Acquiring Minimalist Grammars via Constraint Satisfaction

by

Sagar Indurkhya

S.B., C.S. M.I.T. 2012

Submitted to the
Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

Massachusetts Institute of Technology

July 2015
[September 2015]

Author: Signature redacted
Sagar Indurkhya, Department of Electrical Engineering and Computer Science
July 23, 2015

Certified by: Signature redacted
Robert C. Berwick, Prof. of Computational Linguistics, Thesis Supervisor
July 23, 2015

Accepted by: Signature redacted
Prof. Christopher J. Terman, Master of Engineering Thesis Committee

Acquiring Minimalist Grammars via Constraint Satisfaction

By Sagar Indurkhya
Submitted to the Department of Electrical Engineering and Computer Science

July 27, 2015
In Partial Fulfillment of the Requirements for the Degree of Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

This thesis shows how to algorithmically construct a Minimalist Grammar lexicon that produces a specified set of MG derivations. This thesis introduces a mathematical structure, a Collection of Constraints, that captures the logical constraints, including those that arise as a consequence of the shortest move constraint, imposed upon the syntactic features of lexical items as they are merged together in a derivation produced by a given Minimalist Grammar lexicon. Methods are then developed that (a) map Minimalist Grammar lexicons to be Collections of Constraints, (b) map Collections of Constraints to Minimalist Grammar lexicons and (c) may combine two or more Collections of Constraints into a single Collection of Constraints. The thesis then demonstrates via a series of examples, framed as a simplified acquisition process, how these methods may be used together to iteratively construct a Minimalist Grammar lexicon starting from an empty Collection of Constraints and a sequence of Minimalist Grammar derivations, such that the constructed lexicon is able to generate the set of derivations.

Dedication

This thesis is dedicated to my parents, Gopal and Vandana Indurkhya.

Acknowledgements

I would like to begin by expressing gratitude to my dear family for their love and support throughout the completion of this thesis. I would also like to give a special thank you to several friends and colleagues for their advice, suggestions and encouragement: Apoorva Murarka, Ramita Arora, Dr. Anselm Levskaya, Austen Heinz, Prof. Gerald J. Sussman, Dr. Thomas F. Knight and Prof. Patrick H. Winston. Finally, I would like to express my deep gratitude to my advisor, Prof. Robert C. Berwick, both for introducing me to the subject of *Minimalist Grammars* and for patiently providing me with guidance, motivation and insight throughout the completion of this thesis.

Contents

1 Introduction **11**

1.1 Background 12

1.2 Minimalist Grammars 14

1.3 Outline of the Thesis 16

2 Combining MG Lexicons via Constraint Satisfaction **19**

2.1 Constraints between Syntactic Features 20

2.2 Collections of Constraints 22

2.3 Constructing a Curated Lexicon from a Collection of Constraints 24

2.4 Correspondences between Classes of Lexicons and Collections of Constraints 29

2.5 Combining Collections of Constraints 30

3 Acquisition of an MG Lexicon **33**

3.1 The cat will eat a mouse. 35

3.2 Will a cat eat the mouse? 38

3.3 Which mouse will the cat eat? 43

3.4 Which cat will eat the mouse? 48

3.5 The man will think that the cat will eat a mouse. 53

3.6 Will the man think that the cat will eat a mouse? 59

3.7 Which mouse will the man think that the cat will eat? 65

4 Conclusion **71**

4.1 Summary . . . 71

4.2 Future Work . . . 72

Chapter 1

Introduction

This thesis shows how to algorithmically construct a *Minimalist Grammar* (MG) lexicon that produces a specified set of MG derivations.

I begin by establishing a mathematical framework in which two (or more) MG lexicons may be combined to produce an MG lexicon that generates a superset of the *complete* derivations generated by each of the constituent lexicons. I will introduce a mathematical structure, a *Collection of Constraints*, that captures the logical constraints imposed upon the syntactic features of lexical items as they are *merged* together in a *derivation* produced by a given *lexicon.*[1] After developing a number of properties that this structure exhibits, I will provide and analyze both an algorithm, $\mathbb{P}$, for computing the *Collection of Constraints* from an MG lexicon as well as an algorithm, $\mathbb{Q}$, for constructing an MG lexicon from a *Collection of Constraints.* I will then establish a bijection between the set of *classes of curated MG lexicons*[2] and the set of *Collections of Constraints* that may be computed from a *curated* MG lexicon.[3]

I will then establish a procedure $\mathbb{J}$ that (a) determines whether it is valid to combine several given *Collections of Constraints* into a single *Collection of Constraints* and (b) if such a combination of constraints is valid computes the combined structure. Thus, given a set of MG derivations, I will map each derivation to a *Collection of Constraints* encoding the constraints imposed by that derivation. If it is possible to combine these structures [4] (i.e. the specified set of derivations is not inconsistent) then they are combined into a single *Collection of Constraints* that encodes the constraints imposed by all of specified derivations. Finally, from this aggregate structure an MG lexicon may be constructed that is able to generate the entire set of specified derivations.

[1] In this thesis I will restrict the MG lexicons under consideration to (a) those that do not have any redundant phonetic forms and (b) those in which every lexical item participates in at least one *complete* derivation. See Definition 8.

[2] I will then establish an equivalence relation between lexicons: a *Class of Lexicons.* Two lexicons are said to be in the same class if and only if they produce the same set of *complete* derivations. See Definition 13.

[3] The maps for this bijection are provided by $\mathbb{P}$ and $\mathbb{Q}$.

[4] One particularly interesting reason it may not be possible to combine several MG lexicons is if they have conflicting *anti-licensing constraints*, which are constraints encoding the restrictions imposed upon the *merge* operation by the shortest move constraint. An advantage of the *Collection of Constraints* representation is the set of *anti-licensing constraints* are explicitly enumerated and accessible; hence, validating that *licensing* and *anti-licensing* constraints do not conflict is a relatively simple operation.

- "The cat will eat a mouse."
- "Will the cat eat a mouse?"
- "Which mouse will the cat eat?"
- "Which cat will eat the mouse?"
- "The man will think that the cat will eat the mouse."
- "Will the man think that the cat will eat the mouse?"
- "Which mouse will the man think that the cat will eat?"

Figure 1.1: The set of sentences produced by the seven derivations enumerated in Chapter 3. The methods developed in Chapter 2 will be used in Chapter 3 to construct a lexicon that may generate the derivations associated with these sentences. The choice of example sentences follows Winston's principle that a learner may only learn that which he almost already knows; the sentences the learner encounters have been carefully selected to differ only slightly from one to another in their respective set of syntactic feature constraints.

I will then exhibit an example of how such an MG lexicon may be constructed. I will provide a set of MG derivations that correspond to the sentences in Figure 1.1 and compute the *Collection of Constraints* for each derivation. I will then aggregate the *Collections of Constraints* into a single *Collection of Constraints*, and demonstrate how the MG lexicon in Figure 1.2, which is able to generate all of the derivations in the specified set, may be constructed from the aggregate *Collection of Constraints*. This process may be interpreted as a simplified *acquisition* process in which the learner begins with an empty *Collection of Constraints*, corresponding to no prior knowledge, and is then be sequentially exposed to a series of MG derivations, acquiring new knowledge in the form of accumulating new constraints. In keeping with this interpretation, I will carefully point out when new constraints may be deduced only from the combination of two *Collections of Constraints* and which new MG derivations may be produced due to the introduction of additional constraints. I will also use this acquisition framework to investigate whether the (linguistically) appropriate generalizations may be learned: specifically I will study the phenomenon of successive cyclic wh-movement.

1.1 Background

The Minimalist Program (MP) [3] puts forward the hypothesis that the human language faculty has the minimalist properties of perfection (i.e., universal grammar has a computationally optimal design) and economy (e.g., economy of derivation and representation). Inquiry guided by MP studies what aspects of a theory of generative grammar may be explained by the minimalist properties of the human language faculty, a hypothesis in part inspired by arguments from biolinguistics. [5] Some aspects of early theories of grammar

[5] Prior the Minimalist Program, generative grammar research cumulated in the *Principles and Parameters* framework, in which theories of generative linguistics are described by a set of principles (i.e. grammatical rules or laws) that are universal to all natural languages as well as a finite set of binary parameters that allow for the observed diversity in the syntactic structure of natural languages. From a philosophical perspective, the development of the *Minimalist Program* reflects a shift from addressing *Plato's problem*, which examines how we may account for our tacit knowledge of language despite a poverty of relevant primary linguistic data, to *Darwin's problem*, which addresses the question of how the faculty of language arose within

	$a :: = z1, \sim z2$
	$cat :: \sim z1$
	$eat :: = z2, \sim z6$
	$man :: \sim z1$
	$mouse :: \sim z1$
$\epsilon_C :: = z3, C$	$that :: = z3, \sim z4$
$\epsilon_{C_2} :: = z3, +z8, C$	$that2 :: = z3, +z7, \sim z4$
$\epsilon_{C_3} :: = z3, +z8, +z9, C$	$the :: = z1, \sim z2$
$\epsilon_T :: = z5, = z6, +z10, = z2, \sim z3$	$think :: = z4, \sim z6$
	$which :: = z1, \sim z2, -z9$
	$which2 :: = z1, \sim z2, -z7, -z9$
	$will :: \sim z5, -z10$
	$will2 :: \sim z5, -z10, -z8$

Figure 1.2: This lexicon is able to generate all of the specified derivations that may be found in Chapter 3. Note that due to some restrictions on the form of the lexicons I will consider in this thesis, lexical items with the same phonetic form must have their phonetic forms made distinct in an artificial manner (via indices/subscripts). (See Definition 8) Note that lexical items with the same syntactic features tend to also fall into the same part-of-speech classes. For example, the lexical items for "the" and "a" both have the same syntactic features and thus they are effectively interchangeable within derivations generated by this lexicon, which is expected as they are both determiners. Similarly, the lexical items for the nouns "man", "cat" and "mouse" all have the same syntactic features and are interchangeable. Also note that lexical items that are variations of the same phonetic form (e.g. ϵ_C, ϵ_{C_2} and ϵ_{C_3}) have syntactic feature sequences that are slight variations of each other; this demonstrates that they are related but not identical in functionality (and thus not freely interchangeable). The procedure by which this lexicon is assembled from a set of derivations is worked out in detail in Chapter 3.

developed within the MP have been modeled by Stabler in a formalism known as *Minimalist Grammars.* [15]

Minimalist grammars are a class of formal languages that are mildly context-sensitive [6] [14] and have been shown to be strongly equivalent to *multiple context free grammars.* [9, 18] The formalism has been extended to handle linguistic phenomenon such as head movement, affix hopping and adjunction. [5, 7, 16] Extensive work has been carried out in developing parsing methodologies for both the standard MG formalism (including top-down, bottom-up and chart parsing) [9] as well as various extensions of the formalism. [19] *Minimalist Grammars* have also been assigned probabilistic models, both directly and via translation to *multiple context-free grammars.* [8, 10] Kobele has developed a compositional semantics for the standard MG formalism. [12] It has been shown that constraints upon *derivations* that fall within the domain of second order monadic logic may be encoded as extensions of the MG feature system. [6]

1.2 Minimalist Grammars

This section provides a formal definition of Minimalist Grammars following the *chain*-based model as set out by Stabler and Keenan in [17] along with an example of both a *lexicon* and a *complete derivation* produced by the said lexicon to illustrate the definitions laid out in this section.

Definition 1 (Syntactic Feature)
A syntactic feature *is a symbol (often referred to as the* underlying *or* base *symbol) that belongs to one of the four disjoint subtypes:* selectors, selectees, licensors *and* licensees. *A feature that is a* selector *(written as* $= x_i$*) may* select *a feature that is a* selectee *(written as* $\sim x_i$*). A feature that is a* licensor *(written as* $+x_i$*) may* license *a feature that is a* licensee *(written as* $-x_i$*).* [7] □

Definition 2 (Chain)
A chain *is defined as a a* phonetic form *paired with a sequence of* syntactic features *to be consumed (via feature matching) in order. A* chain *that has not yet had a syntactic feature consumed is a* lexical chain, *which is indicated by a double-colon between the phonetic form and syntactic features. (e.g.* $[the :: = x, \sim y]$*) A* chain *that has already had one or more of its syntactic features consumed is a* derived chain, *which is denoted by a single-colon between the phonetic form and syntactic features.* □

Definition 3 (Lexicon)
An MG lexicon *is a tuple* (Σ, X, Lex) *where* Σ *is a non-empty alphabet,* X *is a finite, non-empty set of* syntactic features *and Lex is a non-empty set of* chains *of type* $[\Sigma^* :: X^* * X^+]$*. For an example of a* lexicon *see Figure 1.3.* □

---

the species.

[6] It has been proposed that *mildly context sensitive grammars* as a class of grammars characterize the complexity of natural languages. See [11]

[7] Note that licensees are traditionally written without any prefix symbol (e.g. x_i). Here we assign a distinct prefix as it will frequently be necessary to distinguish between the underlying symbol (x_i) and the syntactic feature $-x_i$.

$\epsilon_C :: = x10, +x7, +x6, C$	$eat :: = x2, \sim x4$
$\epsilon_T :: = x3, = x4, +x5, = x9, \sim x10$	$will :: \sim x3, -x5, -x7$
$mouse :: \sim x1$	$cat :: \sim x8$
$which :: = x1, \sim x2, -x6$	$the :: = x8, \sim x9$

Figure 1.3: A lexicon from which the interrogative "which mouse will a cat eat?" may be derived. Every syntactic feature in this lexicon, with the exception of the special symbol C, may *select* or *license* only one other syntactic feature. Hence, this lexicon can generate at most one derivation; in fact, this lexicon exactly one *complete* derivation (see Figure 1.4). Given a set of such lexicons, the methods developed in Chapter 2 may be used to combine these lexicons into a single lexicon (see Figure 1.2).

Definition 4 (Derivation Term)
A derivation term *is a non-empty ordered list of* chains. *The first* chain *designated as the* head *of the term. A* derivation term *may be constructed by recursive application of binary composition operation,* merge. □

Definition 5 (Merge)
The merge *operation is a recursive function that attempts to combine two* derivation terms *to produce another* derivation term. *In the context of derivation terms being recursively constructed from a common collection of lexical atoms, the* merge *operation has two disjoint subcases: either both arguments of* merge *are disjoint from one another, in which case the operation is denoted* external merge, *or one of the arguments is a subset of the other, in which case the operation is denoted* internal merge. [8]

Let $s, t \in \Sigma^*$, $f, g \in X$, $\gamma, \phi, \psi \in X^*$, $\delta \in X^+$. *Furthermore,* $\alpha_1, ..., \alpha_k$ *and* $\iota_1, ..., \iota_l$ *are chains for* $0 \leq k, l$. *the separation of the phonetic form and syntactic features by the symbol* $\cdot$ *designates that the chain may be* lexical *or* derived*). External Merge has three disjoint subcases (*EM_1, EM_2 *and* EM_3*), all of which involve feature* selection*:*

$$\frac{[s :: \phi \ * = f, \ \gamma] \qquad [t \cdot \psi \ * \sim f], \ \iota_1, ..., \iota_l}{[st : \phi, = f \ * \gamma], \ \iota_1, ..., \iota_l} \ EM1$$

$$\frac{[s : \phi \ * = f, \ \gamma], \ \alpha_1, ..., \alpha_k \qquad [t \cdot \psi \ * \sim f], \ \iota_1, ..., \iota_l}{[ts : \phi, = f \ * \gamma], \ \alpha_1, ..., \alpha_k, \iota_1, ..., \iota_l} \ EM2$$

$$\frac{[s \cdot \phi \ * = f, \ \gamma], \ \alpha_1, ..., \alpha_k \qquad [t \cdot \psi \ * \sim f, \ \delta], \ \iota_1, ..., \iota_l}{[s : \phi, = f \ * \gamma], \ \alpha_1, ..., \alpha_k, \ [t : \psi, \sim f \ * \delta], \ \iota_1, ..., \iota_l} \ EM3$$

The internal merge *operation consists of the following two disjoint cases (*IM_1 *and* IM_2*) involving feature* licensing*:*

$$\frac{[s : \phi \ * + f, \ \gamma], \ \alpha_1, ..., \alpha_{i-1}, \ [t : \psi \ * - f], \ \alpha_{i+1}, ..., \alpha_k}{[ts : \phi, \ + f \ * \gamma], \ \alpha_1, ..., \alpha_{i-1}, \alpha_{i+1}, ..., \alpha_k} \ IM1$$

[8] *Internal Merge* is sometimes referred to as *Move*, referring to the role it plays in enabling movement.

$$\frac{[s:\phi\ {}^*{+}f,\ \gamma],\ \alpha_1,...,\alpha_{i-1},\ [t:\psi,\ {}^*{-}f,\ \delta],\ \alpha_{i+1},...,\alpha_k}{[s:\phi,\ {+}f\ {}^*\gamma],\ \alpha_1,...,\alpha_{i-1},\ [t:\psi,\ {-}f\ {}^*{-}f,\ \delta],\ \alpha_{i+1},...,\alpha_k}\ IM2$$

Both subcases of internal merge *(IM_1 and IM_2) are assumed to be subject to the* shortest move constraint *(SMC): licensing must be deterministic; that is, when a licensor binds to a licensee, it must be the case that that is the only licensee to which it can bind.* □

Definition 6 (Derivation)
A derivation *is a logical deduction process in which a finite collection of lexical items (i.e. single chain* derivation term*) are recursively combined via the* merge *operation into a single* derivation term. *A* complete derivation *is a* derivation *in which the final* derivation term *has a single* syntactic feature *left to consume and that feature is the special symbol C.* □

See Figure 1.4 for an example of a *derivation* generated by the lexicon in Figure 1.3 that engages all three cases of *external merge* and both cases of *internal merge.*

1.3 Outline of the Thesis

In this section I will give a brief outline of the remainder of this thesis.

In Chapter 2, I will first formally define the logical constraints that relate the syntactic features that belong to the constituent lexical items of a lexicon. I will then define a mathematical structure, a *Collection of Constraints*, that will encode these logical constraints. I will also derive a number of properties that hold for these structures. (See Section 2.1) After that I will provide both a procedure for constructing a lexicon from a *Collection of Constraints* as well a procedure for computing the *Collection of Constraints* for a given lexicon. (See Section 2.2 and Section 2.3) I will then establish a bijection between *classes of curated lexicons* and the set of *Collections of Constraints* that may be computed from an MG lexicon. (See Section 2.4) Finally, I will describe conditions under which multiple *Collections of Constraints* may be combined to form an aggregate *Collection of Constraints* and then provide a procedure for constructing such a combined *Collection of Constraints*; this procedure will serve as the basis for the acquisition example in Chapter 3. (See Section 2.5)

Chapter 3 provides an example of how, for a given set of MG derivations, an MG lexicon may be constructed in an iterative fashion by mapping each derivation to a *curated* lexicon and then combining these lexicons via the procedure $\mathbb{J}$ into an aggregate MG lexicon. Each section of Chapter 3 will introduce a new MG derivation, map it to a *Collection of Constraints* and then combine it with the *Collection of Constraints* that was computed in the previous section. In each section I will also map the aggregate *Collection of Constraints* to an MG lexicon so that the reader has a clear picture of how the MG lexicon is being built step by step. In particular, emphasis is given on studying how the learner may capture generalizations.

Finally, Chapter 4 provides a discussion of both the results derived in Chapter 3 as well as relevant topics of interest for future investigation.

Figure 1.4: This is the only derivation that may be generated from the lexicon in Figure 1.3. The sequence of derivation steps is ordered by dependency. The spellout of this derivation is: "which mouse will the cat eat?"

Chapter 2

Combining MG Lexicons via Constraint Satisfaction

In this chapter, I will develop a mathematical framework for representing Minimalist Grammar lexicons in terms of the constraints between the syntactic features of a lexicon. Section 2.1 establishes formal definitions for these constraints and how they appear in *complete* derivations produced by lexicons. In Section 2.2 and Section 2.3, I will provide procedures for both computing these constraints from lexicons and computing lexicons from constraints. In Section 2.4, I will establish that these procedures are bijections. Finally, in Section 2.5 I will develop a procedure that can recursively combine lexicons; this procedure will serve as the basis for the acquisition example in Chapter 3.

2.1 Constraints between Syntactic Features

In this section I will introduce a restriction on the form of an MG lexicon (See Definition 8) and then define four types of logical relations (constraints) between pairs of syntactic features: *selection constraints*, *licensing constraints*, *start constraints* and *anti-licensing constraints.* (See Definitions 9-12) I will then illustrate how these constraints may be derived from the *external* and *internal merge* operations that are part of an MG derivation. Finally, I will define an equivalence class over the set of *curated* MG lexicons. (See Definition 13)

Definition 7
Given an MG lexicon L, the function $\mathbb{G}(L)$ denotes the set of complete *derivations derived from L.* [1] □

Definition 8 (Curated Lexicon)
An MG lexicon L of the form $L = \{p_i :: x_{1,i}, x_{2,i}, x_{3,i}, ..., x_{k_i,i} | i \in [1 \dots q]\}$ is a curated *lexicon if and only if the following two conditions are met:*

(a) The phonetic forms $p_i \in P$ form a set.

(b) Every lexical item is a subconstituent of at least one complete *derivation generated by L.*

Since P is a set, the syntactic feature $x_{j,i}$ may be uniquely referenced by the tuple (p_i, j). □

Definition 9 (Selection Constraint)
Let L be a curated *lexicon with distinct syntactic features (p_x, i) and (p_y, j). Suppose there is a* complete *derivation D produced by L such that there is a* merge *operation in D in which (p_x, i)* selects *(p_y, j). Then the the tuple of syntactic features $((p_x, i), (p_y, j))$ is a* selection constraint *with respect to L.* □

Definition 10 (Licensing Constraint)
Let L be a curated *lexicon with distinct syntactic features (p_x, i) and (p_y, j). Suppose there is a* complete *derivation D produced by L such that there is a* merge *operation in D in which (p_x, i)* licenses *(p_y, j). Then the the tuple of syntactic features $((p_x, i), (p_y, j))$ is a* licensing constraint *with respect to L.* □

Definition 11 (Anti-Licensing Constraint)
Given a curated lexicon L with distinct syntactic features (p_x, i), (p_y, j) and (p_z, k), if there is some complete *derivation $D \in \mathbb{G}(L)$ in which the shortest move constraint prohibits (p_x, i) from* licensing *(p_y, j) because (p_x, i)* licenses *(p_z, k), then the tuple of syntactic features $((p_x, i), (p_y, j))$ is an* anti-licensing constraint *with respect to L.* □

Definition 12 (Start Constraint)
Given a curated lexicon L with a syntactic feature (p, i) that corresponds to the special symbol C, if there is no syntactic feature $(p, i + j)$ for $j > 0$ in L then (p, i) is a start constraint *with respect to L.* □

[1] Note that $\mathbb{G}(L)$ is an enumerable set.

Having introduced these four types of logical relations, I will now address how these relations may be computed given a *complete* derivation, D, generated from a *curated* Lexicon L. (i.e. $D \in \mathbb{G}(L)$) Then each derivation step[2] $d_i \in D$ imposes *constraints* upon the syntactic features involved in that step. In the case of external merge, the derivation step takes the form:

$$\frac{([p_i \cdot x_{1,i}, ..., x_{m_i-1,i} * x_{m_i,i}, ..., x_{k_i,i}], ...) \qquad ([p_j \cdot x_{1,j}, ..., x_{m_j-1,j} * x_{m_j,j}, ..., x_{k_j,j}], ...)}{([\square \cdot x_{1,i}, ..., x_{m_i,i} * x_{m_i+1,i}, ..., x_{k_i,i}], ...)} \text{ EM}$$

where $1 \leq m_i \leq k_i$ and $1 \leq m_j \leq k_j$. Since the syntactic feature (p_i, m_i) *selects* the syntactic feature (p_j, m_j), the tuple $((p_i, m_i), (p_j, m_j))$ is a *selection constraint* with respect to L. In the case of internal merge, the derivation step takes the form:

$$\frac{([p_i \cdot x_{1,i}, ..., x_{m_i-1,i} * x_{m_i,i}, ..., x_{k_i,i}], ..., [p_j \cdot x_{1,j}, ..., x_{m_i-1,j} * x_{m_i,j}, ..., x_{k_i,j}])}{([\square \cdot x_{1,i}, ..., x_{m_i,i} * x_{m_i+1,i}, ..., x_{k_i,i}], ...)} \text{ IM}$$

where $1 \leq m_i \leq k_i$ and $1 \leq m_j \leq k_j$. Since the syntactic feature (p_i, m_i) *licenses* the syntactic feature (p_j, m_j), the tuple $((p_i, m_i), (p_j, m_j))$ is a *licensing constraint* with respect to L. Furthermore, as a result of the shortest move constraint, we know that for $h \neq j$, the syntactic feature (p_i, m_i) *cannot license* the syntactic feature (p_h, m_h); hence $((p_i, m_i), (p_h, m_h))$ is an *anti-licensing constraint* with respect to L.

In the case of both internal and external merge, if $m_i + 1 = k_i$ and the syntactic feature (p_i, m_i) is the special symbol C, then we say that the tuple (p_i, m_i) is a *start constraint* with respect to lexicon L.

The next definition is an equivalence class that captures the functional equivalence of two MG lexicons that may be superficially different. (i.e. two lexicons may have very different symbols assigned to their syntactic features but generate equivalent sets of *complete* derivations)

Definition 13 (Class of Lexicons)
Two curated lexicons L_1 and L_2 are in the same class of lexicons *(denoted $L_1 \diamond L_2$) if and only if they generate identical sets of* complete *derivations. i.e.* $\mathbb{G}(L_1) = \mathbb{G}(L_2)$ $\square$

[2]Note that since D is *complete*, D consists of a sequence of derivation steps, d_i, that must eventually terminate and thus the sequence of steps is finite.

2.2 Collections of Constraints

In this section I will introduce a new representation: a *Collection of Constraints* (See Definition 14). Given a curated lexicon L, the *collection of constraints* with respect to L captures the constraints between syntactic features that are satisfied in $\mathbb{G}(L)$. I then outline a procedure for computing the *collection of constraints* for a given *curated* lexicon. (See Procedure 1) Finally I prove a number of properties that hold for this new representation. (See Proposition 1-3)

Definition 14 (Collection of Constraints)
Given a curated lexicon L, the collection of constraints *with respect to L is the tuple $(C_S, C_L, C_A, C_{Start})$, where C_S is the set of all* selection *constraints with respect to L, C_L is the set of all* licensing *constraints with respect to L, C_A is the set of all* anti-licensing *constraints with respect to L, and C_{Start} is the set of all* start *constraints with respect to L. Note that these constraints do not in any way involve the particular linearization scheme introduced in subcasing* external merge *into the three subcases EM_1, EM_2, EM_3.* □

Procedure 1 (Constructing a collection of constraints from a curated lexicon.)
Given a curated lexicon L, we may compute the collection of constraints *with respect to L as:*

$$\left(\bigcup_{d\in\mathbb{G}} X_S(d), \bigcup_{d\in\mathbb{G}} X_L(d), \bigcup_{d\in\mathbb{G}} X_A(d), \bigcup_{d\in\mathbb{G}} X_{Start}(d)\right)$$

where for a given complete *derivation $d \in \mathbb{G}$, $X_S(d)$ is the set of* selection constraints *with respect to L observed in d, $X_L(d)$ is the set of* licensing constraints *with respect to L observed in d, $X_A(d)$ is the set of* anti-licensing constraints *with respect to L observed in d, and $X_{Start}(d)$ is the set of* start constraints *with respect to L observed in d. This procedure is denoted $\mathbb{P}$.* [3] □

Next I will prove several properties that a collection of constraints must satisfy. Note that Proposition 2 and Proposition 3 are related to the structure of the lexicon.

Proposition 1
If there exists syntactic features $(x, y) \in C_A$ with respect to a lexicon L, then the syntactic feature y must be a licensee.

Proof: First note that in complete derivations, *all non-leading*[4] *chains of all derivation terms must eventually be resolved; since* external merge *only relates two different derivation terms, only* internal merge *can resolve non-leading chains (via IM_1 and IM_2). Next, consider that by definition, anti-licensing constraints arise from the application of the shortest move constraint within a* complete *derivation. The shortest move constraint is a relation that holds between the leading chain (corresponding to x) of a derivation term, and a non-leading chain (corresponding to y) from the same derivation term. Since the non-leading chain may only be resolved via* internal merge *and the leading-chain has the* licensor *feature, hence the non-leading chain must have a* licensee *feature (i.e. y must be a* licensee*).* □

[3] Note that since $\mathbb{P}$ only depends on $\mathbb{G}(L)$ and not L directly, the procedure $\mathbb{P}$ may be viewed as mapping *classes of lexicons* to collections of constraints.
[4] The term *leading* here is used to indicate the first element of the ordered list of chains that a derivation term is composed of.

Proposition 2

Given a collection of constraints $(C_S, C_L, C_A, C_{Start})$, *the sets* C_S, C_L, C_A *and* C_{Start} *are pairwise disjoint.*

Proof: First I prove that the set C_{Start} *is disjoint with each of the sets* C_S, C_L *and* C_A. *Consider that* C_{Start} *is a set of syntactic features, whereas* C_S, C_L *and* C_A *are sets of pairs of syntactic features (i.e. (*selector, selectee*) or (*licensor, licensee*)). Hence, the elements of* C_{Start} *are of a different* type *than the elements of the sets* C_S, C_L *and* C_A. *Therefore, the set* C_{Start} *is disjoint with each of the sets* C_S, C_L *and* C_A.

Next I prove via contradiction that $C_L \cap C_A = \emptyset$. *Suppose that* $C_L \cap C_A \neq \emptyset$. *Then there exist syntactic features* x *and* y *such that* $(x, y) \in C_L \cap C_A$. *Since* $(x, y) \in C_L$, x *and* y *must have the same underlying symbol. (e.g. if* $(x, y) = (+f, -f) \in C_L$ *then* x *and* y *both have the same underlying symbol* f.*) However,* $(x, y) \in C_A$ *implies that* x *and* y cannot *have the same underlying symbol without violating the shortest move constraint.* [5] *This is a condradiction; hence, there is no* $(x, y) \in C_L \cap C_A$ *and thus* $C_L \cap C_A = \emptyset$.

Next I prove via contradiction that $C_S \cap C_L = \emptyset$. *Suppose that* $C_S \cap C_L \neq \emptyset$. *Then there exists syntactic features* x *and* y *such that* $(x, y) \in C_S \cap C_L$. *Hower,* $(x, y) \in C_S$ *implies that* x *is a* selector *and* y *is a* selectee *while* $(x, y) \in C_L$ *implies that* x *is a* licensor *and* y *is a* licensee. *Since* selectors *cannot also be* licensors *and vice versa, and similarly* selectees *cannot also be* licensees *and vice versa, this is a contradiction; hence, there is no* $(x, y) \in C_S \cap C_L$ *and thus* $C_S \cap C_L = \emptyset$.

Finally, By Proposition 1, if $(x, y) \in C_A$ *then* y *is a* licensee *(it is clear that* x *is a* licensor*); hence, the same reasoning that demonstrated that* $C_S \cap C_L = \emptyset$ *also demonstrates that* $C_S \cap C_A = \emptyset$. *Thus, I have shown that the sets* C_S, C_L, C_A *and* C_{Start} *are pairwise disjoint.* □

Proposition 3

Given a collection of constraints $Y = (C_S, C_L, C_A, C_{Start})$, *for* $i \geq 2$, *if the syntactic feature* (x, i) *appears in some constraint in* Y, *then* $(x, i-1)$ *also appears in some constraint in* Y.

Proof: The syntactic features in a chain form a linear sequence such that the syntactic feature (x, i) *is not accessible until after* $(x, i-1)$ *is consumed, except in the case of the initial syntactic feature. Hence, if the syntactic feature* (x, i) *is observed in* Y *then* $(x, i-1)$ *must have also been observed in* Y. □

[5] Note that by Proposition 1, $(x, y) \in C_A$ implies that y is a licensee, so if x and y had the same underlying symbol then the shortest move constarint would be violated.

2.3 Constructing a Curated Lexicon from a Collection of Constraints

This section focuses on developing a procedure, $\mathbb{Q}$, that constructs a *curated* lexicon from a *collection of constraints.* (See Procedure 3) In developing this procedure, I will introduce a new structure: the *template* of a *curated* lexicon. (See Definition 15) I will also provide a procedure for computing the *template* of a lexicon from a *collection of constraints.* (See Procedure 2) I will then prove that for any *curated* lexicon L, $\mathbb{Q}(\mathbb{P}(L))\diamond L$. (See Proposition 7)

Definition 15 (Template Lexicon)
Given a lexicon L, if the underlying symbols of the syntactic features are erased so that they are not specified, then what results is the template *of L, denoted $\mathbb{T}(L)$. Intuitively, the information contained in the* template *of L consists of pairings of phonetic forms and sequences of syntactic feature types.* □

The following proposition illustrates how the *template* of a lexicon affects the set of *complete* derivations that lexicon may produce.

Proposition 4
Let L_1 and L_2 be curated *lexicons. If $\mathbb{T}(L_1) \neq \mathbb{T}(L_2)$ then $\mathbb{G}(L_1) \neq \mathbb{G}(L_2)$ and $\mathbb{P}(L_1) \neq \mathbb{P}(L_2)$.*

Proof: I will first prove that if L_1 and L_2 do not have the same set of phonetic forms then $\mathbb{G}(L_1) \neq \mathbb{G}(L_2)$. Let $Ph(L_1)$ and $Ph(L_2)$ be the sets of phonetic forms in L_1 and L_2. If $Ph(L_1) \neq Ph(L_2)$, then there exists some $p \in Ph(L_1)$ such that $p \notin Ph(L_2)$ (or vice-versa). Hence, the phonetic form p will not appear in any derivation generated by L_2. However, since L_1 is a curated *lexicon, every phonetic form in $Ph(L_1)$ has an associated lexical item in L_1 and that lexical item must participate in at least one complete derivation generated by L_1; hence p* must *appear in at least one* complete *derivation generated by L_1. Furthermore, since p will not appear in any derivations generated by L_2, constraints involving p will not appear in $\mathbb{P}(L_2)$. Thus if $Ph(L_1) \neq Ph(L_2)$ then $\mathbb{G}(L_1) \neq \mathbb{G}(L_2)$ and $\mathbb{P}(L_1) \neq \mathbb{P}(L_2)$.*

Next, I will prove that if $\mathbb{T}(L_1) \neq \mathbb{T}(L_2)$ and $Ph(L_1) = Ph(L_2)$ then $\mathbb{G}(L_1) \neq \mathbb{G}(L_2)$. If $\mathbb{T}(L_1) \neq \mathbb{T}(L_2)$ and $Ph(L_1) = Ph(L_2)$, then there exists some element $t_1 \in \mathbb{T}(L_1)$ such that its corresponding entry $t_2 \in \mathbb{T}(L_2)$ does not have the same form *as a lexical item: either l_1 and l_2 have a different numbers of syntactic features or the* category *of some feature in l_1 is not the same as the* category *of the corresponding feature in l_2. In both cases, l_1 will not be able to participate in a derivation in the same manner as l_2, either due to differences in (a) the total number of* selection licensing *operations that l_1 and l_2 can participate in or (b) differences in how corresponding features (that have same index) in l_1 and l_2 particpate in feature matching operations. (i.e. the third element of l_1 might be a* selector *feature while the third element of l_2 might be a* licensee *feature)*

Since L_1 is a curated *lexicon, l_1 must participate in at least one* complete *derivation generated by L_1 (call this derivation d). Then, although l_1 and l_2 share the same phonetic form, since l_2 cannot be substituted in place of l_1 in constructing d, hence $d \notin \mathbb{G}(L_2)$ and thus $\mathbb{G}(L_1) \neq \mathbb{G}(L_2)$. Furthermore, both cases (a) and (b)*

will result in differing sets of constraints in derivations in $\mathbb{G}(L_1)$ *and* $\mathbb{G}(L_2)$ *involving* l_1 *and* l_2 *respectively. Hence* $Y_1 \neq Y_2$. $\square$

Procedure 2

This procedure $\mathbb{T}$ *constructs a* template *for a* curated *lexicon from a given* collection of constraints $(C_S, C_L, C_A, C_{Start})$. *Since the* template *of a lexicon is void of all underlying symbols, constructing one requires (a) the set of phonetic forms* $p_i \in P$, *(b) the number of syntactic features each phonetic form is associated with and (c) assigning each syntactic feature to one of the following five mutually exclusive categories:* selectors, selectees, licensors, licensees *and the special symbol* C. *The computation of each of these requirements is described below:*

(a) The set of phonetic forms P *is:*

$$\left(\bigcup_{((p_i,x),(p_j,y)) \in C_S} \{p_i, p_j\} \right) \cup \left(\bigcup_{((p_i,x),(p_j,y)) \in C_L} \{p_i, p_j\} \right) \cup \{p | (p,x) \in C_{Start}\}$$

(b) For a given phonetic form $p^{'}$, *the number of syntactic features* $p^{'}$ *has is:*

$$maximum\,(N_{Selectors} \cup N_{Selectees} \cup N_{Licensors} \cup N_{Licensees} \cup N_{Start})$$

where

$$\begin{aligned} N_{Selectors} &= \{x | ((p^{'}, x), (p_j, y)) \in C_S\} \\ N_{Selectees} &= \{y | ((p_j, x), (p^{'}, y)) \in C_S\} \\ N_{Licensors} &= \{x | ((p^{'}, x), (p_j, y)) \in C_L\} \\ N_{Licensees} &= \{y | ((p_j, x), (p^{'}, y)) \in C_L\} \\ N_{Start} &= \{x | (p_i, x) \in C_{Start}, p_i = p^{'}\} \end{aligned}$$

(c) Given a syntactic feature (p_i, x), *exactly one of the following five conditions will hold, which in turn determines which of the five categories* (p_i, x) *belongs to.*

(a) If there exists a syntactic feature w *such that* $((p_i, x), w) \in C_S$ *then* (p_i, x) *is a* selector.

(b) If there exists a syntactic feature w *such that* $(w, (p_i, x)) \in C_S$ *then* (p_i, x) *is a* selectee.

(c) If there exists a syntactic feature w *such that* $((p_i, x), w) \in C_L$ *then* (p_i, x) *is a* licensor.

(d) If there exists a syntactic feature w *such that* $(w, (p_i, x)) \in C_L$ *then* (p_i, x) *is a* licensee.

(e) If $(p_i, x) \in C_{Start}$, *then* (p_i, x) *is the special start symbol.*

Note that this information can be entirely derived from the sets C_S, C_L *and* C_{Start}; *for the purposes of computing the* template *of a lexicon from a collection of constraints, there is no need for the* anti-licensing *constraints.* $\square$

Proposition 5

Given a curated *lexicon* L, *let* $Y = \mathbb{P}(L) = (C_S, C_L, C_A, C_{Start})$ *be the* collection of constraints *computed*

from L and let T designate the template *computed from Y via Procedure 2. Then $T = \mathbb{T}(L)$.*

Proof: In order to prove that $T = \mathbb{T}(L)$, I will show that they have (a) the same set of phonetic forms, (b) the same number of syntactic features for corresponding phonetic forms and (c) the same assignment of syntactic features to feature categories (e.g. selector, selectee, licensor, *etc).*

(a) I will prove that the set of phonetic forms in T, P_T is the same as the set of phonetic forms in $\mathbb{T}(L)$, P_L.

Given a phonetic form $p_i \in P_L$, since L is a curated *lexicon, the lexical item l_i that corresponds to p_i participates in at least one derivation $d \in \mathbb{G}(L)$. Therefore, l_i participates in at least one* merge *operation involving either* selection *or* licensing*; thus, $\mathbb{P}(L)$ will capture l_i (and thus p_i) in at least one constraint in either C_S or C_L. By Procedure 2, since p_i is involved in either C_S or C_L, $p_i \in P_T$. Next, consider that if a phonetic form p_i is not a member of P_L then there is no corresponding lexical item l_i that participates in derivations produced by L. Therefore, $\mathbb{P}(L)$ will not capture p_i in either C_S or C_L. Hence, if $p_i \notin P_L$ then $p_i \notin P_T$. Thus, since $p_i \in P_L$ if and only if $p_i \in P_T$, therefore $P_L = P_T$.*

(b) Since $P_L = P_T$, both T and $\mathbb{T}(L)$ have a unique lexical item for each phonetic form $p_i \in P_L$. Let $l_i \in \mathbb{T}(L)$ and $l_i' \in T$ be the unique lexical items corresponding to p_i. I will prove that the number of syntactic features in l_i is the same as the number of syntactic features in l_i'. The number of syntactic features for l_i is equal to the maximum index of any syntactic feature belonging to l_i. Let (p_i, x) be the syntactic feature belonging to l_i such that its index is maximal (i.e. l_i has a total of x syntactic features). Since L is a curated *lexicon, l_i participates in at least one* complete *derivation produced by L; furthermore, every syntactic feature of every lexical item in a* complete *derivation is consumed. Therefore (p_i, x) will be captured by either C_S, C_L or C_{Start}. By Procedure 2, the number of syntactic features belonging to l_i' is:*

$$maximum\,(N_{Selectors} \cup N_{Selectees} \cup N_{Licensors} \cup N_{Licensees} \cup N_{Start})$$

and thus l_i and l_i' have the same number of syntactic features.

(c) Thus far, for every syntactic feature (p_i, x) that is a member of $\mathbb{T}(L)$ is also a member of T. I will next prove that (p_i, x) is the same type of feature (e.g. selector, selectee, licensor, *etc) in both $\mathbb{T}(L)$ and T. Without loss of generality, suppose that (p_i, x) is a* selector *with respect to $\mathbb{T}(L)$. Then (p_i, x) will appear as a* selector *in a constraint in C_S; Procedure 2 will therefore determine that (p_i, x) is a* selector *with respect to T. By a similar argument it can be shown that this also holds for the other categories of* selectee, licensor, licensee *and* start *constraints.*

This proves that $T = \mathbb{T}(L)$. □

Procedure 3

This procedure, denoted $\mathbb{Q}$, describes how to construct a curated *lexicon, L, from a collection of constraints $Y = (C_S, C_L, C_A, C_{Start})$, in two steps. First, the template of L, $\mathbb{T}(L)$, is computed using Procedure 2. Second, since $\mathbb{T}(L)$ is void of any assignment of a "symbol" to a syntactic feature, these assignments* [6] *are*

[6] Note that only *selectors*, *selectees*, *licensors* and *licensees* need to have a symbol assigned to them.

made as follows: two syntactic features x and y in L have the same underlying symbol if and only if either $(x, y) \in C_S$ or $(x, y) \in C_L$. Then $L = \mathbb{Q}(Y)$. □

Proposition 6

Given two curated *lexicons L_1 and L_2, let $Y_1 = \mathbb{P}(L_1)$ and $Y_2 = \mathbb{P}(L_2)$. Then $Y_1 = Y_2$ if and only if $L_1 \diamond L_2$.*

Proof: First we prove that if $L_1 \diamond L_2$ then $Y_1 = Y_2$. Suppose $L_1 \diamond L_2$, then $\mathbb{G}(L_1) = \mathbb{G}(L_2)$. Since Y_1 is computed from $\mathbb{G}(L_1)$, this implies that Y_1 may be computed from $\mathbb{G}(L_2)$ as well. Thus both Y_1 and Y_2 may be computed from $\mathbb{G}(L_2)$. Since the procedure $\mathbb{P}$ is deterministic, this means that $Y_1 = Y_2$.

Next we prove that if L_1 is not in the same class of lexicons *as L_2, then $Y_1 \neq Y_2$. By Proposition 4, if $\mathbb{T}(L_1) \neq \mathbb{T}(L_2)$ then $\mathbb{P}(L_1) \neq \mathbb{P}(L_2)$ and thus $Y_1 \neq Y_2$. If $\mathbb{T}(L_1) = \mathbb{T}(L_2)$ then the differences between $\mathbb{G}(L_1)$ and $\mathbb{G}(L_2)$ must arise from the differences in assignment of underlying symbols to the syntactic features in L_1 and L_2, and the effect that has on which* selection, licensing *and* anti-licensing *operations are observed in the generation of* complete *derivations.*

Since $\mathbb{G}(L_1) \neq \mathbb{G}(L_2)$, there is some derivation $d \in \mathbb{G}(L_1)$ such that $d \notin \mathbb{G}(L_2)$ (or vice versa; we may proceed without a loss of generality). Let us designate the selection, licensing *and* anti-licensing *constraints required for L_1 to generate d as C_S', C_L' and C_A' respectively. Then $d \notin \mathbb{G}(L_2)$ because the assignment of underlying symbols to L_2 does not permit the constraints in C_S', C_L' or C_A'. Hence the constraints in C_S', C_L' or C_A' will appear in Y_1 but not in Y_2. Therfore, $Y_1 \neq Y_2$.* □

Proposition 7

Given a curated *lexicon L, let $L' = \mathbb{P}(\mathbb{Q}(L))$. Then $L \diamond L'$.*

Proof: Let $Y = (C_S, C_L, C_A, C_{Start}) = \mathbb{P}(L)$ and let $Y' = (C_S', C_L', C_A', C_{Start}') = \mathbb{P}(L')$. I will prove that $Y = Y'$ and thus by Proposition 6, $L \diamond L'$.

Since $\mathbb{T}(L) = \mathbb{T}(L')$ (see Proposition 5), differences between $\mathbb{G}(L)$ and $\mathbb{G}(L')$ must arise from differences in the underlying symbols assigned to the syntactic features in L and L'. The scheme for the assignment, provided in Section **??**, *has the invariant that the syntactic features x' and y' in L' have the same underlying symbol if and only if either $(x', y') \in C_S$ or $(x', y') \in C_L$.*

First, we prove that $C_S = C_S'$. If two syntactic features x and y in L participated in a selection *operation in some derivation $d \in \mathbb{G}(L)$, then $(x, y) \in C_S$ and the corresponding syntactic features x' and y' will have the same underlying symbols, allowing x' and y' to participate in* selection *operations that appear in $\mathbb{G}(L')$, and thus $C_S \subseteq C_S'$. Next, suppose there is a* selection *constraint $(x', y') \in C_S'$. Then the syntactic features x' and y' must have the same underlying symbols in L' and thus $(x', y') \in C_S$. Hence, $C_S = C_S'$.*

Next, we prove that $C_L = C_L'$. If two syntactic features x and y in L participated in a licensing *operation in some derivation $d \in \mathbb{G}(L)$, then $(x, y) \in C_L$ and the corresponding syntactic features x' and y' will have the same underlying symbols, allowing x' and y' to participate in* licensing *operations that appear in $\mathbb{G}(L')$, and thus $C_L \subseteq C_L'$. Next, suppose there is a* licensing *constraint $(x', y') \in C_L'$. Then the syntactic features x' and y' must have the same underlying symbols in L' and thus $(x', y') \in C_L$. Hence, $C_L = C_L'$.*

Finally, we prove that $C_A = C_A^{'}$. *If some derivation in* $\mathbb{G}(L)$ *required that a syntactic feature* x not license *a syntactic feature* y *(so that the shortest move constraint is not violated) then* $(x, y) \in C_A$ *and thus* $(x, y) \notin C_L$. *Hence,* x *and* y *cannot have the same underlying symbols in* $L^{'}$. *Thus, any derivations in* $\mathbb{G}(L^{'})$ *in which the shortest move constraint requires that it not be possible for* x *to* license y *will not be obstructed. Since derivations are constructed by ensuring that the desired set of* selection *and* licensing *constraints are satisfied, and that no* anti-licensing *constraints obstruct the construction, hence every derivation in* $\mathbb{G}(L) \subseteq \mathbb{G}(L^{'})$ *and thus* $C_A \subseteq C_A^{'}$.

To show that $C_A^{'} \subseteq C_A$ *(via contradiction), suppose there exists some* anti-licensing *constraint* $(x^{'}, y^{'}) \in C_A^{'}$ *such that* $(x^{'}, y^{'}) \notin C_A$. *Then the syntactic features* $x^{'}$ *and* $y^{'}$ *in* $L^{'}$ *cannot have the same underlying symbol, and thus by Section* **??**, $(x^{'}, y^{'}) \notin C_L$. *In order to enforce that* $x^{'}$ *does not* license $y^{'}$ *in any derivation in* $\mathbb{G}(L)$, *we must have* $(x^{'}, y^{'}) \in C_A$. *This is a contradiction, and thus there does not exist any* anti-licensing *constraint* $(x^{'}, y^{'}) \in C_A^{'}$ *such that* $(x^{'}, y^{'}) \notin C_A$. *Hence* $C_A^{'} \subseteq C_A$ *and therefore* $C_A = C_A^{'}$.

Since $C_S = C_S^{'}$, $C_L = C_L^{'}$ *and* $C_A = C_A^{'}$, *and since* $C_{Start} = C_{Start}^{'}$ *(because* $\mathbb{T}(L) = \mathbb{T}(L^{'})$*),* $Y = Y^{'}$ *and therefore* $L \diamond L^{'}$. $\square$

2.4 Correspondences between Classes of Lexicons and Collections of Constraints

In the previous section, Proposition 6 and Proposition 7 established a correspondence between classes of *curated* lexicons and *collections of constraints.* This section further explores this correspondence and the relation between the procedures $\mathbb{P}$ and $\mathbb{Q}$. (See Proposition 8 and Proposition 9)

Proposition 8
Given two collections of constraints, Y_1 *and* Y_2, *let* $L_1^{'} = \mathbb{Q}(Y_1)$ *and* $L_2^{'} = \mathbb{Q}(Y_2)$. *Then* $Y_1 = Y_2$ *if and only if* $L_1^{'} \diamond L_2^{'}$.

Proof: For every collection of constraint Y, *there exists some* curated *lexicon* L *such that* $Y = \mathbb{P}(L)$. *Suppose that* $Y_1 = \mathbb{P}(L_1)$ *and* $Y_2 = \mathbb{P}(L_2)$ *for* curated *lexicons* L_1 *and* L_2. *By Proposition 7,* $L_1 \diamond L_1^{'}$ *and* $L_2 \diamond L_2^{'}$. *Then, by Proposition 6,* $L_1 \diamond L_2$ *if and only if* $Y_1 = Y_2$. *Therefore,* $L_1^{'} \diamond L_2^{'}$ *if and only if* $Y_1 = Y_2$. □

Proposition 9
Let L^* *be the set of all* curated *lexicons,* K *be the set of* classes of lexicons *and* Y^* *be the set of all* collections of constraints. *Then both* $\mathbb{P}$ *and* $\mathbb{Q}$ *are both bijections between* K *and* Y^*.

Proof: First note that for every $Y \in Y^*$ *there exists an* $L \in L^*$ *such that* $Y = \mathbb{P}(L)$. *Thus,* Y^* *is the range of* $\mathbb{P}$; *hence,* $\mathbb{P}$ *is surjective.*

To prove that $\mathbb{P}$ *is a bijection between* K *and* Y^*, *note that by Proposition 6,* $\mathbb{P}$ *provides a one to one correspondence between* K *and* Y^* *and thus* $\mathbb{P}$ *is injective. Therefore,* $\mathbb{P}$ *is a bijection between* K *and* Y^*.

Next, by Proposition 8, $\mathbb{Q}$ *provides a one to one correspondence between* Y^* *and* K. *Hence,* $\mathbb{Q}$ *is an injective map. To show that* $\mathbb{Q}$ *is surjective, consider that for any* class of lexicons $k \in K$ *and any lexicon* $L \in k$, $\mathbb{P}(L) \in Y^*$ *and* $\mathbb{Q}(\mathbb{P}(L)) \diamond L$. *Hence the* class of lexicons k *is in the range of* $\mathbb{Q}$. *Thus,* K *is the range of* $\mathbb{Q}$. *Therefore,* $\mathbb{Q}$ *is a bijection between* K *and* Y^*. □

2.5 Combining Collections of Constraints

This section focuses on developing a procedure $\mathbb{J}$ (see Procedure 4) that takes two curated lexicons, L_1 and L_2, and assembles them into a single curated lexicon, $\mathbb{J}(L_1, L_2)$, such that $\mathbb{G}(L_1) \cup \mathbb{G}(L_2) \subseteq \mathbb{G}(\mathbb{J}(L_1, L_2)$. (See Proposition 10)

Definition 16

Let L_1 and L_2 be curated *lexicons and let P_1 and P_2 be their set of phonetic forms. For any phonetic form $p_i \in P_1$, let T_1^i designate the unique entry in $\mathbb{T}(L_1)$ whose phonetic form is p_i. Likewise, for any phonetic form $p_i \in P_2$, let T_2^i designate the unique entry in $\mathbb{T}(L_2)$ whose phonetic form is p_i. Then L_1 and L_2 have* compatible templates *if and only if for each phonetic form $p_i \in P_1 \cap P_2$, $T_1^i = T_2^i$.* □

Procedure 4

Let L_1 and L_2 be curated *lexicons with* compatible templates, *and let Y_1 and Y_2 be their associated* collections of constraints*:*

$$Y_1 = \mathbb{P}(L_1) = (C_S^1, C_L^1, C_A^1, C_{Start}^1)$$
$$Y_2 = \mathbb{P}(L_2) = (C_S^2, C_L^2, C_A^2, C_{Start}^2)$$

and let:

$$Z_{Selector} = \{x | (x, y) \in C_S^1 \cup C_S^2\}$$
$$Z_{Selectee} = \{y | (x, y) \in C_S^1 \cup C_S^2\}$$
$$Z_{Licensor} = \{x | (x, y) \in C_L^1 \cup C_L^2\}$$
$$Z_{Licensee} = \{y | (x, y) \in C_L^1 \cup C_L^2\}$$

Next, suppose it is possible to define the following sets:

$$D_S^{'} \subseteq \{(x, y) | x \in Z_{Selector}, y \in Z_{Selectee}\}$$
$$D_L^{'} \subseteq \{(x, y) | x \in Z_{Licensor}, y \in Z_{Licensee}\}$$
$$D_A^{'} \subseteq \{(x, y) | x \in Z_{Licensor}, y \in Z_{Licensee}\}$$

such that the following conditions are satisfied:

1. $C_S^1 \cup C_S^2 \subseteq D_S^*$.
2. $C_L^1 \cup C_L^2 \subseteq D_L^*$.
3. $C_A^1 \cup C_A^2 \subseteq D_A^*$.
4. $D_L^* \cap D_A^* = \emptyset$.
5. *If* $(x_1, y_2), (x_1, y_1), (x_2, y_1) \in D_S^*$ *then* $(x_2, y_2) \in D_S^*$.
6. *If* $(x_1, y_2), (x_1, y_1), (x_2, y_1) \in D_L^*$ *then* $(x_2, y_2) \in D_L^*$.

7. *If* $(x_1, y_1), (x_1, y_2) \in D_L^*$ *and* $(x_2, y_1) \in D_A^*$ *then* $(x_2, y_2) \in D_A^*$.

8. *If* $(x_1, y_1), (x_2, y_1) \in D_L^*$ *and* $(x_1 y_2) \in D_A^*$ *then* $(x_2, y_2) \in D_A^*$.

9. *If* $(x_1, y_1), (x_2, y_2) \in D_L^*$ *and* $(x_1, y_2) \in D_A^*$ *then* $(x_2, y_1) \in D_A^*$.

10. *It is not possible to remove any elements from the sets* D_S^*, D_L^* *and* D_A^* *without the previous conditions.*

If D_S^* *and* D_L^* *cannot be defined so as to fulfill these conditions, then the procedure terminates with a failure state. Otherwise, proceed to (a) construct a* template *lexicon* $T' = \mathbb{T}(L_1) \cup \mathbb{T}(L_2)$ *and then (b) assign underlying symbols to* T' *such that two syntactic features* x *and* y *have the same symbol if and only if* $(x, y) \in D_S^* \cup D_L^*$. *The resulting lexicon is the output of the procedure and is designated* $\mathbb{J}(L_1, L_2)$. [7] $\square$

Proposition 10

Let L_1 *and* L_2 *be* curated *lexicons with* compatible templates *and suppose the lexicon* $L' = J(L_1, L_2)$ *may be successfully computed. Then* $\mathbb{G}(L_1) \cup \mathbb{G}(L_2) \subseteq \mathbb{G}(L')$.

Proof: I will prove that $\mathbb{G}(L_1) \subseteq \mathbb{G}(L')$. *First note that* $\mathbb{T}(L') = \mathbb{T}(L_1) \cup \mathbb{T}(L_2)$ *is* compatible *with* L_1. *Hence, any reason that a derivation* $d \in \mathbb{G}(L_1)$ *might not be generated by* L' *is due to the differences in assignment of symbols to the syntactic features in* L_1 *and* L'.

Next, consider that the symbol assignment scheme of $\mathbb{J}$ *preserves the* selection *and* licensing *constraints since* $C_S^1 \subseteq D_S^*$ *and* $C_L^1 \subseteq D_L^*$. *Futhermore, the introduction of other constraints from* L_2 *will not cause any interference with the constraints from* L_1 *since* $D_L^* \cap D_A^* = \emptyset$. *Hence,* $\mathbb{G}(L_1) \subseteq \mathbb{G}(L')$. *The same reasoning shows that* $\mathbb{G}(L_2) \subseteq \mathbb{G}(L')$; *thus,* $\mathbb{G}(L_1) \cup \mathbb{G}(L_2) \subseteq \mathbb{G}(L')$. $\square$

Proposition **??** constitutes the main result of this chapter: a method for assembling together lexicons while preserving the derivations they produce. Specifically, given two *curated* lexicons L_1 and L_2 with *compatible templates*, let Y_1 and Y_2 be their respective *collections of constraints*; if we can construct Y^*, the *combination* of Y_1 and Y_2, then we can construct a *curated* lexicon $L^* = \mathbb{Q}(Y^*)$ via Procedure 10 such that $\mathbb{G}(L_1) \cup \mathbb{G}(L_2) \subseteq \mathbb{G}(\mathbb{Q}(Y^*)$. The next chapter of this thesis will focus on examining the set of *complete* derivations that may be generated by L^* but not by either L_1 or L_2, (i.e. the set of derivations $\mathbb{G}(L^*) - (\mathbb{G}(L_1) \cup \mathbb{G}(L_2))$) and explore the interpretation of this set of derivations as being "learned" or "generalized" from L_1 and L_2.

[7] Note that the procedure $\mathbb{J}$ is symmetric in its arguments.

Chapter 3

Acquisition of an MG Lexicon

In this chapter I present a running example that demonstrates how the mathematical framework for combining *curated* lexicons developed in the previous chapter may be used to simulate the process of a *learner acquiring* the lexicon of a language. Specifically, the learner is presented with a sequence of sentences in the form of MG *derivations* $d_1, \ldots, d_n$ and has the task of producing (if at all possible) a *curated* lexicon L_D such that $d_i \in \mathbb{G}(L_D)$ for $1 \leq i \leq n$. The learner is then said to have *acquired* the lexicon L_D from the derivations $d_1, \ldots, d_n$.

The state of the learner (i.e. a representation of his knowledge up to that point) is encoded as a *collection of constraints* (from which may be computed a *curated* lexicon, representing the set of sentences he is able to produce, via $\mathbb{Q}$). Let Y_i be the *collection of constraints* reflecting the learner's knowledge after exposure to d_i for $1 \leq i \leq n$ and let $Y_0 = (\emptyset, \emptyset, \emptyset, \emptyset)$ (i.e. the learner begins with no knowledge of the language). When the learner encounters a *derivation* d_i, the learner proceeds to:

1. Construct the *curated* lexicon L_i such that $d_i \in \mathbb{G}(L_i)$.
2. Compute the *collection of constraints* $Y_i = \mathbb{J}(\mathbb{Q}(Y_{i-1}), L_i)$.

If these two steps are successful then the learner is said to have acquired the *curated* lexicon $\mathbb{Q}(Y_i)$ from the derivations $d_1, \ldots, d_i$. In this manner, the learner may acquire the lexicon $L_D = \mathbb{Q}(Y_n)$.

I will present a sequence of derivations that have associated derivation trees that are *X-bar* phrase markers for the following sentences:

- "The cat will eat a mouse."
- "Will the cat eat a mouse?"
- "Which mouse will the cat eat?"
- "Which cat will eat the mouse?"

- "The man will think that the cat will eat the mouse."
- "Will the man think that the cat will eat the mouse?"
- "Which mouse will the man think that the cat will eat?"

and point out how the acquisition process outlined above allows the learner to capture generalizations from the examples (i.e. $\{d_1, \ldots, d_7\} \subset \mathbb{G}(L_D)$).

3.1 The cat will eat a mouse.

The first derivation, d_1, that the learner is exposed to (see Page 36) produces a *derived tree* that corresponds to the phrase structure shown in Figure 3.1 for the sentence "the cat will eat a mouse". This derivation is simple in that it does not involve any movement (i.e. *internal merge*).

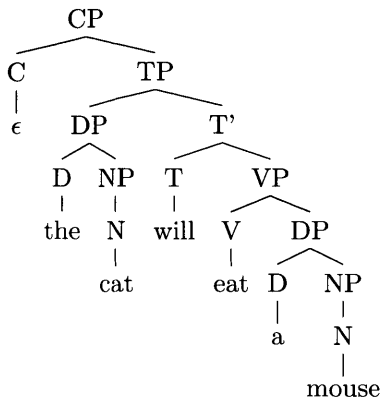

Figure 3.1: Phrase Structure for the sentence "the cat will eat a mouse."

From d_1 the learner computes the lexicon L_1 (See Figure 3.2) such that $d_1 \in \mathbb{G}(L_1)$ (in fact $\{d_1\} = \mathbb{G}(L_1)$). The learner then computes their next state of knowledge: $Y_1 = \mathbb{P}(\mathbb{J}(\mathbb{Q}(Y_0), L_1))$. Since $Y_0 = (\emptyset, \emptyset, \emptyset, \emptyset)$, the learner's next state of knowledge is $Y_1 = \mathbb{P}(L_1)$. The *selection constraints* in Y_1 are presented in Table 3.1 and the *(anti)-licensing constraints* are presented in Table 3.2.

	$a :: = x1, \sim x2$
	$cat :: \sim x6$
$\epsilon_C :: = x8, C$	$eat :: = x2, \sim x4$
$\epsilon_T :: = x3, = x4, +x5, = x7, \sim x8$	$mouse :: \sim x1$
	$the :: = x6, \sim x7$
	$will :: \sim x3, -x5$

Figure 3.2: The *curated* lexicon L_1 from which the derivation d_1 (which produces the sentence "the cat will eat a mouse.") may be derived. Note that $\{d_1\} = \mathbb{G}(L_1)$.

$$\frac{[\text{the} :: = x6, \sim x7] \qquad [\text{cat} :: \sim x6]}{[\text{the cat} : \sim x7]}\ \text{EM1}$$

$$\frac{[\text{a} :: = x1, \sim x2] \qquad [\text{mouse} :: \sim x1]}{[\text{a mouse} : \sim x2]}\ \text{EM1}$$

$$\frac{[\text{eat} :: = x2, \sim x4] \qquad [\text{a mouse} : \sim x2]}{[\text{eat a mouse} : \sim x4]}\ \text{EM1}$$

$$\frac{[\epsilon_T :: = x3, = x4, +x5, = x7, \sim x8] \qquad [\text{will} :: \sim x3, -x5]}{[\epsilon_T : = x4, +x5, = x7, \sim x8][\text{will} : -x5]}\ \text{EM3}$$

$$\frac{[\epsilon_T : = x4, +x5, = x7, \sim x8][\text{will} : -x5] \qquad [\text{eat a mouse} : \sim x4]}{[\text{eat a mouse } \epsilon_T : +x5, = x7, \sim x8][\text{will} : -x5]}\ \text{EM2}$$

$$\frac{[\text{eat a mouse } \epsilon_T : +x5, = x7, \sim x8][\text{will} : -x5]}{[\text{will eat a mouse } \epsilon_T : = x7, \sim x8]}\ \text{IM1}$$

$$\frac{[\text{will eat a mouse } \epsilon_T : = x7, \sim x8] \qquad [\text{the cat} : \sim x7]}{[\text{the cat will eat a mouse } \epsilon_T : \sim x8]}\ \text{EM2}$$

$$\frac{[\epsilon_C :: = x8, \text{C}] \qquad [\text{the cat will eat a mouse } \epsilon_T : \sim x8]}{[\epsilon_C \text{ the cat will eat a mouse } \epsilon_T : \text{C}]}\ \text{EM1}$$

		Selectees						
		(a, 1)	(cat, 0)	(eat, 1)	(mouse, 0)	(the, 1)	(will, 0)	(ϵ_T, 4)
Selectors	(a, 0)	-	-	-	✓	-	-	-
	(eat, 0)	✓	-	-	-	-	-	-
	(the, 0)	-	✓	-	-	-	-	-
	(ϵ_C, 0)	-	-	-	-	-	-	✓
	(ϵ_T, 0)	-	-	-	-	-	✓	-
	(ϵ_T, 1)	-	-	✓	-	-	-	-
	(ϵ_T, 3)	-	-	-	-	✓	-	-

Table 3.1: The Selection Constraints of Y_1. Notice that since no two selection constraints (indicated by a check mark) are in the same row (thus sharing a common selector) or the same column (thus sharing a common selectee) no further constraints may be derived from those immediately produced by the derivation sequence.

		Licensees
		(will, 1)
Licensors	(ϵ_T, 2)	✓

Table 3.2: The (Anti-)Licensing Constraints of Y_1: Since ϵ_T merges with "will" before "eat a mouse", "will" must then be moved in order to place it to the left of "eat a mouse." The reader should note that since the first and second external merge operations have fixed linearization schemes, at times internal merge is necessary to correct for this.

3.2 Will a cat eat the mouse?

The next derivation encountered by the learner, d_2 (see Page 39), produces a derived tree that corresponds to the phrase structure shown in Figure 3.3 for the interrogative sentence "will a cat eat the mouse?" This sentence is a variation on the first sentence: the order of the subject and auxiliary verb are inverted via head raising in order to form a simple question. Note that the determiners have been switched in d_2 ("a cat" and "the mouse") in comparison to d_1 ("the cat" and "a mouse").

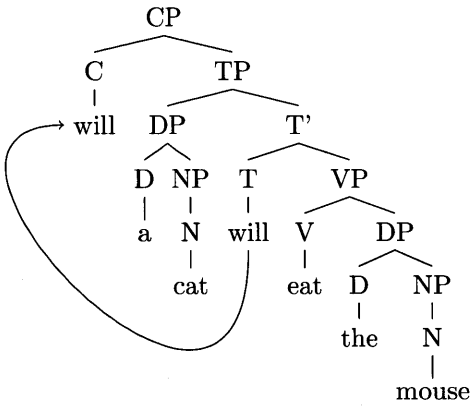

Figure 3.3: Phrase Structure for the sentence "will a cat eat the mouse?"

The learner computes the *curated* lexicon L_2 (see Figure 3.4) which uniquely generates d_2. The learner then comptutes his next state of knowledge: $Y_2 = \mathbb{P}(\mathbb{J}(\mathbb{Q}(Y_1), L_2))$. The *selection constraints* and *(anti-)licensing constraints* for $\mathbb{P}(L_2)$ are presented in Tables 3.3 and 3.4 respectively. The *selection constraints* and *(anti-)licensing constraints* for Y_2 are presented in Tables 3.5 and 3.6 respectively. Finally, the lexicon representation of the learner's new state of knowledge (i.e. $\mathbb{Q}(Y_2)$) is presented in Figure 3.5.

Note that the learner may be said to have captured the following generalizations: first, that the nouns "mouse" and "cat" are the same part of speech and second that the determiners "the" and "a" are also the the same part of speech. Furthermore, the overlap in usage between the two variants of "will" (e.g. [$will_1$ and $will_2$) is captured by their first and second syntactic features each sharing the same symbols (z_4 and z_6 respectively).

Furthermore, note that each constraint in Tables 3.5 and 3.6 may be found in either Y_1 (see Tables 3.1 and 3.2) or $\mathbb{P}(L_2)$ (see Tables 3.3 and 3.4). Thus the learner has not learned any new constraints in combining L_1 and L_2. The next section will provide an example in which a new constraint is acquired.

	$a :: = x7, \sim x8$
	$cat :: \sim x7$
$\epsilon_{C_2} :: = x9, +x10, C$	$eat :: = x2, \sim x5$
$\epsilon_T :: = x3, = x5, +x6, = x8, \sim x9$	$mouse :: \sim x1$
	$the :: = x1, \sim x2$
	$will2 :: \sim x3, -x6, -x10$

Figure 3.4: The *curated* lexicon L_2 from which the derivation d_2 (which produces the interrogative sentence "will a cat eat the mouse?") may be derived. Note that $\{d_2\} = \mathbb{G}(L_2)$.

$$\frac{[\text{a} :: = x7, \sim x8] \qquad [\text{cat} :: \sim x7]}{[\text{a cat} : \sim x8]}\ \text{EM1}$$

$$\frac{[\text{the} :: = x1, \sim x2] \qquad [\text{mouse} :: \sim x1]}{[\text{the mouse} : \sim x2]}\ \text{EM1}$$

$$\frac{[\text{eat} :: = x2, \sim x5] \qquad [\text{the mouse} : \sim x2]}{[\text{eat the mouse} : \sim x5]}\ \text{EM1}$$

$$\frac{[\epsilon_T :: = x3, = x5, +x6, = x8, \sim x9] \qquad [\text{will2} :: \sim x3, -x6, -x10]}{[\epsilon_T : = x5, +x6, = x8, \sim x9][\text{will2} : -x6, -x10]}\ \text{EM3}$$

$$\frac{[\epsilon_T : = x5, +x6, = x8, \sim x9][\text{will2} : -x6, -x10] \qquad [\text{eat the mouse} : \sim x5]}{[\text{eat the mouse } \epsilon_T : +x6, = x8, \sim x9][\text{will2} : -x6, -x10]}\ \text{EM2}$$

$$\frac{[\text{eat the mouse } \epsilon_T : +x6, = x8, \sim x9][\text{will2} : -x6, -x10]}{[\text{eat the mouse } \epsilon_T : = x8, \sim x9][\text{will2} : -x10]}\ \text{IM2}$$

$$\frac{[\text{eat the mouse } \epsilon_T : = x8, \sim x9][\text{will2} : -x10] \qquad [\text{a cat} : \sim x8]}{[\text{a cat eat the mouse } \epsilon_T : \sim x9][\text{will2} : -x10]}\ \text{EM2}$$

$$\frac{[\epsilon_{C_2} :: = x9, +x10, \text{C}] \qquad [\text{a cat eat the mouse } \epsilon_T : \sim x9][\text{will2} : -x10]}{[\epsilon_{C_2} \text{ a cat eat the mouse } \epsilon_T : +x10, \text{C}][\text{will2} : -x10]}\ \text{EM1}$$

$$\frac{[\epsilon_{C_2} \text{ a cat eat the mouse } \epsilon_T : +x10, \text{C}][\text{will2} : -x10]}{[\text{will2 } \epsilon_{C_2} \text{ a cat eat the mouse } \epsilon_T : \text{C}]}\ \text{IM1}$$

		Selectees						
		(a, 1)	(cat, 0)	(eat, 1)	(mouse, 0)	(the, 1)	(will2, 0)	(ϵ_T, 4)
Selectors	(a, 0)	-	✓	-	-	-	-	-
	(eat, 0)	-	-	-	-	✓	-	-
	(the, 0)	-	-	-	✓	-	-	-
	(ϵ_{C_2}, 0)	-	-	-	-	-	-	✓
	(ϵ_T, 0)	-	-	-	-	-	✓	-
	(ϵ_T, 1)	-	-	✓	-	-	-	-
	(ϵ_T, 3)	✓	-	-	-	-	-	-

Table 3.3: The Selection Constraints of $\mathbb{P}(L_2)$: As in Table 3.1, no further constraints are derived from those immediately produced by the derivation sequence.

		Licensees	
		(will2, 1)	(will2, 2)
Licensors	(ϵ_{C_2}, 1)	-	✓
	(ϵ_T, 2)	✓	-

Table 3.4: The (Anti-)Licensing Constraints of $\mathbb{P}(L_2)$: The licensing constraint between the third syntactic feature of ϵ_T and the second syntactic feature of *will* reflects an *internal merge* operation that is responsible for the head-raising process underlying subject-auxiliary verb inversion.

Selectors \ Selectees	(a, 1)	(cat, 0)	(eat, 1)	(mouse, 0)	(the, 1)	(will, 0)	(will2, 0)	(ϵ_T, 4)
(a, 0)	-	✓	-	✓	-	-	-	-
(eat, 0)	✓	-	-	-	✓	-	-	-
(the, 0)	-	✓	-	✓	-	-	-	-
(ϵ_C, 0)	-	-	-	-	-	-	-	✓
(ϵ_{C_2}, 0)	-	-	-	-	-	-	-	✓
(ϵ_T, 0)	-	-	-	-	-	✓	✓	-
(ϵ_T, 1)	-	-	✓	-	-	-	-	-
(ϵ_T, 3)	✓	-	-	-	✓	-	-	-

Table 3.5: The Selection Constraints of Y_2: Although there are no logical inconsistencies that arise in combining the selection constraints from Tables 3.1 and 3.3, no new constraints may be derived from the unified constraints. Nevertheless the learner is able to make certain generalizations that become clear in the lexicon derived from the unified selection and licensing constraints.

Licensors \ Licensees	(will, 1)	(will2, 1)	(will2, 2)
(ϵ_{C_2}, 1)	-	-	✓
(ϵ_T, 2)	✓	✓	-

Table 3.6: The (Anti-)Licensing Constraints of Y_2: Notice that since the third syntactic feature of ϵ_T may select the second syntactic feature of both $will_1$ and $will_2$, this means that the second syntactic features of both $will_1$ and $will_2$ must share the same symbol. This is verified in the lexicon in Figure 3.5.

	$a :: = z1, \sim z2$
	$cat :: \sim z1$
$\epsilon_C :: = z3, C$	$eat :: = z2, \sim z5$
$\epsilon_{C_2} :: = z3, +z8, C$	$mouse :: \sim z1$
$\epsilon_T :: = z4, = z5, +z6, = z2, \sim z3$	$the :: = z1, \sim z2$
	$will :: \sim z4, -z6$
	$will2 :: \sim z4, -z6, -z8$

Figure 3.5: The *curated* lexicon that the learner has acquired from derivations d_1 and d_2. Note that through the acquisition process, the learner may conclude thus far that the nouns "mouse" and "cat" are the same part of speech (since their lexical chains are different only in phonetic forms). Similarly the learner may conclude that the determiners "the" and "a" are also the the same part of speech. Finally, we note that the first and second syntactic features of *will* and $will_2$ are both the same symbols (z_4 and z_6) respectively.

3.3 Which mouse will the cat eat?

The learner is next exposed to derivation d_3 (see Page 44), which is a modification of d_2 that invokes wh-fronting (see Figure 3.6) to produce the sentence: "which mouse will the cat eat?" This derivation will invoke the *shortest move constraint*, a consequence of which will be the appearance of *anti-licensing* constraints.

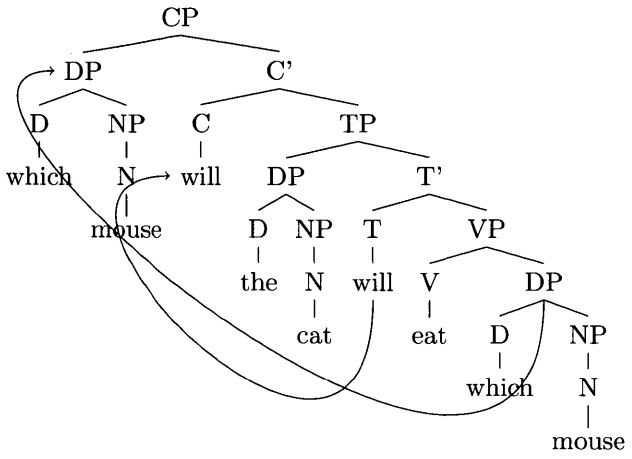

Figure 3.6: Phrase Structure for the sentence "which mouse will the cat eat?"

As before, the learner computes a *curated* lexicon L_3 such that $d_3 \in \mathbb{G}(L_3)$. (see Figure 3.7) The learner then computes their next state of knowledge: $Y_3 = \mathbb{P}(\mathbb{J}(\mathbb{Q}(Y_2), L_3))$. The *selection constraints* and *(anti-)licensing constraints* for $\mathbb{P}(L_3)$ are presented in Tables 3.7 and 3.8 respectively. The *selection constraints* and *(anti-)licensing constraints* for Y_3, the learner's new state of knowledge, are presented in Tables 3.9 and 3.10 respectively. Finally, the lexicon representation of the learner's new state of knowledge (i.e. $\mathbb{Q}(Y_3)$) is presented in Figure 3.8.

	$cat :: \sim x8$
	$eat :: = x2, \sim x4$
$\epsilon_{C_3} :: = x10, +x7, +x6, C$	$mouse :: \sim x1$
$\epsilon_T :: = x3, = x4, +x5, = x9, \sim x10$	$the :: = x8, \sim x9$
	$which :: = x1, \sim x2, -x6$
	$will2 :: \sim x3, -x5, -x7$

Figure 3.7: The *curated* lexicon L_3 from which the derivation d_3 (which produces the interrogative sentence "which mouse will the cat eat?") may be derived. Note that $\{d_3\} = \mathbb{G}(L_3)$.

$$\frac{[\text{the} :: = x8, \sim x9] \qquad [\text{cat} :: \sim x8]}{[\text{the cat} : \sim x9]}\ \text{EM1}$$

$$\frac{[\text{which} :: = x1, \sim x2, -x6] \qquad [\text{mouse} :: \sim x1]}{[\text{which mouse} : \sim x2, -x6]}\ \text{EM1}$$

$$\frac{[\text{eat} :: = x2, \sim x4] \qquad [\text{which mouse} : \sim x2, -x6]}{[\text{eat} : \sim x4][\text{which mouse} : -x6]}\ \text{EM3}$$

$$\frac{[\epsilon_T :: = x3, = x4, +x5, = x9, \sim x10] \qquad [\text{will2} :: \sim x3, -x5, -x7]}{[\epsilon_T : = x4, +x5, = x9, \sim x10][\text{will2} : -x5, -x7]}\ \text{EM3}$$

$$\frac{[\epsilon_T : = x4, +x5, = x9, \sim x10][\text{will2} : -x5, -x7] \qquad [\text{eat} : \sim x4][\text{which mouse} : -x6]}{[\text{eat } \epsilon_T : +x5, = x9, \sim x10][\text{which mouse} : -x6][\text{will2} : -x5, -x7]}\ \text{EM2}$$

$$\frac{[\text{eat } \epsilon_T : +x5, = x9, \sim x10][\text{which mouse} : -x6][\text{will2} : -x5, -x7]}{[\text{eat } \epsilon_T : = x9, \sim x10][\text{will2} : -x7][\text{which mouse} : -x6]}\ \text{IM2}$$

$$\frac{[\text{eat } \epsilon_T : = x9, \sim x10][\text{will2} : -x7][\text{which mouse} : -x6] \qquad [\text{the cat} : \sim x9]}{[\text{the cat eat } \epsilon_T : \sim x10][\text{which mouse} : -x6][\text{will2} : -x7]}\ \text{EM2}$$

$$\frac{[\epsilon_{C_3} :: = x10, +x7, +x6, \text{C}] \qquad [\text{the cat eat } \epsilon_T : \sim x10][\text{which mouse} : -x6][\text{will2} : -x7]}{[\epsilon_{C_3} \text{ the cat eat } \epsilon_T : +x7, +x6, \text{C}][\text{which mouse} : -x6][\text{will2} : -x7]}\ \text{EM1}$$

$$\frac{[\epsilon_{C_3} \text{ the cat eat } \epsilon_T : +x7, +x6, \text{C}][\text{which mouse} : -x6][\text{will2} : -x7]}{[\text{will2 } \epsilon_{C_3} \text{ the cat eat } \epsilon_T : +x6, \text{C}][\text{which mouse} : -x6]}\ \text{IM1}$$

$$\frac{[\text{will2 } \epsilon_{C_3} \text{ the cat eat } \epsilon_T : +x6, \text{C}][\text{which mouse} : -x6]}{[\text{which mouse will2 } \epsilon_{C_3} \text{ the cat eat } \epsilon_T : \text{C}]}\ \text{IM1}$$

Selectors \ Selectees	(cat, 0)	(eat, 1)	(mouse, 0)	(the, 1)	(which, 1)	(will2, 0)	(ϵ_T, 4)
(eat, 0)	-	-	-	-	✓	-	-
(the, 0)	✓	-	-	-	-	-	-
(which, 0)	-	-	✓	-	-	-	-
(ϵ_{C_3}, 0)	-	-	-	-	-	-	✓
(ϵ_T, 0)	-	-	-	-	-	✓	-
(ϵ_T, 1)	-	✓	-	-	-	-	-
(ϵ_T, 3)	-	-	-	✓	-	-	-

Table 3.7: The Selection Constraints of $\mathbb{P}(L_3)$.

Licensors \ Licensees	(which, 2)	(will2, 1)	(will2, 2)
(ϵ_{C_3}, 1)	✗	-	✓
(ϵ_{C_3}, 2)	✓	✗	✗
(ϵ_T, 2)	✗	✓	-

Table 3.8: (Anti-)Licensing Constraints of $\mathbb{P}(L_3)$: The learner must now process not only *licensing* constraints, denoted by a ✓, but also *anti-licensing* constraints, denoted by an ✗. The *anti-licensing* constraints explicitly indicate that certain syntactic features *cannot* be licensed together; intuitively, a combination of *licensing* and *anti-licensing* constraints will ensure that during a derivation, *internal-merge* operations are deterministic in the choice of *licensee*.

Unlike in the previous acquisition step, the learner now acquires new knowledge (constraints) in the process of combining the constraints in Y_2 and $\mathbb{P}(L_3)$ that could not acquire from Y_2 or $\mathbb{P}(L_3)$ alone. (See the designated constraints in Tables 3.9 and 3.10) A specific example of a sentence the learner may now produce from the lexicon $\mathbb{Q}(Y_3)$ (see Figure 3.8) that the learner could not produce from the lexicon $\mathbb{Q}(Y_2)$ (see Figure 3.5) is the sentence "which cat will eat the mouse?" The next section will expose the learner to a derivation that produces this sentence and demonstrate that the learner does not acquire any new constraints; hence $\mathbb{Q}(Y_4) = \mathbb{Q}(Y_3)$.

Selectors \ Selectees	(a, 1)	(cat, 0)	(eat, 1)	(mouse, 0)	(the, 1)	(which, 1)	(will, 0)	(will2, 0)	(ϵ_T, 4)
(a, 0)	-	✓	-	✓	-	-	-	-	-
(eat, 0)	✓	-	-	-	✓	✓	-	-	-
(the, 0)	-	✓	-	✓	-	-	-	-	-
(which, 0)	-	✓	-	✓	-	-	-	-	-
(ϵ_C, 0)	-	-	-	-	-	-	-	-	✓
(ϵ_{C_2}, 0)	-	-	-	-	-	-	-	-	✓
(ϵ_{C_3}, 0)	-	-	-	-	-	-	-	-	✓
(ϵ_T, 0)	-	-	-	-	-	-	✓	✓	-
(ϵ_T, 1)	-	-	✓	-	-	-	-	-	-
(ϵ_T, 3)	✓	-	-	-	✓	✓	-	-	-

Table 3.9: The Selection Constraints of Y_3: This table displays the selection constraints that arise in combining $\mathbb{Q}(Y_2)$ and L_3 (see Tables 3.5 and 3.7), thus representing the selection constraints accumulated from d_1, d_2 and d_3. New constraints that may be derived only from the combination of the two sets of selection constraints are underlined. If two syntactic features share either the same row or column then they share the same base symbol (and vice-versa).

Licensors	Licensees: (which, 2)	(will, 1)	(will2, 1)	(will2, 2)
$(\epsilon_{C_2}, 1)$	✗	-	-	✓
$(\epsilon_{C_3}, 1)$	✗	-	-	✓
$(\epsilon_{C_3}, 2)$	✓	✗	✗	✗
$(\epsilon_T, 2)$	✗	✓	✓	-

Table 3.10: The (Anti-)Licensing Constraints Y_3: This table displays the (anti-)licensing constraints that arise in combining $\mathbb{Q}(Y_2)$ and L_3 (see Tables 3.6 and 3.8), thus representing the (anti-)licensing constraints accumulated from d_1, d_2 and d_3. Licensing constraints are designated by an ✓whereas anti-licensing constraints are designated by an ✗. New constraints that may be derived only from the combination of the two sets of selection constraints are underlined. If two syntactic features share either the same row or column then they share the same base symbol (and vice-versa). Note the substantial number of anti-licensing constraints that will prohibit the learner from acquiring derivations that involve certain licensing operations. (e.g. the *anti-licensing* constraints $((which, 2), (\epsilon_{C_2}, 1))$, $((which, 2), (\epsilon_{C_3}, 1))$, $((which, 2), (\epsilon_T, 2))$ and $((will2, 2), (\epsilon_{C_3}, 2))$ ensure that any lexical item that licenses "which" may never license "will2" and vice-versa).

$$
\begin{array}{rr}
 & a :: = z1, \sim z2 \\
 & cat :: \sim z1 \\
\epsilon_C :: = z3, C & eat :: = z2, \sim z5 \\
\epsilon_{C_2} :: = z3, +z6, C & mouse :: \sim z1 \\
\epsilon_{C_3} :: = z3, +z6, +z7, C & the :: = z1, \sim z2 \\
\epsilon_T :: = z4, = z5, +z8, = z2, \sim z3 & which :: = z1, \sim z2, -z7 \\
 & will :: \sim z4, -z8 \\
 & will2 :: \sim z4, -z8, -z6
\end{array}
$$

Figure 3.8: The lexicon acquired by the learner from derivations $d_1, \cdots, d_3$. Notice that "which", "the" and "a" share the same syntactic features in the first and second position, which alludes to a generalization that these words all belong to the same category (determiners). Similarly note how common indices of ϵ_C share the same symbol, which suggests the generalization that ϵ_C, ϵ_{C_2} and ϵ_{C_3} all belong to the same category (CP).

3.4 Which cat will eat the mouse?

Whereas the previous derivation that the learner encountered involved *wh-fronting* from the object position, the learner will now encounters a sentence that involves *wh-fronting* from the subject position, as shown by the phrase structure in Figure 3.9. The derivation d_4, which produces the sentence "which cat will eat the mouse?" (see Figure 49) is produced by the lexicon L_4 in 3.10.

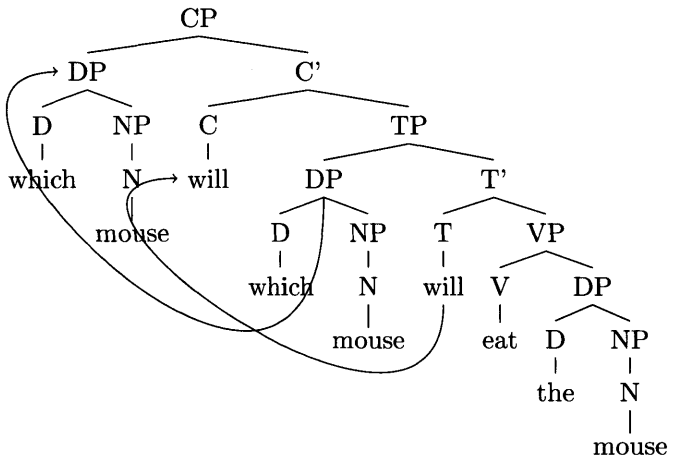

Figure 3.9: Phrase Structure for the sentence "which cat will eat the mouse?"

As before, the learner computes a *curated* lexicon L_4 such that $d_4 \in \mathbb{G}(L_4)$. (see Figure 3.10) The learner then computes their next state of knowledge: $Y_4 = \mathbb{P}(\mathbb{J}(\mathbb{Q}(Y_3), L_4))$. The *selection constraints* and *(anti-)licensing constraints* for $\mathbb{P}(L_4)$ are presented in Tables 3.11 and 3.12 respectively. The *selection constraints* and *(anti-)licensing constraints* for Y_4 are presented in Tables 3.13 and 3.14 respectively. Finally, the lexicon representation of the learner's new state of knowledge (i.e. $\mathbb{Q}(Y_4)$) is presented in Figure 3.11.

	$cat :: \sim x8$
	$eat :: = x2, \sim x4$
$\epsilon_{C_3} :: = x10, +x7, +x6, C$	$mouse :: \sim x1$
$\epsilon_T :: = x3, = x4, +x5, = x9, \sim x10$	$the :: = x1, \sim x2$
	$which :: = x8, \sim x9, -x6$
	$will2 :: \sim x3, -x5, -x7$

Figure 3.10: The *curated* lexicon L_4 from which the derivation d_4 (which produces the interrogative sentence "which cat will eat the mouse?") may be derived. Note that $\{d_4\} = \mathbb{G}(L_4)$.

$$\frac{[\text{which} :: = x8,\sim x9,-x6] \qquad [\text{cat} :: \sim x8]}{[\text{which cat} : \sim x9,-x6]}\ \text{EM1}$$

$$\frac{[\text{the} :: = x1,\sim x2] \qquad [\text{mouse} :: \sim x1]}{[\text{the mouse} : \sim x2]}\ \text{EM1}$$

$$\frac{[\text{eat} :: = x2,\sim x4] \qquad [\text{the mouse} : \sim x2]}{[\text{eat the mouse} : \sim x4]}\ \text{EM1}$$

$$\frac{[\epsilon_T :: = x3,= x4,+x5,= x9,\sim x10] \qquad [\text{will2} :: \sim x3,-x5,-x7]}{[\epsilon_T : = x4,+x5,= x9,\sim x10][\text{will2} : -x5,-x7]}\ \text{EM3}$$

$$\frac{[\epsilon_T : = x4,+x5,= x9,\sim x10][\text{will2} : -x5,-x7] \qquad [\text{eat the mouse} : \sim x4]}{[\text{eat the mouse } \epsilon_T : +x5,= x9,\sim x10][\text{will2} : -x5,-x7]}\ \text{EM2}$$

$$\frac{[\text{eat the mouse } \epsilon_T : +x5,= x9,\sim x10][\text{will2} : -x5,-x7]}{[\text{eat the mouse } \epsilon_T : = x9,\sim x10][\text{will2} : -x7]}\ \text{IM2}$$

$$\frac{[\text{eat the mouse } \epsilon_T : = x9,\sim x10][\text{will2} : -x7] \qquad [\text{which cat} : \sim x9,-x6]}{[\text{eat the mouse } \epsilon_T : \sim x10][\text{which cat} : -x6][\text{will2} : -x7]}\ \text{EM3}$$

$$\frac{[\epsilon_{C_3} :: = x10,+x7,+x6,\text{C}] \qquad [\text{eat the mouse } \epsilon_T : \sim x10][\text{which cat} : -x6][\text{will2} : -x7]}{[\epsilon_{C_3} \text{ eat the mouse } \epsilon_T : +x7,+x6,\text{C}][\text{which cat} : -x6][\text{will2} : -x7]}\ \text{EM1}$$

$$\frac{[\epsilon_{C_3} \text{ eat the mouse } \epsilon_T : +x7,+x6,\text{C}][\text{which cat} : -x6][\text{will2} : -x7]}{[\text{will2 } \epsilon_{C_3} \text{ eat the mouse } \epsilon_T : +x6,\text{C}][\text{which cat} : -x6]}\ \text{IM1}$$

$$\frac{[\text{will2 } \epsilon_{C_3} \text{ eat the mouse } \epsilon_T : +x6,\text{C}][\text{which cat} : -x6]}{[\text{which cat will2 } \epsilon_{C_3} \text{ eat the mouse } \epsilon_T : \text{C}]}\ \text{IM1}$$

Selectors \ Selectees	(cat, 0)	(eat, 1)	(mouse, 0)	(the, 1)	(which, 1)	(will2, 0)	(ϵ_T, 4)
(eat, 0)	-	-	-	✓	-	-	-
(the, 0)	-	-	✓	-	-	-	-
(which, 0)	✓	-	-	-	-	-	-
(ϵ_{C_3}, 0)	-	-	-	-	-	-	✓
(ϵ_T, 0)	-	-	-	-	-	✓	-
(ϵ_T, 1)	-	✓	-	-	-	-	-
(ϵ_T, 3)	-	-	-	-	✓	-	-

Table 3.11: The Selection Constraints of $\mathbb{P}(L_4)$.

Licensors \ Licensees	(which, 2)	(will2, 1)	(will2, 2)
(ϵ_{C_3}, 1)	✗	-	✓
(ϵ_{C_3}, 2)	✓	-	✗
(ϵ_T, 2)	-	✓	-

Table 3.12: The (Anti-)Licensing Constraints of $\mathbb{P}(L_4)$.

Note that neither Table 3.13 nor Table 3.14 have any *new* constraints that cannot be found in either Y_3 or $\mathbb{P}(L_4)$. This indicates that the learner has not learned any new information from this derivation. The reader can verify by this noting that the lexicon in Figure 3.11 is the same as the lexicon in Figure 3.8. Hence the learner was able to generalize *wh-movement* to apply to determiners moving from both object *and* subject position from the first three derivations alone.

Selectors \ Selectees	(a, 1)	(cat, 0)	(eat, 1)	(mouse, 0)	(the, 1)	(which, 1)	(will, 0)	(will2, 0)	(ϵ_T, 4)
(a, 0)	-	✓	-	✓	-	-	-	-	-
(eat, 0)	✓	-	-	-	✓	✓	-	-	-
(the, 0)	-	✓	-	✓	-	-	-	-	-
(which, 0)	-	✓	-	✓	-	-	-	-	-
(ϵ_C, 0)	-	-	-	-	-	-	-	-	✓
(ϵ_{C_2}, 0)	-	-	-	-	-	-	-	-	✓
(ϵ_{C_3}, 0)	-	-	-	-	-	-	-	-	✓
(ϵ_T, 0)	-	-	-	-	-	-	✓	✓	-
(ϵ_T, 1)	-	-	✓	-	-	-	-	-	-
(ϵ_T, 3)	✓	-	-	-	✓	✓	-	-	-

Table 3.13: The Selection Constraints of Y_4: this table displays the selection constraints that arise in combining $\mathbb{Q}(Y_3)$ and L_4. Note that there are no newly derived (underlined) constraints; this is because the learner has thus far acquired the generalization that determiners involved in *wh-fronting* are subject to movement regardless of whether they are located in the specifier or complement position. Note that there is no difference between this table and Table 3.9.

Licensors \ Licensees	(which, 2)	(will, 1)	(will2, 1)	(will2, 2)
(ϵ_{C_2}, 1)	✗	-	-	✓
(ϵ_{C_3}, 1)	✗	-	-	✓
(ϵ_{C_3}, 2)	✓	✗	✗	✗
(ϵ_T, 2)	✗	✓	✓	-

Table 3.14: The (Anti-)Licensing Constraints of Y_4: this table displays the selection constraints that arise in combining $\mathbb{Q}(Y_3)$ and L_4. Note that there are no newly derived (underlined) constraints; this is because the learner has thus far acquired the generalization that determiners involved in *wh-fronting* are subject to movement regardless of whether they are located in the specifier or complement position. Note that there is no difference between this table and Table 3.10.

	$a :: = z1, \sim z2$
	$cat :: \sim z1$
$\epsilon_C :: = z3, C$	$eat :: = z2, \sim z5$
$\epsilon_{C_2} :: = z3, +z6, C$	$mouse :: \sim z1$
$\epsilon_{C_3} :: = z3, +z6, +z7, C$	$the :: = z1, \sim z2$
$\epsilon_T :: = z4, = z5, +z8, = z2, \sim z3$	$which :: = z1, \sim z2, -z7$
	$will :: \sim z4, -z8$
	$will2 :: \sim z4, -z8, -z6$

Figure 3.11: The lexicon acquired by the learner from derivations $d_1, \cdots, d_4$. Notice that this lexicon is exactly the same as the lexicon acquired from sentences 1-3 (Figure 3.8), indicating that the learner has already learned that *wh-movement* (via *internal merge*) may originate from either the *subject* or *object* position.

3.5 The man will think that the cat will eat a mouse.

The learner next encounters a sentence that has an embedded sentence within it (see Figure 3.12). Specifically, the learner is presented with derivation d_5 (see Figure 55), which produces the sentence "the man will think that the cat will eat a mouse." and is produced in turn by the lexicon L_5 in 3.13.

As before, the learner computes a lexicon L_5 such that $d_5 \in \mathbb{G}(L_5)$. (see Figure 3.13) The learner then computes their next state of knowledge: $Y_5 = \mathbb{P}(\mathbb{J}(\mathbb{Q}(Y_4), L_5))$. The *selection constraints* and *(anti-)licensing constraints* for $\mathbb{P}(L_5)$ are presented in Tables 3.15 and 3.16 respectively. The *selection constraints* and *(anti-)licensing constraints* for Y_5 are presented in Tables 3.17 and 3.18 respectively. Finally, the lexicon representation of the learner's new state of knowledge (i.e. $\mathbb{Q}(Y_5)$) is presented in Figure 3.14.

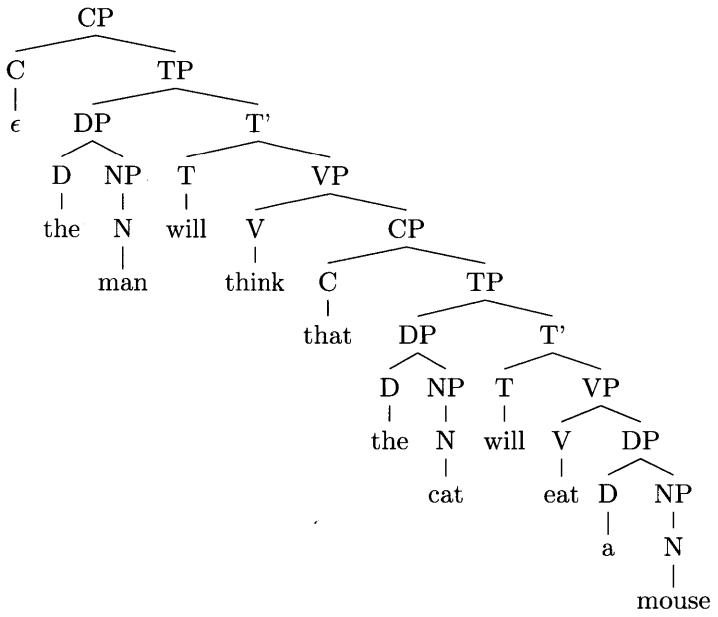

Figure 3.12: Phrase Structure for the sentence "the man will think that the cat will eat a mouse."

	$a :: = x1, \sim x2$
	$cat :: \sim x6$
	$eat :: = x2, \sim x4$
	$man :: \sim x15$
$\epsilon_C :: = x14, C$	$mouse :: \sim x1$
$\epsilon_T :: = x10, = x11, +x12, = x13, \sim x14$	$that :: = x8, \sim x9$
$\epsilon_T :: = x3, = x4, +x5, = x7, \sim x8$	$the :: = x15, \sim x13$
	$the :: = x6, \sim x7$
	$think :: = x9, \sim x11$
	$will :: \sim x10, -x12$
	$will :: \sim x3, -x5$

Figure 3.13: The lexicon L_5 from which the derivation d_5 (which produces the sentence "the man will think that the cat will eat a mouse.") may be derived. Note that $\{d_5\} = \mathbb{G}(L_5)$. Furthermore, note that this lexicon is not *curated*; nevertheless, because the constraints between syntactic features are a function of the phonetic form and the index of the feature, the two copies of the "will" lexical item will be unified into a single copy by the procedure $\mathbb{J}$.

$$\frac{[\text{the} :: = x15, \sim x13] \qquad [\text{man} :: \sim x15]}{[\text{the man} : \sim x13]}\ \text{EM1}$$

$$\frac{[\text{the} :: = x6, \sim x7] \qquad [\text{cat} :: \sim x6]}{[\text{the cat} : \sim x7]}\ \text{EM1}$$

$$\frac{[\text{a} :: = x1, \sim x2] \qquad [\text{mouse} :: \sim x1]}{[\text{a mouse} : \sim x2]}\ \text{EM1}$$

$$\frac{[\text{eat} :: = x2, \sim x4] \qquad [\text{a mouse} : \sim x2]}{[\text{eat a mouse} : \sim x4]}\ \text{EM1}$$

$$\frac{[\epsilon_T :: = x3, = x4, +x5, = x7, \sim x8] \qquad [\text{will} :: \sim x3, -x5]}{[\epsilon_T : = x4, +x5, = x7, \sim x8][\text{will} : -x5]}\ \text{EM3}$$

$$\frac{[\epsilon_T : = x4, +x5, = x7, \sim x8][\text{will} : -x5] \qquad [\text{eat a mouse} : \sim x4]}{[\text{eat a mouse } \epsilon_T : +x5, = x7, \sim x8][\text{will} : -x5]}\ \text{EM2}$$

$$\frac{[\text{eat a mouse } \epsilon_T : +x5, = x7, \sim x8][\text{will} : -x5]}{[\text{will eat a mouse } \epsilon_T : = x7, \sim x8]}\ \text{IM1}$$

$$\frac{[\text{will eat a mouse } \epsilon_T : = x7, \sim x8] \qquad [\text{the cat} : \sim x7]}{[\text{the cat will eat a mouse } \epsilon_T : \sim x8]}\ \text{EM2}$$

$$\frac{[\text{that} :: = x8, \sim x9] \qquad [\text{the cat will eat a mouse } \epsilon_T : \sim x8]}{[\text{that the cat will eat a mouse } \epsilon_T : \sim x9]}\ \text{EM1}$$

$$\frac{[\text{think} :: = x9, \sim x11] \qquad [\text{that the cat will eat a mouse } \epsilon_T : \sim x9]}{[\text{think that the cat will eat a mouse } \epsilon_T : \sim x11]}\ \text{EM1}$$

$$\frac{[\epsilon_T :: = x10, = x11, +x12, = x13, \sim x14] \qquad [\text{will} :: \sim x10, -x12]}{[\epsilon_T : = x11, +x12, = x13, \sim x14][\text{will} : -x12]}\ \text{EM3}$$

$$\frac{[\epsilon_T : = x11, +x12, = x13, \sim x14][\text{will} : -x12] \qquad [\text{think that the cat will eat a mouse } \epsilon_T : \sim x11]}{[\text{think that the cat will eat a mouse } \epsilon_T \epsilon_T : +x12, = x13, \sim x14][\text{will} : -x12]}\ \text{EM2}$$

$$\frac{[\text{think that the cat will eat a mouse } \epsilon_T \epsilon_T : +x12, = x13, \sim x14][\text{will} : -x12]}{[\text{will think that the cat will eat a mouse } \epsilon_T \epsilon_T : = x13, \sim x14]}\ \text{IM1}$$

$$\frac{[\text{will think that the cat will eat a mouse } \epsilon_T \epsilon_T : = x13, \sim x14] \qquad [\text{the man} : \sim x13]}{[\text{the man will think that the cat will eat a mouse } \epsilon_T \epsilon_T : \sim x14]}\ \text{EM2}$$

$$\frac{[\epsilon_C :: = x14, \text{C}] \qquad [\text{the man will think that the cat will eat a mouse } \epsilon_T \epsilon_T : \sim x14]}{[\epsilon_C \text{ the man will think that the cat will eat a mouse } \epsilon_T \epsilon_T : \text{C}]}\ \text{EM1}$$

Selectors \ Selectees	(a, 1)	(cat, 0)	(eat, 1)	(man, 0)	(mouse, 0)	(that, 1)	(the, 1)	(think, 1)	(will, 0)	(ϵ_T, 4)
(a, 0)	-	-	-	-	✓	-	-	-	-	-
(eat, 0)	✓	-	-	-	-	-	-	-	-	-
(that, 0)	-	-	-	-	-	-	-	-	-	✓
(the, 0)	-	✓	-	✓	-	-	-	-	-	-
(think, 0)	-	-	-	-	-	✓	-	-	-	-
(ϵ_C, 0)	-	-	-	-	-	-	-	-	-	✓
(ϵ_T, 0)	-	-	-	-	-	-	-	-	✓	-
(ϵ_T, 1)	-	-	✓	-	-	-	-	✓	-	-
(ϵ_T, 3)	-	-	-	-	-	-	✓	-	-	-

Table 3.15: The Selection Constraints of $\mathbb{P}(L_5)$.

Licensors \ Licensees	(will, 1)
(ϵ_T, 2)	✓

Table 3.16: The (Anti-)Licensing Constraints of $\mathbb{P}(L_5)$.

Selectors	Selectees: (a, 1)	(cat, 0)	(eat, 1)	(man, 0)	(mouse, 0)	(that, 1)	(the, 1)	(think, 1)	(which, 1)	(will, 0)	(will2, 0)	(ϵ_T, 4)
(a, 0)	-	✓	-	✓	✓	-	-	-	-	-	-	-
(eat, 0)	✓	-	-	-	-	-	✓	-	✓	-	-	-
(that, 0)	-	-	-	-	-	-	-	-	-	-	-	✓
(the, 0)	-	✓	-	✓	✓	-	-	-	-	-	-	-
(think, 0)	-	-	-	-	-	✓	-	-	-	-	-	-
(which, 0)	-	✓	-	✓	✓	-	-	-	-	-	-	-
(ϵ_C, 0)	-	-	-	-	-	-	-	-	-	-	-	✓
(ϵ_{C_2}, 0)	-	-	-	-	-	-	-	-	-	-	-	✓
(ϵ_{C_3}, 0)	-	-	-	-	-	-	-	-	-	-	-	✓
(ϵ_T, 0)	-	-	-	-	-	-	-	-	-	✓	✓	-
(ϵ_T, 1)	-	-	✓	-	-	-	-	✓	-	-	-	-
(ϵ_T, 3)	✓	-	-	-	-	-	✓	-	✓	-	-	-

Table 3.17: The Selection Constraints of Y_5. The newly derived constraints (underlined) indicate that "man", "mouse" and "cat" all belong to the same category (NP) and may all merge with the determiners ("which", "the" and "a").

Licensors	Licensees: (which, 2)	(will, 1)	(will2, 1)	(will2, 2)
(ϵ_{C_2}, 1)	✗	-	-	✓
(ϵ_{C_3}, 1)	✗	-	-	✓
(ϵ_{C_3}, 2)	✓	✗	✗	✗
(ϵ_T, 2)	✗	✓	✓	-

Table 3.18: The (Anti-)Licensing Constraints of Y_5.

Note that the learner acquires new *selection constraints* which allow him to determine that the words "man", "mouse" and "cat" all belong to the same category (NP) and may all merge with the determiners ("which", "the" and "a"). However the learner does *not* acquire any new *(anti-)licensing constraints.*

	$a :: = z1, \sim z2$
	$cat :: \sim z1$
	$eat :: = z2, \sim z6$
	$man :: \sim z1$
$\epsilon_C :: = z3, C$	$mouse :: \sim z1$
$\epsilon_{C_2} :: = z3, +z7, C$	$that :: = z3, \sim z4$
$\epsilon_{C_3} :: = z3, +z7, +z8, C$	$the :: = z1, \sim z2$
$\epsilon_T :: = z5, = z6, +z9, = z2, \sim z3$	$think :: = z4, \sim z6$
	$which :: = z1, \sim z2, -z8$
	$will :: \sim z5, -z9$
	$will2 :: \sim z5, -z9, -z7$

Figure 3.14: The lexicon acquired by the learner from derivations $d_1, \cdots, d_5$.

3.6 Will the man think that the cat will eat a mouse?

The learner next encounters a derivation that is the interrogative variant of d_5 that results from subject auxiliary-verb inversion (see Figure 3.15). The learner is presented with derivation d_6 (see Figure 61), which produces the sentence "will the man think that the cat will eat a mouse?" and is produced in turn by the lexicon L_6 in 3.16.

As before, the learner computes a lexicon L_6 such that $d_6 \in \mathbb{G}(L_6)$. (see Figure 3.16) The learner then computes their next state of knowledge: $Y_6 = \mathbb{P}(\mathbb{J}(\mathbb{Q}(Y_5), L_6))$. The *selection constraints* and *(anti-)licensing constraints* for $\mathbb{P}(L_6)$ are presented in Tables 3.19 and 3.20 respectively. The *selection constraints* and *(anti-)licensing constraints* for Y_6 are presented in Tables 3.21 and 3.22 respectively. Finally, the lexicon representation of the learner's new state of knowledge (i.e. $\mathbb{Q}(Y_6)$) is presented in Figure 3.17.

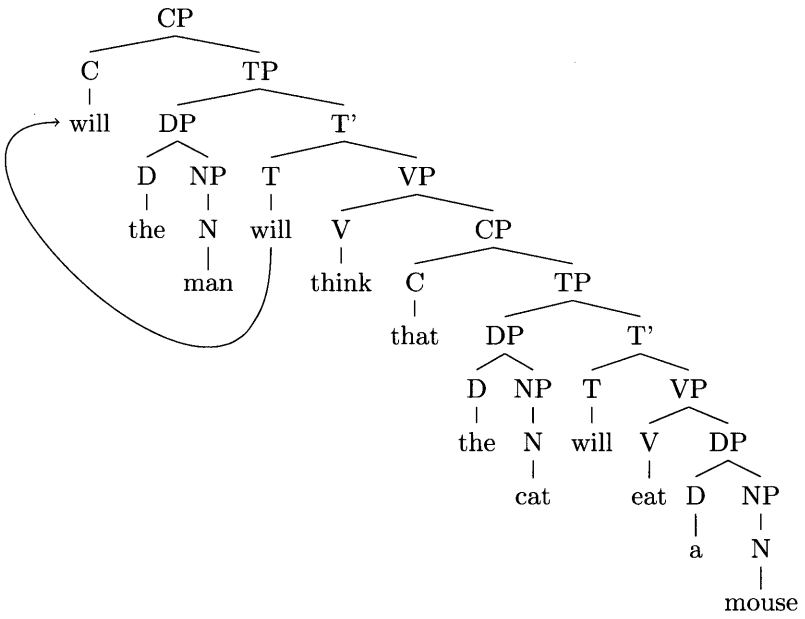

Figure 3.15: Phrase Structure for the sentence "will the man think that the cat will eat a mouse?"

	$a :: = x1, \sim x2$
	$cat :: \sim x6$
	$eat :: = x2, \sim x4$
	$man :: \sim x15$
$\epsilon_{C_2} :: = x14, +x16, C$	$mouse :: \sim x1$
$\epsilon_T :: = x10, = x11, +x12, = x13, \sim x14$	$that :: = x8, \sim x9$
$\epsilon_T :: = x3, = x4, +x5, = x7, \sim x8$	$the :: = x15, \sim x13$
	$the :: = x6, \sim x7$
	$think :: = x9, \sim x11$
	$will :: \sim x3, -x5$
	$will2 :: \sim x10, -x12, -x16$

Figure 3.16: The lexicon L_6 from which the derivation d_6 (which produces the sentence "will the man think that the cat will eat a mouse?") may be derived. Note that $\{d_6\} = \mathbb{G}(L_6)$.

$$\frac{[\text{the} :: = x15, \sim x13] \qquad [\text{man} :: \sim x15]}{[\text{the man} : \sim x13]}\ \text{EM1}$$

$$\frac{[\text{the} :: = x6, \sim x7] \qquad [\text{cat} :: \sim x6]}{[\text{the cat} : \sim x7]}\ \text{EM1}$$

$$\frac{[\text{a} :: = x1, \sim x2] \qquad [\text{mouse} :: \sim x1]}{[\text{a mouse} : \sim x2]}\ \text{EM1}$$

$$\frac{[\text{eat} :: = x2, \sim x4] \qquad [\text{a mouse} : \sim x2]}{[\text{eat a mouse} : \sim x4]}\ \text{EM1}$$

$$\frac{[\epsilon_T :: = x3, = x4, +x5, = x7, \sim x8] \qquad [\text{will} :: \sim x3, -x5]}{[\epsilon_T : = x4, +x5, = x7, \sim x8][\text{will} : -x5]}\ \text{EM3}$$

$$\frac{[\epsilon_T : = x4, +x5, = x7, \sim x8][\text{will} : -x5] \qquad [\text{eat a mouse} : \sim x4]}{[\text{eat a mouse } \epsilon_T : +x5, = x7, \sim x8][\text{will} : -x5]}\ \text{EM2}$$

$$\frac{[\text{eat a mouse } \epsilon_T : +x5, = x7, \sim x8][\text{will} : -x5]}{[\text{will eat a mouse } \epsilon_T : = x7, \sim x8]}\ \text{IM1}$$

$$\frac{[\text{will eat a mouse } \epsilon_T : = x7, \sim x8] \qquad [\text{the cat} : \sim x7]}{[\text{the cat will eat a mouse } \epsilon_T : \sim x8]}\ \text{EM2}$$

$$\frac{[\text{that} :: = x8, \sim x9] \qquad [\text{the cat will eat a mouse } \epsilon_T : \sim x8]}{[\text{that the cat will eat a mouse } \epsilon_T : \sim x9]}\ \text{EM1}$$

$$\frac{[\text{think} :: = x9,\sim x11] \qquad [\text{that the cat will eat a mouse } \epsilon_T : \sim x9]}{[\text{think that the cat will eat a mouse } \epsilon_T : \sim x11]}\ \text{EM1}$$

$$\frac{[\epsilon_T :: = x10,= x11,+x12,= x13,\sim x14] \qquad [\text{will2} :: \sim x10,-x12,-x16]}{[\epsilon_T : = x11,+x12,= x13,\sim x14][\text{will2} : -x12,-x16]}\ \text{EM3}$$

$$\frac{[\epsilon_T : = x11,+x12,= x13,\sim x14][\text{will2} : -x12,-x16] \qquad [\text{think that the cat will eat a mouse } \epsilon_T : \sim x11]}{[\text{think that the cat will eat a mouse } \epsilon_T\epsilon_T : +x12,= x13,\sim x14][\text{will2} : -x12,-x16]}\ \text{EM2}$$

$$\frac{[\text{think that the cat will eat a mouse } \epsilon_T\epsilon_T : +x12,= x13,\sim x14][\text{will2} : -x12,-x16]}{[\text{think that the cat will eat a mouse } \epsilon_T\epsilon_T : = x13,\sim x14][\text{will2} : -x16]}\ \text{IM2}$$

$$\frac{[\text{think that the cat will eat a mouse } \epsilon_T\epsilon_T : = x13,\sim x14][\text{will2} : -x16] \qquad [\text{the man} : \sim x13]}{[\text{the man think that the cat will eat a mouse } \epsilon_T\epsilon_T : \sim x14][\text{will2} : -x16]}\ \text{EM2}$$

$$\frac{[\epsilon_{C_2} :: = x14,+x16,\text{C}] \qquad [\text{the man think that the cat will eat a mouse } \epsilon_T\epsilon_T : \sim x14][\text{will2} : -x16]}{[\epsilon_{C_2} \text{ the man think that the cat will eat a mouse } \epsilon_T\epsilon_T : +x16,\text{C}][\text{will2} : -x16]}\ \text{EM1}$$

$$\frac{[\epsilon_{C_2} \text{ the man think that the cat will eat a mouse } \epsilon_T\epsilon_T : +x16,\text{C}][\text{will2} : -x16]}{[\text{will2 } \epsilon_{C_2} \text{ the man think that the cat will eat a mouse } \epsilon_T\epsilon_T : \text{C}]}\ \text{IM1}$$

Selectors \ Selectees	(a, 1)	(cat, 0)	(eat, 1)	(man, 0)	(mouse, 0)	(that, 1)	(the, 1)	(think, 1)	(will, 0)	(will2, 0)	(ϵ_T, 4)
(a, 0)	-	-	-	-	✓	-	-	-	-	-	-
(eat, 0)	✓	-	-	-	-	-	-	-	-	-	-
(that, 0)	-	-	-	-	-	-	-	-	-	-	✓
(the, 0)	-	✓	-	✓	-	-	-	-	-	-	-
(think, 0)	-	-	-	-	-	✓	-	-	-	-	-
(ϵ_{C_2}, 0)	-	-	-	-	-	-	-	-	-	-	✓
(ϵ_T, 0)	-	-	-	-	-	-	-	-	✓	✓	-
(ϵ_T, 1)	-	-	✓	-	-	-	-	✓	-	-	-
(ϵ_T, 3)	-	-	-	-	-	-	✓	-	-	-	-

Table 3.19: The Selection Constraints of $\mathbb{P}(L_6)$.

Licensors \ Licensees	(will, 1)	(will2, 1)	(will2, 2)
(ϵ_{C_2}, 1)	-	-	✓
(ϵ_T, 2)	✓	✓	-

Table 3.20: The (Anti-)Licensing Constraints of $\mathbb{P}(L_6)$.

Selectors \ Selectees	(a, 1)	(cat, 0)	(eat, 1)	(man, 0)	(mouse, 0)	(that, 1)	(the, 1)	(think, 1)	(which, 1)	(will, 0)	(will2, 0)	(ϵ_T, 4)
(a, 0)	-	✓	-	✓	✓	-	-	-	-	-	-	-
(eat, 0)	✓	-	-	-	-	-	✓	-	✓	-	-	-
(that, 0)	-	-	-	-	-	-	-	-	-	-	-	✓
(the, 0)	-	✓	-	✓	✓	-	-	-	-	-	-	-
(think, 0)	-	-	-	-	-	✓	-	-	-	-	-	-
(which, 0)	-	✓	-	✓	✓	-	-	-	-	-	-	-
(ϵ_C, 0)	-	-	-	-	-	-	-	-	-	-	-	✓
(ϵ_{C_2}, 0)	-	-	-	-	-	-	-	-	-	-	-	✓
(ϵ_{C_3}, 0)	-	-	-	-	-	-	-	-	-	-	-	✓
(ϵ_T, 0)	-	-	-	-	-	-	-	-	-	✓	✓	-
(ϵ_T, 1)	-	-	✓	-	-	-	-	✓	-	-	-	-
(ϵ_T, 3)	✓	-	-	-	-	-	✓	-	✓	-	-	-

Table 3.21: The Selection Constraints of Y_6. Note that there are no newly derived constraints (i.e. no underlined entries). This is a consequence of the fact that $d_6 \in \mathbb{G}(\mathbb{Q}(Y_5))$.

Licensors \ Licensees	(which, 2)	(will, 1)	(will2, 1)	(will2, 2)
(ϵ_{C_2}, 1)	✗	-	-	✓
(ϵ_{C_3}, 1)	✗	-	-	✓
(ϵ_{C_3}, 2)	✓	✗	✗	✗
(ϵ_T, 2)	✗	✓	✓	-

Table 3.22: The (Anti-)Licensing Constraints of Y_6. Note that there are no newly derived constraints (i.e. no underlined entries). This is a consequence of the fact that $d_6 \in \mathbb{G}(\mathbb{Q}(Y_5))$.

Note that, as in Section 3.4, the learner has not acquired any new knowledge (i.e. constraints) after exposure to d_6; this indicates that $d_6 \in \mathbb{G}(\mathbb{Q}(Y_5))$. The reader may confirm this by observing that Figure 3.14 and Figure 3.17 are exactly the same.

	$a :: = z1, \sim z2$
	$cat :: \sim z1$
	$eat :: = z2, \sim z6$
	$man :: \sim z1$
$\epsilon_C :: = z3, C$	$mouse :: \sim z1$
$\epsilon_{C_2} :: = z3, +z7, C$	$that :: = z3, \sim z4$
$\epsilon_{C_3} :: = z3, +z7, +z8, C$	$the :: = z1, \sim z2$
$\epsilon_T :: = z5, = z6, +z9, = z2, \sim z3$	$think :: = z4, \sim z6$
	$which :: = z1, \sim z2, -z8$
	$will :: \sim z5, -z9$
	$will2 :: \sim z5, -z9, -z7$

Figure 3.17: The lexicon acquired from derivations $d_1, \cdots, d_6$. Note that this lexicon is exactly the same as the lexicon presented in Figure 3.14. This indicates that the learner already acquired all knowledge necessary for producing sentence 6 from the acquisition process involving sentences 1-5. (i.e. $d_6 \in \mathbb{G}(\mathbb{Q}(Y_5))$

3.7 Which mouse will the man think that the cat will eat?

The learner next encounters a derivation that is a variant of d_6 that results from movement (*wh-fronting*) from the object position of the embedded clause subject to the specifier position while respecting the subjacency principle. (see Figure 3.15) The learner is presented with derivation d_7 (see Figure 67), which produces the sentence "which mouse will the man think that the cat will eat?" and is produced in turn by the lexicon L_7 in 3.19.

As before, the learner computes a lexicon L_7 such that $d_7 \in \mathbb{G}(L_7)$. (see Figure 3.19) The learner then computes their next state of knowledge: $Y_7 = \mathbb{P}(\mathbb{J}(\mathbb{Q}(Y_6), L_7))$. The *selection constraints* and *(anti-)licensing constraints* for $\mathbb{P}(L_7)$ are presented in Tables 3.23 and 3.24 respectively. The *selection constraints* and *(anti-)licensing constraints* for Y_7 are presented in Tables 3.25 and 3.26 respectively. Finally, the lexicon representation of the learner's new state of knowledge (i.e. $\mathbb{Q}(Y_7)$) is presented in Figure 3.20.

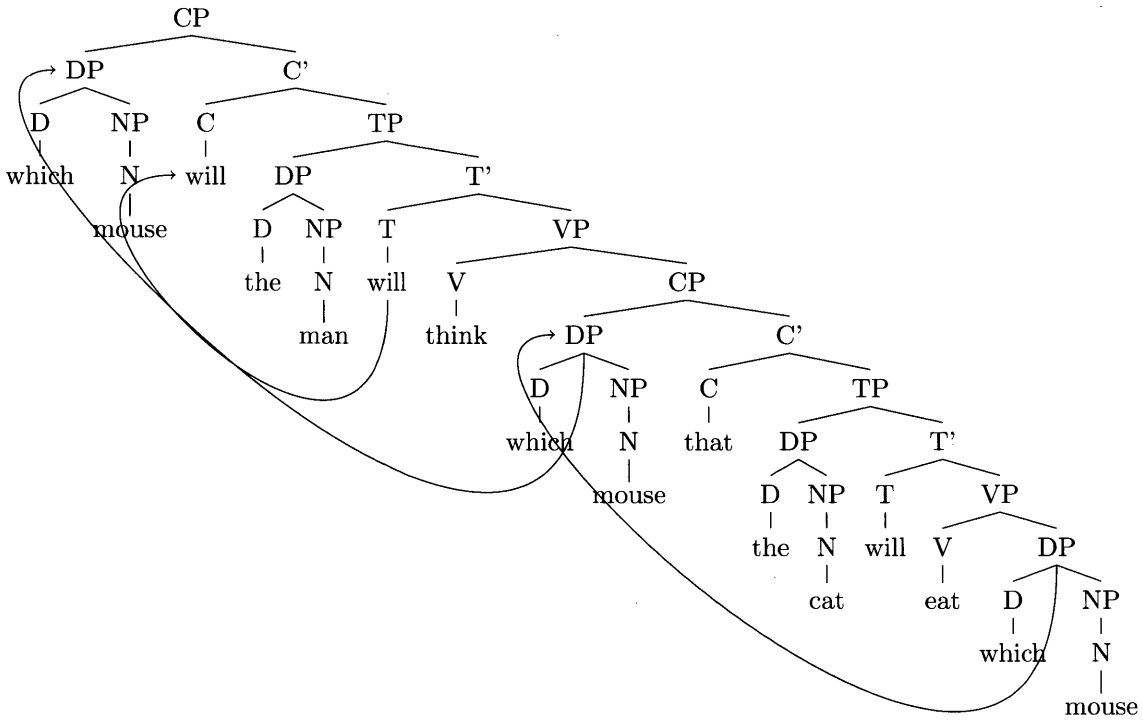

Figure 3.18: Phrase Structure for the sentence "which mouse will the man think that the cat will eat?"

	$cat :: \sim x6$
	$eat :: = x2, \sim x4$
	$man :: \sim x15$
	$mouse :: \sim x1$
$\epsilon_{C_3} :: = x14, +x16, +x18, C$	$that2 :: = x8, +x17, \sim x9$
$\epsilon_T :: = x10, = x11, +x12, = x13, \sim x14$	$the :: = x15, \sim x13$
$\epsilon_T :: = x3, = x4, +x5, = x7, \sim x8$	$the :: = x6, \sim x7$
	$think :: = x9, \sim x11$
	$which2 :: = x1, \sim x2, -x17, -x18$
	$will :: \sim x3, -x5$
	$will2 :: \sim x10, -x12, -x16$

Figure 3.19: The lexicon L_7 from which the derivation d_7 (which produces the sentence "will the man think that the cat will eat a mouse?") may be derived. Note that $\{d_7\} = \mathbb{G}(L_7)$. Furthermore, note that this lexicon is not *curated*; nevertheless, because the constraints between syntactic features are a function of the phonetic form and the index of the feature, the two copies of the "the" lexical item will be unified into a single copy by the procedure $\mathbb{J}$.

$$\frac{[\text{the} :: = x15, \sim x13] \qquad [\text{man} :: \sim x15]}{[\text{the man} : \sim x13]}\ \text{EM1}$$

$$\frac{[\text{the} :: = x6, \sim x7] \qquad [\text{cat} :: \sim x6]}{[\text{the cat} : \sim x7]}\ \text{EM1}$$

$$\frac{[\text{which2} :: = x1, \sim x2, -x17, -x18] \qquad [\text{mouse} :: \sim x1]}{[\text{which2 mouse} : \sim x2, -x17, -x18]}\ \text{EM1}$$

$$\frac{[\text{eat} :: = x2, \sim x4] \qquad [\text{which2 mouse} : \sim x2, -x17, -x18]}{[\text{eat} : \sim x4][\text{which2 mouse} : -x17, -x18]}\ \text{EM3}$$

$$\frac{[\epsilon_T :: = x3, = x4, +x5, = x7, \sim x8] \qquad [\text{will} :: \sim x3, -x5]}{[\epsilon_T : = x4, +x5, = x7, \sim x8][\text{will} : -x5]}\ \text{EM3}$$

$$\frac{[\epsilon_T : = x4, +x5, = x7, \sim x8][\text{will} : -x5] \qquad [\text{eat} : \sim x4][\text{which2 mouse} : -x17, -x18]}{[\text{eat } \epsilon_T : +x5, = x7, \sim x8][\text{which2 mouse} : -x17, -x18][\text{will} : -x5]}\ \text{EM2}$$

$$\frac{[\text{eat } \epsilon_T : +x5, = x7, \sim x8][\text{which2 mouse} : -x17, -x18][\text{will} : -x5]}{[\text{will eat } \epsilon_T : = x7, \sim x8][\text{which2 mouse} : -x17, -x18]}\ \text{IM1}$$

$$\frac{[\text{will eat } \epsilon_T : = x7, \sim x8][\text{which2 mouse} : -x17, -x18] \qquad [\text{the cat} : \sim x7]}{[\text{the cat will eat } \epsilon_T : \sim x8][\text{which2 mouse} : -x17, -x18]}\ \text{EM2}$$

$$\frac{[\text{that2} :: = x8,+x17,\sim x9] \qquad [\text{the cat will eat } \epsilon_T : \sim x8][\text{which2 mouse} : -x17,-x18]}{[\text{that2 the cat will eat } \epsilon_T : +x17,\sim x9][\text{which2 mouse} : -x17,-x18]}\ \text{EM1}$$

$$\frac{[\text{that2 the cat will eat } \epsilon_T : +x17,\sim x9][\text{which2 mouse} : -x17,-x18]}{[\text{that2 the cat will eat } \epsilon_T : \sim x9][\text{which2 mouse} : -x18]}\ \text{IM2}$$

$$\frac{[\text{think} :: = x9,\sim x11] \qquad [\text{that2 the cat will eat } \epsilon_T : \sim x9][\text{which2 mouse} : -x18]}{[\text{think that2 the cat will eat } \epsilon_T : \sim x11][\text{which2 mouse} : -x18]}\ \text{EM1}$$

$$\frac{[\epsilon_T :: = x10,= x11,+x12,= x13,\sim x14] \qquad [\text{will2} :: \sim x10,-x12,-x16]}{[\epsilon_T : = x11,+x12,= x13,\sim x14][\text{will2} : -x12,-x16]}\ \text{EM3}$$

$$\frac{[\epsilon_T : = x11,+x12,= x13,\sim x14][\text{will2} : -x12,-x16] \qquad [\text{think that2 the cat will eat } \epsilon_T : \sim x11][\text{which2 mouse} : -x18]}{[\text{think that2 the cat will eat } \epsilon_T\epsilon_T : +x12,= x13,\sim x14][\text{which2 mouse} : -x18][\text{will2} : -x12,-x16]}\ \text{EM2}$$

$$\frac{[\text{think that2 the cat will eat } \epsilon_T\epsilon_T : +x12,= x13,\sim x14][\text{which2 mouse} : -x18][\text{will2} : -x12,-x16]}{[\text{think that2 the cat will eat } \epsilon_T\epsilon_T : = x13,\sim x14][\text{will2} : -x16][\text{which2 mouse} : -x18]}\ \text{IM2}$$

$$\frac{[\text{think that2 the cat will eat } \epsilon_T\epsilon_T : = x13,\sim x14][\text{will2} : -x16][\text{which2 mouse} : -x18] \qquad [\text{the man} : \sim x13]}{[\text{the man think that2 the cat will eat } \epsilon_T\epsilon_T : \sim x14][\text{which2 mouse} : -x18][\text{will2} : -x16]}\ \text{EM2}$$

$$\frac{[\epsilon_{C_3} :: = x14,+x16,+x18,\text{C}] \qquad [\text{the man think that2 the cat will eat } \epsilon_T\epsilon_T : \sim x14][\text{which2 mouse} : -x18][\text{will2} : -x16]}{[\epsilon_{C_3} \text{ the man think that2 the cat will eat } \epsilon_T\epsilon_T : +x16,+x18,\text{C}][\text{which2 mouse} : -x18][\text{will2} : -x16]}\ \text{EM1}$$

$$\frac{[\epsilon_{C_3} \text{ the man think that2 the cat will eat } \epsilon_T\epsilon_T : +x16,+x18,\text{C}][\text{which2 mouse} : -x18][\text{will2} : -x16]}{[\text{will2 } \epsilon_{C_3} \text{ the man think that2 the cat will eat } \epsilon_T\epsilon_T : +x18,\text{C}][\text{which2 mouse} : -x18]}\ \text{IM1}$$

$$\frac{[\text{will2 } \epsilon_{C_3} \text{ the man think that2 the cat will eat } \epsilon_T\epsilon_T : +x18,\text{C}][\text{which2 mouse} : -x18]}{[\text{which2 mouse will2 } \epsilon_{C_3} \text{ the man think that2 the cat will eat } \epsilon_T\epsilon_T : \text{C}]}\ \text{IM1}$$

Selectors \ Selectees	(cat, 0)	(eat, 1)	(man, 0)	(mouse, 0)	(that2, 2)	(the, 1)	(think, 1)	(which2, 1)	(will, 0)	(will2, 0)	(ϵ_T, 4)
(eat, 0)	-	-	-	-	-	-	-	✓	-	-	-
(that2, 0)	-	-	-	-	-	-	-	-	-	-	✓
(the, 0)	✓	-	✓	-	-	-	-	-	-	-	-
(think, 0)	-	-	-	-	✓	-	-	-	-	-	-
(which2, 0)	-	-	-	✓	-	-	-	-	-	-	-
(ϵ_{C_3}, 0)	-	-	-	-	-	-	-	-	-	-	✓
(ϵ_T, 0)	-	-	-	-	-	-	-	-	✓	✓	-
(ϵ_T, 1)	-	✓	-	-	-	-	✓	-	-	-	-
(ϵ_T, 3)	-	-	-	-	-	✓	-	-	-	-	-

Table 3.23: The Selection Constraints of $\mathbb{P}(L_7)$.

Licensors \ Licensees	(which2, 2)	(which2, 3)	(will, 1)	(will2, 1)	(will2, 2)
(that2, 1)	✓	-	✗	✗	-
(ϵ_{C_3}, 1)	-	✗	-	-	✓
(ϵ_{C_3}, 2)	-	✓	✗	✗	✗
(ϵ_T, 2)	✗	✗	✓	✓	-

Table 3.24: The (Anti-)Licensing Constraints of $\mathbb{P}(L_7)$.

Selectors \ Selectees	(a, 1)	(cat, 0)	(eat, 1)	(man, 0)	(mouse, 0)	(that, 1)	(that2, 2)	(the, 1)	(think, 1)	(which, 1)	(which2, 1)	(will, 0)	(will2, 0)	(ϵ_T, 4)
(a, 0)	-	✓	-	✓	✓	-	-	-	-	-	-	-	-	-
(eat, 0)	✓	-	-	-	-	-	-	✓	-	✓	✓	-	-	-
(that, 0)	-	-	-	-	-	-	-	-	-	-	-	-	-	✓
(that2, 0)	-	-	-	-	-	-	-	-	-	-	-	-	-	✓
(the, 0)	-	✓	-	✓	✓	-	-	-	-	-	-	-	-	-
(think, 0)	-	-	-	-	-	✓	✓	-	-	-	-	-	-	-
(which, 0)	-	✓	-	✓	✓	-	-	-	-	-	-	-	-	-
(which2, 0)	-	✓	-	✓	✓	-	-	-	-	-	-	-	-	-
(ϵ_C, 0)	-	-	-	-	-	-	-	-	-	-	-	-	-	✓
(ϵ_{C_2}, 0)	-	-	-	-	-	-	-	-	-	-	-	-	-	✓
(ϵ_{C_3}, 0)	-	-	-	-	-	-	-	-	-	-	-	-	-	✓
(ϵ_T, 0)	-	-	-	-	-	-	-	-	-	-	-	✓	✓	-
(ϵ_T, 1)	-	-	✓	-	-	-	-	-	✓	-	-	-	-	-
(ϵ_T, 3)	✓	-	-	-	-	-	-	✓	-	✓	✓	-	-	-

Table 3.25: The Selection Constraints of Y_7. Note that the newly acquired constraints (designated by an underline) have to do with which noun phrases the lexical item $which_2$ may merge with.

Licensors \ Licensees	(which, 2)	(which2, 2)	(which2, 3)	(will, 1)	(will2, 1)	(will2, 2)
(that2, 1)	-	✓	-	✗	✗	-
(ϵ_{C_2}, 1)	✗	-	✗	-	-	✓
(ϵ_{C_3}, 1)	✗	-	✗	-	-	✓
(ϵ_{C_3}, 2)	✓	-	✓	✗	✗	✗
(ϵ_T, 2)	✗	✗	✗	✓	✓	-

Table 3.26: The (Anti-)Licensing Constraints of Y_7.

	$a :: = z1, \sim z2$
	$cat :: \sim z1$
	$eat :: = z2, \sim z6$
	$man :: \sim z1$
	$mouse :: \sim z1$
$\epsilon_C :: = z3, C$	$that :: = z3, \sim z4$
$\epsilon_{C_2} :: = z3, +z8, C$	$that2 :: = z3, +z7, \sim z4$
$\epsilon_{C_3} :: = z3, +z8, +z9, C$	$the :: = z1, \sim z2$
$\epsilon_T :: = z5, = z6, +z10, = z2, \sim z3$	$think :: = z4, \sim z6$
	$which :: = z1, \sim z2, -z9$
	$which2 :: = z1, \sim z2, -z7, -z9$
	$will :: \sim z5, -z10$
	$will2 :: \sim z5, -z10, -z8$

Figure 3.20: The lexicon acquired from derivations $d_1, \cdots, d_7$.

This concludes the process by which the learner acquires the lexicon $\mathbb{Q}(Y_7)$ from the derivations $d_1, \cdots, d_7$, such that $d_i \in \mathbb{Q}(Y_7)$ for $1 \leq i \leq 7$.

Chapter 4

Conclusion

4.1 Summary

This thesis has shown how to construct a *curated* MG lexicon that produces a desired set of MG derivations. In order to do this, it has developed a mathematical framework in which two or more *curated* lexicons may be (recursively) combined to produce a *curated* lexicon that generates a superset of the *complete* derivations generated by each of the constituent lexicons. In the process, it defined a mathematical structure, the *Collection of Constraints*, that captures the logical constraints imposed upon the syntactic features of lexical items as they are *merged* together in a *derivation* produced by a given *lexicon*. This thesis then further developed the relation between *curated lexicons* and *Collections of Constraints* by establishing two procedures: the procedure $\mathbb{P}$, which maps a *curated* lexicon to a *collections of constraints* and the procedure $\mathbb{Q}$, which maps a *collections of constraints* produced by $\mathbb{P}$ to a *curated* lexicon. This thesis then showed that these two procedures provide a bijective correspondence between the set of *classes of curated lexicons* and the set of *collections of constraints*.

I then observed that combining multiple *curated lexicons* together requires that no *lexicons' anti-licensing constraints* may violate any *lexicons' licensing constraints*. **Since a *Collections of Constraints* structure explicitly enumerates both the set of *licensing constraints* and the set of *anti-licensing constraints* that hold in the *complete derivations* produced by a given *lexicon*, this mathematical structure is well suited for efficiently detecting violations of *licensing constraints* by *anti-licensing constraints* that could arise when trying to combine several lexicons together.** This thesis capitalized upon this insight by establishing the procedure $\mathbb{J}$, which determines whether it is valid to combine several specified *Collections of Constraints* into a single *Collection of Constraints* and if it is, proceeds to compute the aggregate *Collection of Constraints*. Thus, given a set of MG derivations, each derivation may be mapped via the procedure $\mathbb{P}$ to a *Collection of Constraints* encoding the constraints imposed by that derivation. The procedures $\mathbb{J}$ and $\mathbb{Q}$ may then be applied to the resulting *Collection of Constraints* to produce a *curated lexicon* that generates a superset of the original MG derivations it was constructed from.

This thesis then provided a running example demonstrating how the procedures $\mathbb{J}$, $\mathbb{P}$ and $\mathbb{Q}$ may be

used together to model how a "learner" may acquire a sequence of derivations.[1] The "learner" begins with no prior knowledge, in the form of an empty *Collection of Constraints*. The "learner" is then sequentially exposed to a series of MG derivations, acquiring new knowledge in the form of *constraints*. Each step in the acquisition process involves enumerating the "learner's" total state of knowledge as well as the *new* knowledge that the "learner" acquired in the previous acquisition step. In the process, this thesis was able to investigate which kinds of generalizations that the "learner" could and could not capture. In particular, this thesis demonstrated how the "learner" is able to infer *wh-movement* from the subject position in an interrogative given previous exposure to *wh-movement* from the object position in an interrogative.

4.2 Future Work

The ability to algorithmically construct a lexicon that generates a desired set of derivations greatly eases the process of exploring the space of grammars that covers a given corpora of phrase structures. The methods developed in Chapter 2 and the exploration presented in Chapter ?? are restricted to *curated* lexicons. A consequence of this, each lexical item in a set of related lexical items must be assigned a unique phonetic component. (e.g. ϵ_C, ϵ_{C_2} and ϵ_{C_3}). In this section I will motivate an extension of the MG formalism in which lexical items will structure their syntactic features as non-deterministic finite state automata (NDA), in effect generalizing the standard linear ordering. Specifically, the states of a lexical item's syntactic feature NDA represent the currently exposed syntactic feature while the edges of the NDA correspond to the syntactic feature that will be *selected* or *licensed* (See Figures 4.2, 4.3, 4.4 and 4.5 for examples of such NDAs). I will present three kinds of patterns observed in the lexicon that the learner ends with in Chapter 3 (See Figure 3.20) that may all be captured under the proposed extension to organize the syntactic features of a lexical item as a non-deterministic finite state automata.

Figure 4.1 presents the final state of the learner's knowledge in lexicon form; note the following three lexical items that are variations of ϵ_C:

$$\epsilon_C :: = z3, C$$
$$\epsilon_{C_2} :: = z3, +z8, C$$
$$\epsilon_{C_3} :: = z3, +z8, +z9, C$$

These three lexical items are all capturing aspects of the CP part of speech. They are related by the syntactic features they share in common; specifically, one can view them as a set of decision sequences characterizing decision tree. Generalizing the linear sequence of syntactic features of a chain into an NDA would allow the learner to encode such a decision tree into the structure of the syntatic features of a single lexical item, $\epsilon_C^{'}$ (See Figure 4.2). As a consequence of this, the features $(\epsilon_C, 0)$, $(\epsilon_{C2}, 0)$ and $(\epsilon_{C3}, 0)$ (which

[1] In this thesis I have taken the liberty of supplying derivations that map to derived trees that have a linguistically plausible interpretation. In general, it is an open question as to how the "learner" is supplied with derivations. Note that if the procedures $\mathbb{J}$ is implemented efficiently and there is some procedure $\mathbb{W}$ provided that enumerates all possible derivations, then if one of the arguments of $\mathbb{J}$ is fixed with the current state of the learner's knowledge, $\mathbb{J}$ may effectively be used as a filter to search for other *possible* derivations to assign to a newly seen sentence that is consistent with the current state of the learner's knowledge.

$\epsilon_C :: = z3, C$
$\epsilon_{C_2} :: = z3, +z8, C$
$\epsilon_{C_3} :: = z3, +z8, +z9, C$
$\epsilon_T :: = z5, = z6, +z10, = z2, \sim z3$

$a :: = z1, \sim z2$
$cat :: \sim z1$
$eat :: = z2, \sim z6$
$man :: \sim z1$
$mouse :: \sim z1$
$that :: = z3, \sim z4$
$that2 :: = z3, +z7, \sim z4$
$the :: = z1, \sim z2$
$think :: = z4, \sim z6$
$which :: = z1, \sim z2, -z9$
$which2 :: = z1, \sim z2, -z7, -z9$
$will :: \sim z5, -z10$
$will2 :: \sim z5, -z10, -z8$

Figure 4.1: The lexicon acquired from derivations the seven derivations presented in Chapter 3. Notice the following groups of lexical items that have related phonetic forms as well as syntactic features: (a) ϵ_C, ϵ_{C_2} and ϵ_{C_3}, (b) "that" and "that2," and (c) "which" and "which2." The proposed extension would allow for each of the above groups of lexical items to be represented by a single lexical item.

all share the representation $= z_3$) would collapse into a single syntactic feature $(\epsilon_C^{'}, 0)$.[2] Similarly, the two features $(\epsilon_C, 1)$ and $(\epsilon_{C2}, 1)$, which share the representation $= z_3$, would collapse into the single syntactic feature $(\epsilon_C^{'}, 1)$. [3]

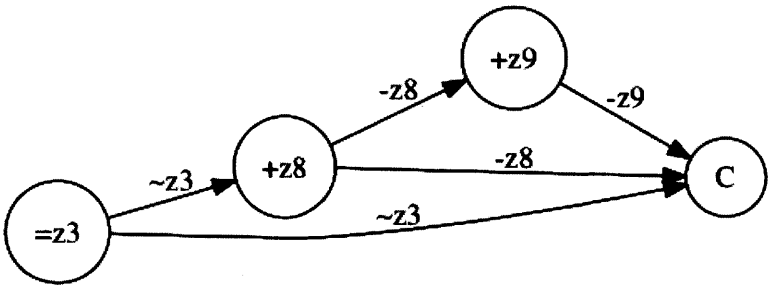

Figure 4.2: A finite state machine encoding the feature system of ϵ_C, ϵ_{C_2} and ϵ_{C_3} into a single non-deterministic finite state automata for the lexical item $\epsilon_C^{'}$. Each node corresponds to a syntactic feature of $\epsilon_C^{'}$ and each edge corresponds to a syntactic feature that may check or be checked by $\epsilon_C^{'}$.

Next, consider the following two related lexical items.

$$that :: = z3, \sim z4$$
$$that2 :: = z3, +z7, \sim z4$$

These two lexical items are also capturing aspects of the CP part of speech. The difference between

[2] A consequence of these features collapsing into a single feature is that if weights were assigned to the features in a lexical item (i.e. in a probabilistic model), the weights for those three features must have the same value.

[3] Note that the indexing system used for syntactic features throughout this thesis (e.g. $(\epsilon_C, 0)$) is intended for linear sequences of syntactic features; generalizing to the case of a finite state transducer would require that a more elaborate indexing system be utilized.

"that" and "that2" is the *licensor* $+z7$ in "that2" to which "which2" may bind. [4] We would like for $(that, 1)$ and $(that2, 2)$ to be the same feature since they share the same representation, $\sim z_4$. One manner of achieving that is to consider the syntactic feature $-z7 = (that2, 1)$ to be an "optional" feature. Modeling the syntactic features of a lexical item as an NDA as proposed above would allow for such "optional" features (See Figure ??).

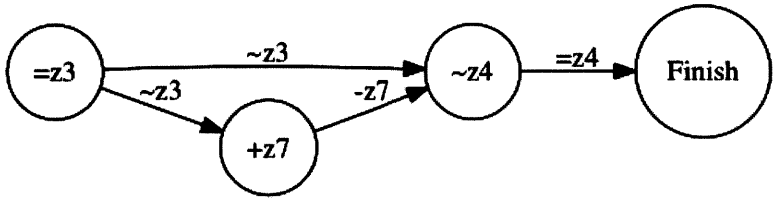

Figure 4.3: A finite state machine encoding the feature system of "that" and "that2" into a single non-deterministic finite state automata. Note that the syntactic feature $-z_7$ is *optional* in this context.

Finally, consider the following two lexical items:

$$which :: = z1, \sim z2, -z9$$
$$which2 :: = z1, \sim z2, -z7, -z9$$

These two lexical items are capturing aspects of the DP part of speech. The difference between "which" and "which2" is the *licensee* $-z7$ in "which2" that will bind with the feature $+z7$ in "that2." Once again, we would like for $(that, 1)$ and $(that2, 2)$ to be the same feature since they share the same representation, $\sim z_4$ (See Figure 4.4).

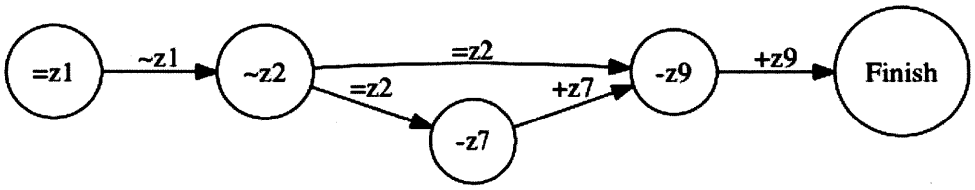

Figure 4.4: A finite state machine encoding the feature system of "which" and "which2" into a single non-deterministic finite state automata.

Although we could handle this case similar to the case of "that" above, consider what the lexical items for "which" would look like for further successive cyclic wh-movement:

$$which3 :: = z1, \sim z2, -z7, -z7, -z9$$
$$which4 :: = z1, \sim z2, -z7, -z7, -z7, -z9$$
$$\ldots$$
$$which_n :: = z1, \sim z2, (-z7)^{n-1}, -z9$$

[4] The licensor $+z7$ serves as a landing site for *wh-movement* so as not to violate the subjacency principle, which is defined in [2] as: "A cyclic rule cannot move a phrase from position Y to position X (or conversely) in ... X ... [α ... [β ... Y ...] ...] ... X ..., where α and β are cyclic nodes. Cyclic nodes are S and NP."

This suggests that the feature $-z7$ should be a "cyclic" feature that allows for unlimited licensing. Once again, modeling the syntactic features of a lexical item as a finite state transducer as proposed above would allow for such "cyclic" features (See Figure 4.5).

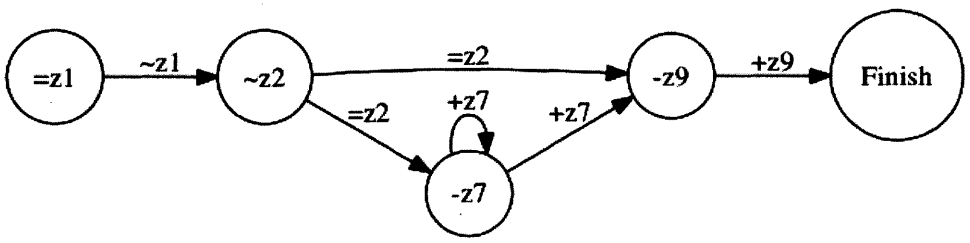

Figure 4.5: A finite state machine extending the *wh-movement* captured by "which" and "which2" to *cyclically successive wh-movement.*

Thus far we have seen how NDA's may capture commonalities in the network topology of the syntactic feature structures of a group of related lexical items. The addition of weights to the edges of the NDA encoding the syntactic features of a lexical item would allow for the development of probabilistic models for Minimalist Grammars based upon rational kernels for support vector machines. [4] Furthermore, the topology of the NDA encoding the syntactic features of a lexical item may be viewed as prior knowledge; thus, it may be possible to identify a syntactic feature NDA topology for each of the syntactic categories (i.e. parts of speech). Then each particular instance of a lexical item will have a unique assignment of *weights* to it's syntactic feature NDA. In contrast to methods developed in machine learning that propose to automatically discover key features and properties when no prior knowledge of the acquisition task is at hand, rational kernels may allow us to explicitly capture our prior knowledge about universal grammar in the structure of the NDA. [1] The extensions to the MG formalism presented in this section may provide a path by which the methods developed in this thesis may be adapted to probabilistic models of Minimalist Grammars.

Bibliography

[1] Bengio, Y. (2009). Learning deep architectures for AI. Foundations and trends® in Machine Learning, 2(1), 1-127.

[2] Chomsky, N. (1973). "Conditions on Transformations", in Anderson and Kiparsky, A Festschrift for Morris Halle, New York: Holt, Rinehart & Winston, pp. 232–286.

[3] Chomsky, N. (1995). The minimalist program (Vol. 28). Cambridge, MA: MIT press.

[4] Cortes, C., Haffner, P., & Mohri, M. (2004). Rational kernels: Theory and algorithms. The Journal of Machine Learning Research, 5, 1035-1062.

[5] Fowlie, M. (2014). Adjuncts and minimalist grammars. In Formal Grammar (pp. 34-51). Springer Berlin Heidelberg.

[6] Graf, T. (2013). Local and transderivational constraints in syntax and semantics (Doctoral dissertation, UCLA).

[7] Graf, T. (2014). Models of Adjunction in Minimalist Grammars. In Formal Grammar (pp. 52-68). Springer Berlin Heidelberg.

[8] John Hale. 2006. Uncertainty about the rest of the sentence. Cognitive Science, 30:643–672.

[9] Harkema, H. (2001). A characterization of minimalist languages. In Logical aspects of computational linguistics (pp. 193-211). Springer Berlin Heidelberg.

[10] Hunter, T., & Dyer, C. (2013). Distributions on Minimalist Grammar Derivations. In The 13th Meeting on the Mathematics of Language (p. 1).

[11] Aravind Joshi. (1985). How much context-sensitivity is necessary for characterizing structural descriptions. In D. Dowty, L. Karttunen, and A. Zwicky, ed., Natural Language Processing: Theoretical, Computational and Psychological Perspectives, pgs 206–250. Cambridge University Press, NY.

[12] Kobele, G. (2006). Generating copies (Doctoral dissertation, PhD thesis, UCLA, Los Angeles).

[13] Mainguy, T. (2010). A probabilistic top-down parser for minimalist grammars. arXiv preprint arXiv:1010.1826.

[14] Michaelis, J. (2001). Derivational minimalism is mildly context–sensitive. In Logical aspects of computational linguistics (pp. 179-198). Springer Berlin Heidelberg.

[15] Stabler, E. P. (1997). Derivational minimalism. In Logical aspects of computational linguistics (pp. 68-95). Springer Berlin Heidelberg.

[16] Stabler, E. P. (2001). Recognizing head movement. In Logical aspects of computational linguistics (pp. 245-260). Springer Berlin Heidelberg.

[17] Stabler, E. P., & Keenan, E. L. (2003). Structural similarity within and among languages. Theoretical Computer Science 293(2), 345-363.

[18] Stabler, E. P. (2011). Top-down recognizers for MCFGs and MGs. In Proceedings of the 2nd workshop on cognitive modeling and computational linguistics (pp. 39-48). Association for Computational Linguistics.

[19] Stabler, E. P. (2013). Two models of minimalist, incremental syntactic analysis. Topics in cognitive science, 5(3), 611-633.