array}{lllll}
Asp_1{}^0 < D_1{}^0 & Top^0 < C^0 & C^0 < D_2{}^0 & D_2{}^0 < Adv^0 & Adv^0 < Neg^0 \\
<_S \mathbf{Asp_1{}^0D_1{}^0}> < Top^0 & Top^0 < D_2{}^0 & C^0 < Adv^0 & D_2{}^0 < Neg^0 & Adv^0 < T^0 \\
<_S \mathbf{Asp_1{}^0D_1{}^0}> < C^0 & Top^0 < Adv^0 & C^0 < Neg^0 & D_2{}^0 < T^0 & Adv^0 < Asp_1{}^0 \\
<_S \mathbf{Asp_1{}^0D_1{}^0}> < D_2{}^0 & Top^0 < Neg^0 & C^0 < T^0 & D_2{}^0 < Asp_1{}^0 & Adv^0 < D_1{}^0 \\
<_S \mathbf{Asp_1{}^0D_1{}^0}> < Adv^0 & Top^0 < T^0 & C^0 < Asp_1{}^0 & D_2{}^0 < D_1{}^0 & \\
<_S \mathbf{Asp_1{}^0D_1{}^0}> < Neg^0 & & C^0 < D_1{}^0 & & \\
<_S \mathbf{Asp_1{}^0D_1{}^0}> < T^0 & & & & \\
 & & & & \\
\quad Neg^0 < T^0 & T^0 < Asp_1{}^0 & & & \\
\quad Neg^0 < Asp_1{}^0 & T^0 < D_1{}^0 & & & \\
\quad Neg^0 < D_1{}^0 & & & &
\end{array}\right\}$$

(45) **The final string**

$<_S <_S \mathrm{Asp}_1{}^0\mathrm{D}_1{}^0>\mathrm{Top}^0\mathrm{C}^0\mathrm{D}_2{}^0\mathrm{Adv}^0\mathrm{Neg}^0\mathrm{T}^0\mathrm{Asp}_1{}^0\mathrm{D}_1{}^0>$

As a quick summary, for the previous non successive-cyclic case and V-doubling case, and also the VP-doubling case, what the Set-to-String algorithm does is to turn their final union set into a string. The only difference is that during the linearization process, the previous non successive-cyclic case and V-doubling case only deal with ordering statements that linearize words (i.e., X^0s) but for the VP-doubling case, it also deals with ordering statements that linearize words and strings (i.e., XPs). In other words, the Set-to-

String algorithm turns the final union set of the previous non successive-cyclic case and V-doubling case into a string without forming a sub-string first, while for the VP-doubling case, he Set-to-String algorithm first generates a string out of the subset of the ordering statements and then forms the final string, which contains the sub-string.

4.4 Summary

In this chapter, I introduced the basic algorithm that contains the Candidates Generator G, the Constraints and the Set-to-String algorithm. The Candidates Generator G generates all possible sets of ordered pairs using the Maximal X^0 Nodes; the Constraints filter out most of the sets and leave one survivor that satisfies all the Constraints; and the Set-to-String algorithm proceeds with the survived set and the survived set becomes a string.[16] Finally, lexical items are inserted and the final utterance is produced. In the next chapter, I will incorporate this basic linearization process with Fox and Pesetsky (2005)'s cyclic linearization and show how it works. I will also introduce the Ordering Deletion rule that gets rid of the redundant ordering statements caused by the cyclic linearization process.

[16]Here, I assume that there is always only one legitimate utterance generated by the linearization process. However, the linearization process proposed in this dissertation can potentially generate more than one legitimate utterance.

CHAPTER 5

The cyclic linearization process

So far, I have been making an implicit assumption that the linearization process of p is applied to the whole structure. In this chapter, I show how the basic linearization process works in a cyclic way by first presenting Fox and Pesetsky (2005)'s proposal about cyclic linearization in section 5.1 and then showing how the basic linearization algorithm is implemented in a cyclic way in section 5.2.

5.1 Fox and Pesetsky (2005)'s cyclic linearization

Fox and Pesetsky (2005) propose that for a given linearization algorithm, it is applied to each Spell-out domain, where Spell-out domain is defined in (1). They propose that the list of Spell-out domains includes at least CP, VP and DP. For instance, when a Spell-out domain, say DP, is constructed in the derivation, Spell-out applies to DP and linearizes DP using the linearization algorithm; then build the structure for the next Spell-out domain, say VP, and then apply Spell-out to VP and linearizes VP using the linearization algorithm, and repeat this procedure until the structure in the last Spell-out domain is linearized.

(1) **Spell-out domain**

Spell-out domain refers to the constituents that are mapped by Spell-out, where Spell-out refers to the mapping between syntax and phonology.

Fox and Pesetsky (2005)

One key property of Fox and Pesetsky (2005)'s cyclic linearization is Order Preservation, which is stated in (2).

(2) **Order Preservation**

Information about linearization, once established at the end of a given Spell-out domain, is never deleted in the course of a derivation. The sole function of Spell-out is to add information.

Fox and Pesetsky (2005)

For instance, Order Preservation requires that once the linearization of a Spell-out domain, say DP, is generated, the linearization of DP will never be deleted and will be added to be part of the information about linearization. After the next Spell-out domain, namely VP, is linearized, the linearization information of VP will also be added to be part of the information about linearization and cannot be deleted either.

In the following, I discuss the Object Shift phenomenon (Holmberg 1986) in Swedish which Fox and Pesetsky (2005) argue to be explained by cyclic linearization. Before going in to the details of Object Shift in Swedish, I provide a brief discussion about the verb-second property of Swedish.

Verb-second refers to the phenomenon that the finite verb is obligatorily the second constituent and is typical for Germanic languages. As one of the Germanic languages, Swedish also has this property, which are shown in (3). It can be seen that regardless of what the first constituent (i.e., the italicized material) is, the finite verb (i.e., the auxiliary verb *har* 'have') is the second constituent. Traditionally, verb-second is believed to be a result of verb movement — the verb moves to C^0 (cf. Koster 1975, Besten 1983).

(3) a.

Jag	**har**	ärligt	talat	aldri	sett	huggormar	i	den	här	skogen
I	have	honestly	speaking	never	seen	adders	in	this	here	forest

b.

Huggormar	**har**	Jag	ärligt	talat	aldri	sett	i	den	här	skogen
adders	have	I	honestly	speaking	never	seen	in	this	here	forest

c.

I	*den*	*här*	*skogen*	**har**	Jag	ärligt	talat	aldri	sett	Huggormar
in	this	here	forest	have	I	honestly	speaking	never	seen	adders

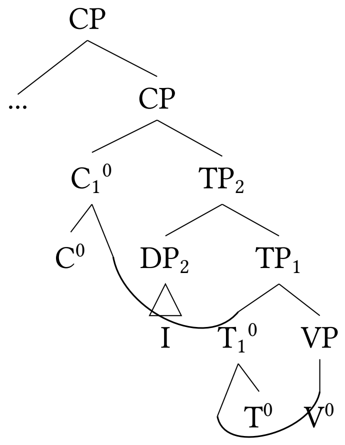
d.

ärligt	*talat*	**har**	Jag	aldri	sett	Huggormar	i	den	här	skogen
honestly	speaking	have	I	never	seen	adders	in	this	here	forest

'To be honest I've never seen adders in this forest.'

(Holmberg 2015: 2, ex. 1)

(4)



As shown in (4), the verb moves to C^0, preceding the subject *Jag* 'I' and after the first constituent at Spec, CP (e.g., *huggormar* 'adders').

However, verb-second can be blocked in an embedded clause, which is shown in (5), where the finite verb *kysste* 'kissed' is not the second constituent in the embedded clause. In this case, verb-movement is considered to be blocked. However, one exception for this is found in Icelandic (6), where the verb precedes the adverb, which is evidence that the verb moves out of VP; and since it is below the subject, which is considered to be in spec-TP, the verb moves to T^0.

(5)

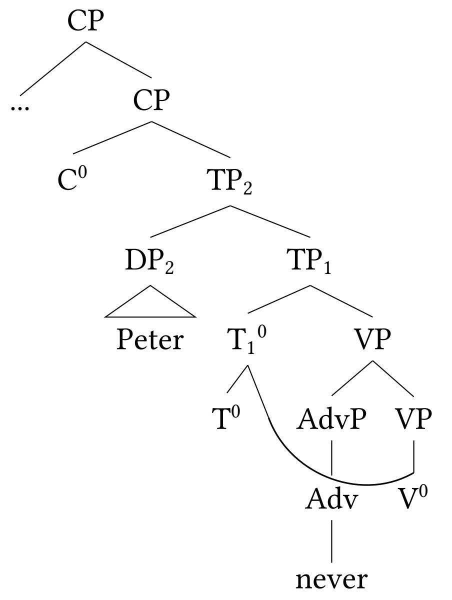
att	jag	inte	kysste	henne
that	I	not	kissed	her

(Holmberg 1999: 1, ex.1c')

(6)	a.	Ég spurði af-hverju Pétur læsi$_v$ aldrei [$_{VP}$ t$_V$ þessa bók].
		I asked why Peter read never this book

(Vikner 2005: 395-396, ex. 12(a))

b.



Now, I go back to Object Shift. Object Shift is found in the North Germanic (Icelandic, Faroese, Norwegian, Danish, Swedish) languages and refers to the rule that moves an object leftwards under the condition that the verb selecting the object also moves out of VP, which is also known as Holmberg's Generalization. An example for object shift is in (17) from Swedish: when the verb in V^0 does not move to C^0, Object Shift is impossible. To be more specific, (7a) is grammatical, where the verb moves to C^0 and the object moves out of the VP. In contrast, in (7b) and (7c), the object moves out of the VP while the verb stays inside the VP. To be more specific, in (7b), verb movement is blocked and the verb *kysste* 'kissed' has to stay in VP but the object moves out of VP across the verb; in (7c), since there is an auxiliary verb in the clause, verb-second applies to the auxiliary verb, so the verb *kysst* 'kissed' also stays in VP but the object moves out of the VP across the verb.[1]

[1]According to Holmberg (1986), usually Object Shift only applies to weakly stressed pronouns, but Object Shift applies to all NPs in Icelandic, provided they are not very heavy.

(7) **Object Shift**

a. Jag kysste henne inte [$_{VP}$ t$_V$ t$_O$]
I kissed her not

b. *att jag henne inte [$_{VP}$ kysste t$_O$]
that I her not kissed

c. *Jag har henne inte [$_{VP}$ kysst t$_O$] (Holmberg 1999: 1, ex.1)
I have her not kissed

Note that Object Shift does not hinge on the verb movement to C^0. In Icelandic embedded clauses, the verb moves to T^0, and Object Shift is still possible. For instance, in (8), the verb *læsi* 'read' moves to T^0, and the object can precede the adverb *þessa* 'never', which is evidence that the object moves out of VP, too.

(8) Ég spurði af-hverju Pétur læsi$_v$ bók$_i$ aldrei þessa [$_{VP}$ t$_V$ t$_i$].
I asked why Peter read this book never
(Vikner 2005: 395-396, ex. 12(b))

The first modern analysis for Object Shift is in Holmberg (1986), which ties the facts to case assignment: when the verb stays in V^0, it assigns case obligatorily to its object and thus, the object cannot move freely into a different position and be assigned case there. On the contrary, if the verb moves to C^0, the trace of the verb optionally assigns case to its object and thus, the object can move to a different place and be assigned case there. However, there seem to be some problems with this case analysis. First, according to works such as Burzio (1986), Johnson (1991) and Kratzer (1996), it is v^0 that assigns accusative case to the object, not the verb. Even if one adopts the theory that it is indeed the verb that assigns case, Holmberg (1999) made an observation that any phonologically visible category inside VP preceding the object can block Object Shift, not just an unmoved verb, including prepositions (9), verb particles (10), and indirect object (11). For example, (9a) has verb-second word order (i.e., the verb *talade* "spoke" moves to C^0) and is grammatical; while in (9b), despite the fact that the verb moves to C^0, it is ungrammatical if the object moves out of VP when there is a preposition preceding the object in its original position. This is also true of (10) and (11), where there is a verb particle or an indirect object preceding the object in its original position. Note that unlike the example in (17), in (9)-(10), object shift is still not allowed, though the verb has moved. For (9) and (10), one might argue that it is the preposition and the particle that assign case to the object, so moving the preposition and the particle, instead of the verb, is required for Object Shift. However, it is unclear why (11b) is ungrammatical, especially given the facts in (12) —under the case analysis, it is unclear why Object Shift is disallowed when the indirect object does not move (11b) but it is allowed when the indirect object moves (12), given that the indirect object is clearly not a case assigner like the preposition and the particle.

(9) a. Jag talade inte [$_{VP}$ t$_V$ med henne]
I spoke not with her
b. *Jag talade henne inte [$_{VP}$ t$_V$ **med** t$_O$]
I spoke her not with

(10) a. Dom kastade inte [$_{VP}$ t$_V$ ut mej]
they threw not out me
b. *Dom kastade mej inte [$_{VP}$ t$_V$ **ut** t$_O$]
they threw me not out

(11) a. Jag gav inte [$_{VP}$ t$_V$ Elsa den]
I gave not Elsa it
b. *Jag gav den inte [$_{VP}$ t$_V$ **Elsa** t$_O$]
I gave it not Elsa

(Holmberg 1999: 2, ex. 2(a) - (c'))

(12) a. Vem$_{io}$ gav du den$_o$ inte [$_{VP}$ t$_{io}$ t$_o$]?
who gave you it not
'Who didn't you give it to?'

b. Henne$_{io}$ visar jag den$_o$ helst inte [$_{VP}$ t$_{io}$ t$_o$].
her show I it rather not
'I'd rather not show it to HER.'

(Holmberg 1999: 17, ex. 43)

In fact, Holmberg (1999) provides another view to explain the facts in (17) and (9) - (12): it is the linear position of an element that is responsible for the intervention effect - Object Shift is prevented from moving across any phonologically visible element in VP. Along this line, Fox and Pesetsky (2005) propose that Object Shift is related to linearization. To be more specific, assuming that Object Shift does not proceed through Spec, VP, before Object Shift, the object follows certain items; and after Object Shift, the object precedes those items. Due to Order Preservation, information about the object preceding those items and following those items is preserved, which results in ordering contradiction.

Before going into the details of explaining the facts in (17), I would like to point out that in Fox and Pesetsky (2005), they do not make a distinction between vP and VP in most cases, including the Object Shift examples in (17). So, in the following discussion of (17), I will also treat VP, not vP, as a Spell-out domain. Also, in the Object Shift example, whether DP being a Spell-out domain is not crucial to the explanation, so I will only focus on VP and CP being a Spell-out domain in the following discussion. Also, following Fox and Pesetsky (2005), for the ease of presentation, the base-generated position of the external argument is ignored for the Object Shift examples.

Now, I continue with the discussion about how Fox and Pesetsky (2005) explains Object Shift using (7a) as an example. To begin with, Spell-out applies to the VP Spell-out domain and linearizes it. The linearization of the VP Spell-out domain is shown in (13).

(13) **The ordering statements in the VP domain, labeled as $\mathbf{L_{VP}}$:**

{*kysste* 'kissed' < *henne* 'her'}

Then, Spell-out applies to the next Spell-out domain, which is CP. The linearization of the CP Spell-out domain is shown in (14). Note that for Fox and Pesetsky (2005), "linearization" means ordered pairs, not strings.

(14) **The ordering statements in the CP domain, labeled as $\mathbf{L_{CP}}$:**

$$\left\{\begin{array}{lll} \textit{jag `I'} < \textit{kysste `kissed'} & \boldsymbol{\textit{kysste}}\ \textbf{`kissed'} < \boldsymbol{\textit{henne}}\ \textbf{`her'} & \textit{henne `her'} < \textit{inte `not'} \\ \textit{jag `I'} < \textit{henne `her'} & \textit{kysste kissed} < \textit{inte `not'} & \\ \textit{jag `I'} < \textit{inte `not'} & & \end{array}\right\}$$

It can be seen that L_{VP} and L_{CP} are compatible with each other, so (7a) is linearized as *Jag kysste henne inte* ('I kissed her not').

Now, let's look at (7b). Still, Spell-out applies to VP and linearizes it. The linearization of VP is shown in (15), which is the same as the linearization of VP in (13).

(15) **The ordering statements in the VP domain, labeled as L_{VP}:**

{*kysste* 'kissed' < *henne* 'her'}

Next, Spell-out applies to CP and linearizes it. The linearization of CP is shown in (16), which is different from the linearization of CP in (14) especially regarding the ordering of the object *henne* 'her' and the verb *kysste* 'kissed'.

(16) **The ordering statements in the CP domain, labeled as L_{CP}:**

att 'that' < jag 'I', att	*jag 'I' < henne 'her'*	*henne 'her' < inte 'not'*
att 'that' < henne 'her'	*jag 'I' < inte 'not'*	***henne* 'her' < *kysste* 'kissed'**
att 'that' < inte 'not'	*jag 'I' < kysste 'kissed'*	
att 'that' < kysste 'kissed'		
inte 'not' < kysste *'kissed'*		

It can be seen that L_{VP} contradicts L_{CP}: unlike (7a), repeated below as (17a), for (7b), repeated below as (17b), *henne* 'her' < *kysste* 'kissed' in the CP domain but *kysste* 'kissed' < *henne* 'her' in the VP domain, which creates an ordering contradiction. Since L_{VP} cannot be deleted based on Order Preservation, the contradiction cannot be resolved. As a result, the linearization of (7b) fails. The same logic applies to (7c).

(17) **Object Shift**

a. Jag kysste henne inte [$_{VP}$ t_V t_O]
   I kissed her not

b. *att jag henne inte [$_{VP}$ kysste t_O]
   that I her not kissed

As a quick summary, under the assumption that the Object Shift does not go through Spec, VP, since linearization is implemented to each Spell-out domain and observes Order Preservation, the ordering for the Spell-out domain CP must be compatible with the ordering for its previous Spell-out domain VP. In other words, the ordering of the object *henne* 'her' and the verb *kysste* must remain the same in the VP Spell-out domain and the CP Spell-out domain. If the object moves out of VP, the verb must also move to C so that the relative ordering of the object and the verb can remain as *kysste* 'kissed' $<$ *henne* 'her' in the CP Spell-out domain. In other words, despite the fact that the object phrase is base-generated from a non-initial position in VP, but when the verb moves out of VP, the object becomes a non-initial position.

It is worth pointing out that a crucial ingredient of Fox and Pesetsky (2005)'s cyclic linearization is Order Preservation that preserves the precedence relations, which are used to evaluate whether there is ordering contradiction; for instance, in the case of Object Shift, the precedence relations are used to see whether the ordering statements in the VP Spell-out domain are preserved in the CP Spell-out domain. In this sense, for Fox and Pesetsky (2005), it is crucial for their linearization to include precedence relations instead of just including a string. In the work of this dissertation, I follow Fox and Pesetsky (2005) and assume that linearization should include a representation of precedence relations.

In the next section, I propose a detailed mechanism of how Fox and Pesetsky (2005)'s cyclic linearization works in combination with the basic linearization process that I proposed in the previous chapter.

5.2 The cyclic linearization process of p

In this section, I propose a cyclic linearization process that incorporates the basic linearization process with Fox and Pesetsky (2005)'s cyclic linearization. It is worth pointing out that different from Fox and Pesetsky (2005), I assume that

1. Though linearization is applied in a cyclic fashion, it is applied cyclically to a syntactic structure that is fully built.[2]

2. In addition to DP, VP, and CP, the root node is always a Spell-out domain.

3. All the material inside the current Spell-out domain is linearized (in Fox and Pesetsky (2005), material that is linearized in the previous Spell-out domain is not linearized again in the current Spell-out domain).

In general, after the whole structure is built, do the following: start linearizing the lowest Spell-out domain, and then continue to linearize the next higher Spell-out domain. If there is a previous Spell-out domain, take the union of the set of ordered pairs from the current Spell-out domain and the set of ordered pairs from the previous Spell-out domain. Repeat the procedure until the final union set is formed. Then, run the Set-to-String algorithm to the final union set. The detailed cyclic linearization process is stated as the following. Note that since the linearization process also includes the string (of nodes) representation, in addition to precedence relation, a Spell-out domain will be first linearized as a string of nodes, and then lexical items will be inserted to the string and get pronounced.

1. **Generate sets and filter - starting set:** Apply the the Candidates Generator G and the Constraints (i.e., the Totality Constraint, the Anti-reflexivity Constraint, the Asymmetry Constraint and the Language Specific Constraints) to the first Spell-out domain XP.

[2]In Fox and Pesetsky (2005), the application of linearization is interleaved in the process of building the structure (e.g., first build VP and then linearize VP; and later build CP and then linearize CP). However, in my analysis, linearization is applied cyclically after the whole structure is built (e.g., first build the whole structure, then linearize VP, and then linearize CP). Assuming that movement is interleaved in the process of building structures (e.g., V-to-v movement happens while vP is built up, instead of happening after the whole structure is built.), applying linearization after the whole structure is built in my analysis and applying linearization during the course of building structures in Fox and Pesetsky (2005) are effectively the same, at least for cases discussed in this dissertation. Choosing to linearize after the whole structure is built in my analysis is for ease of presentation. To be more specific, this dissertation does not focus on the theories of how things move, but focus on how a structure with movement is linearized. Applying linearization after the whole structure is built leaves out the details of how movement happens in each Spell-out domain, which helps one focusing on how each Spell-out domain is linearized.

2. **Cyclic process steps**

   (a) **Cyclic process step I: generate sets and filter** Apply the Candidates Generator G and the Constraints to the next Spell-out domain YP.

   (b) **Cyclic process step II: Union** Apply the operation Union to the set of ordered pairs that survives the Constraints from XP and the set of ordered pairs that survives the Constraints from the previous Spell-out domain YP. I refer to the set generated by the Union operation as the union set.

   (c) **Cyclic process step III: Filter on the union set** Impose the Constraints on the union set. Note that the constraints are evaluated against the structure of the most recent Spell-out domain.

   (d) **Cyclic process step IV: check point** If the union set survives all the constraints, the linearization process can continue; otherwise, the linearization process will fail.

3. **Repeat until the last union:** Repeat the cyclic process steps. Once the last union set is formed, apply Ordering Deletion, which is defined in (18), and instead of implementing step III and IV, apply the Set-to-String algorithm.

4. **Vocabulary insertion:** Form lexical insertion sites and insert vocabulary items to the sites.

(18) **Ordering Deletion**

For a given set of ordering statements, Y $< \alpha$ must be deleted if X dominates Y and X $< \alpha$, and $\alpha <$ Y must be deleted if X dominates Y and $\alpha <$ X; unless there exists β such that $\beta <$ X and Y $< \beta$, or X $< \beta$ and $\beta <$ Y.

In the following, I first use a simple sentence in (19) to illustrate how the cyclic linearization process works, and then use a more complex sentence in (20) to show how a non successive-cyclic movement is ruled out. Lastly, I use the example in (21) to illustrate how the Ordering Deletion rule works.

Note that since the linearization of the Spell-out domain DP is trivial for the current as well as the future cases and leaving it out will not affect the discussion, I omit the linearization of the DP Spell-out domain for all cases.

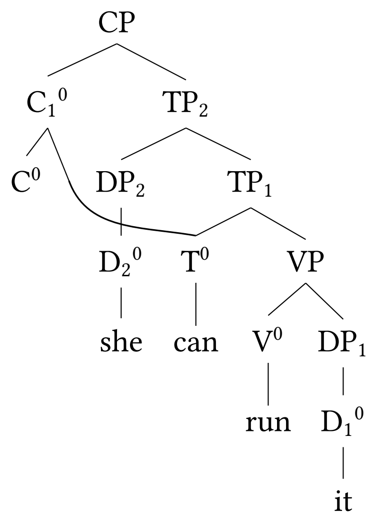
(19) *Can she run it?*

(20) *To whom will she give it?*

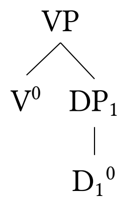
(21) *She can run it.*

5.2.1 Apply the cyclic linearization process

Let's first look at example (19), the structure of which is shown in (22).

(22)


The first Spell-out domain is the VP domain, which is shown in (23). The Maximal X^0 Nodes in (23) are shown in (24).

(23)


(24) Maximal Nodes of VP = {V^0, ${D_1}^0$}

After applying G(VP) and the Constraints on the generated sets, the surviving set of ordered pairs is shown in (25).

(25) {$V^0 < {D_1}^0$}

Next, apply the cyclic process steps. The next Spell-out domain is CP, the structure of which is shown in (22). The Maximal X^0 Nodes of the CP Spell-out domain are shown in (26). Note that T^0 is not a Maximal X^0 Node since it is dominated by ${C_1}^0$. Thus, only the nodes in (26) will serve as the input for the Candidates Generator G; the node T^0 will not occur in any ordered pairs in any set generated by G.

(26) Maximal Nodes of CP = {${C_1}^0$, ${D_2}^0$, V^0, ${D_1}^0$}

Apply the Candidates Generator G to CP and impose the Constraints on the generated sets, and the surviving set of ordered pairs is shown in (27).

(27) $$\begin{Bmatrix} {C_1}^0 < {D_2}^0 & {D_2}^0 < V^0 & V^0 < {D_1}^0 \\ {C_1}^0 < V^0 & {D_2}^0 < {D_1}^0 & \\ {C_1}^0 < {D_1}^0 & & \end{Bmatrix}$$

Table 5.1 summarizes the relevant structural relationship in CP, and the required orders by the Language Specific Constraints for all Maximal X^0 Nodes α and β of CP if α and β are ordered as a pair in a given set.

		IDM	**relationship**	**ordered pairs**
(${C_1}^0$	TP_2)	CP	sisters	${C_1}^0 < {D_2}^0$, ${C_1}^0 < V^0$, ${C_1}^0 < {D_1}^0$
(DP_2	TP_1)	TP_2	sisters	${D_2}^0 < V^0$, ${D_2}^0 < {D_1}^0$
(V^0	DP_1)	VP	sisters	$V^0 < {D_1}^0$

Table 5.1: CP relation

$C_1{}^0$ and TP_2 are sisters because their immediate dominating mother CP fully dominates them. The reason CP fully dominates $C_1{}^0$ and TP_2 is that every path for $C_1{}^0$ and TP_2 includes CP. The same logic applies to DP and TP_1. The relevant paths and full-dominance relationships are shown in (28) and (29).

(28) a. $p(C_1{}^0) = (CP)$

b. $p(TP_2) = (CP)$

c. $p(DP_2) = (TP_2, CP)$

d. $p(TP_1) = (TP_2, CP)$

e. $p(V^0) = (VP, TP_1, TP_2, CP)$

f. $p(DP_1) = (VP, TP_1, TP_2, CP)$

g. $p(D_1{}^0) = (DP_1, VP, TP_1, TP_2, CP)$

(29) a. CP fully dominates $C_1{}^0$ and TP_2

b. TP_2 fully dominates DP_2, TP_1, $D_2{}^0$, V^0, $D_1{}^0$

c. DP_2 fully dominates $D_2{}^0$

d. TP_1 fully dominates V^0, $D_1{}^0$

e. VP fully dominates V^0, DP_1

f. DP_1 fully dominates $D_1{}^0$

Now, apply the operation Union to the set of ordered pairs from the VP domain and to the one from the CP Spell-out domain. The union set of VP and CP is identical to the one in the CP Spell-out domain and is shown in (30).

(30)

$$\left\{\begin{array}{lll} C_1{}^0 < D_2{}^0 & D_2{}^0 < V^0 & V^0 < D_1{}^0 \\ C_1{}^0 < V^0 & D_2{}^0 < D_1{}^0 & \\ C_1{}^0 < D_1{}^0 & & \end{array}\right\}$$

Since this union set is the final union set, apply Ordering Deletion. Since no ordering statement fits the criteria in Ordering Deletion, nothing is deleted. Note that in a later case in this section, I will illustrate how Ordering Deletion works. Note that the final union set satisfies all the constraints. The reason is the following. The Totality Constraint is evaluated against the structure of the most recent Spell-out domain, which is the structure of the CP domain. Thus, the Maximal X^0 Nodes for the union of VP and CP domain is the same as the one in the CP domain in (26). Since all the Maximal X^0 Nodes are ordered relative to each other in the union set in (30), this set satisfies the Totality Constraint. It satisfies the Anti-reflexivity Constraint since for each statement in the union set, only distinct nodes are ordered. It satisfies the Asymmetry Constraint since there are no paradoxical orders. It satisfies the Language Specific Constraints since no required order is missing.

Finally, implement the Set-to-String algorithm. The algorithm starts with determining whether the set (30) violates the Asymmetry Constraint by finding the intersection sets for each node in this set (i.e., Stage 1). The relevant part of the algorithm is shown below, and the intersection sets are in (31).

Stage 1:

1. **Find the intersection sets:** For each distinct node n_1, n_2, n_3, ... in the set of ordered pairs O = $\{n_1 < n_3, n_3 < n_2, ...\}$, get the set X = $\{..., n_1, ...\}$ that contains all the nodes that precedes n and the set Y = $\{..., n_2, ...\}$ that contains all the nodes that follows n, and get the set I = X $\cap$ Y, which is the intersection of set X and set Y.

2. **Check the intersection sets:** If all the intersection sets are empty, run the String Forming Mechanism on set O and the set will be turned into a string. **Otherwise, proceed to Stage 2.**

(31) a. $C_1{}^0$: $\varnothing = \varnothing \cap \{D_2{}^0, V^0, D_1{}^0\}$

b. $D_2{}^0$: $\varnothing = \{C_1{}^0\} \cap \{V^0, D_1{}^0\}$

c. V^0: $\varnothing = \{C_1{}^0, D_2{}^0\} \cap \{D_1{}^0\}$

d. $D_1{}^0$: $\varnothing = \{C_1{}^0, D_2{}^0, V^0\} \cap \varnothing$

Since all the intersection sets are empty, the linearization stops at Stage 1. Then, the next step is to run the String Forming Mechanism algorithm, repeated below as (32), which yields the string $<_S C_1{}^0D_2{}^0V^0D_1{}^0>$.

(32) **String Forming Mechanism**

For a given set of ordered pairs x, it is associated with a string str that is by default empty. Concatenate node/string α to string str if the node/string α is not preceded by any other node, and then delete all the ordered pairs that contain α. Repeat this procedure unless there is only one ordered pair $\gamma < \delta$ left in set x, then, first concatenate γ and then concatenate δ to string str.

The following is how the string is formed by the String Forming Mechanism algorithm. First, find the node that is not preceded by any other nodes (i.e., $C_1{}^0$), concatenate the node to the empty string str, and delete the ordering statements that contain $C_1{}^0$. The next step is to repeat this procedure (i.e., find the next node that is not preceded by any other nodes (i.e., $D_2{}^0$), concatenate the node to string $<_S C_1{}^0 >$, and delete all the statements that contain $D_2{}^0$). Now, there is only one statement left, which is $V^0 < D_1{}^0$, first V^0 and then $D_1{}^0$ will be concatenated to $<_S C_1{}^0D_2 >$, which yields the final string $<_S C_1{}^0D_2{}^0V^0D_1{}^0>$.

Next, it is the last step of the linearization process: insert vocabulary items. First, for each node, form the lexical insertion site, which is shown in (33). $A \sim B$ represents that The lexical insertion site for A is B. Then, the string becomes $<_S \#C_1{}^0\#\#D_2\#V^0\#\#D_1{}^0\#>$.

(33) a. ${C_1}^0 \sim \#{C_1}^0\#$

b. ${D_2}^0 \sim \#{D_2}^0\#$

c. $V^0 \sim \#V^0\#$

d. ${D_1}^0 \sim \#{D_1}^0\#$

Then, insert lexical items in the lexical insertion site, which is shown in (34). A = B represents that the lexical insertion for A is B. As a result, the string $<_S \#{C_1}^0\#\#D_2\#V^0\#\#{D_1}^0\#>$ is updated as $<_S$ can she run it$>$.

(34) a. $\#{C_1}^0\#$ = *can*

b. $\#{D_2}^0\#$ = *she*

c. $\#V^0\#$ = *run*

d. $\#{D_1}^0\#$ = *it*

Finally, we get string *can she run it* as the utterance.

Now, let's look at example (20), which is repeated below in (35). I will discuss the linearization of both the derivation that has successive-cyclic movement, which is shown in (36a), and the one that has non successive-cyclic movement, which is shown in (36b).

(35) *To whom will she give it?*

(36) Successive-cyclic movement v.s. Non successive-cyclic movement

a. [$_{CP}$ **To whom** will she [$_{VP}$ **to whom** give it **to whom**]]?

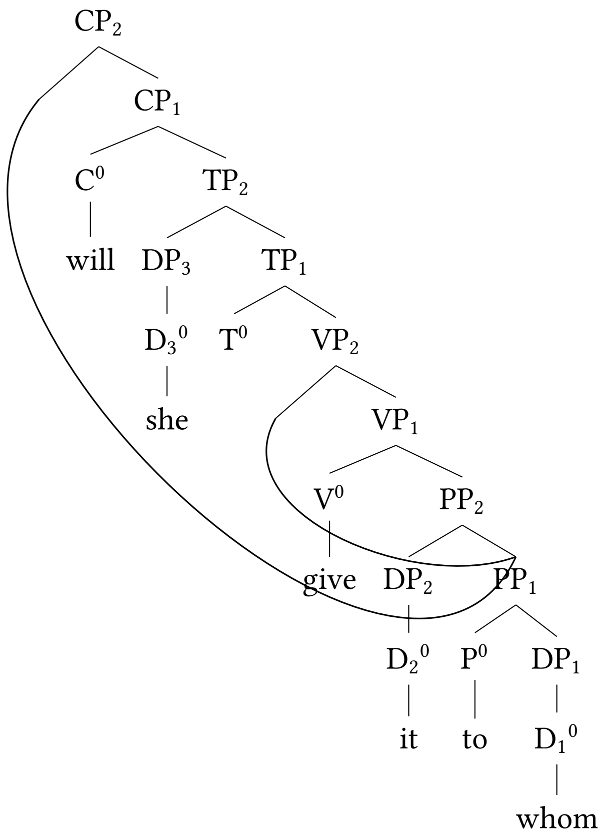
successive-cyclic movement

b. *[$_{CP}$ **To whom** will she [$_{VP}$ ______ give it **to whom**]]?

non successive-cyclic movement

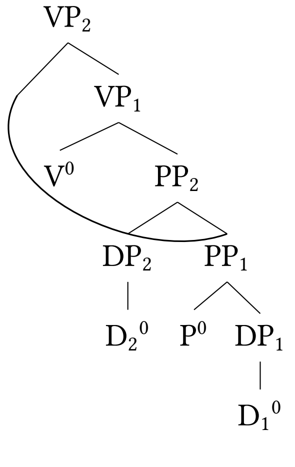
I start with the successive-cyclic derivation, the structure of which is shown in (37).

(37) Successive-cyclic movement



There are two Spell-out domains: VP_2 and CP_2. Let's start with the first Spell-out domain VP_2, the structure of which is shown in (38).

(38) Successive-cyclic movement



The Maximal X^0 Nodes are shown in (39)

(39) The Maximal X^0 Nodes of VP_2 = $\{P^0, {D_1}^0, V^0, {D_2}^0\}$

The set that satisfies all the constraints is shown in (40).

(40)

$$\begin{Bmatrix} P^0 < {D_1}^0 & {D_1}^0 < V^0 & V^0 < {D_2}^0 \\ P^0 < V^0 & {D_1}^0 < {D_2}^0 & \\ P^0 < {D_2}^0 & & \end{Bmatrix}$$

Below, Table 5.2 summarizes the relevant structural relationship in VP_2, regarding the Language Specific Constraints.

		IDM	**relationship**	**ordered pairs**
(PP_1	VP_1)	VP_2	sisters	$P^0 < V^0$, $P^0 < {D_2}^0$, ${D_1}^0 < V^0$, ${D_1}^0 < {D_2}^0$
(P^0	DP_1)	PP_1	sisters	$P^0 < {D_1}^0$
(V^0	PP_2)	VP_1	sisters	$V^0 < {D_2}^0$
(DP_2	PP_1)	PP_2	not sisters	N/A

Table 5.2: VP relation

Note that in this case, the distinction between full-dominance and dominance is important. The ordering statement $V^0 < P^0$ is not required because despite the fact that (i) the set has $P^0 < V^0$, (ii) V is the head, and (iii) PP_2 is the sister of V^0, PP_2 does not fully dominate P^0 (there is a path of P^0 that does not go through PP_2, which is shown in (42e-i)). The same logic applies to the reason $V^0 < {D_1}^0$ is not required (i.e., ${D_1}^0$ is also not fully dominated by PP_2 - see the path information in (42c-i)).

In addition, ${D_2}^0 < P^0$ is not required because despite the fact that (i) the set has $\{P^0 < {D_2}^0\}$, (ii) DP_2 is a specifier, (iii) ${D_2}^0$ is dominated by DP_2, and (iv) PP_1 fully dominates P^0, PP_1 is not a sister of DP_2 (i.e., their immediate dominating mother PP_2 does not fully dominate PP_1 - there is a path for PP_1 that does not go through PP_2 - see the path information in (42d-i)). The same logic applies to why ${D_2}^0 < {D_1}^0$ is not required.

The relevant full-dominance relationships and paths are shown in (41) and (42).

(41) a. PP_1 fully dominates P^0 and $D_1{}^0$

b. VP_1 fully dominates V^0 and D^0

c. DP_1 fully dominates $D_1{}^0$

d. PP_2 fully dominates $D_2{}^0$

(42) a. $p(V^0) = (VP_1, VP_2)$

b. $p(D_2{}^0) = (DP_2, PP_2, VP_1, VP_2)$

c. $p(D_1{}^0\ ) =$

(i) (DP_1, PP_1, VP_2)

(ii) $(DP_1, PP_1, PP_2, VP_1, VP_2)$

d. $p(PP_1) =$

(i) (VP_2)

(ii) (PP_2, VP_1, VP_2)

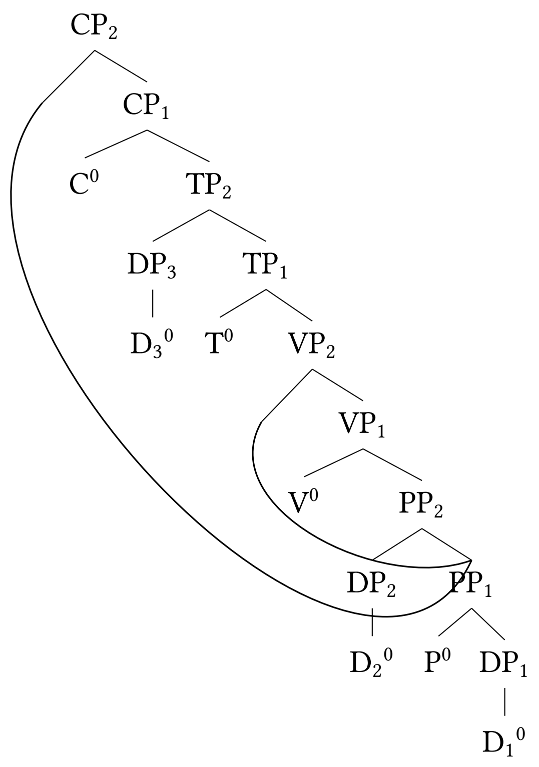
e. $p(P^0) =$

(i) (PP_1, VP_2)

(ii) (PP_1, PP_2, VP_1, VP_2)

Now, let's proceed to the CP_2 Spell-out domain, the structure of which is repeated below in (43).

(43)



The Maximal X^0 Nodes are shown in (57)

(44) The Maximal X^0 Nodes of CP_2 = $\{P^0, D_1{}^0, C^0, D_3{}^0, T^0, V^0, D_2{}^0\}$

The set that satisfies all the constraints are shown in (45).

(45)

$$\left\{\begin{array}{llllll}
P^0 < D_1{}^0 & D_1{}^0 < C^0 & C^0 < D_3{}^0 & D_3{}^0 < T^0 & T^0 < V^0 & V^0 < D_2{}^0 \\
P^0 < C^0 & D_1{}^0 < D_3{}^0 & C^0 < T^0 & D_3{}^0 < V^0 & T^0 < D_2{}^0 & \\
P^0 < D_3{}^0 & D_1{}^0 < T^0 & C^0 < V^0 & D_3{}^0 < D_2{}^0 & & \\
P^0 < T^0 & D_1{}^0 < V^0 & C^0 < D_2{}^0 & & & \\
P^0 < V^0 & D_1{}^0 < D_2{}^0 & & & & \\
P^0 < D_2{}^0 & & & & &
\end{array}\right\}$$

Below, Table 5.3 summarizes the relevant structural relationship in CP_2, regarding the Language Specific Constraints.

		IDM	relationship	ordered pairs
(PP_1	CP_1)	CP_2	sisters	$P^0 < C^0$, $P^0 < D_3{}^0$, $P^0 < T^0$, $P^0 < V^0$, $P^0 < D_2{}^0$ $D_1{}^0 < C^0$, $D_1{}^0 < D_3{}^0$, $D_1{}^0 < T^0$, $D_1{}^0 < V^0$, $D_1{}^0 < D_2{}^0$
(P^0	DP_1)	PP_1	sisters	$P^0 < D_1{}^0$
(C^0	TP_2)	TP_1	sisters	$C^0 < D_3{}^0$, $C^0 < T^0$, $C^0 < V^0$, $C^0 < D_2{}^0$
($D_3{}^0$	TP_1)	TP_2	sisters	$D_3{}^0 < T^0$, $D_3{}^0 < V^0$, $D_3{}^0 < D_2{}^0$
(T^0	VP_2)	TP_1	sisters	$T^0 < V^0$, $T^0 < D_2{}^0$
(V^0	PP_2)	VP_1	sisters	$V^0 < D_2{}^0$
(DP_2	PP_1)	PP_2	not sisters	N/A
(PP_1	VP_1)	VP_2	not sisters	N/A

Table 5.3: CP relation

Note that the ordering statement $C^0 < P^0$ is not required because despite the fact that (i) the set has $P^0 < C^0$, (ii) C is the head, and (iii) TP_2 is the sister of C^0, TP_2 does not fully dominate P^0 (there is a path of P^0 that does not go through PP_2, which is shown in (46b-i)). The same logic applies to the reason $T^0 < P^0$, $V^0 < P^0$, as well as $C^0 < D_1{}^0$, $T^0 < D_1{}^0$, $V^0 < D_1{}^0$ are not required.

In addition, $D_3{}^0 < P^0$ is not required because despite the fact that (i) the set has $P^0 < D_3{}^0$, (ii) DP_3 is a specifier, and (iii) DP_3 dominates $D_3{}^0$, TP_1 does not fully dominate P^0 (i.e., there is a path for P^0 that does not go through TP_1 - see (46b-i)). The same logic applies to the reason $D_3{}^0 < D_1{}^0$ is not required.

Furthermore, $D_2{}^0 < P^0$ is not required because despite the fact that (i) the set has $P^0 < D_2{}^0$, (ii) DP_2 is a specifier, (iii) DP_2 dominates $D_2{}^0$, and (iv) PP_1 fully dominates P^0,

DP_2 and PP_1 are not sisters (i.e., there are paths for PP_1 that does not go through PP_2 - see (46a-i) and (46a-ii)). The same logic applies to the reason ${D_2}^0 < {D_1}^0$ is not required.

(46) a. p(PP_1) =

(i) (CP_2)

(ii) (VP_2, TP_1, TP_2, CP_1, CP_2)

(iii) (PP_2, VP_1, VP_2, TP_1, TP_2, CP_1, CP_2)

b. p(P^0) =

(i) (PP_1, CP_2)

(ii) (PP_1, VP_2, TP_1, TP_2, CP_1, CP_2}

(iii) (PP_1, PP_2, VP_1, VP_2, TP_1, TP_2, CP_1, CP_2)

c. p(${D_1}^0$) =

(i) (DP_1, PP_1, CP_2)

(ii) (DP_1, PP_1, VP_2, TP_1, TP_2, CP_1, CP_2)

(iii) (DP_1, PP_1, PP_2, VP_1, VP_2, TP_1, TP_2, CP_1, CP_2)

Next, take the union of the set of ordered pairs from the VP Spell-out domain and the one from the CP Spell-out domain, both of which are repeated below in (47). Note that the set from CP contains all the ordering statements in the set from VP. Thus, the union set of CP and VP is the same as the one from CP.

(47)

a. $$\left\{\begin{array}{lll} P^0 < {D_1}^0 & {D_1}^0 < V^0 & V^0 < {D_2}^0 \\ P^0 < V^0 & {D_1}^0 < {D_2}^0 & \\ P^0 < {D_2}^0 & & \end{array}\right\}$$

b.
$$\left\{\begin{array}{llllll} P^0 < {D_1}^0 & {D_1}^0 < C^0 & C^0 < {D_3}^0 & {D_3}^0 < T^0 & T^0 < V^0 & V^0 < {D_2}^0 \\ P^0 < C^0 & {D_1}^0 < {D_3}^0 & C^0 < T^0 & {D_3}^0 < V^0 & T^0 < {D_2}^0 & \\ P^0 < {D_3}^0 & {D_1}^0 < T^0 & C^0 < V^0 & {D_3}^0 < {D_2}^0 & & \\ P^0 < T^0 & {D_1}^0 < V^0 & C^0 < {D_2}^0 & & & \\ P^0 < V^0 & {D_1}^0 < {D_2}^0 & & & & \\ P^0 < {D_2}^0 & & & & & \end{array}\right\}$$

Since the final Spell-out domain has been reached, apply Ordering Deletion. Since no ordering statements fit the criteria in Ordering Deletion, nothing is deleted. Then, apply the Set-to-String algorithm to the union set of CP and VP, which is the set in 2. Since the union set satisfies all the constraints, the Set-to-String algorithm proceeds with it. Next, get the intersection sets for it. Notice that for each node, the intersection sets are empty. Thus, run the String Forming Mechanism algorithm on the union set, which yields the string $<_S P_0{D_1}^0C_0{D_3}^0T_0V_0{D_2}^0>$.

The next step is to form lexical insertion sites, which is shown in (48).

(48) a. $P^0 \sim \#C^0\#$

b. ${D_1}^0 \sim \#{D_1}^0\#$

c. $C^0 \sim \#C^0\#$

d. ${D_3}^0 \sim \#{D_3}^0\#$

e. $T^0 \sim \#T^0\#$

f. $V^0 \sim \#V^0\#$

g. ${D_2}^0 \sim \#{D_2}^0\#$

Now, implement lexical insertion, which is shown in (49).

(49) a. P^0 = ‘to’

b. ${D_1}^0$ = ‘whom’

c. C^0 = ‘will’

d. $D_3{}^0$ = ‘she’

e. $T^0 = \varnothing$

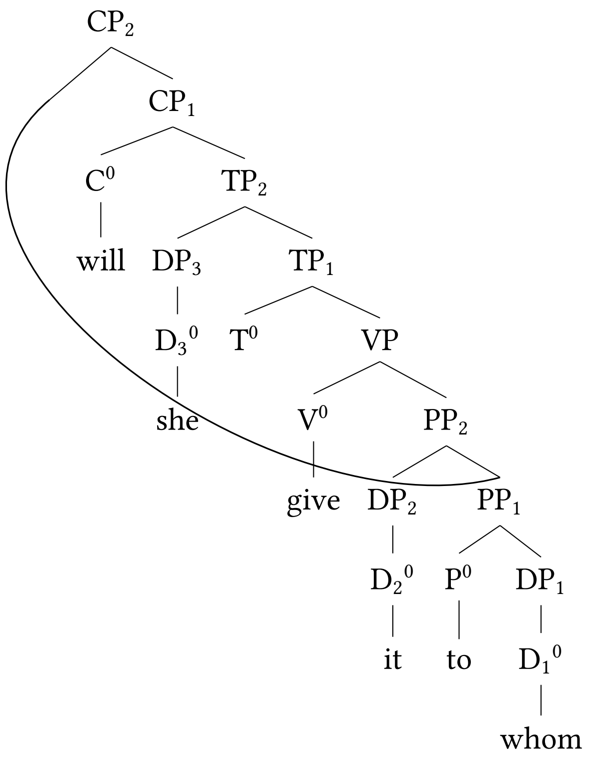
f. V^0 = ‘give’

g. $D_2{}^0$ = ‘it’

Thus, the string is updated as $<_S$ to whom will she give it>. Finally, the string becomes the utterance *to whom will she give it?*

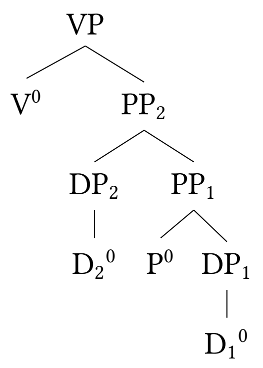
In the following, I will discuss the non successive-cyclic derivation of the sentence *to whom will she give it?* and show how the linearization of it fails. The structure is shown in (50).

(50) Non successive-cyclic movement



There are two Spell-out domains: VP and CP_2. Let’s start with the first constructed Spell-out domain VP, the structure of which is shown in (51).

(51)



The Maximal X^0 Nodes are shown in (52).

(52) The Maximal X^0 Nodes of VP = $\{V^0, {D_2}^0, P^0, {D_1}^0\}$ The set that satisfies all the constraints is shown in (53).

(53)
$$\left\{\begin{matrix} V^0 < {D_2}^0 & {D_2}^0 < P^0 & P^0 < {D_1}^0 \\ V^0 < P^0 & {D_2}^0 < {D_1}^0 & \\ V^0 < {D_1}^0 & & \end{matrix}\right\}$$

Below, Table 5.4 summarizes the relevant structural relationship in VP, regarding the Language Specific Constraints.

		IDM	**relationship**	**ordered pairs**
(V^0	PP_2)	VP	sisters	$V^0 < {D_2}^0$, $V^0 < P^0$, $V^0 < {D_1}^0$
(DP_2	PP_1)	PP_2	sisters	${D_2}^0 < P^0$, ${D_2}^0 < {D_1}^0$
(P^0	DP_1)	PP_1	sisters	$P^0 < {D_1}^0$

Table 5.4: VP relation

The relevant full-dominance relationship and paths are shown in (54) and (55).

(54) a. PP_2 fully dominates ${D_2}^0$, P^0 and ${D_1}^0$

b. PP_1 fully dominates P^0 and ${D_1}^0$

c. DP_1 fully dominates ${D_1}^0$

(55) a. $p(V^0) = (VP)$

b. $p(PP_2) = (VP)$

c. $p(DP_2) = (PP_2, VP)$

d. $p(D_2{}^0) = (DP_2, PP_2, VP)$

e. $p(PP_1) = (PP_2, VP)$

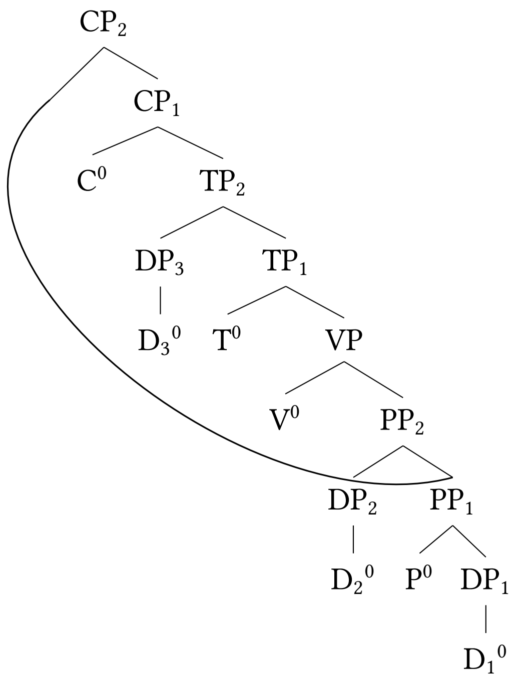
f. $p(P^0) = (PP_1, PP_2, VP)$

g. $p(DP_1) = (PP_1, PP_2, VP)$

h. $p(D_1{}^0) = (DP_1, PP_1, PP_2, VP)$

Now, let's proceed to the CP_2 Spell-out domain, the structure of which is repeated below in (56).

(56)



The Maximal X^0 Nodes are shown in (57).

(57) The Maximal X^0 Nodes of $CP_2 = \{P^0, D_1{}^0, C^0, D_3{}^0, T^0, V^0, D_2{}^0\}$

The set that satisfies all the constraints is shown in (58).

(58)
$$\begin{Bmatrix} P^0 < {D_1}^0 & {D_1}^0 < C^0 & C^0 < {D_3}^0 & {D_3}^0 < T^0 & T^0 < V^0 & V^0 < {D_2}^0 \\ P^0 < C^0 & {D_1}^0 < {D_3}^0 & C^0 < T^0 & {D_3}^0 < V^0 & T^0 < {D_2}^0 & \\ P^0 < {D_3}^0 & {D_1}^0 < T^0 & C^0 < V^0 & {D_3}^0 < {D_2}^0 & & \\ P^0 < T^0 & {D_1}^0 < V^0 & C^0 < {D_2}^0 & & & \\ P^0 < V^0 & {D_1}^0 < {D_2}^0 & & & & \\ P^0 < {D_2}^0 & & & & & \end{Bmatrix}$$

Below, Table 5.5 summarizes the relevant structural relationship in CP_2, regarding the Language Specific Constraints.

		IDM	**relationship**	**ordered pairs**
(PP_1	CP_1)	CP_2	sisters	$P^0 < C^0$, $P^0 < {D_3}^0$, $P^0 < T^0$, $P^0 < V^0$, $P^0 < {D_2}^0$ ${D_1}^0 < C^0$, ${D_1}^0 < {D_3}^0$, ${D_1}^0 < T^0$, ${D_1}^0 < V^0$, ${D_1}^0 < {D_2}^0$
(P^0	DP_1)	PP_1	sisters	$P^0 < {D_1}^0$
(C^0	TP_2)	TP_1	sisters	$C^0 < {D_3}^0$, $C^0 < T^0$, $C^0 < V^0$, $C^0 < {D_2}^0$
(${D_3}^0$	TP_1)	TP_2	sisters	${D_3}^0 < T^0$, ${D_3}^0 < V^0$, ${D_3}^0 < {D_2}^0$
(T^0	VP)	TP_1	sisters	$T^0 < V^0$, $T^0 < {D_2}^0$
(V^0	PP_2)	VP	sisters	$V^0 < {D_2}^0$
(DP_2	PP_1)	PP_2	not sisters	N/A

Table 5.5: CP_2 relation

Note that the ordering statement $C^0 < P^0$ is not required because despite the fact that (i) $P^0 < C^0$, (ii) C is a head, and (iii) TP_2 is the sister of C^0, TP_2 does not fully dominate P^0 (there is a path of P^0 that does not go through PP_2 - see (59b-i)). The same logic applies to the reason $T^0 < P^0$, $V^0 < P^0$, as well as $C^0 < {D_1}^0$, $T^0 < {D_1}^0$, $V^0 < {D_1}^0$, are not required.

In addition, $D_3{}^0 < P^0$ is not required because despite the fact that (i) the set has $P^0 < D_3{}^0$, (ii) DP_3 is a specifier, and (iii) DP_3 dominates $D_3{}^0$, (iv) DP_3 and TP_1 are sisters, TP_1 does not fully dominate P^0 (i.e., there is a path for P^0 that does not go through TP_1 - see (59b-i)). The same logic applies to the reason $D_3{}^0 < D_1{}^0$ are not required.

Furthermore, $D_2{}^0 < P^0$ is not required because despite the fact that (i) the set has $P^0 < D_2{}^0$, (ii) DP_2 is a specifier, (iii) DP_2 dominates $D_2{}^0$, and (iv) PP_1 fully dominates P^0, DP_2 and PP_1 are not sisters (i.e., there are paths for PP_1 that does not go through PP_2 - see (59a-i)). The same logic applies to the reason $D_2{}^0 < D_1{}^0$ is not required.

(59) a. $p(PP_1)$ =

(i) (CP_2)

(ii) $(PP_2, VP, TP_1, TP_2, CP_1, CP_2)$

b. $p(P^0)$ =

(i) (PP_1, CP_2)

(ii) $(PP_1, PP_2, VP, TP_1, TP_2, CP_1, CP_2)$

c. $p(D_1{}^0)$ =

(i) (DP_1, PP_1, CP_2)

(ii) $(DP_1, PP_1, PP_2, VP, TP_1, TP_2, CP_1, CP_2)$

Next, take the union of the set of ordered pairs from the VP Spell-out domain and the one from the CP Spell-out domain, both of which are repeated below in (60). The union set is shown in (61).

(60)

a. $$\begin{Bmatrix} V^0 < D_2{}^0 & \mathbf{D_2{}^0 < P^0} & P^0 < D_1{}^0 \\ \mathbf{V^0 < P^0} & \mathbf{D_2{}^0 < D_1{}^0} & \\ \mathbf{V^0 < D_1{}^0} & & \end{Bmatrix}$$

b.

$$\left\{\begin{array}{llllll}
P^0 < {D_1}^0 & {D_1}^0 < C^0 & C^0 < {D_3}^0 & {D_3}^0 < T^0 & T^0 < V^0 & V^0 < {D_2}^0 \\
P^0 < C^0 & {D_1}^0 < {D_3}^0 & C^0 < T^0 & {D_3}^0 < V^0 & T^0 < {D_2}^0 & \\
P^0 < {D_3}^0 & {D_1}^0 < T^0 & C^0 < V^0 & {D_3}^0 < {D_2}^0 & & \\
P^0 < T^0 & \mathbf{{D_1}^0} < \mathbf{V^0} & C^0 < {D_2}^0 & & & \\
\mathbf{P^0} < \mathbf{V^0} & \mathbf{{D_1}^0} < \mathbf{{D_2}^0} & & & & \\
\mathbf{P^0} < \mathbf{{D_2}^0} & & & & &
\end{array}\right\}$$

(61)

$$\left\{\begin{array}{lllllll}
P^0 < {D_1}^0 & {D_1}^0 < C^0 & C^0 < {D_3}^0 & {D_3}^0 < T^0 & T^0 < V^0 & V^0 < {D_2}^0 & \mathbf{{D_2}^0} < \mathbf{P^0} \\
P^0 < C^0 & {D_1}^0 < {D_3}^0 & C^0 < T^0 & {D_3}^0 < V^0 & T^0 < {D_2}^0 & \mathbf{V^0} < \mathbf{P^0} & \mathbf{{D_2}^0} < \mathbf{{D_1}^0} \\
P^0 < {D_3}^0 & {D_1}^0 < T^0 & C^0 < V^0 & {D_3}^0 < {D_2}^0 & & \mathbf{V^0} < \mathbf{{D_1}^0} & \\
P^0 < T^0 & \mathbf{{D_1}^0} < \mathbf{V^0} & C^0 < {D_2}^0 & & & & \\
\mathbf{P^0} < \mathbf{V^0} & \mathbf{{D_1}^0} < \mathbf{{D_2}^0} & & & & & \\
\mathbf{P^0} < \mathbf{{D_2}^0} & & & & & &
\end{array}\right\}$$

Note that the union set satisfies all the constraints except for the Asymmetry Constraint (e.g., the set has both $V^0 < P^0$ and $P^0 < V^0$). It satisfies the Totality Constraint because all the Maximal X^0 Nodes in the most recent Spell-out domain, namely, the CP Spell-out domain, are ordered relative to each other. It satisfies the Anti-reflexivity Constraint because no node is ordered relative to itself. It satisfies the Language Specific Constraints because all the required ordering statements are in the union set. Note that based on the Language Specific Constraints, the existence of $V^0 < P^0$ does not require the existence of any ordering statement: despite the fact that (i) the set has $V^0 < P^0$, (ii) V^0 is a head, and (iii) PP_2 is the sister of V^0, PP_2 does not fully dominate P^0, so $V^0 < P^0$ is not required and at the same time nothing prohibits the existence of $V^0 < P^0$ based on the Language Specific Constraints (though having both $V^0 < P^0$ and $P^0 < V^0$ violates the Asymmetry Constraint) — the Language Specific Constraints are about the conditions under which existing ordering statements require another ordering statement but not about preventing certain ordering statements. Similarly, $D_2^0 < P^0$, as well as $V^0 < D_1^0$ and $D_2^0 < D_1^0$ do not require the existence of any ordering statement either.

Since the final Spell-out domain has been reached, apply Ordering Deletion. Since no ordering statement fits the criteria in Ordering Deletion, nothing is deleted. Then, apply the Set-to-String algorithm to the union set of CP and VP, which is the set in (61). First, implement Stage 1 in the Set-to-String algorithm (see below for the content of Stage 1).

Stage 1:

1. **Find the intersection sets:** For each distinct node n_1, n_2, n_3, ... in the set of ordered pairs O = $\{n_1 < n_3, n_3 < n_2, ...\}$, get the set X = $\{..., n_1, ...\}$ that contains all the nodes that precedes n and the set Y = $\{..., n_2, ...\}$ that contains all the nodes that follows n, and get the set I = X $\cap$ Y, which is the intersection of set X and set Y.

2. **Check the intersection sets:** If all the intersection sets are empty, run the String Forming Mechanism, which is defined in (35) below, on set O and the set will be turned into a string. **Otherwise, proceed to Stage 2.**

It turns out that there are non-empty intersection sets, which are shown in (62). Thus, the algorithm goes to Stage 2.

(62) a. $\{P^0, D_1^0\}$

b. $\{V^0, D_2^0\}$

Stage 2:

1. **Collect all the non-empty intersection sets:** Form a set M = $\{I_1, I_2,...\}$ that contains all the non-empty intersection sets.

2. **Try forming a string:**

   (a) **No qualified constituent:** For all given intersection sets $I_1 = \{n_1, n_2, ...\}$, $I_1 = \{n_3, ...\}$, ... in set M, if none of them forms a qualified constituent, the linearization process crashes.

(b) **Qualified constituent:** If there exists intersection set I^* that forms a qualified constituent, **proceed to Stage 3.**

In Stage 2, the algorithm checks whether there exists an intersection set that can form a qualified constituent. In this case, {V^0, $D_2{}^0$} does not form a constituent; {P^0, $D_1{}^0$} forms a constituent but it is not a qualified constituent in English. So, none of the non-empty intersection sets can form a qualified constituent. As a result, the linearization process crashes.

As a quick summary, the non successive-cyclic derivation results in the final union set violating the Asymmetry Constraint, which cannot be resolved by the Set-to-String algorithm, which eventually makes the linearization process crash.

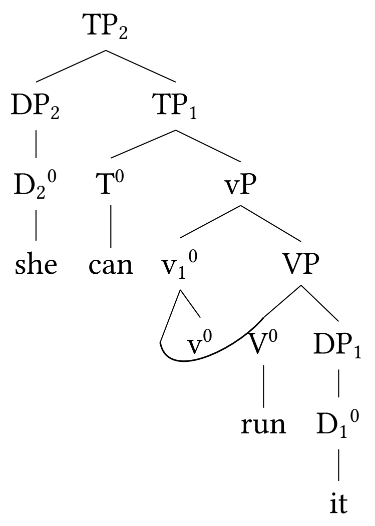
5.2.2 Ordering Deletion

Lastly, I discuss the example in (21) (*She can run it.*) to illustrate how the Ordering Deletion rule (63) works. The structure of it is shown in (64).

(63) **Ordering Deletion**

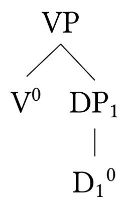
For a given set of ordering statements, $Y < \alpha$ must be deleted if X dominates Y and $X < \alpha$, and $\alpha < Y$ must be deleted if X dominates Y and $\alpha < X$; unless there exists β such that $\beta < X$ and $Y < \beta$, or $X < \beta$ and $\beta < Y$.

(64)



The set of ordered pairs that satisfies all the constraints in the VP Spell-out domain is shown in (66).

(65)



(66) $\{V^0 < D_1{}^0\}$

The Maximal X^0 Nodes in TP is shown in (67).

(67) Maximal Nodes of TP = $\{D_2{}^0, T^0, v_1{}^0, D_1{}^0\}$

The set of ordered pairs that satisfies all the constraints in the TP Spell-out domain is shown in (68).

(68)
$$\left\{\begin{array}{lll} D_2{}^0 < T^0 & T^0 < v_1{}^0 & v_1{}^0 < D_1{}^0 \\ D_2{}^0 < v_1{}^0 & T^0 < D_1{}^0 & \\ D_2{}^0 < D_1{}^0 & & \end{array}\right\}$$

The union set of VP and TP is shown in (69).

(69)
$$\left\{\begin{array}{llll} {D_2}^0 < T^0 & T^0 < {v_1}^0 & {v_1}^0 < {D_1}^0 & \mathbf{V^0} < \mathbf{{D_1}^0} \\ {D_2}^0 < {v_1}^0 & T^0 < {D_1}^0 & & \\ {D_2}^0 < {D_1}^0 & & & \end{array}\right\}$$

Since the final union is reached, Ordering Deletion, which is repeated below as (70), should apply.

3. **Repeat until the last union:**

   Repeat the cyclic process steps. Once the last union set is formed, apply Ordering Deletion and instead of implementing step III and IV, apply the Set-to-String algorithm.

(70) **Ordering Deletion**

For a given set of ordering statements, Y $< \alpha$ must be deleted if X dominates Y and X $< \alpha$, and $\alpha <$ Y must be deleted if X dominates Y and $\alpha <$ X; unless there exists β such that $\beta <$ X and Y $< \beta$, or X $< \beta$ and $\beta <$ Y.

Based on Ordering Deletion, $V^0 < {D_1}^0$ should be deleted because (i) ${v_1}^0$ dominates V^0, (ii) ${v_1}^0 < {D_1}^0$, and (iii) there does not exist β such that $\beta < {v_1}^0$ and $V^0 < \beta$, or ${v_1}^0 < \beta$ and $\beta < V^0$.

After applying Ordering Deletion to the union set, it is updated as (71). Note that this updated union set is the same as the set of ordered pairs in the TP Spell-out domain.

(71)
$$\left\{\begin{array}{lll} {D_2}^0 < T^0 & T^0 < {v_1}^0 & {v_1}^0 < {D_1}^0 \\ {D_2}^0 < {v_1}^0 & T^0 < {D_1}^0 & \\ {D_2}^0 < {D_1}^0 & & \end{array}\right\}$$

Once the final union set is updated, the Set-to-String algorithm should be implemented. One will get the string $<_S {D_2}^0 T^0 {v_1}^0 {D_1}^0 >$. The lexical insertion sites are shown in (72). Note that ${v_1}^0$ has the complex form of $\#V^0v^0\#$ and I assume that the ordering between V^0

and v^0 is decided by morphology, not the linearization process.

(72) a. ${D_2}^0 \sim \#{D_2}^0\#$

b. $T^0 \sim \#T^0\#$

c. ${v_1}^0 \sim \#V^0v^0\#$

d. ${D_1}^0 \sim \#{D_1}^0\#$

Now, implement lexical insertion, which is shown in (73).

(73) a. ${D_2}^0$ = 'she'

b. T^0 = 'can'

c. ${v_1}^0$ = 'run-$\varnothing$'

d. ${D_1}^0$ = 'it'

Thus, the final utterance is *She can run it.*

In the following, I discuss two questions: (i) Why is Ordering Deletion necessary in the linearization process? (ii) How to understand Ordering Deletion, especially regarding Fox and Pesetsky (2005)'s Order Preservation?

(74) **Order Preservation**

Information about linearization, once established at the end of a given Spell-out domain, is never deleted in the course of a derivation. The sole function of Spell-out is to add information.

Fox and Pesetsky (2005)

I start with the first question. Note that in the union set in (69), if one orders the set via the String Forming Mechanism, ${D_2}^0$ will be made the first element of the string; then, T^0 will be made the next element of the string. Now, the set is left with the ordering statements ${v_1}^0 < {D_1}^0$ and $V^0 < {D_1}^0$. Note that now, both ${v_1}^0$ and V^0 are not preceded by any other nodes, but there is no statement about the ordering relation between ${v_1}^0$ and V^0. Thus, the set in (69) cannot be linearized.[3]

The fact that the set in (69) cannot be linearized is due to the interaction between cyclic linearization and Maximal X^0 Nodes in the current linearization process. To be more specific, in the VP Spell-out domain, V^0 is one of the Maximal X^0 Nodes. However, in the TP Spell-out domain, V^0 is merged with v^0 and dominated by ${v_1}^0$. Thus, V^0 is no longer a Maximal X^0 Node in the TP Spell-out domain. However, since cyclic linearization collects ordering statements in each Spell-out domain, the ordering statement involving V^0 from the VP Spell-out domain is still collected and becomes part of the union set of VP and TP. Note that without the ordering statement $V^0 < {D_1}^0$, the set can be linearized. In this sense, the Ordering Deletion rule helps getting rid of the problematic ordering statement $V^0 < {D_1}^0$ so that the set can be linearized by the String Forming Mechanism.

Regarding the second question, Ordering Deletion gets rid of the ordering statement collected, which seems to be contradictory to Ordering Preservation. However, the ordering information about V^0 and ${D_1}^0$ is still preserved in the sense that in the TP Spell-out domain, V^0 becomes part of ${v_1}^0$ and thus, the position of V^0 is now represented by ${v_1}^0$; since ${v_1}^0 < {D_1}^0$ is in the union set, the information that V^0 preceding ${D_1}^0$ can be represented by the mother of V^0, namely, ${v_1}^0$, preceding ${D_1}^0$.[4]

5.3 Summary

In this chapter, I introduced the cyclic linearization process and showed how it works on real examples. Instead of linearizing the structure as a whole, the cyclic linearization process linearizes the structure in a cyclic way, and the set of ordered pairs that gets linearized by the Set-to-String algorithm ends up being the union set of all the Spell-out domains. In the next chapter, I will look at the V(P)-doubling cases and show how this

[3]Another way to view the problem is this set only shows that V^0 should be ordered before ${D_1}^0$, but no information about how V^0 should be ordered relative to the other nodes is provided.

[4]Another approach to make the linearization process coherent is to revise Fox and Pesetsky (2005)'s Order Preservation such that only information about Maximal X^0 Nodes in the final Spell-out domain must be preserved.

cyclic linearization process works for these cases.

CHAPTER 6

Analyzing V(P)-doubling constructions

In this chapter, I discuss the linearization of verb-doubling cases in Hebrew and Yiddish, which are shown in (1) and (2), and the VP-doubling case in Mandarin, which is shown in (3). Before going into the details of how the linearization process works on these cases, I would like to point out that the reasons why the verbs are doubled vary among these cases, and are related to how the V(P)s move.

(1) **Verb-doubling (with verb-fronting)**

a. **Hebrew (infinitival)**

Li**kn**o**t**, hi **k**a**nt**a et ha-praxim
buy.INF she **buy**.PST ACC the-flowers
'As for buying, she bought the flowers.'

b. **Yiddish (aspectual form)**

Gegessen, hot Maks **gegessen** fish
eaten has Max **eaten** fish
'As for having eaten, Max has eaten fish.'

(2) **Verb-doubling (with VP-fronting)**

Hebrew (infinitival form)

Li**sht**o**f** maher et ha-kelim, hu **sh**a**t**a**f**
to.**wash** quickly ACC the-dishes he **wash**.PST
'As for washing the dishes quickly, he washed.'

(3) **VP-doubling**

Mandarin (aspectual form)

Chi-guo	**bale**,	Lili	dique	mei	**chi-guo bale**
eat-ASP	**Guava**	Lili	indeed	not	**eat**-ASP **Guava**

'As for having eaten Guava before, Lili indeed hasn't eaten Guava before.'

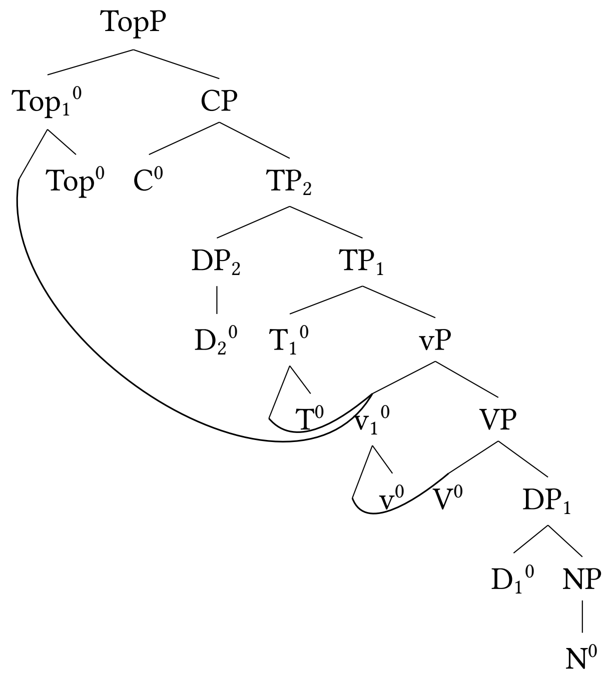
6.1 V-doubling with verb fronting

I start with the verb-doubling case in Hebrew, which is repeated below as (4). The structure of which is shown in (5). Note that I will ignore the linearization of the accusative marker.

(4) **Hebrew (infinitival)**

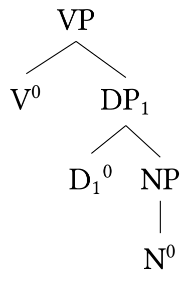
Li**kn**o**t**,	hi	**ka**n**t**a	et	ha-praxim
buy.INF	she	**buy**.PST	ACC	the-flowers

'As for buying, she bought the flowers.'

(5)


There are three Spell-out domains in (5): VP, CP and TopP. The structure of the VP Spell-out domain is shown in (6).

(6)



The Maximal X^0 Nodes of VP are shown in (7).

(7) Maximal X^0 Nodes of VP = $\{V^0, {D_1}^0, N^0\}$

Table 6.1 summarizes all the sisterhood relationship and the required orders by the Language Specific Constraints for all Maximal X^0 Nodes α and β of VP if α and β are ordered as a pair in a given set.

		relationship	**ordering statements**
V^0	DP_1	sisters	$\{V^0 < {D_1}^0, V^0 < N^0\}$
${D_1}^0$	NP	sisters	$\{{D_1}^0 < N^0\}$

Table 6.1: VP Spell-out domain

The relevant paths and full-dominance relationship are shown in (8) and (9).

(8)
a. $p(V^0)$ = (VP)
b. $p(DP_1)$ = (VP)
c. p(NP) = (DP_1, VP)
d. $p({D_1}^0)$ = (DP_1, VP)
e. $p(N^0)$ = (NP, DP_1, VP)

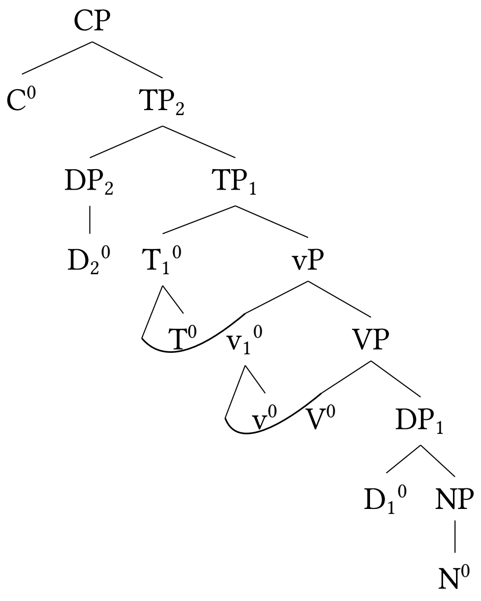
(9)
a. DP_1 fully dominates ${D_1}^0$ and N^0
b. NP fully dominates N^0

The following set in (10) for the VP Spell-out domain satisfies all the constraints.

(10) $$\begin{Bmatrix} V^0 < {D_1}^0 & {D_1}^0 < N^0 \\ V^0 < N^0 & \end{Bmatrix}$$

Now, let's look at the CP Spell-out domain. The structure of the CP Spell-out domain is shown in (11).

(11)



The Maximal X^0 Nodes of CP is shown in (12). Table 6.2 shows the sisterhood relationship and the required orders by the Language Specific Constraints for all Maximal X^0 Nodes α and β of VP if α and β are ordered as a pair in a given set.

(12) Maximal X^0 Nodes of CP = $\{C^0, {D_2}^0, {T_1}^0, {D_1}^0, N^0\}$

		relationship	**ordering statements**
C^0	TP_2	sisters	$\{C^0 < {D_2}^0, C^0 < {T_1}^0, C^0 < {D_1}^0, C^0 < N^0\}$
DP_2	TP_1	sisters	$\{{D_2}^0 < {T_1}^0, {D_2}^0 < {D_1}^0, {D_2}^0 < N^0\}$
${T_1}^0$	vP	sisters	$\{{T_1}^0 < {D_1}^0, {T_1}^0 < N^0\}$
${D_1}^0$	NP	sisters	$\{{D_1}^0 < N^0\}$

Table 6.2: CP Spell-out domain

The relevant paths and full-dominance relationship are shown in (13) and (14).

(13) a. $p(C^0) = (CP)$

b. $p(TP_2) = (CP)$

c. $p(DP_2) = (TP_2, CP)$

d. $p(TP_1) = (TP_2, CP)$

e. $p({T_1}^0) = (TP_1, TP_2, CP)$

f. $p(vP) = (TP_1, TP_2, CP)$

g. $p({D_1}^0) = (DP_1, VP, vP, TP_1, TP_2, CP)$

h. $p(N^0) = (NP, DP_1, VP, vP, TP_1, TP_2, CP)$

i. $p({D_2}^0) = (DP_2, TP_2, CP)$

(14) a. CP fully dominates C^0 and TP_2.

b. TP_2 fully dominates DP_2, TP_1, ${D_2}^0$, ${T_1}^0$, ${D_1}^0$ and N^0.

c. TP_1 fully dominates vP, ${T_1}^0$, ${D_1}^0$ and N^0.

d. vP_1 fully dominates ${D_1}^0$ and N^0.

e. DP_1 fully dominates ${D_1}^0$ and N^0.

f. NP_1 fully dominates N^0.

The following set in (15) for the CP Spell-out domain satisfies all the constraints.

(15)

$$\left\{\begin{array}{llll} C^0 < {D_2}^0 & {D_2}^0 < {T_1}^0 & {T_1}^0 < {D_1}^0 & {D_1}^0 < N^0 \\ C^0 < {T_1}^0 & {D_2}^0 < {D_1}^0 & {T_1}^0 < N^0 & \\ C^0 < {D_1}^0 & {D_2}^0 < N^0 & & \\ C^0 < N^0 & & & \end{array}\right\}$$

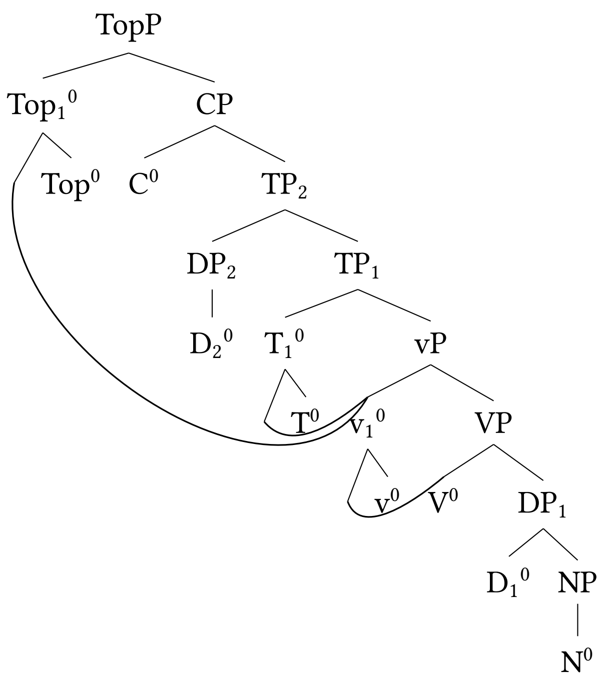
The union set of CP and VP is shown in (16) and it satisfies all the constraints.

(16)

$$\left\{\begin{array}{lllll} C^0 < {D_2}^0 & {D_2}^0 < {T_1}^0 & {T_1}^0 < {D_1}^0 & {D_1}^0 < N^0 & \mathbf{V^0} < \mathbf{{D_1}^0} \\ C^0 < {T_1}^0 & {D_2}^0 < {D_1}^0 & {T_1}^0 < N^0 & & \mathbf{V^0} < \mathbf{N^0} \\ C^0 < {D_1}^0 & {D_2}^0 < N^0 & & & \\ C^0 < N^0 & & & & \end{array}\right\}$$

Before going into the next Spell-out domain, I discuss why the union set of CP and VP in (16) satisfies all the constraints. This union set satisfies the Totality Constraint since (i) the Totality Constraint is evaluated against the most recent Spell-out domain, which is the CP Spell-out domain, and (ii) all the Maximal X^0 Nodes in the CP Spell-out domain is ordered relative to each other in the union set of CP and VP. It satisfies the Anti-reflexivity Constraint since none of the Maximal X^0 Nodes is ordered relative to itself. It satisfies the Anti-symmetry constraint since there is no ordering contradiction. It satisfies the Language Specific Constraint because no required order is missing. Note that the existence of $V^0 < {D_1}^0$ does not require the existence of any ordering statements: despite the fact that V^0 is a head and DP_1 fully dominates ${D_1}^0$, DP_1 is not the sister of V^0 (i.e., the immediate dominating mother VP does not fully dominate V^0), and thus, no ordering statement is required by the Language Specific Constraint. At the same time, the Language Specific Constraint does not prevent $V^0 < {D_1}^0$ from being in the set, either. The same logic also applies to $V^0 < N^0$.

Now, let's look at the TopP domain, the structure of which is shown in (17).

(17)


In the TopP Spell-out domain, the Maximal X^0 Nodes of TopP are shown in (18).

(18) Maximal X^0 Nodes of TopP = {Top$_1{}^0$, C^0, D$_2{}^0$, T$_1{}^0$, D$_1{}^0$, N^0}

Table 6.3 shows the sisterhood relationship and the required orders by the Language Specific Constraints for all Maximal X^0 Nodes α and β of VP if α and β are ordered as a pair in a given set.

		relationship	**ordering statements**
Top$_1{}^0$	CP	sisters	{Top$_1{}^0$ $<$ C^0, Top$_1{}^0$ $<$ D$_2{}^0$, Top$_1{}^0$ $<$ T$_1{}^0$, Top$_1{}^0$ $<$ D$_1{}^0$, Top$_1{}^0$ $<$ N^0 }
C^0	TP$_2$	sisters	{C^0 $<$ D$_2{}^0$, C^0 $<$ T$_1{}^0$, C^0 $<$ D$_1{}^0$, C^0 $<$ N^0 }
DP$_2$	TP$_1$	sisters	{D$_2{}^0$ $<$ T$_1{}^0$, D$_2{}^0$ $<$ D$_1{}^0$, D$_2{}^0$ $<$ N^0 }
T$_1{}^0$	vP	sisters	{T$_1{}^0$ $<$ D$_1{}^0$, T$_1{}^0$ $<$ N^0 }
D$_1{}^0$	NP	non-sisters	{D$_1{}^0$ $<$ N^0 }

Table 6.3: TopP Spell-out domain

The relevant paths and full-dominance relationship are shown in (19) and (20).

(19) a. $Top_1{}^0$ = (TopP)

b. CP = (TopP)

c. $p(C^0)$ = (CP, TopP)

d. TP_2 = (CP, TopP)

e. TP_1 = (TP_2, CP, TopP)

f. $p(D_2{}^0)$ = (DP_2, TP_2, CP, TopP)

g. $p(T_1{}^0)$ = (TP_1, TP_2, CP, TopP)

h. $p(D_1{}^0)$ = (DP_1, VP, vP, TP_1, TP_2, CP, TopP)

i. $p(N^0)$ = (NP, DP_1, VP, vP, TP_1, TP_2, CP, TopP)

(20) a. TopP fully dominates $Top_1{}^0$ and CP.

b. CP fully dominates C^0, TP_2, $D_2{}^0$, $T_1{}^0$, $D_1{}^0$ and N^0.

c. TP_2 fully dominates $D_2{}^0$, $T_1{}^0$, $D_1{}^0$ and N^0.

d. TP_1 fully dominates $T_1{}^0$, $D_1{}^0$ and N^0.

e. vP fully dominates $D_1{}^0$ and N^0.

f. DP_1 $D_1{}^0$ and N^0.

The following set in (21) for the TopP Spell-out domain satisfies all the constraints.

(21)

$$\begin{Bmatrix} Top_1{}^0 < C^0 & C^0 < D_1{}^0 & D_2{}^0 < T_1{}^0 & T_1{}^0 < D_1{}^0 & D_1{}^0 < N^0 \\ Top_1{}^0 < D_2{}^0 & C^0 < T_1{}^0 & D_2{}^0 < D_1{}^0 & T_1{}^0 < N^0 & \\ Top_1{}^0 < T_1{}^0 & C^0 < D_2{}^0 & D_2{}^0 < N^0 & & \\ Top_1{}^0 < D_1{}^0 & C^0 < N^0 & & & \\ Top_1{}^0 < N^0 & & & & \end{Bmatrix}$$

The union set of TopP, CP and VP is shown in (22).

(22)

$$\left\{\begin{array}{llllll} Top_1{}^0 < C^0 & C^0 < D_1{}^0 & D_2{}^0 < T_1{}^0 & T_1{}^0 < D_1{}^0 & D_1{}^0 < N^0 & \mathbf{V^0} < \mathbf{D_2{}^0} \\ Top_1{}^0 < D_2{}^0 & C^0 < T_1{}^0 & D_2{}^0 < D_1{}^0 & T_1{}^0 < N^0 & & \mathbf{V^0} < \mathbf{N^0} \\ Top_1{}^0 < T_1{}^0 & C^0 < D_2{}^0 & D_2{}^0 < N^0 & & & \\ Top_1{}^0 < D_1{}^0 & C^0 < N^0 & & & & \\ Top_1{}^0 < N^0 & & & & & \end{array}\right\}$$

Since the final union set is formed, Ordering Deletion should be applied, which is repeated below in (23). After Ordering Deletion, the union set becomes the one in (24). Note that $V^0 < D_2{}^0$ is deleted since (i) T^0 dominates V^0 and (ii) T^0 also precedes $D_2{}^0$. The same logic applies to the reason $V^0 < N^0$ is deleted.

(23) **Ordering Deletion**

For a given set of ordering statements, Y < α must be deleted if X dominates Y and X < α, and α < Y must be deleted if X dominates Y and α < X; unless there exists β such that β < X and Y < β, or X < β and β < Y.

(24)

$$\left\{\begin{array}{lllll} Top_1{}^0 < C^0 & C^0 < D_1{}^0 & D_2{}^0 < T_1{}^0 & T_1{}^0 < D_1{}^0 & D_1{}^0 < N^0 \\ Top_1{}^0 < D_2{}^0 & C^0 < T_1{}^0 & D_2{}^0 < D_1{}^0 & T_1{}^0 < N^0 & \\ Top_1{}^0 < T_1{}^0 & C^0 < D_2{}^0 & D_2{}^0 < N^0 & & \\ Top_1{}^0 < D_1{}^0 & C^0 < N^0 & & & \\ Top_1{}^0 < N^0 & & & & \end{array}\right\}$$

Now, implement the Set-to-String algorithm, starting with Stage 1.

Stage 1:

1. **Find the intersection sets:** For each distinct node n_1, n_2, n_3, ... in the set of ordered pairs O = {$n_1 < n_3$, $n_3 < n_2$, ...}, get the set X = {..., n_1, ...} that contains all the nodes that precedes n and the set Y = {..., n_2, ...} that contains all the nodes that follows n, and get the set I = X ∩ Y, which is the intersection of set X and set Y.

2. **Check the intersection sets:** If all the intersection sets are empty, run the String Forming Mechanism on set O and the set will be turned into a string. **Otherwise, proceed to Stage 2.**

The intersection sets for all the nodes in the union set in (24) are shown in (25). Since all the intersection sets are empty, String Forming Mechanism should be applied to the union set and yields a string, and the Set-to-String algorithm stops at Stage 1.

(25)
a. Top_1^0: $\varnothing = \varnothing \cap \{C^0, D_2^0, T_1^0, D_1^0, N^0\}$
b. C^0: $\varnothing = \{Top_1^0\} \cap \{D_2^0, T_1^0, D_1^0, N^0\}$
c. D_2^0: $\varnothing = \{Top_1^0, C^0\} \cap \{T_1^0, D_1^0, N^0\}$
d. T_1^0: $\varnothing = \{Top_1^0, C^0, D_2^0\} \cap \{D_1^0, N^0\}$
e. D_1^0: $\varnothing = \{Top_1^0, C^0, D_2^0, T_1^0\} \cap \{N^0\}$
f. N^0: $\varnothing = \{Top_1^0, C^0, D_2^0, T_1^0, D_1^0\} \cap \varnothing$

After applying the String Forming Mechanism on the union set, the string $<_S C^0D_2^0T_1^0D_1^0N^0 >$ is generated.

Next, form lexical insertion sites, which are shown in (26).

(26) a. $\text{Top}_1{}^0 \sim \#\text{V}^0\text{v}^0\text{Top}^0\#$

b. $\text{C}^0 \sim \#\text{C}^0\#$

c. $\text{D}_2{}^0 \sim \#\text{D}_2{}^0\#$

d. $\text{T}_1{}^0 \sim \#\text{V}^0\text{v}^0\text{T}^0\#$

e. $\text{D}_1{}^0 \sim \#\text{D}_1{}^0\#$

f. $\text{N}^0 \sim \#\text{N}^0\#$

The final step is to implement lexical insertion, which is shown in (27).

(27) a. $\#\text{V}^0\text{v}^0\text{Top}^0\# =$ *liknot* 'to buy'

b. $\text{C}^0 = \varnothing$

c. $\text{D}_1{}^0 =$ *hi* 'she'

d. $\text{T}_1{}^0 =$ *kanta* 'bought'

e. $\text{D}_2{}^0 =$ *ha* 'the'

f. $\text{N}^0 =$ *praxim* 'flowers'

The string is updated as $<_S$ liknot hi kanta et ha-praxim $>$, which is repeated below as (28).

(28) **Hebrew (infinitival)**

Li**kn**o**t**,	hi	**k**a**nt**a	et	ha-praxim
buy.INF	she	**buy**.PST	ACC	the-flowers

'As for buying, she bought the flowers.'

In the following, I discuss the Yiddish verb-doubling case, where the verb upstairs has an aspectual form. the example is repeated below in (29), the structure of which is shown in (30).[1]

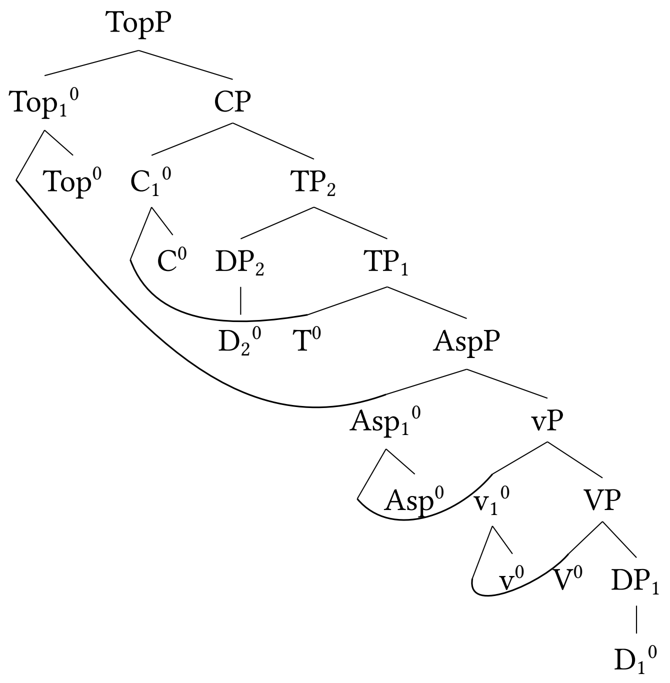
[1]The auxiliary verb should have an extra VP layer, but for ease of presentation, I ignore this VP layer, which will not affect the analysis.

(29) **Yiddish (aspectual form)**

Gegessen,	hot Maks	**gegessen**	fish
eaten	has Max	**eaten**	fish

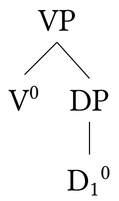
‘As for having eaten, Max has eaten fish.’

(30)



There are three Spell-out domains in (30): VP, CP and TopP. The structure of the VP Spell-out domain is shown in (31) (note that for ease of presentation, I simplify the structure of the DPs).

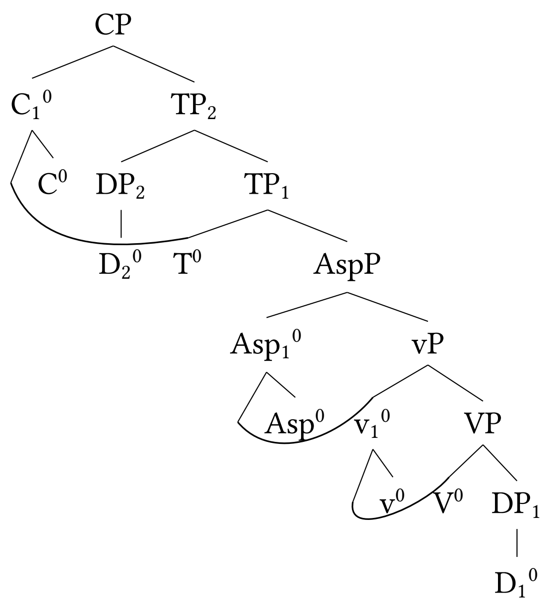
(31)



The following set in (32) for the VP Spell-out domain satisfies all the constraints.

(32) $\{V^0 < D_1{}^0\}$

Now, let’s proceed to the CP Spell-out domain, the structure of which is shown in (33).

(33)


In the CP Spell-out domain, the Maximal X^0 Nodes are shown in (34).

(34) Maximal X^0 Nodes of CP = $\{C_1{}^0, D_2{}^0, Asp_1{}^0, D_1{}^0\}$

Table 6.4 shows the sisterhood relationship and and the required orders by the Language Specific Constraints for all Maximal X^0 Nodes α and β of CP if α and β are ordered as a pair in a given set.

		relationship	**ordering statements**
$C_1{}^0$	TP_2	sisters	$\{C_1{}^0 < D_2{}^0, C_1{}^0 < Asp_1{}^0, C_1{}^0 < D_1{}^0\}$
DP_2	TP_1	sisters	$\{D_2{}^0 < Asp_1{}^0, D_2{}^0 < D_1{}^0\}$
$Asp_1{}^0$	vP	sisters	$\{Asp_1{}^0 < D_1{}^0\}$

Table 6.4: CP Spell-out domain

The following set in (35) for the CP Spell-out domain satisfies all the constraints.

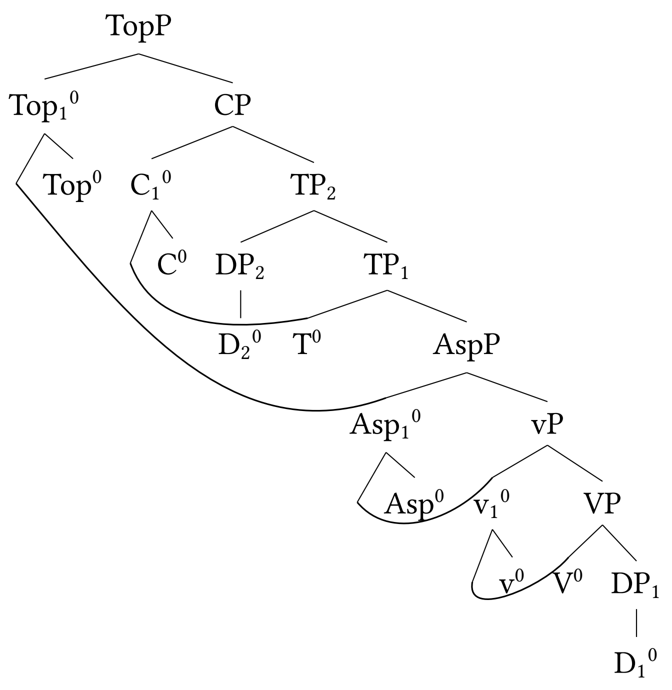
(35)
$$\left\{\begin{array}{lll} C_1{}^0 < D_2{}^0 & D_2{}^0 < Asp_1{}^0 & Asp_1{}^0 < D_1{}^0 \\ C_1{}^0 < Asp_1{}^0 & D_2{}^0 < D_1{}^0 & \\ C_1{}^0 < D_1{}^0 & & \end{array}\right\}$$

The union set of CP and VP is shown in (36).

(36)
$$\begin{Bmatrix} C_1{}^0 < D_2{}^0 & D_2{}^0 < Asp_1{}^0 & Asp_1{}^0 < D_1{}^0 & \mathbf{V^0 < D_1{}^0} \\ C_1{}^0 < Asp_1{}^0 & D_2{}^0 < D_1{}^0 & & \\ C_1{}^0 < D_1{}^0 & & & \end{Bmatrix}$$

Now, let's go to the TopP Spell-out domain. The structure of it is shown in (37).

(37)



In the TopP Spell-out domain, the Maximal X^0 Nodes are shown in (38).

(38) Maximal X^0 Nodes of TopP = {Top$_1{}^0$, C$_1{}^0$, D$_2{}^0$, D$_1{}^0$}

Table 6.5 shows the sisterhood relationship and the ordering statements allowed by the Language Specific Constraint in the TopP Spell-out domain.

		relationship	**ordering statements**
$Top_1{}^0$	CP	sisters	$\{Top_1{}^0 < C_1{}^0, Top_1{}^0 < D_2{}^0, Top_1{}^0 < D_1{}^0\}$
$C_1{}^0$	TP_2	sisters	$\{C_1{}^0 < D_2{}^0, C_1{}^0 < D_1{}^0\}$
DP_2	TP_1	sisters	$\{D_2{}^0 < D_1{}^0\}$

Table 6.5: TopP Spell-out domain

The following set in (39) for the TopP Spell-out domain satisfies all the constraints.

(39)
$$\left\{\begin{array}{lll} Top_1{}^0 < C_1{}^0 & C_1{}^0 < D_2{}^0 & D_2{}^0 < D_1{}^0 \\ Top_1{}^0 < D_2{}^0 & C_1{}^0 < D_1{}^0 & \\ Top_1{}^0 < D_1{}^0 & & \end{array}\right\}$$

The union set of TopP, CP and VP is shown in (40).

(40)
$$\left\{\begin{array}{lllll} Top_1{}^0 < C_1{}^0 & C_1{}^0 < D_2{}^0 & \mathbf{D_2{}^0 < Asp_1{}^0} & \mathbf{Asp_1{}^0 < D_1{}^0} & \mathbf{V^0 < D_1{}^0} \\ Top_1{}^0 < D_2{}^0 & \mathbf{C_1{}^0 < Asp_1{}^0} & D_2{}^0 < D_1{}^0 & & \\ Top_1{}^0 < D_1{}^0 & C_1{}^0 < D_1{}^0 & & & \end{array}\right\}$$

Since the final union set is formed, Ordering Deletion should be applied.

(41) **Ordering Deletion**

For a given set of ordering statements, $Y < \alpha$ must be deleted if X dominates Y and $X < \alpha$, and $\alpha < Y$ must be deleted if X dominates Y and $\alpha < X$; unless there exists β such that $\beta < X$ and $Y < \beta$, or $X < \beta$ and $\beta < Y$.

Since (i) $Asp_1{}^0$ dominates V^0, (ii) both $Asp_1{}^0$ and V^0 precede $D_1{}^0$, and (iii) there does not exist β such that $\beta < Asp_1{}^0$ and $V^0 < \beta$, or $Asp_1{}^0 < \beta$ and $\beta < V^0$, the ordering statement $V^0 < D_1{}^0$ should be deleted. However, despite the fact that (i) $Top_1{}^0$ dominates $Asp_1{}^0$, and (ii) they both precede $D_1{}^0$, there exists $C_1{}^0$ such that $C_1{}^0 < Asp_1{}^0$ and $Top_1{}^0 < C_1{}^0$, and thus the ordering statement $Asp_1{}^0 < D_1{}^0$ cannot be deleted. After applying Ordering Deletion, the union set is changed to the one in (42).

(42)
$$\begin{Bmatrix} Top_1{}^0 < C_1{}^0 & C_1{}^0 < D_2{}^0 & \mathbf{D_2{}^0 < Asp_1{}^0} & \mathbf{Asp_1{}^0 < D_1{}^0} \\ Top_1{}^0 < D_2{}^0 & \mathbf{C_1{}^0 < Asp_1{}^0} & D_2{}^0 < D_1{}^0 & \\ Top_1{}^0 < D_1{}^0 & C_1{}^0 < D_1{}^0 & & \end{Bmatrix}$$

Notice that both $Asp_1{}^0$ and V^0 become dominated by another node during the derivation, but only the ordering statement regarding V^0 is deleted but not the ordering statements regarding $Asp_1{}^0$. This corresponds to the condition in the Ordering Deletion rule that there cannot exist β such that $\beta <$ X and Y $< \beta$, or X $< \beta$ and $\beta <$ Y if deletion is to apply. In other words, every node preceding Y must also precede X and every node following Y must also follow X for deletion to apply. Thus, the fact that $Top_1{}^0$ and $Asp_1{}^0$ are ordered differently regarding $C_1{}^0$ prevents the ordering statements regarding $Asp_1{}^0$ to be deleted. Conceptually, deleting $C_1{}^0 < Asp_1{}^0$ will cause information loss since $C_1{}^0 < Asp_1{}^0$ cannot be represented by $Top_1{}^0 < C_1{}^0$, which is ordered differently relative to $C_1{}^0$. In other words, in this case, the position of $Asp_1{}^0$ cannot be represented by $Top_1{}^0$ since they are ordered differently relative to some nodes. This is why the verb will show up twice in the string.[2]

Next, implement the Set-to-String algorithm. Similar to the Hebrew verb-doubling case that has been previously discussed, the set does not have non-empty intersection sets (i.e., no violation of the Asymmetry Constraint) and the algorithm does not need to proceed to Stage 2 but only needs to run the String Forming Mechanism on (42), which yields the string $<_S Top_1{}^0 C_1{}^0 D_2{}^0 Asp_1{}^0 D_1{}^0 >$.

The next step is to form lexical insertion sites, which are shown in (43).

[2]In chapter 4, I mentioned that unlike Kayne (1994), my analysis does not include the "'Transitivity" constraint, and the reason is that having it does not add new information. This can be seen in (42) — there is no ordering statement $Top_1{}^0 < Asp_1{}^0$ despite the fact that the set has $Top_1{}^0 < C_1{}^0$ and $C_1{}^0 < Asp_1{}^0$. In fact, since the set already has $Top_1{}^0 < C_1{}^0$ and $C_1{}^0 < Asp_1{}^0$, it can be deduced that $Top_1{}^0$ should be linearly placed before $Asp_1{}^0$ in order to form a string. In other words, having the ordering statement $Top_1{}^0 < Asp_1{}^0$ does not add any new information, and thus, transitivity is not needed as a constraint. Note that at least for the cases discussed in this dissertation, transitivity is not needed for forming a string.

(43) a. $\text{Top}_1{}^0 \sim \#\text{V}^0\text{v}^0\text{Asp}^0\text{Top}^0\#$

b. $\text{C}_1{}^0 \sim \#\text{T}^0\text{C}^0\#$

c. $\text{D}_2{}^0 = \#\text{D}_2{}^0\#$

d. $\text{Asp}_1{}^0 \sim \#\text{V}^0\text{v}^0\text{Asp}^0\#$

e. $\text{D}_1{}^0 = \#\text{D}_2{}^0\#$

Now, implement lexical insertion, which is shown in (44).

(44) a. $\#\text{V}^0\text{v}^0\text{Asp}^0\text{Top}^0\# =$ *gegessen* 'eaten'

b. $\#\text{T}^0\text{C}^0\# =$ *hot* 'has'

c. $\text{D}_2{}^0 =$ *Maks* 'Max'

d. $\#\text{V}^0\text{v}^0\text{Asp}^0\# =$ *gegessen* 'eaten'

e. $\text{D}_1{}^0 =$ *fish* 'fish'

Thus, the string is updated as $<_S$ gegessen hot Max gegessen fish$>$. The utterance is shown in (45).

(45) **Yiddish (aspectual form)**

Gegessen,	hot	Maks	**gegessen**	fish
eaten	has	Max	**eaten**	fish

'As for having eaten, Max has eaten fish.'

As a quick summary, in Hebrew, the verb root is doubled since during the derivation, $\text{v}_1{}^0$ becomes part of $\text{T}_1{}^0$ and later becomes part of $\text{Top}_1{}^0$, which makes $\text{T}_1{}^0$ and $\text{Top}_1{}^0$ both Maximal X^0 Nodes in the TopP Spell-out domain, and they are both ordered in the linearization. The Yiddish case is a bit different from the Hebrew case in the sense that the doubling comes from the fact that the union set gathers ordering statements regarding both $\text{Top}_1{}^0$ and $\text{Asp}_1{}^0$ and the ordering statements regarding $\text{Asp}_1{}^0$ cannot be deleted (i.e., despite the fact that in the TopP Spell-out domain, $\text{Asp}_1{}^0$ is part of $\text{Top}_1{}^0$, which makes only $\text{Top}_1{}^0$ but not $\text{Asp}_1{}^0$ a Maximal X^0 Node, the Deletion Rule cannot be applied since there exists some nodes to which $\text{Asp}_1{}^0$ and $\text{Top}_1{}^0$ are ordered differently).

6.2 V-doubling with VP fronting

In the previous section, I discussed the verb-doubling examples with only the verb being fronted. In this section, I examine the Hebrew verb-doubling case with the VP being fronted in (105), which is repeated below in (46). For ease of presentation, I assume *et ha-kelim* are all together under $D_1{}^0$.

(46) **Hebrew (infinitival form)**

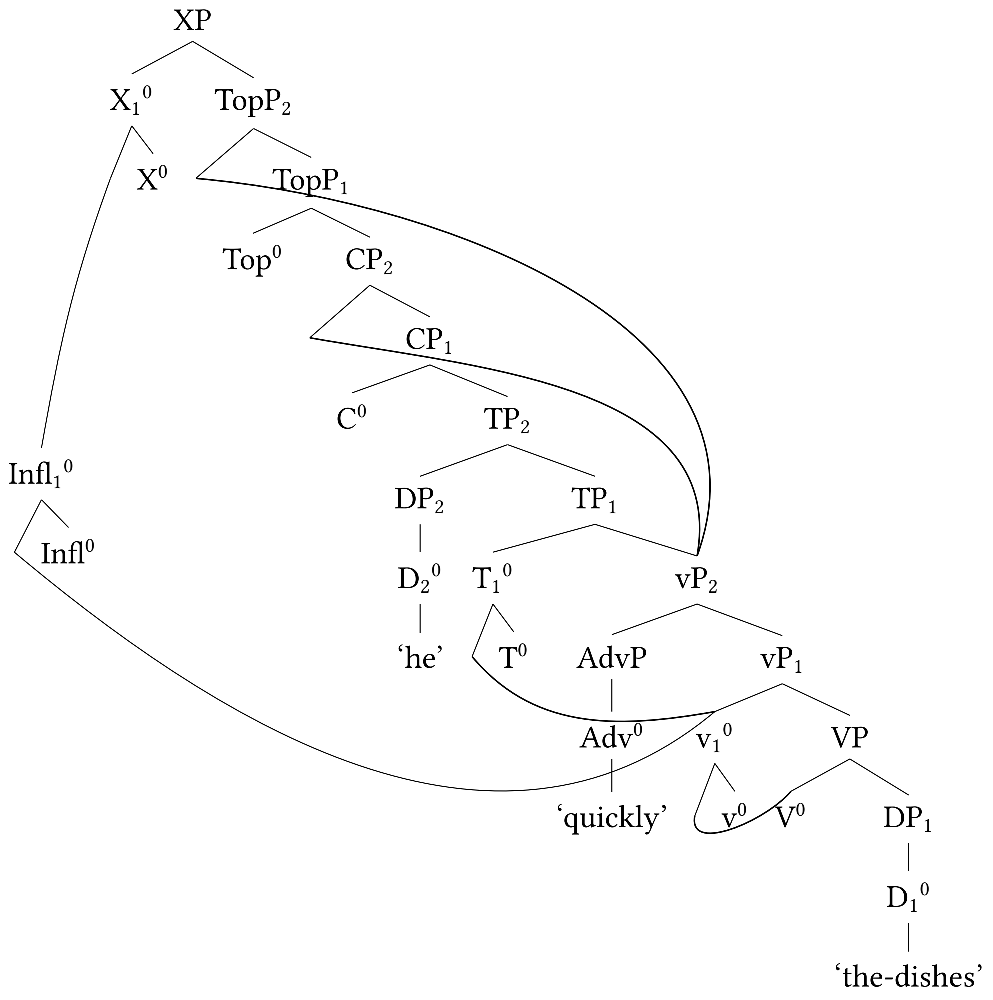
Li**sht**o**f**	maher	et	ha-kelim,	hu	**sh**a**t**a**f**
washINF	quickly	ACC	the-dishes	he	**wash**.PST

'As for washing the dishes quickly, he washed.'

The structure of (46) is shown in (47). I will provide more discussions about the key steps in (47) when discussing the XP Spell-out domain.[3]

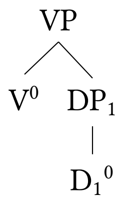
[3]Not that in (47), CP is an empty projection. However, I keep the empty CP projection here because in the next section, the empty CP layer is required in order to produce VP-doubling. To be consistent, I keep the empty CP projection in (47), too.

(47)



There are three Spell-out domains in (47): VP, CP$_2$ and XP. The structure of VP Spell-out domain is shown in (48).

(48)



In the VP Spell-out domain, the Maximal X^0 Nodes are shown in (49).

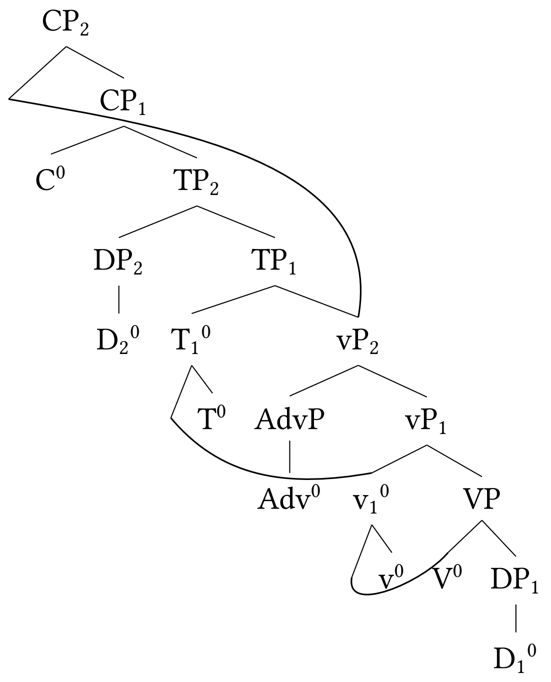
(49) Maximal X^0 Nodes of VP = $\{V^0, D_1^0\}$

The following set in (50) for the VP Spell-out domain satisfies all the constraints.

(50) $\{V^0 < D_1^0\}$

Now, let's look at the CP Spell-out domain. The structure of the CP Spell-out domain is shown in (51), where there are V-to-v-to-T movement and vP-to-Spec, CP movement.

(51)



In the CP Spell-out domain, the Maximal X^0 Nodes are shown in (52).

(52) Maximal X^0 Nodes of CP = $\{Adv^0, D_1{}^0, C^0, D_2{}^0, T_1{}^0\}$

Table 6.6 shows the sisterhood relationship and the ordering statements allowed by the Language Specific Constraint in the CP Spell-out domain.

		relationship	ordering statements
vP_2	CP_1	sisters	$\{Adv^0 < C^0, Adv^0 < {D_2}^0, Adv^0 < {T_1}^0, {D_1}^0 < C^0, {D_1}^0 < {D_2}^0, {D_1}^0 < {T_1}^0\}$
AdvP	vP_1	sisters	$\{Adv^0 < {D_1}^0\}$
C^0	TP_2	sisters	$\{C^0 < {D_2}^0, C^0 < {T_1}^0\}$
DP_2	TP_1	sisters	$\{{D_2}^0 < {T_1}^0\}$
${T_1}^0$	vP_2	non-sisters	N/A

Table 6.6: CP Spell-out domain

Note that CP_1, TP_2 and TP_1 do not fully dominate Adv^0 and ${D_1}^0$ (see (53a-i) and (53b-i)). Also, ${T_1}^0$ and vP_2 are not sisters since their immediate dominating mother TP_1 does not fully dominate vP_2 (see (53f-i)). The relevant paths are shown in (53).

(53) a. (i) $p_2(Adv^0) = \{AdvP, vP_2, CP_2\}$

(ii) $p_1(Adv^0) = \{AdvP, vP_2, TP_1, TP_2, CP_1, CP_2\}$

b. (i) $p_2({D_1}^0) = \{DP_1, VP, vP_1, vP_2, CP_2\}$

(ii) $p_1({D_1}^0) = \{DP_1, VP, vP_1, vP_2, TP_1, TP_2, CP_1, CP_2\}$

c. $p(C^0) = \{CP_1, CP_2\}$

d. $p({D_2}^0) = \{DP_2, TP_2, CP_1, CP_2\}$

e. $p({T_1}^0) = \{TP_1, TP_2, CP_1, CP_2\}$

f. (i) $p_2(vP_2) = \{CP_2\}$

(ii) $p_1(vP_2) = \{TP_1, TP_2, CP_1, CP_2\}$

The following set in (54) for the CP Spell-out domain satisfies all the constraints.

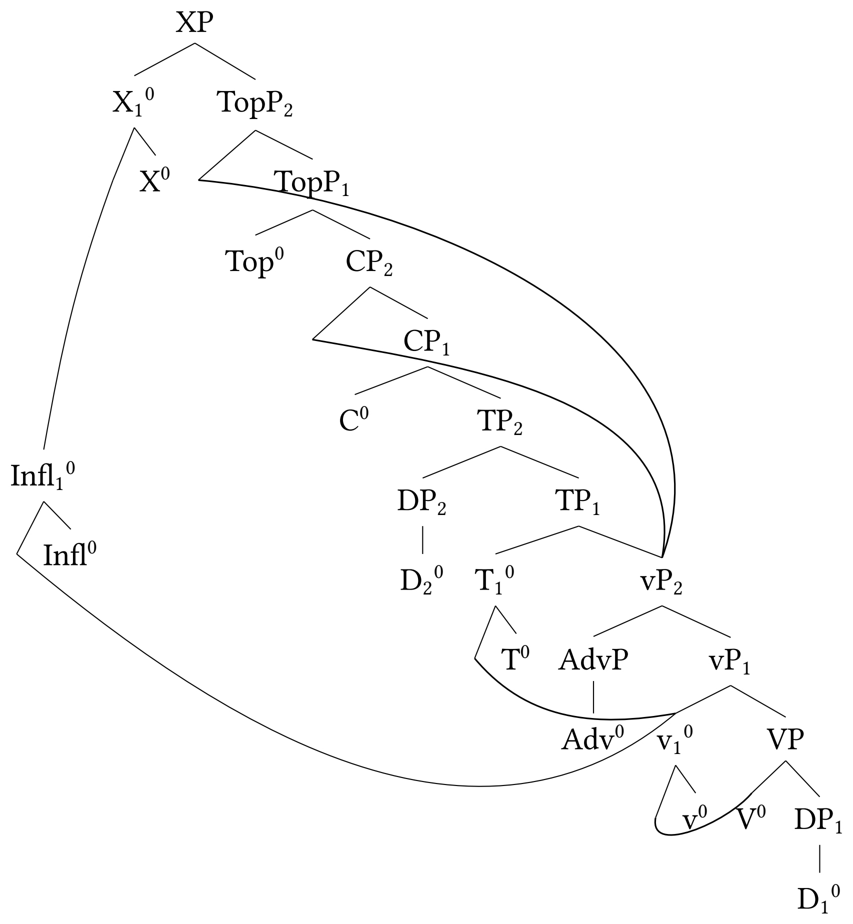
(54)
$$\left\{\begin{array}{llll} Adv^0 < {D_1}^0 & {D_1}^0 < C^0 & C^0 < {D_2}^0 & {D_2}^0 < {T_1}^0 \\ Adv^0 < C^0 & {D_1}^0 < {D_2}^0 & C^0 < {T_1}^0 & \\ Adv^0 < {D_2}^0 & {D_1}^0 < {T_1}^0 & & \\ Adv^0 < {T_1}^0 & & & \end{array}\right\}$$

The union set of CP and VP is shown in (55) and it satisfies all the constraints.

(55)
$$\left\{\begin{array}{lllll} Adv^0 < {D_1}^0 & {D_1}^0 < C^0 & C^0 < {D_2}^0 & {D_2}^0 < {T_1}^0 & \mathbf{V^0 < {D_1}^0} \\ Adv^0 < C^0 & {D_1}^0 < {D_2}^0 & C^0 < {T_1}^0 & & \\ Adv^0 < {D_2}^0 & {D_1}^0 < {T_1}^0 & & & \\ Adv^0 < {T_1}^0 & & & & \end{array}\right\}$$

Now, let's look at the XP domain, the structure of which is shown in (56).

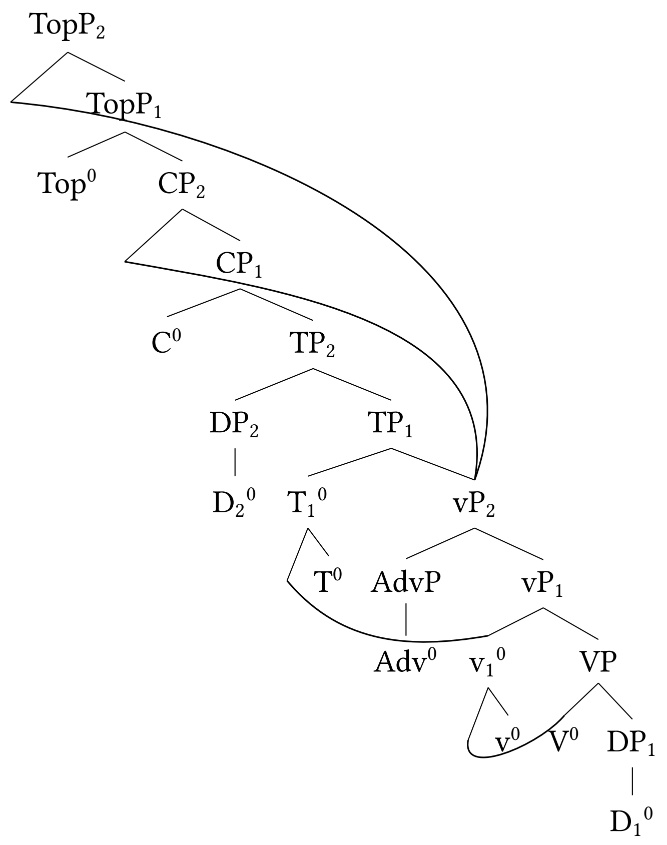
(56)



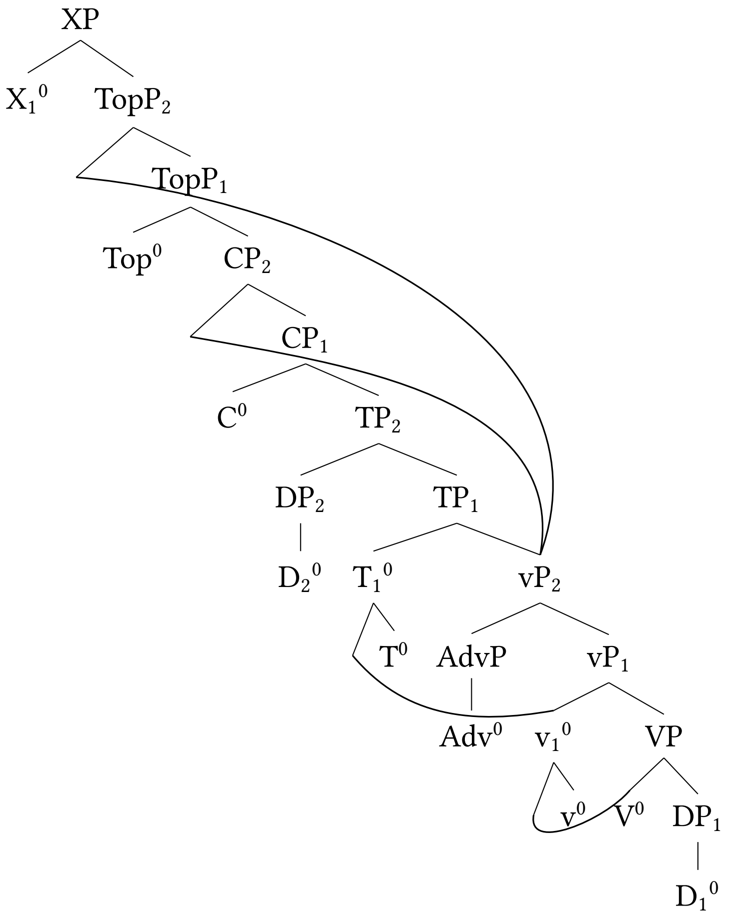
Before going in to the linearization process, I show some key steps from CP to XP below.

In (57), vP$_2$ merges with TopP$_1$, forming TopP$_2$.

(57)

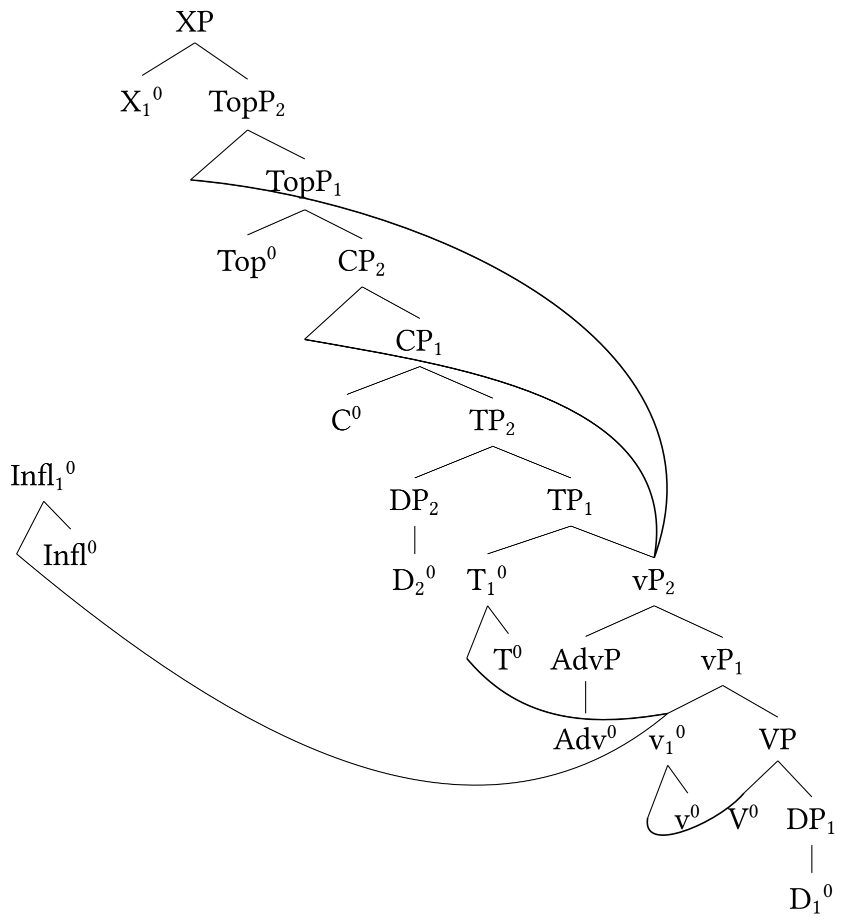


In (58), $X_1{}^0$ merges with $TopP_2$, forming XP.

(58)


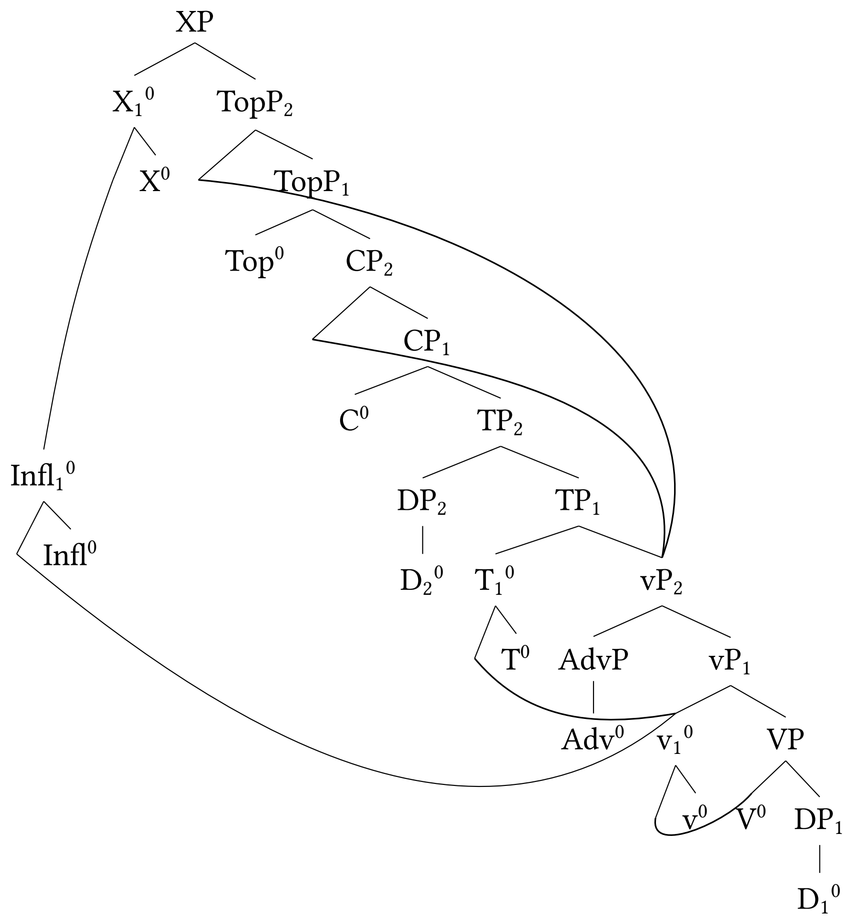
In (59), $v_1{}^0$ merges with I^0, forming $I_1{}^0$.

(59)



In (60), $\text{Infl}_1{}^0$ moves to $X_1{}^0$.

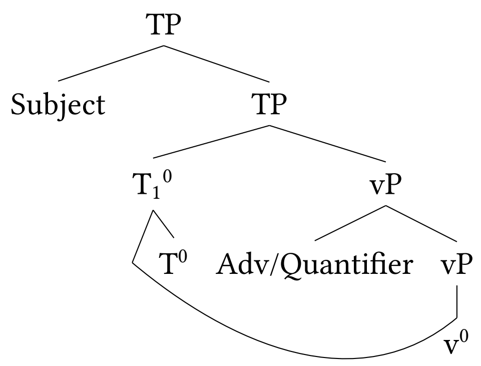
(60)

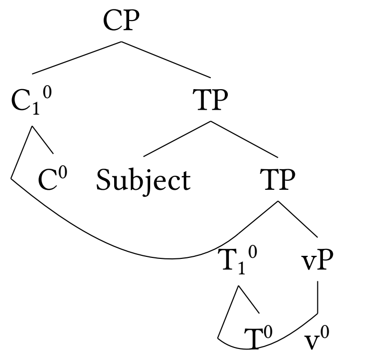


Note that after TopP_2 is built (57), an XP phrase is built and $v_1{}^0$ further moves to adjoin X^0 and ends being at the left edge.

To better understand the V-to-Infl-to-X movement, I would like to start with some observations regarding the position of the verb in Hebrew. In Pollock (1989), it is argued that if one takes the assumption that adverbs and floating quantifiers are located at the edge of VP (in the Hebrew cases, I assume that they are located at vP), if the verb ends up preceding the adverbs/quantifiers and below the subject, it has to move out of the vP, perhaps adjoining with T^0 (61); in addition, if the subject ends up being in the Spec, TP position,[4] the verb also has to move out of vP to precede the subject, perhaps adjoining with C^0 (62).

[4]Even if the subject stays inside the vP, the verb still has to move out of vP to precede the subject

(61)


(62)


Doron (1999) shows that these patterns are found in Hebrew and thus claims that Hebrew has V^0-to-T^0 movement. For instance, in (63a) and (63b), verbs appear in a position that precedes the adverb *lif'amim* 'sometimes' and the quantifier *Hebrew-move-quan* but below the subject, which is evidence showing V-to-T movement; in (63c), the verb occurs before the subject *dani* 'Dani', which is evidence showing V-to-T-to-C movement. Thus, verbs in Hebrew seem to have to move out of vP to adjoin with higher heads.

(63) a. **Adverb placement**

Dani	menasek	lif'amim	et	dina
Dani	kisses	sometimes	ACC	Dina

'Dani kisses Dina sometimes.' (Doron 1999: 126, ex.3(a))

b. **Floating quantifiers**

ha-yeladim	nisku	sneyhem	et	dina
the-children	kissed	both	ACC	Dina

'The children both kissed Dina.' (Doron 1999: 136, ex.4(a))

c. **Subject-verb Inversion**

et mi nisek dani etmol
ACC whom kissed Dani yesterday
'Who did Dani kiss yesterday?' (Doron 1999: 126, ex.2(a))

In addition, Landau (2006) makes the observation that verbs must occur at the left edge of vP in the verb doubling constructions. He discusses two contrastive examples: in (64), negation is not allowed to occur at the left edge, and in (65), the adverb *tamid* 'always' is prohibited in the initial position in the verb-doubling constructions. There are two possible explanations for the contrastive examples in (64) and (65). One possible explanation is that the verb in the fronted vP must further move to the left edge of vP, otherwise, the sentence will be ungrammatical. The other is that the fronted vP cannot be too large (i.e., the fronted vP can only include a verb and an object at most). I found evidence that seems to be against the second explanation: I constructed examples in (66), where the adverb *maher* 'quickly' is a low adverb, and my consultant told me that it is grammatical to have the adverb in the sentence with the verb being at the initial position (66a) and ungrammatical to have the adverb in the initial position (66b). This shows that the fronted vP can include an adverb as long as the verb is at the initial position, which is against the second explanation where the fronted vP cannot be too large. In this sense, I suggest that the facts in (65) - (66) are derived by verbs obligatorily moving out of a fronted vP.

(64) a. le'horid et ha-maym, Gil lo morid
to-flush ACC the-water Gil not flushes
'Flush the toilet, Gil does not.'

b. *lo le'horid et ha-maym, Gil morid
not to-flush ACC the-water Gil flushes

(65) a. le'horid et ha-maym, Gil tamid morid
to-flush ACC the-water Gil always flushes
'Flush the toilet, Gil always flushes.'

b. *tamid le'horid et ha-maym, Gil morid (Landau 2006: 38, ex.10)
always to-flush ACC the-water Gil flushes

(66) a.

Lishtof	maher	et	ha-kelim,	hu	shataf
wash.INF	quickly	ACC	the-dishes	he	wash.PST

‘As for washing the dishes quickly, he washed.’

b.

*maher	lishtof	et	ha-kelim,	hu	shataf
quickly	wash.INF	ACC	the-dishes	he	wash.PST

Here, I assume that low adverbs, such as *maher* ‘quickly’, is at the left edge of vP, not VP; and vP is fronted. Also, note that the verb in the fronted VP is in its infinitival form. Thus, in order to capture the observation that the infinitival verb has to occur at the left edge, I propose the structure in (60), where $v_1{}^0$ moves out of VP and ends up adjoining with X^0 and being at the left edge.[5]

So far, I have shown evidence supporting that in Hebrew,

(i) verbs move out of vP to adjoin with higher heads (i.e., V-to-T(-to-C) movement);

(ii) in verb-doubling constructions, verbs move out of vP and end up adjoining with a higher head (i.e., X^0).

I propose that (i) and (ii) both follow from the fact in (67). In (i), verb roots move to T^0 to get the tense morphology to be spelled out as a tensed verb; and in (ii) verb roots move to Infl^0 so that the verb can be in the infinitival form and there is a lexical item in the lexicon that can be inserted for the infinitival verb.[6]

(67) In Hebrew, verb roots cannot be pronounced.

(Landau 2006)

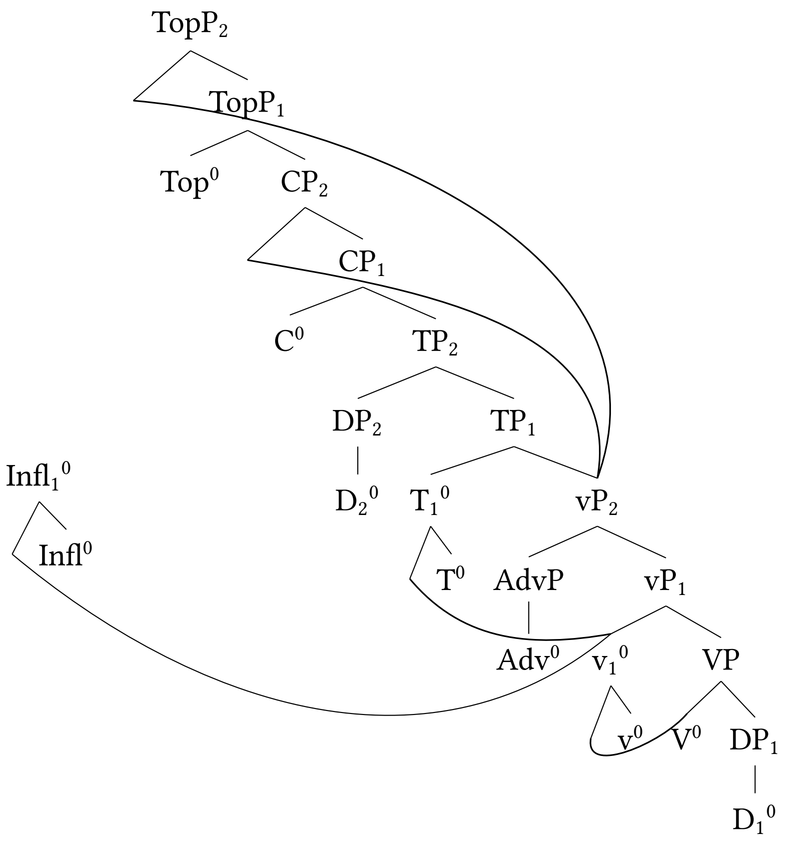
Having explained why v moves to Infl^0, there is still the mystery about what this X head is and what triggers the $I_1{}^0$-to-X^0 movement. Note that if the complex $I_1{}^0$ does not

[5]If one assumes that the adverb is at the left edge of VP and vP is fronted, the verb will always be at the left edge since in vP the verb always precedes VP-adverbs, and the VP-adverb can never occur to the left of the verb regardless of whether there is v(P) fronting, which leaves the movement of the verb out of the moved vP unmotivated.

[6]In Landau (2006), it is claimed that the verb root is spelled out as its “default form”, which is the infinitival form; my proposal here is consistent with this claim but provides a syntactic mechanism for the verb root to be spelled out as an infinitival form.

adjoin with X^0, the structure will have two root nodes, which are shown in (68), where $TopP_2$ and $I_1{}^0$ are two root nodes. I assume that there can only be one root node in a structure, so I propose that $I_1{}^0$ adjoining with X^0 and $X_1{}^0$ adjoins with $TopP_2$ to form XP is a technique to combine two root nodes into one (i.e., there is no semantic interpretation associated with X^0 projections, which can be treated as a functional empty category). I propose that the projection of X^0 is a last resort and can only be implemented when needed (i.e., in this case, to combine two root nodes to one).

(68)



Now that I have clarified the details of the structure, I continue with the linearization in the XP Spell-out domain. In the XP Spell-out domain, the Maximal X^0 Nodes are shown in (69).

(69) Maximal X^0 Nodes of XP = $\{X_1{}^0, Adv^0, D_1{}^0, Top^0, C^0, D_2{}^0, T_1{}^0\}$

Table 6.7 shows the sisterhood relationship and the required orders by the Language

Specific Constraints for all Maximal X^0 Nodes α and β of VP if α and β are ordered as a pair in a given set.

		relationship	**ordering statements**
$X_1{}^0$	$TopP_2$	sisters	$\{X_1{}^0 < Top^0, X_1{}^0 < C^0, X_1{}^0 < D_2{}^0, X_1{}^0 < T_1{}^0\}$, $X_1{}^0 < Adv^0, X_1{}^0 < D_1{}^0$
vP_2	$TopP_1$	sisters	$\{Adv^0 < Top^0, Adv^0 < C^0, Adv^0 < D_2{}^0, Adv^0 < T_1{}^0$, $D_1{}^0 < Top^0, D_1{}^0 < C^0, D_1{}^0 < D_2{}^0, D_1{}^0 < T_1{}^0\}$
AdvP	vP_1	sisters	$\{Adv^0 < D_1{}^0\}$
Top^0	CP_2	sisters	$\{Top^0 < C^0, Top_1{}^0 < D_2{}^0, Top_1{}^0 < T_1{}^0\}$
vP_2	CP_1	non-sisters	N/A
C^0	TP_2	sisters	$\{C^0 < D_2{}^0, C^0 < T_1{}^0\}$
DP_2	TP_1	sisters	$\{D_2{}^0 < T_1{}^0\}$
$T_1{}^0$	vP_2	non-sisters	N/A

Table 6.7: XP Spell-out domain

Note that $TopP_1$, CP_2, CP_1, TP_2 and TP_1 do not fully dominate Adv^0 and $D_1{}^0$. Also, $T_1{}^0$ and vP_2 are not sisters since their immediate dominating mother TP_1 does not fully dominate vP_2; vP_2 and CP_1 are not sisters since their immediate dominating mother CP_2 does not fully dominate vP_2. The relevant paths are shown in (70).

(70) a. (i) $p_3(Adv^0) = \{AdvP, vP_2, TopP_2, XP\}$

(ii) $p_2(Adv^0) = \{AdvP, vP_2, CP_2, TopP_1, TopP_2, XP\}$

(iii) $p_1(Adv^0) = \{AdvP, vP_2, TP_1, TP_2, CP_1, CP_2, TopP_1, TopP_2, XP\}$

b. (i) $p_3(D_1{}^0) = \{DP_1, VP, vP_1, vP_2, TopP_2, XP\}$

(ii) $p_2(D_1{}^0) = \{DP_1, VP, vP_1, vP_2, CP_2, TopP_1, TopP_2, XP\}$

(iii) $p_1(D_1{}^0) = \{DP_1, VP, vP_1, vP_2, TP_1, TP_2, CP_1, CP_2\}$

c. $p(C^0) = \{CP_1, CP_2, TopP_1, TopP_2, XP\}$

d. $p(D_2{}^0) = \{DP_2, TP_2, CP_1, CP_2, TopP_1, TopP_2, XP\}$

e. $p(T_1{}^0) = \{TP_1, TP_2, CP_1, CP_2, TopP_1, TopP_2, XP\}$

f. (i) $p_3(vP_2) = \{TopP_2, XP\}$

(ii) $p_2(vP_2) = \{CP_2, TopP_1, TopP_2, XP\}$

(iii) $p_1(vP_2) = \{TP_1, TP_2, CP_1, CP_2, TopP_1, TopP_2, XP\}$

The following set in (71) for the XP Spell-out domain satisfies all the constraints.

(71)

$$\left\{\begin{array}{llllll} X_1{}^0 < Adv^0 & Adv^0 < D_1{}^0 & D_1{}^0 < Top^0 & Top_1{}^0 < C^0 & C^0 < D_2{}^0 & D_2{}^0 < T_1{}^0 \\ X_1{}^0 < D_1{}^0 & Adv^0 < Top^0 & D_1{}^0 < C^0 & Top_1{}^0 < D_2{}^0 & C^0 < T_1{}^0 & \\ X_1{}^0 < Top^0 & Adv^0 < C^0 & D_1{}^0 < D_2{}^0 & Top_1{}^0 < T_1{}^0 & & \\ X_1{}^0 < C^0 & Adv^0 < D_2{}^0 & D_1{}^0 < T_1{}^0 & & & \\ X_1{}^0 < D_2{}^0 & Adv^0 < T_1{}^0 & & & & \\ X_1{}^0 < T_1{}^0, & & & & & \end{array}\right\}$$

The union set of XP, CP and VP is shown in (72).

(72)

$$\left\{\begin{array}{lllllll} X_1{}^0 < Adv^0 & Adv^0 < D_1{}^0 & D_1{}^0 < Top^0 & Top_1{}^0 < C^0 & C^0 < D_2{}^0 & D_2{}^0 < T_1{}^0 & \mathbf{V^0 < D_1{}^0} \\ X_1{}^0 < D_1{}^0 & Adv^0 < Top^0 & D_1{}^0 < C^0 & Top_1{}^0 < D_2{}^0 & C^0 < T_1{}^0 & & \\ X_1{}^0 < Top^0 & Adv^0 < C^0 & D_1{}^0 < D_2{}^0 & Top_1{}^0 < T_1{}^0 & & & \\ X_1{}^0 < C^0 & Adv^0 < D_2{}^0 & D_1{}^0 < T_1{}^0 & & & & \\ X_1{}^0 < D_2{}^0 & Adv^0 < T_1{}^0 & & & & & \\ X_1{}^0 < T_1{}^0, & & & & & & \end{array}\right\}$$

Since the final union set is reached, Ordering Deletion should be implemented, which changes the union set to the following one in (73). Note that since (i) $X_1{}^0$ dominates V^0, (ii) both $X_1{}^0$ and V^0 precede $D_1{}^0$, and (iii) there is no β such that $\beta < X_1{}^0$ and $V^0 < \beta$ or $\beta < V^0$ and $X_1{}^0 < \beta$, $V^0 < D_1{}^0$ should be deleted.

(73)

$$\left\{\begin{array}{llllll}
{X_1}^0 < Adv^0 & Adv^0 < {D_1}^0 & {D_1}^0 < Top^0 & {Top_1}^0 < C^0 & C^0 < {D_2}^0 & {D_2}^0 < {T_1}^0 \\
{X_1}^0 < {D_1}^0 & Adv^0 < Top^0 & {D_1}^0 < C^0 & {Top_1}^0 < {D_2}^0 & C^0 < {T_1}^0 & \\
{X_1}^0 < Top^0 & Adv^0 < C^0 & {D_1}^0 < {D_2}^0 & {Top_1}^0 < {T_1}^0 & & \\
{X_1}^0 < C^0 & Adv^0 < {D_2}^0 & {D_1}^0 < {T_1}^0 & & & \\
{X_1}^0 < {D_2}^0 & Adv^0 < {T_1}^0 & & & & \\
{X_1}^0 < {T_1}^0, & & & & &
\end{array}\right\}$$

Now, implement the Set-to-String algorithm. This union set in (73) satisfies all the constraints (i.e., all the intersection sets are empty), so as before, the linearization algorithm stops at Stage 1. Then, the String Forming Mechanism runs on (73), which yields the string $<_S {X_1}^0 Adv^0 {D_1}^0 Top^0 C^0 {D_2}^0 {T_1}^0 >$.

Next, lexical insertion forms sites. (74) shows the sites for complex heads.

(74) a. ${X_1}^0 \sim \#V^0v^0I^0X^0\#$

b. ${T_1}^0 \sim \#V^0v^0T^0\#$

Lastly, implement lexical insertion, which is shown in (75).

(75) a. $\#V^0v^0I^0X^0\# =$ *lishtof* ‘to wash’

b. $Adv^0 =$ *maher* ‘quickly’

c. ${D_1}^0 =$ *et ha-kelim* ‘acc the dishes’

d. $Top^0 = \varnothing$

e. $C^0 = \varnothing$

f. ${D_2}^0 =$ *hu* ‘he’

g. ${T_1}^0 =$ *shataf* ‘washed’

Finally, we get the utterance in (105), which is repeated below in (76).

(76) **Hebrew (infinitival form)**

li**sht**o**f**	maher	et	ha-kelim,	hu	**sh**a**t**a**f**
wash.INF	quickly	ACC	the-dishes	he	**wash**.PST

'As for washing the dishes quickly, he washed.'

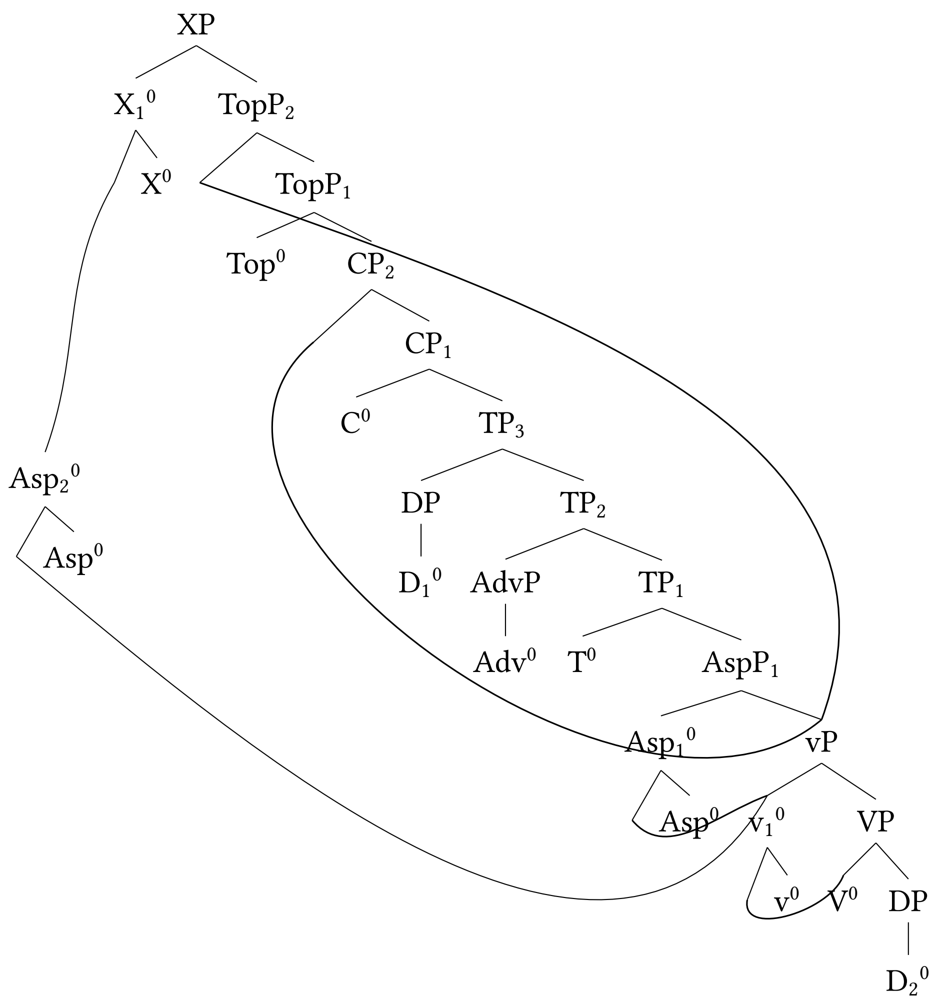
Note that in this Hebrew example, the verb upstairs is in the infinitival form and the verb downstairs is in the past form. In Mandarin, there are also cases where both the verb upstairs and the verb downstairs are marked with aspect. An example is shown in (77), the structure of which is shown in (78). I will not go through the linearization process again since it is quite similar to the Hebrew example in (76).

(77)

Chi-guo	bale,	Lili	dique	**chi-guo**
eat-ASP	Guava	Lili	indeed	**eat-**ASP

'As for having eaten Guava before, Lili indeed has eaten (them) before.'

(78)



6.3 VP-doubling

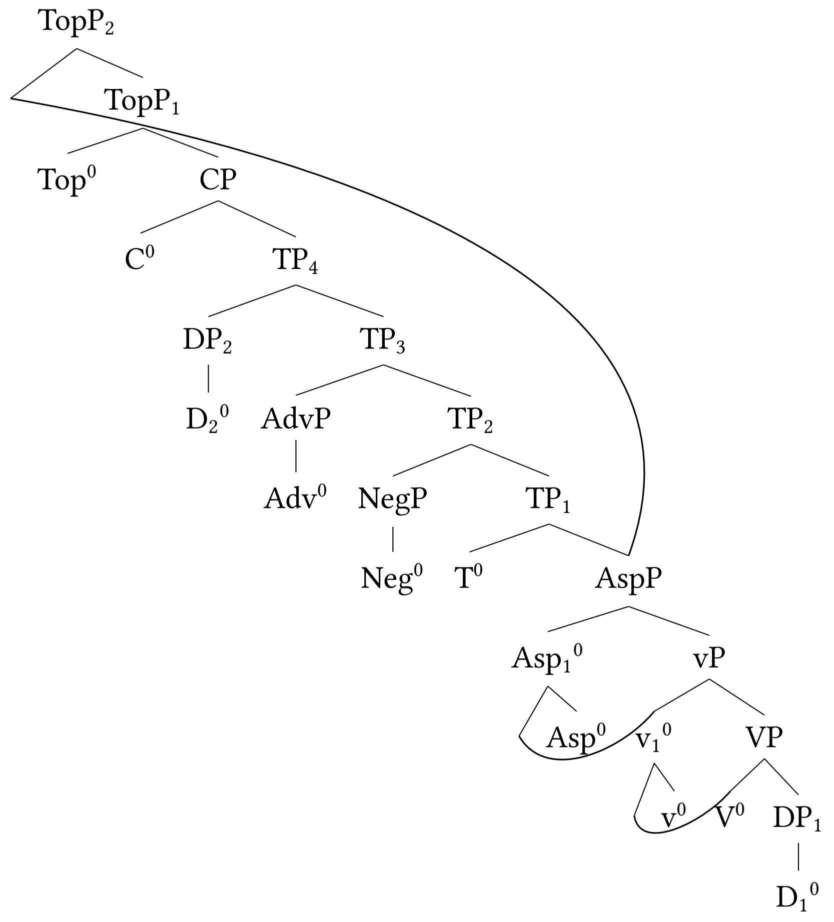
Now, let's turn to the Mandarin VP-doubling example in (79). The structure of it is shown in (80), where AspP moves directly to Spec, TopP.

(79) **Mandarin (aspectual form)**

Chi-guo bale,		Lili	dique	mei	**chi-guo bale**
eat-ASP	**Guava**	Lili	indeed	not	**eat-**ASP **Guava**

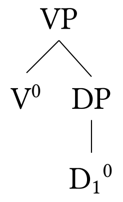
'As for having eaten Guava before, Lili indeed hasn't eaten Guava before.'

(80)



There are three Spell-out domains in (80): VP, CP and TopP$_2$. The structure of the VP Spell-out domain is shown in (81) (note that for ease of presentation, I simplify the structure of the DPs).

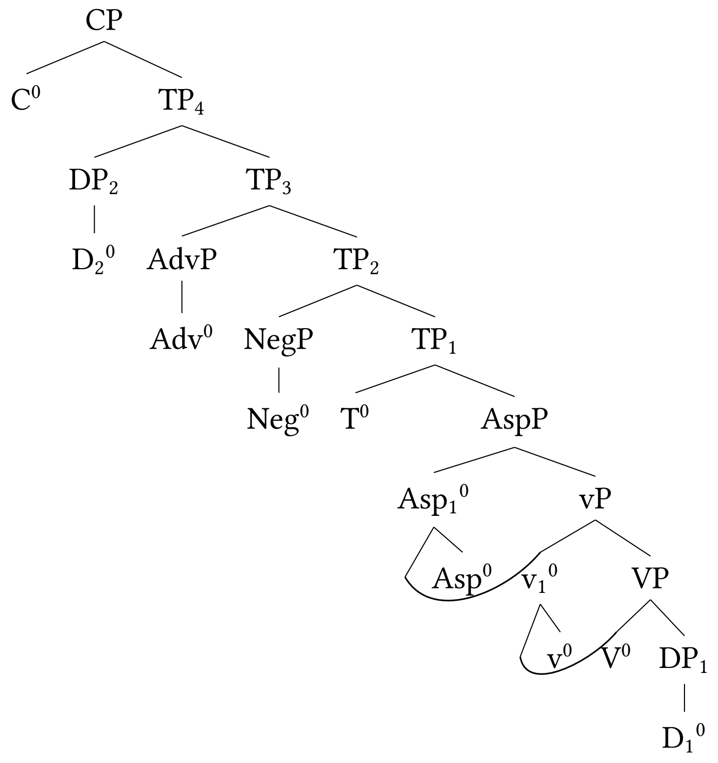
(81)



The following set in (82) for the VP Spell-out domain satisfies all the constraints.

(82) $\{V^0 < {D_1}^0\}$

Now, let's proceed to the CP Spell-out domain, the structure of which is shown in (83).

(83)


In the CP Spell-out domain, the Maximal X^0 Nodes are shown in (84).

(84) Maximal X^0 Nodes of CP = $\{C^0, {D_2}^0, Adv^0, Neg^0, T^0, {Asp_1}^0, {D_1}^0\}$

The set for the CP Spell-out domain that satisfies all the constraints is shown in (85).

(85)

$$\left\{\begin{array}{llllll} C^0 < {D_2}^0 & {D_2}^0 < Adv^0 & Adv^0 < Neg^0 & Neg^0 < T^0 & T^0 < {Asp_1}^0 & {Asp_1}^0 < {D_1}^0 \\ C^0 < Adv^0 & {D_2}^0 < Neg^0 & Adv^0 < T^0 & Neg^0 < {Asp_1}^0 & T^0 < {D_1}^0 & \\ C^0 < Neg^0 & {D_2}^0 < T^0 & Adv^0 < {Asp_1}^0 & Neg^0 < {D_1}^0 & & \\ C^0 < T^0 & {D_2}^0 < {Asp_1}^0 & Adv^0 < {D_1}^0 & & & \\ C^0 < {Asp_1}^0 & {D_2}^0 < {D_1}^0 & & & & \\ C^0 < {D_1}^0 & & & & & \end{array}\right\}$$

The union set of CP and VP is shown in (86).

(86)

$$\left\{\begin{array}{llllll} C^0 < {D_2}^0 & {D_2}^0 < Adv^0 & Adv^0 < Neg^0 & Neg^0 < T^0 & T^0 < {Asp_1}^0 & {Asp_1}^0 < {D_1}^0 \\ C^0 < Adv^0 & {D_2}^0 < Neg^0 & Adv^0 < T^0 & Neg^0 < {Asp_1}^0 & T^0 < {D_1}^0 & \\ C^0 < Neg^0 & {D_2}^0 < T^0 & Adv^0 < {Asp_1}^0 & Neg^0 < {D_1}^0 & & \\ C^0 < T^0 & {D_2}^0 < {Asp_1}^0 & Adv^0 < {D_1}^0 & & & \\ C^0 < {Asp_1}^0 & {D_2}^0 < {D_1}^0 & & & & \\ C^0 < {D_1}^0 & & & & & \\ & & & & & \\ \mathbf{V^0} < \mathbf{{D_1}^0} & & & & & \end{array}\right\}$$

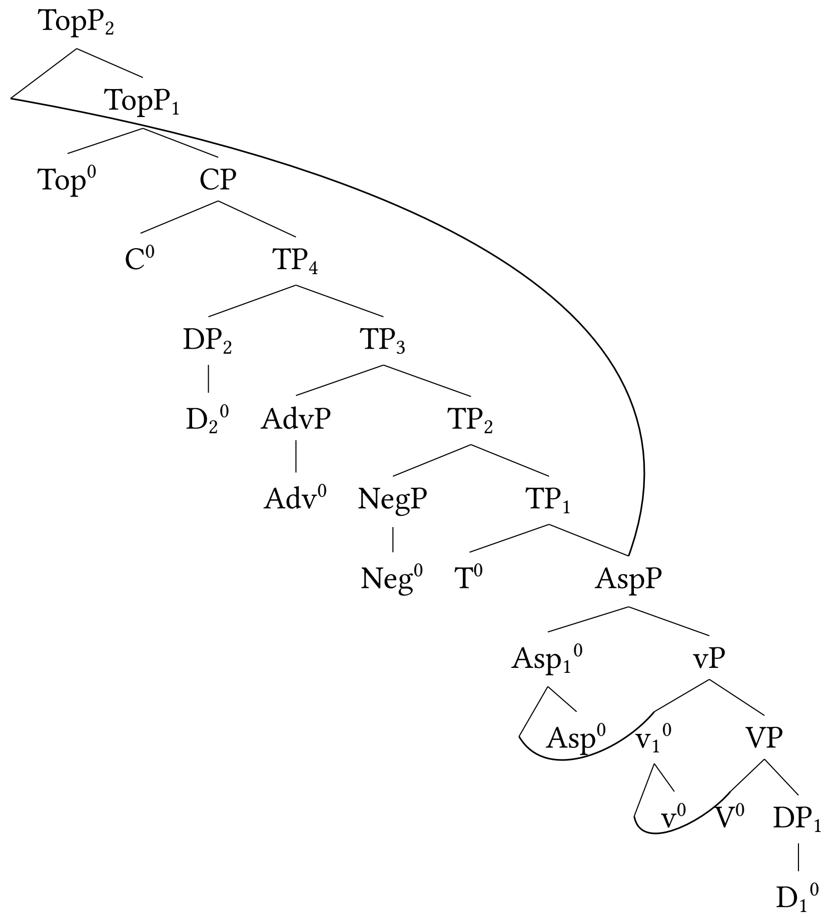
This union set satisfies all the constraints. Note that for the union set, the Totality Constraint and the Language Specific Constraints are evaluated against the structure in the CP Spell-out domain. Regarding the Totality Constraint, the Maximal X^0 Nodes are the same as the ones in the CP Spell-out domain, which are shown in (87). Note that V^0 is not a Maximal X^0 Node in the CP Spell-out domain, so the Totality Constraint does not consider whether V^0 is evaluated relative to the other Maximal X^0 Nodes.

(87) Maximal X^0 Nodes of CP = {C^0, ${D_2}^0$, Adv^0, Neg^0, T^0, ${Asp_1}^0$, ${D_1}^0$}

In addition, regarding the Language Specific Constraints, especially the head constraint (i.e., if x contains $\alpha < \beta$ or $\beta < \alpha$, and α is a head, there must be an ordering statement of α preceding β if β is fully dominated by YP, where YP is the sister of α.), having the ordering statement $V^0 < {D_1}^0$ does not make the set violate the Language Specific Constraints. The reason is that though (i) the set has the ordering statement $V^0 < {D_1}^0$, (ii) V^0 is a head, and (iii) ${D_1}^0$ is fully dominated by DP_1, V^0 and DP_1 are not sisters (i..e, VP, which is the immediate dominating mother of V^0 and DP_1, does not fully dominate V^0 - V^0 does not have to go through VP to reach the root node since it is multidominated). Thus, the existence of the ordering statement $V^0 < {D_1}^0$ does not require any ordering statement to be present in the union set of VP and CP. Since the Language Specific Constraints do not forbid $V^0 < {D_1}^0$ to appear in the union set either, having $V^0 < {D_1}^0$ does not violate any constraint.

Now, let's go to the $TopP_2$ Spell-out domain. The structure of it is shown in (88).

(88)



In the TopP$_2$ Spell-out domain, the Maximal X^0 Nodes are shown in (89).

(89) Maximal Nodes of TopP$_2$ = {Asp$_1{}^0$, D$_1{}^0$, Top^0, C^0, D$_2{}^0$, Adv^0, Neg^0, T^0}

Table 6.8 shows the sisterhood relationship and the ordering statements allowed by the Language Specific Constraint in the TopP$_2$ Spell-out domain.

		relationship	**ordering statements**
AspP	Top$_1$	sisters	{Asp$_1$^{0} < Top^0, Asp$_1$^{0} < C^0, Asp$_1$^{0} < D$_2$^{0}, Asp$_1$^{0} < Adv^0, Asp$_1$^{0} < Neg^0, Asp$_1$^{0} < T^0, D$_1$^{0} < Top^0, D$_1$^{0} < C^0, D$_1$^{0} < D$_2$^{0}, D$_1$^{0} < Adv^0, D$_1$^{0} < Neg^0, D$_1$^{0} < T^0 }
Asp$_1$^{0}	vP	sisters	{Asp$_1$^{0} < D$_1$^{0}}
Top^0	CP	sisters	{Top^0 < C^0, Top^0 < D$_2$^{0}, Top^0 < Adv^0, Top^0 < Neg^0, Top^0 < T^0 }
C^0	TP$_4$	sisters	{C^0 < D$_2$^{0}, C^0 < Adv^0, C^0 < Neg^0, C^0 < T^0 }
D$_2$^{0}	TP$_3$	sisters	{D$_2$^{0} < Adv^0, D$_2$^{0} < Neg^0, D$_2$^{0} < T^0}
Adv^0	TP$_2$	sisters	{Adv^0 < Neg^0, Adv^0 < T^0}
Neg^0	TP$_1$	sisters	{Neg^0 < T^0}
T^0	AspP	non-sisters	N/A

Table 6.8: TopP$_2$ Spell-out domain

The following set in (90) for the TopP$_2$ Spell-out domain satisfies all the constraints.

(90)

$$\left\{\begin{array}{llllll}
Asp_1{}^0 < D_1{}^0 & D_1{}^0 < Top^0 & Top^0 < C^0 & C^0 < D_2{}^0 & D_2{}^0 < Adv^0 & Adv^0 < Neg^0 \\
Asp_1{}^0 < Top^0 & D_1{}^0 < C^0 & Top^0 < D_2{}^0 & C^0 < Adv^0 & D_2{}^0 < Neg^0 & Adv^0 < T^0 \\
Asp_1{}^0 < C^0 & D_1{}^0 < D_2{}^0 & Top^0 < Adv^0 & C^0 < Neg^0 & D_2{}^0 < T^0 & \\
Asp_1{}^0 < D_2{}^0 & D_1{}^0 < Adv^0 & Top^0 < Neg^0 & C^0 < T^0 & & \\
Asp_1{}^0 < Adv^0 & D_1{}^0 < Neg^0 & Top^0 < T^0 & & & \\
Asp_1{}^0 < Neg^0 & D_1{}^0 < T^0 & & & & \\
Asp_1{}^0 < T^0 & & & & & \\
\\
\quad Neg^0 < T^0 & & & & &
\end{array}\right\}$$

The union set of CP and VP is repeated in (91).

(91)

$$\left\{\begin{array}{llll}
C^0 < {D_2}^0 & {D_2}^0 < Adv^0 & Adv^0 < Neg^0 & \\
C^0 < Adv^0 & {D_2}^0 < Neg^0 & Adv^0 < T^0 & \\
C^0 < Neg^0 & {D_2}^0 < T^0 & Adv^0 < {Asp_1}^0 & \\
C^0 < T^0 & {D_2}^0 < {Asp_1}^0 & Adv^0 < {D_1}^0 & \\
C^0 < {Asp_1}^0 & {D_2}^0 < {D_1}^0 & & \\
C^0 < {D_1}^0 & & & \\
 & & & \\
Neg^0 < T^0 & T^0 < {Asp_1}^0 & {Asp_1}^0 < {D_1}^0 & \mathbf{V^0 < {D_1}^0} \\
Neg^0 < {Asp_1}^0 & T^0 < {D_1}^0 & & \\
Neg^0 < {D_1}^0 & & &
\end{array}\right\}$$

The union set of TopP$_2$, CP and VP is shown in (92).

(92)

$$\left\{\begin{array}{llllll}
{Asp_1}^0 < {D_1}^0 & {D_1}^0 < Top^0 & Top^0 < C^0 & C^0 < {D_2}^0 & {D_2}^0 < Adv^0 & Adv^0 < Neg^0 \\
{Asp_1}^0 < Top^0 & {D_1}^0 < C^0 & Top^0 < {D_2}^0 & C^0 < Adv^0 & {D_2}^0 < Neg^0 & Adv^0 < T^0 \\
{Asp_1}^0 < C^0 & {D_1}^0 < {D_2}^0 & Top^0 < Adv^0 & C^0 < Neg^0 & {D_2}^0 < T^0 & \mathbf{Adv^0 < {Asp_1}^0} \\
{Asp_1}^0 < {D_2}^0 & {D_1}^0 < Adv^0 & Top^0 < Neg^0 & C^0 < T^0 & \mathbf{{D_2}^0 < {Asp_1}^0} & \mathbf{Adv^0 < {D_1}^0} \\
{Asp_1}^0 < Adv^0 & {D_1}^0 < Neg^0 & Top^0 < T^0 & \mathbf{C^0 < {Asp_1}^0} & \mathbf{{D_2}^0 < {D_1}^0} & \\
{Asp_1}^0 < Neg^0 & {D_1}^0 < T^0 & & \mathbf{C^0 < {D_1}^0} & & \\
{Asp_1}^0 < T^0 & & & & & \\
 & & & & & \\
Neg^0 < T^0 & \mathbf{T^0 < {Asp_1}^0} & \mathbf{V^0 < {D_1}^0} & & & \\
\mathbf{Neg^0 < {Asp_1}^0} & \mathbf{T^0 < {D_1}^0} & & & & \\
\mathbf{Neg^0 < {D_1}^0} & & & & &
\end{array}\right\}$$

Since the final union set is formed, Ordering Deletion should be applied, which is repeated below in (23). After Ordering Deletion, the union set becomes the one in (93) ($V^0 < {D_1}^0$ is deleted since ${Asp_1}^0$ dominates V^0 and ${Asp_1}^0$ also precedes ${D_1}^0$).

(93)

$$\left\{\begin{array}{llllll}
Asp_1{}^0 < D_1{}^0 & D_1{}^0 < Top^0 & Top^0 < C^0 & C^0 < D_2{}^0 & D_2{}^0 < Adv^0 & Adv^0 < Neg^0 \\
Asp_1{}^0 < Top^0 & D_1{}^0 < C^0 & Top^0 < D_2{}^0 & C^0 < Adv^0 & D_2{}^0 < Neg^0 & Adv^0 < T^0 \\
Asp_1{}^0 < C^0 & D_1{}^0 < D_2{}^0 & Top^0 < Adv^0 & C^0 < Neg^0 & D_2{}^0 < T^0 & \mathbf{Adv^0 < Asp_1{}^0} \\
Asp_1{}^0 < D_2{}^0 & D_1{}^0 < Adv^0 & Top^0 < Neg^0 & C^0 < T^0 & \mathbf{D_2{}^0 < Asp_1{}^0} & \mathbf{Adv^0 < D_1{}^0} \\
Asp_1{}^0 < Adv^0 & D_1{}^0 < Neg^0 & Top^0 < T^0 & \mathbf{C^0 < Asp_1{}^0} & \mathbf{D_2{}^0 < D_1{}^0} & \\
Asp_1{}^0 < Neg^0 & D_1{}^0 < T^0 & & \mathbf{C^0 < D_1{}^0} & & \\
Asp_1{}^0 < T^0 & & & & & \\
 & & & & & \\
Neg^0 < T^0 & \mathbf{T^0 < Asp_1{}^0} & & & & \\
\mathbf{Neg^0 < Asp_1{}^0} & \mathbf{T^0 < D_1{}^0} & & & & \\
\mathbf{Neg^0 < D_1{}^0} & & & & &
\end{array}\right\}$$

Next, implement the Set-to-String algorithm. Note that this union set satisfies all the constraints except for the Asymmetry Constraint. For instance, this union set has both $Asp_1{}^0 < C^0$ and $C^0 < Asp_1{}^0$. Based on Stage 1 of the Set-to-String algorithm, when there exists a non-empty intersection set, the Set-to-String algorithm should proceed to Stage 2. The intersection sets for all nodes are shown in (94).

Stage 1:

1. **Find the intersection sets:** For each distinct node n_1, n_2, n_3, ... in the set of ordered pairs O = $\{n_1 < n_3, n_3 < n_2, ...\}$, get the set X = $\{..., n_1, ...\}$ that contains all the nodes that precedes n and the set Y = $\{..., n_2, ...\}$ that contains all the nodes that follows n, and get the set I = X $\cap$ Y, which is the intersection of set X and set Y.

2. **Check the intersection sets:** If all the intersection sets are empty, run the String Forming Mechanism on set O and the set will be turned into a string. **Otherwise, proceed to Stage 2.**

(94) a. $Asp_1{}^0$: $\{C^0, D_2{}^0, Adv^0, Neg^0, T^0\} = \{D_1{}^0, Top^0, C^0, D_2{}^0, Adv^0, Neg^0, T^0\} \cap \{C^0, D_2{}^0, Adv^0, Neg^0, T^0\}$

b. $D_1{}^0$: $\{C^0, D_2{}^0, Adv^0, Neg^0, T^0\} = \{Top^0, C^0, D_2{}^0, Adv^0, Neg^0, T^0\} \cap \{Asp_1{}^0, C^0, D_2{}^0, Adv^0, Neg^0, T^0, V^0\}$

c. Top^0: $\{\varnothing\} = \{C^0, D_2{}^0, Adv^0, Neg^0, T^0\} \cap \{Asp_1{}^0, D_1{}^0\}$

d. C^0: $\{Asp_1{}^0, D_1{}^0\} = \{D_2{}^0, Adv^0, Neg^0, T^0, Asp_1{}^0, D_1{}^0\} \cap \{Asp_1{}^0, D_1{}^0, Top^0\}$

e. $D_2{}^0$: $\{Asp_1{}^0, D_1{}^0\} = \{Adv^0, Neg^0, T^0, Asp_1{}^0, D_1{}^0\} \cap \{Asp_1{}^0, D_1{}^0, Top^0, C^0\}$

f. Adv^0: $\{Asp_1{}^0, D_1{}^0\} = \{Neg^0, T^0, Asp_1{}^0, D_1{}^0\} \cap \{Asp_1{}^0, D_1{}^0, Top^0, C^0, D_2{}^0\}$

g. Neg^0: $\{Asp_1{}^0, D_1{}^0\} = \{T^0, Asp_1{}^0, D_1{}^0\} \cap \{Asp_1{}^0, D_1{}^0, Top^0, C^0, D_2{}^0, Adv^0\}$

h. T^0: $\{Asp_1{}^0, D_1{}^0\} = \{Asp_1{}^0, D_1{}^0\} \cap \{Asp_1{}^0, D_1{}^0, Top^0, C^0, D_2{}^0, Adv^0, Neg^0\}$

In Stage 2, the algorithm examines each non-empty intersection sets and see whether there exists an intersection set that can form a qualified constituent. The set that contains all the non-empty intersections sets are shown in (95).

Stage 2:

1. **Collect all the non-empty intersection sets:** Form a set M = $\{I_1, I_2, ...\}$ that contains all the non-empty intersection sets.

2. **Try forming a string:**

   (a) **No qualified constituent:** For all given intersection sets $I_1 = \{n_1, n_2, ...\}$, $I_1 = \{n_3, ...\}$, ... in set M, if none of them forms a qualified constituent, the linearization process crashes.

   (b) **Qualified constituent:** If there exists intersection set I^* that forms a qualified constituent, **proceed to Stage 3.**

(95) $M = \{I_1 = \{C^0, D_2{}^0, Adv^0, Neg^0, T^0\}, I_2 = \{Asp_1{}^0, D_1{}^0\}\}$

Note that the nodes in I_2 of (95) form a constituent, and it is also a qualified constituent (i.e., in Mandarin, $[_{VP}$ V bare-NP] is a qualified constituent). Thus, proceed to Stage 3.

Stage 3:

1. **Form set S:** Form the subset $S = \{n_1 < n_2, ...\}$ of set O that contains all the ordered pairs $\alpha < \beta$, where both α and β are in the intersection set I^*.

2. **Forming a string:** Run the String Forming Mechanism algorithm, which is stated in (35), on set S and get the string str = $<_S$ n_1n_2...>.

In Stage 3, first, form a subset S of O, which contains all the ordering statements that order the nodes that form a qualified constituent. These nodes are in intersection set I_2: {$Asp_1{}^0$, $D_1{}^0$}}. Thus, the subset S is {$Asp_1{}^0 < D_1{}^0$}. Then, run the String Forming Mechanism on subset S, which yields the string $<_S$ $Asp_1{}^0D_1{}^0$>.

Now that a string is formed out of the nodes in the moved item, the next step is to replace the corresponding nodes with the string, which is the final stage (i.e., Stage 4).

Stage 4:

1. **Replace nodes with string:**

   (a) **Get set D = O - S** Form set D = O - S (i.e., D = {$n_1 < n_3$, $n_3 < n_2$ ...}, O = {$n_1 < n_3$, $n_3 < n_2$, $n_1 < n_2$, ...} and S = {$n_1 < n_2$})

   (b) **Replace** For every ordering statement $\alpha < \beta$ in the original set O, if the ordering statement is in set D, replace the node with the string str = $<_S$ n_1n_2...> if the node is in the intersection set I^* and in the α position. The updated set O = {$\underline{<_S n_1n_2...>}$ $< n_3$, $n_3 < n_2$, $n_1 < n_2$, ...}.

2. **Linearizing O:** Run the String Forming Mechanism algorithm in (35) on the updated set O.

In Stage 4, the algorithm needs to replace all the nodes in the intersection set I_2 with the string $<_S$ $Asp_1{}^0D_1{}^0$> for all the ordering statements in O except for the one that orders $Asp_1{}^0$ and $D_1{}^0$ (i.e., the ordering statement in subset S). To do so, set D is derived by subtracting S from the original set O, which is shown in (96).

(96)

$$\left\{\begin{array}{llllll}
Asp_1{}^0 < Top^0 & D_1{}^0 < Top^0 & Top^0 < C^0 & C^0 < D_2{}^0 & D_2{}^0 < Adv^0 & Adv^0 < Neg^0 \\
Asp_1{}^0 < C^0 & D_1{}^0 < C^0 & Top^0 < D_2{}^0 & C^0 < Adv^0 & D_2{}^0 < Neg^0 & Adv^0 < T^0 \\
Asp_1{}^0 < D_2{}^0 & D_1{}^0 < D_2{}^0 & Top^0 < Adv^0 & C^0 < Neg^0 & D_2{}^0 < T^0 & \mathbf{Adv^0 < Asp_1{}^0} \\
Asp_1{}^0 < Adv^0 & D_1{}^0 < Adv^0 & Top^0 < Neg^0 & C^0 < T^0 & \mathbf{D_2{}^0 < Asp_1{}^0} & \mathbf{Adv^0 < D_1{}^0} \\
Asp_1{}^0 < Neg^0 & D_1{}^0 < Neg^0 & Top^0 < T^0 & \mathbf{C^0 < Asp_1{}^0} & \mathbf{D_2{}^0 < D_1{}^0} & \\
Asp_1{}^0 < T^0 & D_1{}^0 < T^0 & & \mathbf{C^0 < D_1{}^0} & & \\
 & & & & & \\
Neg^0 < T^0 & \mathbf{T^0 < Asp_1{}^0} & & & & \\
\mathbf{Neg^0 < Asp_1{}^0} & \mathbf{T^0 < D_1{}^0} & & & & \\
\mathbf{Neg^0 < D_1{}^0} & & & & &
\end{array}\right\}$$

To make sure that the moved item is only ordered as a string in its higher position, replacement is only applied to the preceding positions for the nodes within the moved item (i.e., nodes in intersection set I_2). Formally, for every ordering statement s in O, if s is in set D, replace the node in s with the string $<_S Asp_1{}^0D_1{}^0>$ if the node is in intersection set $I_2 = \{Asp_1{}^0, D_1{}^0\}$ and in the preceding position. This is shown in (97). Note that since $Asp_1{}^0 < D_1{}^0$ is not in set D, no replacement occurs there.

(97)

$$\left\{\begin{array}{lllll}
Asp_1{}^0 < D_1{}^0 & Top^0 < C^0 & C^0 < D_2{}^0 & D_2{}^0 < Adv^0 & Adv^0 < Neg^0 \\
<_S \mathbf{Asp_1{}^0D_1{}^0}> < Top^0 & Top^0 < D_2{}^0 & C^0 < Adv^0 & D_2{}^0 < Neg^0 & Adv^0 < T^0 \\
<_S \mathbf{Asp_1{}^0D_1{}^0}> < C^0 & Top^0 < Adv^0 & C^0 < Neg^0 & D_2{}^0 < T^0 & Adv^0 < Asp_1{}^0 \\
<_S \mathbf{Asp_1{}^0D_1{}^0}> < D_2{}^0 & Top^0 < Neg^0 & C^0 < T^0 & D_2{}^0 < Asp_1{}^0 & Adv^0 < D_1{}^0 \\
<_S \mathbf{Asp_1{}^0D_1{}^0}> < Adv^0 & Top^0 < T^0 & C^0 < Asp_1{}^0 & D_2{}^0 < D_1{}^0 & \\
<_S \mathbf{Asp_1{}^0D_1{}^0}> < Neg^0 & & C^0 < D_1{}^0 & & \\
<_S \mathbf{Asp_1{}^0D_1{}^0}> < T^0 & & & & \\
 & & & & \\
\quad Neg^0 < T^0 & T^0 < Asp_1{}^0 & & & \\
\quad Neg^0 < Asp_1{}^0 & T^0 < D_1{}^0 & & & \\
\quad Neg^0 < D_1{}^0 & & & &
\end{array}\right\}$$

Now, the last step is to run the String Forming Mechanism on the updated union set in (97), which yields the string $<_S <_S Asp_1{}^0D_1{}^0>Top^0C^0D_2{}^0Adv^0Neg^0T^0Asp_1{}^0D_1{}^0>$.[7]

[7] Notice that in this set, there are no ordering statements between Top^0 and $Asp_1{}^0$ or $D_1{}^0$. However, since

The next step is to form lexical insertion sites, which is shown in (98).

(98) a. $<_S \mathrm{Asp_1}^0 \mathrm{D_1}^0> \sim <_S \#\mathrm{V}^0\mathrm{v}^0\mathrm{Asp}^0\#\#\mathrm{D_1}^0\#>$

b. $\mathrm{Top}^0 \sim \#\mathrm{Top}^0\#$

c. $\mathrm{C}^0 \sim \#\mathrm{C}^0\#$

d. $\mathrm{D_2}^0 \sim \#\mathrm{D_2}^0\#$

e. $\mathrm{Adv}^0 \sim \#\mathrm{Adv}^0\#$

f. $\mathrm{Neg}^0 \sim \#\mathrm{Neg}^0\#$

g. $\mathrm{T}^0 \sim \#\mathrm{T}^0\#$

h. $\mathrm{Asp_1}^0 \sim \#\mathrm{V}^0\mathrm{v}^0\mathrm{Asp}^0\#$

i. $\mathrm{D_1}^0 \sim \#\mathrm{D_1}^0\#$

Now, implement lexical insertion, which is shown in (99).

(99) a. $<_S \mathrm{Asp_1}^0\mathrm{D_1}^0> =$ *chi-∅-guo bale* 'eaten Guava'

b. Top^0 = ∅

c. C^0 = ∅

d. $\mathrm{D_2}^0$ = *Lili* 'Lili'

e. Adv^0 = *dique* 'indeed'

f. Neg^0 = *mei* 'not'

g. T^0 = ∅

h. $\#\mathrm{V}^0\mathrm{v}^0\mathrm{Asp}^0\# =$ *chi-∅-guo* 'eaten'

i. $\mathrm{D_1}^0$ = *bale* 'Guava'

Thus, the string is updated as:

$<_S$ $<_S$ eaten Guava > ∅ ∅ Lili indeed not ∅ eaten Guava>

Finally, the string becomes the utterance in (106), which is repeated below as (100).

the set contains $\mathrm{Top}^0 < \mathrm{C}^0$ and $\mathrm{C}^0 < \mathrm{Asp_1}^0$ and $\mathrm{D_1}^0$, in the actual utterance, Top^0 can only be pronounced before $\mathrm{Asp_1}^0$ and $\mathrm{D_1}^0$ in the linear order. See the chapter for system design for more details.

(100) **Mandarin (aspectual form)**

Chi-guo bale, Lili dique mei **chi-guo bale**
eat-∅-ASP Guava Lili indeed not **eat-∅-ASP Guava**
'As for having eaten Guava before, Lili indeed hasn't eaten Guava before.'

Before ending this section, I would like to present more VP-doubling data in Mandarin. I asked my consultants' judgements about data from (101) to (103). All my consultants accept VP-doubling when the object is a short bare noun (101a) or a pronoun (101b). However, their judgements vary for the rest of the data and it is not clear to me what factor(s) is relevant for their judgment. To be more specific, one consultant consistently accepts objects without a determiner but gives a question mark for objects with a determiner (the objects with determiner in the data have a specific meaning and the ones without a determiner have a non-specific meaning). However, whether an object has a determiner is not sufficient to decide the grammaticality because another consultant accepts sentences with a bare noun object but only when the object has sufficiently few number of syllables (for the bare noun that has the greatest number of syllables (101e), the consultant judges it ungrammatical). However, it is still not sufficient if both determiner and the number of syllables are taken into consideration. One consultant judges (103a) to be better than (103b) despite the fact that (103a)'s object has fewer syllables than (103b)'s object ((103a) has an adjective and a noun, with two syntactic layers; and (103b) has a number, a classifier and a noun, with three syntactic layers).[8] For now, I leave it as an open question as to what factors influence the grammaticality judgment of the VP-doubling cases and why those factors matter.

[8]Note that if in these examples, the main clauses are positive, the degree of acceptance gets much better. In this dissertation, I leave it as an open question why the polarity of the main clause makes a difference and only focuses on cases where the main clause has a negative force.

(101) **Bare nouns**

a. Chi-guo **bale**, Lili dique mei chi-guo **bale**.
eat-∅-ASP **Guava** Lili indeed not eat-∅-ASP **Guava**

‘As for having eaten Guava before, Lili indeed hasn’t eaten Guava before.’

Repeat of ex.(100)

b. Jian-guo **ni**, Lili dique mei jian-guo **ni**.
see-∅-ASP **pronoun** Lili indeed not see-∅-ASP **you**
‘As for having seen you before, Lili indeed hasn’t seen you before.’

c. (?/ok) qu-guo **aisaiebiya** ta kending mei chi-guo **aisaiebiya**
go-ASP Ethiopia 3.SG sure not go-ASP Ethiopia

‘As for having been to Ethiopia before, s/he sure hasn’t been to Ethiopia before(, but s/he has seen it on TV before).’

d. (?/ok) qu-guo **jiekesiluofake** ta kending mei chi-guo **jiekesiluofake**
go-ASP Czechoslovakia 3.SG sure not go-ASP Czechoslovakia

‘As for having been to Czechoslovakia before, s/he sure hasn’t been to Czechoslovakia before(, but s/he has seen it on TV before).’

e. (*/ok) qu-guo **buyinuosiailisi** ta kending mei chi-guo **buyinuosiailisi**
go-ASP Buenos.Aires 3.SG sure not go-ASP Buenos.Aires

‘As for having been to Buenos.Aires before, s/he sure hasn’t been to Buenos.Aires before(, but s/he has seen it on TV before).’

(102) **With determiner**

a. **Det + N**

(*/ok) jian-guo **na ren** ta kending mei jian-guo **na ren**
see-ASP that person 3.SG sure not see-ASP that person

‘As for having seen that person before, s/he sure hasn’t seen that person before (, but s/he has heard about it before).’

b. **Det + CL + N**

(*/ok) jian-guo **na-ge ren** ta kending mei jian-guo **na-ge ren]**
see-ASP that-CL person 3.SG sure not see-ASP that-CL person

‘As for having seen that person before, s/he sure hasn’t seen that person before (, but s/he has heard about it before).’

c. **Det + Adj + N**

(*/?) chi-guo **na fense-de bale** ta kending mei chi-guo **na fense-de**
eat-ASP that pink-Adj Guava 3.SG sure not eat-ASP that pink-Adj
bale
Guava

‘As for having eaten that pink Guava before, s/he sure hasn’t eaten that pink Guava before (, but s/he has seen it before).’

d. **Det + CL + Adj + N**

(*/?) jian-guo **na-ge fense-de bale** wo kending mei jian-guo
see-ASP that-CL pink-Adj Guava 1.SG sure not see-ASP
na-ge fense-de bale
that-CL pink-Adj Guava

‘As for having seen that pink Guava before, I sure hasn’t seen that pink Guava before (, but I have heard about it before).’

e. **Det + Num + CL + N**

(?/ok) jian-guo **zhe san-ge ren** ta kending mei jian-guo **zhe**
see-ASP these three-CL person 3.SG sure not see-ASP these
san-ge ren
three-CL person

‘As for having seen these three persons before, s/he sure hasn’t seen these three persons before (, but s/he has heard the voice of these three persons before).’

f. **Det + Num + CL + Adj + N**

(*/?) jian-guo **na san-ge fense-de bale** ta kending mei jian-guo
see-ASP those three-CL pink-Adj Guava 3.SG sure not see-ASP
na san-ge fense-de bale
those three-CL pink-Adj Guava

‘As for having seen those three pink Guavas before, s/he sure hasn’t seen those three Guavas before (, but s/he has heard about it before).’

(103) **No determiner**

a. **Adj + N**

(?/ok)	chi-guo	**fense-de**	**bale**	ta	kending	mei	chi-guo	**fense-de**	**bale**
	eat-ASP	pink-Adj	Guava	3.SG	sure	not	eat-ASP	pink-Adj	Guava

'As for having eaten pink Guava before, s/he sure hasn't eaten pink Guava before (, but s/he probably has seen it before).'

b. **Num + CL+ N**

(*/?/ok)	bangzhu-guo	**san-ge**	**ren**	ta	kending	mei	bangzhu-guo	**san-ge**	**ren**
	help-ASP	three-CL	person	3.SG	sure	not	help-ASP	three-CL	person

'As for having helped three persons before, s/he sure hasn't helped three persons before (, but s/he has helped two people).'

c. **Num + CL + Adj + N**

(*/ok)	chi-guo	**san-ge**	**fense-de**	**bale**	ta	kending	mei	chi-guo	**san-ge**	**fense-de**	**bale**
	eat-ASP	three-CL	pink-Adj	Guava	3.SG	sure	not	eat-ASP	three-CL	pink-Adj	Guava

'As for having eaten three pink Guavas before s/he sure hasn't eaten three pink Guavas before (, but s/he has eaten three pink peaches before).'

6.4 Summary

In this section, I have discussed cases where the verb appears to be pronounced multiple times. I presented three types of cases. The first type has a doubled verb, where only the verb is fronted (i.e., (104)), repeated below in (104). For the Hebrew case in (104a), the verb is doubled because there are two movement chains (i.e., V^0-to-v^0-to-T^0 and v^0-to-Top^0), where v^0 forms two different Maximal X^0 nodes (i.e., T^0 and Top^0), and these two nodes are both linearized and hence pronounced.[9] For the Yiddish case, the verb moves across a

[9]Note that in this case, v^0-to-Top^0 movement does not result in the verb being pronounced under v^0 and Top^0 (the verb is pronounced under T^0 and Top^0) despite that this movement across a Spell-out domain via a

Spell-out domain via a non-initial position, which results in multiple pronunciations.

(104) **Verb-doubling (with verb-fronting)**

a. **Hebrew (infinitival)**

Li**kn**o**t**,	hi	**k**a**nt**a	et	ha-praxim
buy.INF	she	**buy**.PST	ACC	the-flowers

'As for buying, she bought the flowers.'

b. **Yiddish (aspectual form)**

Gegessen,	hot	Maks	**gegessen**	fish
eaten	has	Max	**eaten**	fish

'As for having eaten, Max has eaten fish.'

The second type has a doubled verb with more than the verb being fronted (i.e., (2)), repeated below in (105). In this case, the verb further moves out of the fronted VP, which results in multiple pronunciations.

(105) **Verb-doubling (with VP-fronting)**

Hebrew (infinitival form)

Li**sht**o**f**	maher	et	ha-kelim,	hu	**sh**a**t**a**f**
to.**wash**	quickly	ACC	the-dishes	he	**wash**.PST

'As for washing the dishes quickly, he washed.'

The third type has VP doubling, where more than the verb is doubled. The example is in (3), repeated below in (106). In this case, the VP moves across a Spell-out domain via a non-initial position, and the VP is a qualified constituent, forming a substring that gets linearized, which results in multiple pronunciations.

non-initial position. This is because the V^0-to-v^0-to-T^0 movement interferes with the v^0-to-Top^0 movement in the sense that V^0-to-v^0-to-T^0 movement makes v^0 dominated by T^0, which makes v^0 not a Maximal X^0 node and thus cannot be linearized. If there is only the v^0-to-Top^0 movement in the structure, it should be predicted that the verb is pronounced under Top^0 and v^0.

(106) **VP-doubling**

Mandarin (aspectual form)

Chi-guo	**bale**,	Lili	dique	mei	**chi-guo**	**bale**
eat-ASP	**Guava**	Lili	indeed	not	**eat-**ASP	**Guava**

'As for having eaten Guava before, Lili indeed hasn't eaten Guava before.'

As a quick summary, the double pronunciations in the verb-doubling cases result from moving across a Spell-out domain via a non-initial position or from two separate chains.

CHAPTER 7

Conclusion

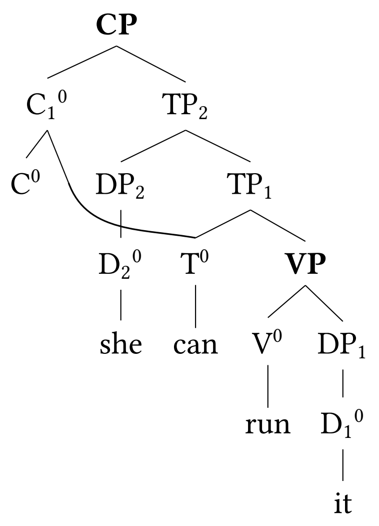
7.1 Summary

In this section, I provide a summary of all the head/phrase movement cases that I have shown in this dissertation. I first focus on the verb movement cases. In example (1) (repeat of (27) in chapter 4) and (3) (repeat of (21) in chapter 5), the moved verbs are pronounced only once. The T-to-C movement in (1) happens within one Spell-out domain, where only the highest head $C_1{}^0$ is collected. The V-to-v movement in (4) happens across Spell-out domains but via the initial positions of the spell-out domains such that the Ordering Deletion rule is triggered and ordering statements containing V are deleted. Thus, both of the moved verbs are pronounced only once.

(1) **Move within one Spell-out domain**

Can she run it?

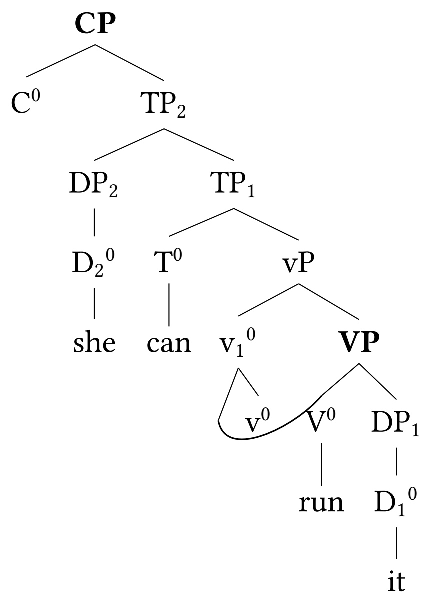
(2)



(3) **Move across Spell-out domains via the edge**

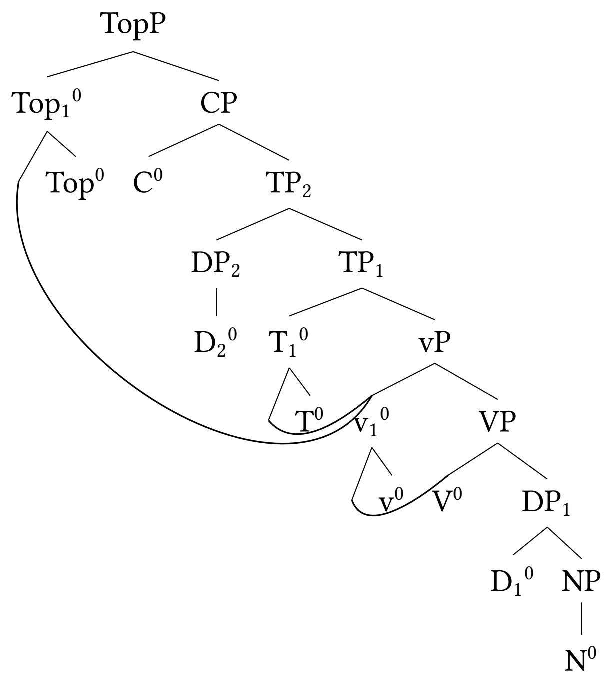
She can run it.

(4)



In contrast, if a head moves across different spell-out domains but not via the initial positions of the spell-out domains, the head is pronounced multiple times. An example for this scenario is the verb-doubling case that I discussed in section 6.1, which is repeated below in (5). In this case, $v_1{}^0$ moves from the CP Spell-out domain to the TopP Spell-out domain via a position that is not the initial position of the CP Spell-out domain. As a result, the Ordering Deletion rule is not applied: despite the fact that $Top_1{}^0$ dominates $v_1{}^0$, there exist ordering statements like $D_2{}^0 < v_1{}^0$ and $Top^0 < D_2{}^0$, where $v_1{}^0$ and its mother node Top^0 are ordered differently regarding $D_2{}^0$.

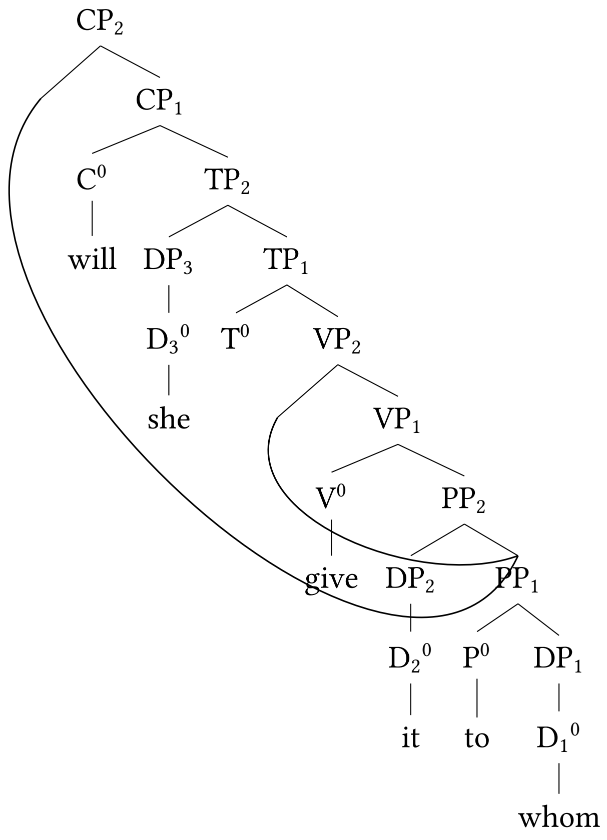
(5) Li**kn**o**t**, hi **k**a**nt**a et ha-praxim
buy.INF she **buy**.PST ACC the-flowers
'As for buying, she bought the flowers.'

(6)


As a quick summary, in my analysis, a moved verb is predicted to be pronounced once if it moves in one spell-out domain or across different spell-out domains via the initial positions of the spell-out domains; otherwise, the moved verb has to be pronounced multiple times.

Now, let's look at the phrasal movement cases. In (7) (repeat of (37) in chapter 5), my analysis predicts that the moved phrase will be pronounced once if it moves within one spell-out domain or across different spell-out domains via the initial positions of the spell-out domains. In this case, the phrase moves across different spell-out domains via the initial positions of the Spell-out domains.

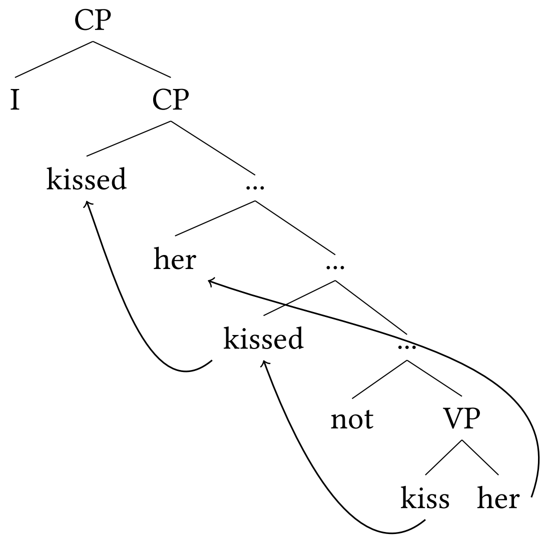
(7)



Note that in the Object Shift case (8) (repeat of (7a) in chapter 5), the object moves from a non-initial position in VP. However, since the verb in VP also moves and to a higher position in CP, the precedence relation between the verb and the object is preserved, so both the verb and the object are predicted to be pronounced once. In other words, during the course of derivation, the object position becomes an initial position when the verb moves out of VP.

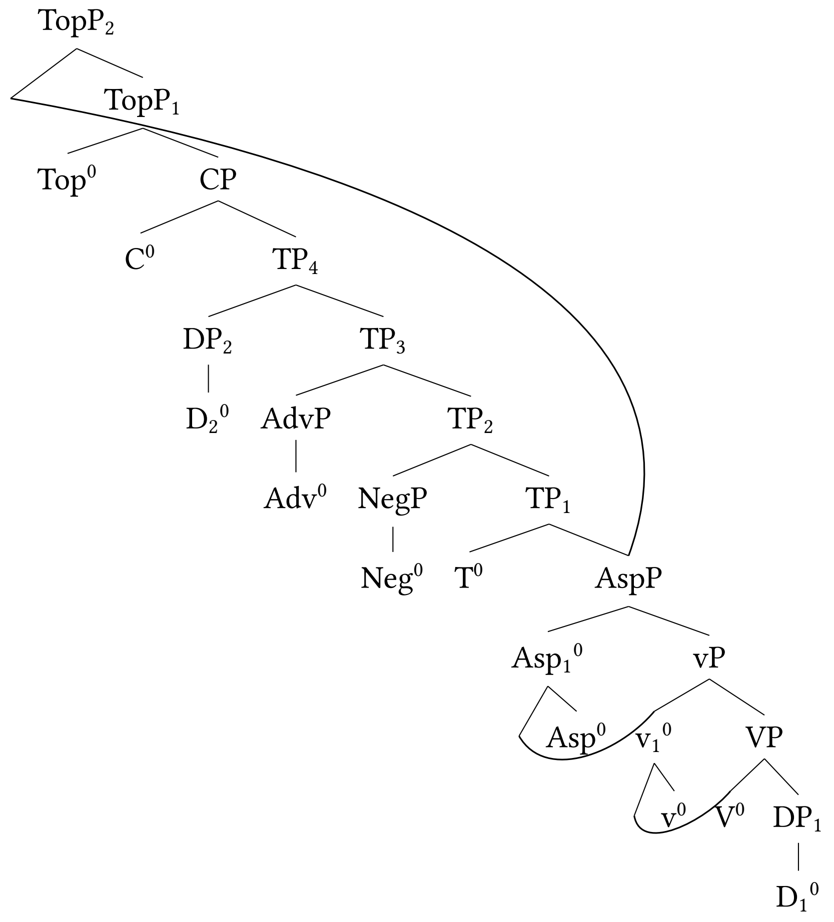
(8)	Jag	kysste	henne	inte	[$_{VP}$ t$_V$ t$_O$]
	I	kissed	her	not	

(9)



In contrast, in the VP-doubling case in (10) (repeat of (88) in chapter 6), the moved phrase is predicted to get pronounced twice since it does not move via the initial positions of the spell-out domains such that the precedence relations between the nodes in the moved item and some nodes are not preserved, which violates the Asymmetry Constraint. However, I stipulate that in Mandarin, it is allowed that the string of the moved item, instead of nodes, can be linearized, so the linearization process linearizes the moved item as a string in its higher position, which gets rid of the violation of the Asymmetry Constraint and produces multiple pronunciations of the moved phrase. Note that I stipulate that in English, strings are not allowed to be linearized, so the movement in (10) will lead the linearization process to crash in English.

(10)



Overall, my analysis states that a moved head/phrase will be pronounced only once if movement happens in the same Spell-out domain or across different Spell-out domains via the initial positions of spell-out domains; otherwise, a moved head is predicted to be pronounced multiple times while a moved phrase is predicted either to be pronounced multiple times or to cause the linearization process to crash depending on the specific languages.

7.2 Predictions about verb movement

In the previous chapters, I have shown that movement in (11) results in multiple pronunciation while movement in (12) results in single pronunciation.

(11) Movement that happens across different spell-out domains via non-initial positions in the spell-out domains

(12) Movement that happens in the same spell-out domain or across different spell-out domains via the initial positions in the spell-out domains

Concretely, focusing on the verb movement cases, the seemingly "long distance/non-local" topicalization movement (i.e., verb-doubling cases) can be a case of movement in (11); while the seemingly "short distance/local" T^0-to-C^0 movement, V^0-to-v^0 movement, etc. are cases of movement in (12). This is summarized in Table 7.2.

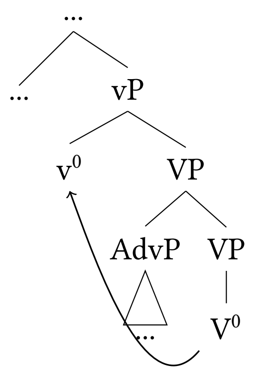
	Movement in (11) (multiple pronunciation)	Movement in (12) (single pronunciation)
long movement (non-local)	topicalization (i.e., verb-doubling)	
short movement (local)		T^0-to-C^0 movement, V^0-to-v^0

Table 7.1: Movements

However, it is worth pointing out that movement in (11) is not equivalent to "long/non-local movement"and (12) is not equivalent to "short/local-movement". To be more specific, the analysis predicts that multiple pronunciation is forced if movement happens in the way defined in (11), and single pronunciation is forced if movement happens in the way defined in (12), regardless of whether the movement is long or short. A hypothetical example of movement being short and is of movement in (11) is shown in (13).[1]

[1]It is not clear to me whether there is empirical data that has the structure in (13).

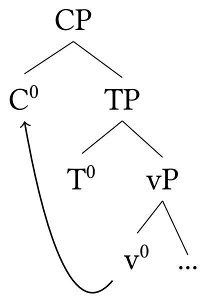
(13)



In (13), the V^0-to-v^0 movement is short/local but this movement is across a Spell-out domain via a non-initial position (the initial position in (13) is AdvP). In this case, since this movement is of the movement in (11), it is predicted that there should be multiple pronunciations of the verb, though in this case the movement is short/local.

On the other hand, a hypothetical example in (14), where the v^0-to-C^0 movement is long/non-local but is of movement in (12) (i.e., movement happens in the same Spell-out domain), should be predicted by the analysis to have a single pronunciation.

(14)



A possible candidate for the hypothetical example in (14) is the Bulgarian data (48) in Chapter 1, repeated below in (15). In this case, it seems that there is v^0-to-C^0 movement, across the auxiliary *sŭm* "I.have", which is under T^0 (cf. Rivero (1994)).

(15) **Bulgarian**

a.

Pročel	sŭm	knigata
read	I.have	book.the

‘I have read the book.’

b. *Sŭm **Pročel** knigata (Rivero 1994: 87, ex. 34)

Assuming that Rivero (1994) is correct about v^0-to-C^0 being long/non-local movement, to explain the single pronunciation of the verb in (15) under my analysis, I need to assume that this movement happens in the same Spell-out domain (perhaps the CP Spell-out domain). If this v^0-to-C^0 movement indeed happens in the same Spell-out domain, my analysis predicts that it should be pronounced once; otherwise, the analysis makes the wrong prediction.

	Movement in (11) (multiple pronunciation)	Movement in (12) (single pronunciation)
long movement (non-local)	topicalization (i.e., verb-doubling)	Bulgarian v^0-to-C^0?
short movement (local)	(13)?	T^0-to-C^0 movement, V^0-to-v^0

Table 7.2: Movements

Recall that I presented an observation in chapter 1, repeated below in (16). If the Bulgarian data is considered, the empirical generalization in (17) is more accurate.

(16) It seems that multiple pronunciations of a moved verb can occur in a long distance/non-local movement, for instance, topicalization movement; and a single pronunciation of a moved verb tends to occur in a short distance/local movement, for instance, T^0-to-C^0 movement.

(17) It seems that multiple pronunciations of a moved verb can occur occur in a long distance/non-local movement, for instance, topicalization movement; and a single pronunciation of a moved verb tends to occur in a short distance/local movement, for instance, T^0-to-C^0 movement, *with a possible exception of the long movement v^0-to-C^0.*

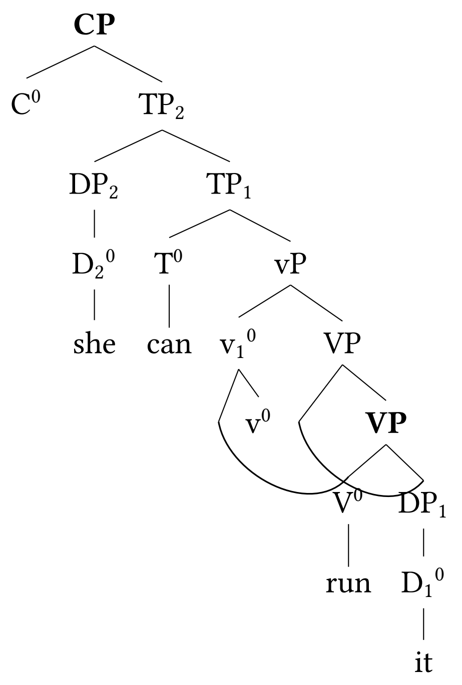
As a quick summary, the analysis predicts a moved verb to be pronounced multiple times if it moves as the way in (11) and predicts a moved verb to be pronounced once if it moves as the way in (12).

7.3 A potential problem

There is a potential problem for cases where head movement happens via the initial position of the Spell-out domain. Take the English example in (18) as an example. If the object DP moves to the initial position of the VP Spell-out domain and does not move further, the analysis predicts that the v^0-to-V^0 movement in this case will have multiple pronunciations. To be more specific, in the VP Spell-out domain, ${D_1}^0 < V^0$ is collected. In the CP Spell-out domain, ${v_1}^0 < {D_1}^0$ is collected. As a result, the Ordering Deletion rule cannot be applied to ${D_1}^0 < V^0$ because despite that ${v_1}^0$ dominates V^0, ${v_1}^0$ precedes but V^0 follows ${D_1}^0$. Thus, the ordering statements that contain V^0 and ${v_1}^0$ are all kept and both V^0 and ${v_1}^0$ are pronounced, resulting in double pronunciation.

(18) *She can run it.*

(19)



However, in the current literature, it had been argued that movement is available only as a last resort; in other words movement is only applied when it is necessary (cf. Chomsky (1991, 1998), Pesetsky (1989)). Thus, I propose that the problem in (18) can be avoided if (20) is adopted.

(20) An item X can only move to a position Y, if Y has a feature that needs to be checked by X or X needs to check the feature of Z, where Z is a position higher than Y.

In this case, I propose that the Spec, VP position does not have any feature that needs to be checked by the object phrase, and there is no higher position than Spec, VP that triggers the movement of the object phrase. Thus, it is illegitimate for the object phrase to move to Spec, VP, and the derivation in (19) is ruled out by (20) and crashes.

7.4 Open questions

7.4.1 How V(P)s move

Once a V(P) is specified regarding how to move, the analysis in this dissertation makes a prediction about whether the moved V(P) will be pronounced once or multiple times. However, it is worth noting that this dissertation does not provide an answer for why a certain V(P) moves the way it does. For instance, for the Yiddish example in (29) in chapter 6, repeated below in (21), this dissertation provides a way to explain the double pronunciation by assuming that (22) is a possible derivation but does not provide an answer for why (21) can only have the derivation in (22). To be more specific, it still needs to be explained why there cannot be $Asp_1{}^0$ -to-T^0-to-C^0-to-Top^0 movement, where $Asp_1{}^0$ moves across the Spell-out domain via an initial position and should be predicted to have a single pronunciation.[2]

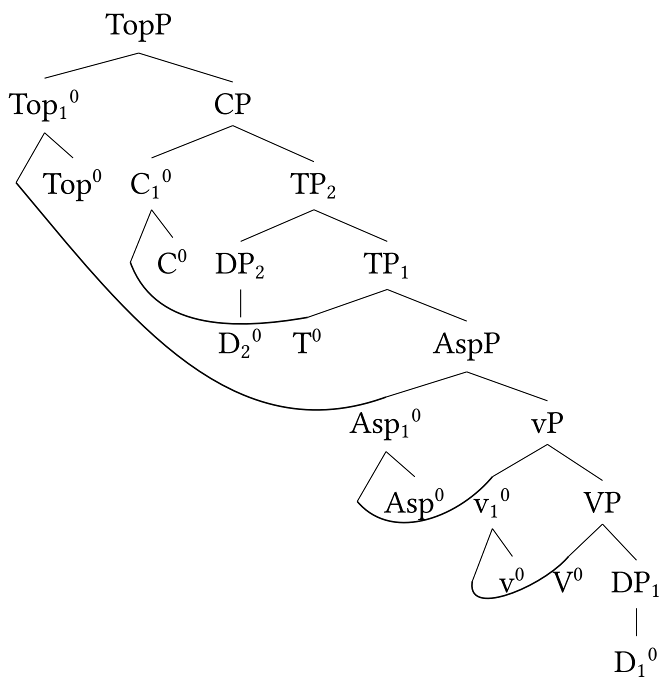
(21) **Yiddish (aspectual form)**

Gegessen,	hot Maks	**gegessen**	fish
eaten	has Max	**eaten**	fish

'As for having eaten, Max has eaten fish.'

[2]One possible way to solve this problem is to say that T^0/C^0 is occupied by the auxiliary *hot* "has" and cannot serve as a landing position for $Asp_1{}^0$. However, it still needs to be explained why the auxiliary *hot* "has" blocks $Asp_1{}^0$ -to-T^0-to-C^0-to-Top^0 movement, considering that excorporation is argued to be possible in some literature (cf. Roberts (2010)).

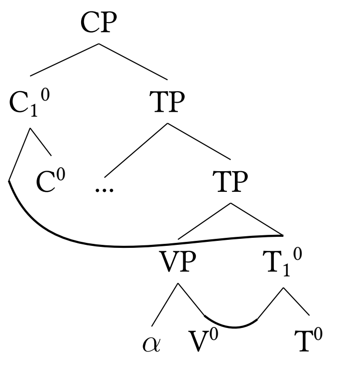
(22)



7.4.2 The problem of mixed head-final/initial

This analysis seems to work fine for languages where heads are consistently final or initial, but if a language has a mixture of head-final and head-initial structures, this analysis makes wrong predictions. One example is German. In German, C^0 is initial but V^0 is final (23).

(23)



In this case, in the VP Spell-out domain, the ordering statement $\alpha < V^0$ will be collected. In the CP Spell-out domain, $C_1{}^0 < \alpha$ will be collected. Note that despite the fact that $C_1{}^0$ dominates V^0, the ordering statement $\alpha < V^0$ cannot be deleted since $C_1{}^0$ and V^0 are ordered differently regarding α. As a result, the analysis wrongly predicts that the verb will be pronounced twice (i.e., once in ${C_1}^0$, once in V^0).[3]

One possible way to solve this problem is to change the Spell-out domains: Spell-out domains include DP, CP and the highest phrase, but not VP. In this way, since there is no VP Spell-out domain, the only domain relevant here is the CP Spell-out domain, where V^0 will not be linearized since it is dominated by ${C_1}^0$ and thus is not a Maximal X^0 node. Along this line, one has to assume that Spell-out domains can be different across languages.

[3]In the CP spell-out domain, there is T^0-to-C^0 movement, and thus T^0 will be dominated by ${C_1}^0$. As a result, T^0 will not be a Maximal X^0 node and will not be linearized.

BIBLIOGRAPHY

Bale, A. (2005). Quantifiers, again and the complexity of Verb Phrases. In E. Georgala and J. Howell (Eds.), *Proceedings of SALT XV*, Cornell University, Ithaca, NY, pp. 1–18. CLC Publications.

Beck, S. and K. Johnson (2004). Double objects again. *Linguisic Inquiry 35*(1), 97–124.

Brass, A. (1986). *Chains and Anaphoric Dependence: On Reconstruction and Its Implications,.* Ph. D. thesis, Massachusetts Institute of Technology.

Burzio, L. (1986). *Italian Syntax.* Reidel Publishers.

Cable, S. (2004). Predicate clefts and base-generation: Evidence from Yiddish and Brazilian Portuguese. Ms., MIT.

Chafe, W. (1976). Givenness. contrastiveness, definiteness, subjects, topics and point of view. In C. N. Li (Ed.), *Subject and Topic*, pp. 27–55. New York: Academic Press.

Chomsky, N. (1981). *Lectures on Government and Binding.* Dordrecht: Foris.

Chomsky, N. (1986). *Barriers.* Cambridge, Massachusetts: MIT Press.

Chomsky, N. (1991). Some notes on economy of derivation and representation. In R. Freidin (Ed.), *Principles and Parameters in Comparative Grammar*, pp. 417–454. Cambridge, Massachusetts: MIT Press.

Chomsky, N. (1993). A minimalist program for linguistic theory. In K. Hale and S. Keyser (Eds.), *The View from Building 20: Essays in Honor of Sylvain Bromberger*, Number 1–52. Cambridge, Massachusetts: MIT Press.

Chomsky, N. (1995). *The Minimalist Program.* Cambridge, MA: The MIT Press.

Chomsky, N. (1998). Some observations on economy in generative grammar. In P. Barbosa, D. Fox, P. Hagstrom, M. McGinnis, and D. Pesetsky (Eds.), *Is the best good enough?*, pp. 115–128. Cambridge, Massachusetts: MIT Press.

Chomsky, N. (2000). Minimalist inquiries: The framework. In R. Martin, D. Michaels, and J. Uriagereka (Eds.), *Step by step: Essays on minimalist syntax in honor of Howard Lasnik*, Number 89-155. Cambridge, Mass: MIT Press.

Chomsky, N. (2001). Derivation by phase. In M. Kenstowicz (Ed.), *Ken Hale: A life in language*, Volume 1-52. Cambridge, Mass: MIT Press.

Citko, B. (2005). On the nature of merge: External merge, internal merge, and parallel merge. *Linguistic Inquiry 36*(4), 475–496.

Doron, E. (1999). V-movement and VP ellipsis. In S. Lappin and E. Benmamoun (Eds.), *Fragments: Studies in ellipsis and gapping*, pp. 124–140.

Embick, D. and R. Noyer (2001). Movement operations after syntax. *Linguisic Inquiry 32*, 555–595.

Engdahl, E. (1980). *The Syntax and Semantics of Questions in Swedish.* Ph. D. thesis, University of Massachusetts, Amherst, Massachusetts.

Fitzpatrick, J. and E. Groat (2005). The timing of syntactic operations: Phases, c-command, remerger, and Lebeaux effects. paper presented at ECO5.

Fox, D. (1999). Reconstruction, binding theory, and the interpretation of chains. *Linguistic Inquiry 30*(157–196).

Fox, D. and D. Pesetsky (2005). Cyclic linearization of syntactic structure. *Theoretical Linguistics 31*, 1–45.

Frampton, J. (2004). Copies, traces, occurrences, and all that: Evidence from Bulgarian multiple wh-phenomena. unpublished manuscript, Northeastern University.

Fukui, N. and Y. Takano (1998). Symmetry in syntax: Merge and demerge. *Journal of East Asian Linguistics 7*(1), 27—86.

Gärtner, H.-M. (1997). *Generalized Transformations and Beyond.* Ph. D. thesis, University of Franfurt/Main.

Halle, M. and A. Marantz (1993). Distributed morphology and the pieces of inflection. In K. Hale and S. Keyser (Eds.), *The view from building 20*, Number 111–176. The MIT Press.

Hein, J. (2018). *Verbal Fronting: Typology an Theory.* Universität Leipzig.

Hornstein, N., J. Nunes, and K. K. Grohmann (2005). *Understanding minimalism.* Cambridge Textbooks in Linguistics. Cambridge: Cambridge University Press.

Johnson, K. (1991). Object positions. *Natural Language and Linguistic Theory 9*(4), 577–636.

Johnson, K. (2004, Fall). Introduction to transformational grammar. Lecture notes collection used at University of Massachusetts at Amherst.

Johnson, K. (2012). Toward deriving differences in how *Wh* movement and QR are pronounced. *Lingua 122*(6), 529–553.

Johnson, K. (2016, 06). Towards a multidominant theory of movement. Lecture notes used at University College London.

Johnson, K. (2018). To give someone their innocence again. In R. Ivan (Ed.), *UMOP 40: The Leader of the Pack: A Festschrift in Honor of Peggy Speas*, Volume 40. Graduate Linguistics Student Association.

Kayne, R. S. (1994). *The Antisymmetry of Syntax*, Volume 25 of *Linguistic Inquiry Monographs*. Cambridge, MA: MIT Press.

Ko, H. (2005). *Syntactic edges and linearization.* Ph. D. thesis, Massachusetts Institute of Technology.

Koopman, H. (1984). *The Syntax of Verbs.* Dordrecht: Foris.

Kratzer, A. (1996). Severing the external argument from its verb. In J. Rooryck and L. Zaring (Eds.), *Phrase Structure and the Lexicon*, pp. 109–137. Dordrecht, The Netherlands: Kluwer Academic Publishers.

Kusmer, L. (2019). *OPTIMAL LINEARIZATION: PROSODIC DISPLACEMENT IN KHOEKHOE-GOWAB AND BEYOND.* Ph. D. thesis, University of Massachusetts Amherst.

Landau, I. (2006). Chain resolution in Hebrew V(P)-fronting. *Syntax 9*(1), 32–66.

Lebeaux, D. (1991). Relative clauses, licensing, and the nature of the derivation. In S. D. Rothstein (Ed.), *Perspectives on Phrase Structure: Heads and Licensing*. San Diego: Academic Press.

Manfredi, V. (1993). Verb focus in the typology of Kwa/Kru and Haitian. In F. Byrne and D. Winford (Eds.), *Focus and grammatical relations in Creole languages*, pp. 3–51. John Benjamins.

May, R. (1985). *Logical Form: Its Structure and Derivation.* Cambridge, Massachusetts: MIT Press.

Nunes, J. (2001). Sideward movement. *Linguistic Inquiry 32*(2), 303–344.

Nunes, J. (2004). *Linearization of chains and sideward movement.* Cambridge: MIT Press.

Pesetsky, D. (1989). The Earliness Principle. Paper presented at GLOW.

Pollock, J.-Y. (1989). Verb movement, Universal Grammar and the structure of IP. *Linguistic Inquiry 20*(3), 365–424.

Rivero, L. M. (1994, 02). Clause structure and V-movement in the languages of the Balkans-movement in the languages of the balkans. *Natural Language and Linguistic Theory 12*(1), 63–120.

Roberts, I. (2010). *Agreement and head movement: clitics, incorporation, and defective goals.* The MIT Press.

Sabbagh, J. (2003). Ordering and linearizing rightward movement. Ms.

Selkirk, E. (1996). The prosodic structure of function words. In J. L. Morgan and K. Demuth (Eds.), *Signal to syntax: Bootstrapping from speech to grammar in early acquisition*, pp. 187—213. Lawrence Erlbaum Associates, Inc.

Sichel, I. (2014, Fall). Resumptive pronouns and competition. *Linguistic Inquiry 45*(4), 655–693.

Starke, M. (2001). *Move Dissolves into Merge: A Theory of Locality.* Ph. D. thesis, University of Geneva.

Stowell, T. (1981). *Origins of phrase structure.* Ph. D. thesis, Massachusetts Institute of Technology, Cambridge.

Thoms, G. (2010). *Poetic language: a Minimalist theory.* Ph. D. thesis, University of Strathclyde.

Travis, L. (1984). *Parameters and Effects of Word Order Variation.* Ph. D. thesis, MIT.

Wilder, C. (1999). Right node raising and the LCA. In S. Bird, A. Carnie, J. D. Haugen, and P. Norquest (Eds.), *Proceedings on the Eighteenth West Coast Conference on Formal Linguistics*, pp. 586–598.