# Solving for Syntax

by

Sagar Indurkhya

B.Sc., Massachusetts Institute of Technology (2012)

M.Eng., Massachusetts Institute of Technology (2015)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2021

Author . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Department of Electrical Engineering and Computer Science
November 9, 2020

Certified by . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Robert C. Berwick
Professor of Computational Linguistics and Computer Science and Engineering
Thesis Supervisor

Accepted by . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

**Solving for Syntax**
by
Sagar Indurkhya

Submitted to the Department of Electrical Engineering and Computer Science
on November 9, 2020, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

## Abstract

Among the key questions that have guided research into the nature of human language for the past sixty years, two have been particularly salient: (1) What constitutes knowledge of language? and (2) How is that knowledge acquired? In particular, children using limited input examples, in effect small "sample size" complexity, all acquire their native language.

This thesis attempts to answer these two questions by developing a novel, explicit, computational implementation of one contemporary approach to human language known as the Minimalist Program. It provides an answer to question (1) via the explicit axiomatization of a declaratively specified logical model of minimalist grammars, consisting of a set of formally specified principles and a single structure-building operation, along with a lexicon. By rendering these axioms along with the lexicon as a set of constraints that are expressed using Satisfiability Modulo Theories (SMT), that must be simultaneously satisfied, the thesis demonstrates how to *"solve for syntax"*: it uses an SMT-solver to computationally deduce the syntactic derivations that associate particular input sentences with their logical forms. In this sense, the thesis demonstrates that the proposed linguistic principles underlying such a system, including the contemporary notion of "economy conditions" in syntax are both coherent and consistent, and, importantly, that minimalist syntax can be placed within a classical "parsing as deduction" framework.

This thesis then extends the system developed to address question (1) to provide an answer to question (2), by modeling acquisition as the construction of a succession of lexicons, starting from some initial, essentially empty lexicon, and then augmenting that lexicon. To do this, it uses a set of (input sentence, skeletal "meaning") pairs intended to reflect minimally cognitively faithful constraints to infer what lexicon would bridge from input to quasi-meaning forms, again using an SMT-solver. Using this approach, the thesis explicitly demonstrates that a wide variety of syntactic sentence constructions in English can be acquired in this way, sufficient to account for the infinite generative capacity of at least one portion of English syntax. Importantly then, the thesis thus demonstrates that a contemporary minimalist syntactic system, with a single, fixed structure-building operation leaving only the lexicon to vary, can serve as the foundation for a contemporary approach to language acquisition. In this sense, the implementation serves as a concrete, computationally realized contemporary instantiation of the model of language acquisition set out in *Aspects of the Theory of Syntax*.

Thesis Supervisor: Robert C. Berwick
Title: Professor of Computational Linguistics and Computer Science and Engineering

This doctoral thesis has been examined by a Committee of the Department of Electrical Engineering and Computer Science as follows:

Professor Robert C. Berwick . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .<br>
Chairman, Thesis Committee & Thesis Supervisor<br>
Professor of Computational Linguistics and Computer Science and Engineering

Professor Sanjoy K. Mitter . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .<br>
Member, Thesis Committee<br>
Professor of Electrical Engineering

Professor Sandiway Fong . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .<br>
Member, Thesis Committee<br>
Associate Professor of Linguistics

*This thesis is dedicated to my wife, Ramita.*

## Acknowledgments

This thesis would not have been possible without the support and guidance of my teachers, colleagues, family, and friends.

To begin, I would like to express my gratitude to my wife, Ramita. This thesis would not have happened without her love, patience, and encouragement.

I want to thank my parents, Gopal and Vandana Indurkhya, for teaching me right from wrong and instructing me on the importance of doing good work. I would also like to thank my brother, Aakash Indurkhya, who continually reminded me to have fun along the way.

My friends have supported and encouraged me throughout my graduate career. I want to thank Apoorva Murarka for thirteen years of camaraderie and the many long walks on weekends to Harvard Square when we talked about everything. I want to thank my labmate, Beracah Yankama, for the many years of discussing scientific puzzles together over lunch. I want to thank my friend Parama Pal for the fun conversations we had while studying together. I would also like to give a special thank you to Nicholas Tang, Austen J. Heinz, Nirav Lakhani, and Hattie Chung for encouraging me to pursue a career in research.

I have been lucky to participate in and learn from a wonderful community of scientists. I would like to thank Barbara Lust, Norbert Hornstein, Charles Yang, Patrick Winston, Aline Villavicencio, Marco Idiart, Robert Ajemian, Hector Vazquez, Annika Heuser, Gabriel Teixeira, Basil Saeed, and Run Chen.

Throughout my graduate career, I have been guided and supported by my mentors. I am indebted to Myra Halpin, Marvin Minsky, Thomas F. Knight, Gerald J. Sussman, Anselm Levskaya, and Vivienne Sze.

I would also like to express my gratitude to my thesis committee members, Sandiway Fong and Sanjoy Mitter, for their help in guiding and reviewing this thesis and for asking many illuminating questions along the way. I am grateful to Sandiway for the many exciting conversations in which he patiently explained to me the intricacies of syntax.

Finally, I would like to acknowledge and thank my mentor, teacher, and friend, Robert C. Berwick, for inspiring me and guiding me throughout my graduate career. His influence and impact on this thesis cannot be overstated. He has taught me what it means to be a scholar and a scientist.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Human beings are unique among other species in part due to their innate endowment of a *faculty of language*, referred to as the Human Language Faculty (HLF), that provides them with the capacity to acquire and exercise *knowledge of language* (Berwick and Chomsky, 2016). (See Fig. 1-1 for a high level system diagram that illustrates this abstract view of the computational system underlying the human language faculty, $C_{HLF}$.) *Knowledge of language*, for a particular language, consists of a generative grammar (i.e. a linguistic theory) that specifies how a countably infinite set of interpretable hierarchical structures,[1] each pairing meaning with sound,[2] may be derived via the recursive combination of terms drawn from a finite lexicon of words, such that related interpretations are assigned structures that are correspondingly systematically related via structural transformations (Chomsky, 1986). Notably, every child has the capacity to acquire knowledge of any natural language if exposed to the appropriate *Primary Linguistic Data* (PLD) during the first several years of their life (Everaert et al., 2015). Scientific investigations of human language that attempt to provide linguistic theories that explain these observations have been driven by two fundamental questions: (1) What constitutes knowledge of language? (2) How is do children acquire knowledge of language, particularly in light of *arguments from the Poverty of the Stimulus*?[3] This thesis provides answer to these two questions by leveraging advances in state-of-the-art, high-performance automatic theorem provers to develop within the framework of the Minimalist Program novel procedures for language parsing and language acquisition that are grounded in the "parsing as deduction" and the "logic as grammar" frameworks respectively.[4][5][6]

The thesis answers the first question, what constitutes knowledge of language, by developing an axiomatization of minimalist syntax that is expressed using a logic, Satisfiability

[1] Each natural language expression must be assigned zero or more (syntactic) structures, each of which is uniquely associated with a distinct interpretations of the expression.

[2] Classically, natural languages are said to pair together the domains of sounds and meaning; modern linguistics generalizes these domains to the notion of interfaces, with sound mapping to the *Sensory-Motor interface* and meaning mapping to what we may call a *Conceptual-Intensional interface.*

[3] See (Berwick et al., 2011) for a review of arguments from the Poverty of the Stimulus.

[4] The Minimalist Program (Chomsky, 1990; Chomsky, 1995) is the leading framework in contemporary linguistics for studying the faculty of language and is a reformulation of an earlier linguistic framework from the 1980's (the theory of Principles and Parameters) that seeks to assess the degree to which the design of the faculty of language is optimal with respect to the specifications of the interface requirements of a natural language.

[5] See (Pereira and Warren, 1983) for an introduction to the "parsing as deduction" framework.

[6] See (Rayner et al., 1988) for an introduction to the "logic as grammar" framework.

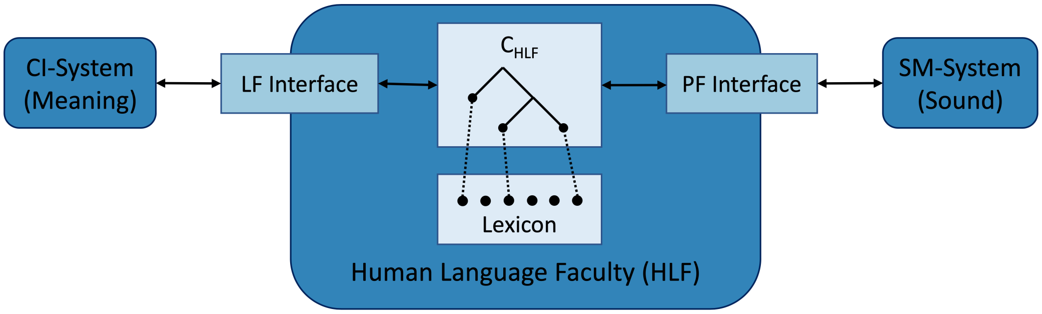


Figure 1-1: Minimalist theories of syntax assert that the Human Language Faculty (HLF) pairs meaning, which is processed by the Conceptual-Intensional (CI) System, with sound, which is processed by the Sensory-Motor (SM) System. Within the Human Language Faculty (HLF), the Computational System ($C_{HLF}$) derives a syntactic structure the syntactic atoms in the Lexicon via the repeated application of the recursive binary function *merge*; after each step in the derivation of the syntactic structure (i.e. application of *merge*), relevant information is sent to the LF Interface via *Transfer*, and to the PF Interface via *Spellout*.

Modulo Theories (SMT), and then introducing a procedure for parsing Minimalist Grammars (MGs) that has been implemented as a working computer program.[7] The procedure takes as input a (partial) specification of LF and PF interface conditions and a specification of an MG lexicon. The procedure outputs a derivation that can be yielded by the specified lexicon and that satisfies the specified interface conditions. The procedure for parsing operates by constructing an SMT-model of a parser, based on this axiomatization of minimalist syntax, that is further constrained by its inputs (i.e. the interface conditions and the lexicon). It then uses an SMT-solver to obtain a satisfiable interpretation (of this SMT-model) from which an MG derivation can be (automatically) recovered. In this way, the thesis demonstrates that the proposed linguistic principles underlying such a system (i.e. an axiomatization of minimalist syntax) can be placed within a classical “parsing as deduction” framework in which the specified interface conditions and lexicon constitute “knowns” and the derivation (i.e. the output of the parser) constitutes the “unknown” that is being solved for. A detailed, technical presentation of this procedure for parsing is provided in Chapter 2.

The thesis answers the second question, how knowledge of language is acquired, by developing a procedure for acquiring minimalist grammars, implemented as a working computer program, that takes the form of a computational model of a child language learner and accords with the criterion for models of language acquisition set out in (Chomsky, 1965). The procedure takes as input a (possibly empty) initial lexicon and a finite sequence of pairings of LF and PF interface conditions that stands in for the PLD that a child language learner is exposed to. The procedure outputs a lexicon that is compatible with the given PLD in that the output lexicon can yield, for each pair of LF and PF interface conditions in the input sequence, a derivation that satisfies those interface conditions. (That is, the procedure augments the input lexicon so that it can be used by the procedure for parsing to successfully parse each pair of LF and PF interface conditions in the input sequence.) This procedure operates by constructing an SMT-model of a (minimalist) grammar that is an extension of the SMT-model of the parser and that is constrained by the input – i.e. the initial lexicon and the sequence of interface conditions – and uses an SMT-solver to obtain

[7] See (Stabler, 1996) for an introduction to the MG formalism; see §2.2 for a review of this formalism.

a satisfiable interpretation of this SMT-model from which the smallest possible MG lexicon (that is compatible with the input) can be (automatically) recovered an MG lexicon; in this way, the procedure falls within the "logic as grammar" framework, with the input (initial) lexicon and sequence of interface conditions constituting "knowns" and the output lexicon (a superset of the initial lexicon) and derivations it yields (that satisfy the input sequence of interface conditions) constituting the "unknowns." The thesis then presents computational experiments that demonstrate the capacity of the model to acquire knowledge of syntax from psychologically plausible inputs, and that a contemporary minimalist syntactic system, with a single, fixed structure-building operation leaving only the lexicon to vary, can serve as the foundation for a contemporary approach to language acquisition – in this sense, the implementation serves as a concrete, computationally realized contemporary instantiation of the model of language acquisition set out in (Chomsky, 1965). A detailed, technical presentation of this procedure for acquisition is given in Chapter 3.

To summarize, in both the case of parsing and the case of acquisition, this thesis provides an implemented python procedure that does the following. First, the procedure constructs an SMT-model – i.e. a conjunction of logical formulae expressed using a first-order quantifier-free multi-sort logic extended with the theory of uninterpreted functions with equality – and constrains the model using the inputs to the procedure. Second, the procedure uses the Z3 SMT-solver (De Moura and Bjørner, 2008) to identify a solution to the system of equations (i.e. a satisfiable interpretation of the SMT-model) from which the desired output may be (automatically) recovered. It is in this sense that, colloquially speaking, the procedures for parsing and acquisition introduced in this thesis may be said to be *"solving for syntax."*

The remainder of this chapter is organized as follows: first, §1.1 previews the procedure for parsing that will be presented in Chapter 2; then, §1.2 previews the procedures for acquisition that will be presented in Chapter 3; finally, §1.3 summarizes the key ideas underlying this thesis as well as the lessons learned.

## 1.1 Parsing

This thesis develops a minimalist parser, developed and implemented as a working computer program, that aims to model the processing and comprehension of a natural language expression by the language faculty of a competent adult. The parser is a competence model – that is, it puts aside considerations of computational performance.

The parser takes as input a pairing of LF and PF interface conditions – i.e. simply put, a sentence annotated with interface conditions (encoding word order, predicate-argument relations, and agreement relations) – and an MG lexicon that consists of a set of lexical entries, each of which pairs a phonological form with a finite sequence of syntactic features (i.e. a feature matrix). The parser outputs a derivation that may be yielded by the input lexicon and that satisfies the supplied LF and PF interface conditions. Notably, the parser can yield derivations that include empty categories, syntactic movement (of phrases) and head-movement. To illustrate the problem that the procedure must solve, consider using the procedure to parse the interface conditions listed in entry $I_{26}$ in Table 1.1 using the lexicon presented in Table 1.2. The parser must yield a derivation – i.e. a syntactic structure – that can be assembled from the lexical entries listed in the lexicon, and that: (i) establishes the local syntactic relations required by LF interface conditions encoding predicate-argument structure and agreement relations, and (ii) derives the externalized form of the sentence (*"John has told Mary a story."*) listed under the PF interface condition in accordance with

| ID | PF Interface Conditions | LF Interface Conditions |
|---|---|---|
| $I_0$ | who has eaten/V icecream/N? | $\theta_{\text{eaten}}[s\colon \text{who}, o\colon \text{icecream}]$, $Agr_{\text{has}}[s\colon \text{who}]$ |
| $I_1$ | icecream/N was eaten/V. | $\theta_{\text{eaten}}[o\colon \text{icecream}]$, $Agr_{\text{was}}[s\colon \text{icecream}]$ |
| $I_2$ | who was eating/V icecream/N? | $\theta_{\text{eating}}[s\colon \text{who}, o\colon \text{icecream}]$, $Agr_{\text{was}}[s\colon \text{who}]$ |
| $I_3$ | was pizza/N eaten/V? | $\theta_{\text{eaten}}[o\colon \text{pizza}]$, $Agr_{\text{was}}[s\colon \text{pizza}]$ |
| $I_4$ | what has john/N eaten/V? | $\theta_{\text{eaten}}[s\colon \text{john}, o\colon \text{what}]$, $Agr_{\text{has}}[s\colon \text{john}]$ |
| $I_5$ | has mary/N eaten/V pizza/N? | $\theta_{\text{eaten}}[s\colon \text{mary}, o\colon \text{pizza}]$, $Agr_{\text{has}}[s\colon \text{mary}]$ |
| $I_6$ | was john/N eating/V pizza/N? | $\theta_{\text{eating}}[s\colon \text{john}, o\colon \text{pizza}]$, $Agr_{\text{was}}[s\colon \text{john}]$ |
| $I_7$ | what was mary/N eating/V? | $\theta_{\text{eating}}[s\colon \text{mary}, o\colon \text{what}]$, $Agr_{\text{was}}[s\colon \text{mary}]$ |
| $I_8$ | what was eaten/V? | $\theta_{\text{eaten}}[o\colon \text{what}]$, $Agr_{\text{was}}[s\colon \text{what}]$ |
| $I_9$ | was mary/N given/V pizza/N? | $\theta_{\text{given}}[o\colon \text{pizza}, i\colon \text{mary}]$, $Agr_{\text{was}}[s\colon \text{mary}]$ |
| $I_{10}$ | what has mary/N given/V john/N? | $\theta_{\text{given}}[s\colon \text{mary}, o\colon \text{what}, i\colon \text{john}]$, $Agr_{\text{has}}[s\colon \text{mary}]$ |
| $I_{11}$ | mary/N has given/V john/N money/N. | $\theta_{\text{given}}[s\colon \text{mary}, o\colon \text{money}, i\colon \text{john}]$, $Agr_{\text{has}}[s\colon \text{mary}]$ |
| $I_{12}$ | who was money/N given/V to/P? | $\theta_{\text{given}}[o\colon \text{money}, i\colon \text{to who}]$, $Agr_{\text{was}}[s\colon \text{money}]$ |
| $I_{13}$ | who has john/N given/V money/N to/P? | $\theta_{\text{given}}[s\colon \text{john}, o\colon \text{money}, i\colon \text{to who}]$, $Agr_{\text{has}}[s\colon \text{john}]$ |
| $I_{14}$ | was the boy/N sleeping/V? | $\theta_{\text{sleeping}}[s\colon \text{the boy}]$, $Agr_{\text{was}}[s\colon \text{the boy}]$ |
| $I_{15}$ | the boy/N has slept/V. | $\theta_{\text{slept}}[s\colon \text{the boy}]$, $Agr_{\text{has}}[s\colon \text{the boy}]$ |
| $I_{16}$ | john/N was told/V nothing/N. | $\theta_{\text{told}}[o\colon \text{nothing}, i\colon \text{john}]$, $Agr_{\text{was}}[s\colon \text{john}]$ |
| $I_{17}$ | someone/N has known/V everything/N. | $\theta_{\text{known}}[s\colon \text{someone}, o\colon \text{everything}]$, $Agr_{\text{has}}[s\colon \text{someone}]$ |
| $I_{18}$ | who was asking/V nothing/N? | $\theta_{\text{asking}}[s\colon \text{who}, o\colon \text{nothing}]$, $Agr_{\text{was}}[s\colon \text{who}]$ |
| $I_{19}$ | nothing/N was asked/V. | $\theta_{\text{asked}}[o\colon \text{nothing}]$, $Agr_{\text{was}}[s\colon \text{nothing}]$ |
| $I_{20}$ | everything/N was known/V. | $\theta_{\text{known}}[o\colon \text{everything}]$, $Agr_{\text{was}}[s\colon \text{everything}]$ |
| $I_{21}$ | who was everything/N told/V to? | $\theta_{\text{told}}[o\colon \text{everything}, i\colon \text{to who}]$, $Agr_{\text{was}}[s\colon \text{everything}]$ |
| $I_{22}$ | john/N has asked/V someone/N everything/N. | $\theta_{\text{asked}}[s\colon \text{john}, o\colon \text{everything}, i\colon \text{someone}]$, $Agr_{\text{has}}[s\colon \text{john}]$ |
| $I_{23}$ | what was someone/N asked/V? | $\theta_{\text{asked}}[o\colon \text{what}, i\colon \text{someone}]$, $Agr_{\text{was}}[s\colon \text{someone}]$ |
| $I_{24}$ | who has told/V someone/N the story/N? | $\theta_{\text{told}}[s\colon \text{who}, o\colon \text{the story}, i\colon \text{someone}]$, $Agr_{\text{has}}[s\colon \text{who}]$ |
| $I_{25}$ | a boy/N was eating/V the pizza/N. | $\theta_{\text{eating}}[s\colon \text{a boy}, o\colon \text{the pizza}]$, $Agr_{\text{was}}[s\colon \text{a boy}]$ |
| $I_{26}$ | john/N has told/V mary/N a story/N. | $\theta_{\text{told}}[s\colon \text{john}, o\colon \text{a story}, i\colon \text{mary}]$, $Agr_{\text{has}}[s\colon \text{john}]$ |
| $I_{27}$ | the story/N was told/V to a boy/N. | $\theta_{\text{told}}[o\colon \text{the story}, i\colon \text{to a boy}]$, $Agr_{\text{was}}[s\colon \text{the story}]$ |
| $I_{28}$ | what was john/N asking/V? | $\theta_{\text{asking}}[s\colon \text{john}, o\colon \text{what}]$, $Agr_{\text{was}}[s\colon \text{john}]$ |

Table 1.1: Primary Linguistic Data (PLD). The learner is presented with a sequence of pairs of PF and LF interface conditions – i.e. a sequence of sentences annotated with syntactic relations. The acquisition procedure takes the PLD as an input. The PF interface conditions consist of a tokenized sentence, with some tokens having their category prespecified (indicated by a suffix of a slash followed by the category). The LF interface conditions consists of: (i) locality constraints that include agreement (*Agr*) and predicate-argument structure (i.e. a $\theta$ grid), with the predicate indicated in the suffix and the subject, object and indirect object components marked by "s:", "o:" and "i:" respectively; (ii) the type of the sentence – i.e. either declarative or interrogative – is also annotated on each sentence, indicated by the end-of-sentence punctuation. The LF interface conditions are entirely hierarchical/structural in the constraints they impose – i.e. the values filling the slots consist of *sets* of tokens, not sequences of tokens.

a specifier-head-complement linear ordering. In particular, note that the derivation output by the procedure (see Fig.1-2) that satisfies these interface conditions utilizes syntactic movement to establish both a relation between the predicate "told" and the argument "John" and an agreement relation between "John" and "has". Note that, although a chart parser for MGs may be used to produce such a derivation that yields the externalized (linear) form stipulated in the PF interface conditions, it remains unclear how one can ensure that the derivation produced also satisfies the specified LF interface conditions. A further question is how would the parser function in the case that the LF and PF interface conditions are only partially stipulated. For example, suppose only the LF interface conditions are provided and we wish to use the parser to obtain a derivation that satisfies these LF interface conditions and from which we will derive the satisfied PF interface conditions (as in the case of someone with a thought they wish to externalize to speech). With these questions in mind, let us now turn examine how the procedure for parsing operates, noting that the design of the parser will enable it to answer such questions.

The parser operates by first constructing an SMT-model from the input lexicon and interface conditions and then uses an SMT-solver to solve the model and recover the output derivation from the solution to the model. Let us now consider this process in more detail. First, the parser constructs an SMT-model of a minimalist derivation (i.e. a conjunction of SMT-formulae) that centers on an axiomatization of minimalist syntax grounded in universal principles of language (i.e. the constituent axioms form an encoding of Universal Grammar). Then, the parser derives axioms (expressed as SMT-formulae) from the supplied interface conditions and adds them to the model of the derivation, thereby constraining the model of the derivation so that any derivation recovered from a solution to the model accords with the supplied interface conditions. Next the parser constructs an SMT-model of a minimalist lexicon and hardcodes some of the free variables in the model so that only the supplied lexicon may be recovered from the solution to the model. The parser then forms the SMT-model as the conjunction of the constrained derivation model, the hardcoded lexicon model, and additional axioms that connect the model of the derivation to the model of the lexicon so that any derivation recovered from a solution to the model must be one that can be yielded by the supplied lexicon. Finally, the parser uses an SMT-solver to check whether or not there exists a satisfiable interpretation of the (constructed) SMT-model – i.e. whether or not there exists an assignment of values to the free-variables in the SMT-formulae that make up the SMT-model such that the conjunction of the formulae evaluates to true; if a satisfiable interpretation of the model exists, then the SMT-model identifies one such satisfiable interpretation of the model and the derivation that constitutes the output of the parser is (automatically) recovered from the (identified) satisfiable model interpretation.

To summarize, the parser constructs and then solves a system of logical equations in which the lexicon and interface conditions (i.e. the inputs to the parser) are known quantities and the derivation is an unknown quantity that will be solved for, so that the approach taken may be described as *"parsing via equation solving."* In this way, the parser is a modern adaptation of the *"parsing as deduction"* framework outlined in (Pereira and Warren, 1983), with an axiomatization of minimalist grammars (MGs) in place of an axiomatization of context free grammars (CFGs), pairings of LF and PF interface conditions replacing strings, and an SMT-solver used instead of a Prolog engine. A declaratively specified logical model of minimalist syntax has a particular advantage: although minimalist syntax is a derivational theory, and thus computations are modeled as flowing bottom-up in constructing the derivation, we can stipulate the interface conditions that are met, and the logic program can work out deductions working backwards through the derivation as required; this is not something a typical bottom-up (chart) parser[8] can do, and it enables the model of the minimalist parser to be adapted to the task of grammar induction.

Notably, the model may be used to search for derivations that satisfy a *partial* specification of the lexicon and interface conditions, thereby enabling alternative uses of the parser

[8] See (Harkema, 2001) and (Niyogi and Berwick, 2005) for examples of agenda-based parsers for Minimalist Grammars.

| | |
|---|---|
| $\{\text{eating}, \text{eaten}, \text{asking}, \text{asked}, \text{known}\}/V$ | $:: = x_0, \sim x_0$ |
| $\{\text{given}, \text{asked}, \text{told}\}/V$ | $:: = x_0, = x_0, \sim x_0$ |
| $\epsilon/C_{declarative}$ | $:: = x_0, C$ |
| $\epsilon/C_{question}$ | $:: <= x_0, +z, C$ |
| $\epsilon/v$ | $:: <= x_0, \sim x_0$ |
| $\epsilon/C_{question}$ | $:: <= x_0, C$ |
| $\epsilon/v$ | $:: <= x_0, = x_0, \sim x_0$ |
| $\text{to}/P$ | $:: = x_0, \sim x_0$ |
| $\{\text{a}, \text{the}\}/D$ | $:: = x_0, \sim x_0, -l$ |
| $\{\text{a}, \text{the}\}/D$ | $:: = x_0, \sim x_0$ |
| $\{\text{what}, \text{who}\}/D$ | $:: \sim x_0, -z$ |
| $\{\text{what}, \text{who}\}/D$ | $:: \sim x_0, -l, -z$ |
| $\{\text{has}, \text{was}\}/T$ | $:: = x_0, +l, \sim x_0$ |
| $\{\text{sleeping}, \text{slept}\}/V$ | $:: \sim x_0$ |
| $\{\text{pizza}, \text{everything}, \text{john}, \text{mary}, \text{nothing}, \text{icecream}, \text{someone}, \text{money}\}/N$ | $:: \sim x_0, -l$ |
| $\{\text{story}, \text{pizza}, \text{everything}, \text{john}, \text{mary}, \text{boy}, \text{nothing}, \text{icecream}, \text{someone}, \text{money}\}/N$ | $:: \sim x_0$ |

Table 1.2: The *optimal* minimalist lexicon that was inferred from the PLD listed in Table 1.1 using the (instantaneous) acquisition procedure developed in this thesis. For each entry in the PLD, the derivation yielded by the lexicon both agrees with the interface conditions stipulated in that entry, and also aligns with the prescriptions of contemporary theories of minimalist syntax. Each lexical entry consists of a phonological form paired with a lexical feature sequence (i.e. a sequence of syntactic features); a double-colon indicates that each member of the set of phonetic forms on the left hand side is paired with the lexical feature sequence on the right hand side. The phonetic form $\epsilon$ is covert (unpronounced). A feature has: (i) a value from a finite set of categories; (ii) a type, which is either *selector*, *selectee*, *licensor* or *licensee*, indicated by the prefix $=$, $\sim$, $+$ and $-$ respectively; a $<$ or $>$ prefixed before a selector prefix indicates that the selector triggers left or right head-movement respectively. There is also a special feature, $C$, that serves to indicate the completion of a parse. See Table 1.3 for a *factored view* of this lexicon.

such as:

(i) parsing a given pairing of LF and PF interface conditions with only a partially specified lexicon, as in the case of a child encountering a new verb, and recovering the (novel) lexical entry used by the derivation output by the parser;

(ii) parsing with a specification of a lexicon and LF interface conditions, but without any PF interface conditions supplied, as in the case of a child externalizing a thought, and recovering the PF interface conditions satisfied by the derivation output by the parser;

(iii) parsing with a specification of a lexicon and PF interface conditions, but without any LF interface conditions supplied, as in the case of a child parsing speech, and recovering the LF interface conditions satisfied by the derivation output by the parser.

In this way, the model enables the study of the extent to which the lexicon and the interface conditions constrain the space of possible derivations that a parser must search through. More generally, a declaratively specified, logical model of a grammar enables one to focus on developing the logical axioms that make up the model and ensuring they are faithfully grounded in universal principles of language, while delegating to the SMT-solver the task of

| ID | Category | Features | Phonological Forms | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | pizza | everything | john | mary | nothing | icecream | someone | money | story | boy | a | the | to | sleeping | slept | has | was | eating | eaten | asking | known | asked | told | given | who | what | $\epsilon$ |
| $\mathfrak{L}_1$ | $V$ | $=x_0, \sim x_0$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | × | × | × | × | × | · | · | · | · | · |
| $\mathfrak{L}_2$ | $V$ | $=x_0, =x_0, \sim x_0$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | × | × | × | · | · | · |
| $\mathfrak{L}_3$ | $C_{Decl.}$ | $=x_0, C$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | × |
| $\mathfrak{L}_4$ | $C_{ques.}$ | $<=x_0, +z, C$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | × |
| $\mathfrak{L}_5$ | $v$ | $<=x_0, \sim x_0$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | × |
| $\mathfrak{L}_6$ | $C_{ques.}$ | $<=x_0, C$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | × |
| $\mathfrak{L}_7$ | $v$ | $<=x_0, =x_0, \sim x_0$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | × |
| $\mathfrak{L}_8$ | $P$ | $=x_0, \sim x_0$ | · | · | · | · | · | · | · | · | · | · | · | · | × | · | · | · | · | · | · | · | · | · | · | · | · | · | · |
| $\mathfrak{L}_9$ | $D$ | $=x_0, \sim x_0, -l$ | · | · | · | · | · | · | · | · | · | · | × | × | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · |
| $\mathfrak{L}_{10}$ | $D$ | $=x_0, \sim x_0$ | · | · | · | · | · | · | · | · | · | · | × | × | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · |
| $\mathfrak{L}_{11}$ | $D$ | $\sim x_0, -z$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | × | × | · |
| $\mathfrak{L}_{12}$ | $D$ | $\sim x_0, -l, -z$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | × | × | · |
| $\mathfrak{L}_{13}$ | $T$ | $=x_0, +l, \sim x_0$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | × | × | · | · | · | · | · | · | · | · | · | · |
| $\mathfrak{L}_{14}$ | $V$ | $\sim x_0$ | · | · | · | · | · | · | · | · | · | · | · | · | · | × | × | · | · | · | · | · | · | · | · | · | · | · | · |
| $\mathfrak{L}_{15}$ | $N$ | $\sim x_0, -l$ | × | × | × | × | × | × | × | × | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · |
| $\mathfrak{L}_{16}$ | $N$ | $\sim x_0$ | × | × | × | × | × | × | × | × | × | × | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · |

Table 1.3: Factored View of the Optimized Inferred Lexicon. This is a *factored* view of the optimized inferred lexicon (listed in Table 1.2) that was the output of the acquisition procedure being applied to the primary linguistic data listed in Table 1.1. The 27 columns each code for a distinct phonological form; there are 26 overt phonological forms, and one covert phonological form (i.e. $\epsilon$). The 16 rows each code for a distinct lexical feature sequence. An entry in the table indicates that the pairing of the associated phonological form and lexical feature sequence is an entry in the lexicon. Since every entry in the lexicon can be uniquely factored apart into a pairing of a phonological form and a lexical feature sequence, there is a one-to-one mapping between the entries in this table and the entries in the lexicon. The rows and columns have been seriated (using the hamming distance metric) so as to visually group together similar entries.

carrying out the appropriate deductions that constitute the model-checking process; importantly, in the case that a parse cannot be solved (i.e. there is no satisfiable interpretation of the model), the SMT-solver can identify which model axioms – i.e. those that encode principles of Universal Grammar or those derived from the supplied interface conditions – contradict one another (e.g. by identifying minimum unsatisfiable cores), so that we might gain insight into how the universal principles of language underlying these axioms are in conflict with one another and with the conditions imposed to be satisfied at the LF and PF interfaces respectively.

## 1.2 Acquisition

This thesis develops a novel procedure for inferring Minimalist Grammars that has been implemented as a working computer program. This procedure takes as input: (i) a finite sequence of pairings of LF and PF interface conditions (i.e. (sentence, skeletal "meaning") pairs) that is referred to as the *Primary Linguistic Data* (PLD), and (ii) a initial (minimalist) lexicon (that may be empty). The procedure outputs a minimalist lexicon that is a superset of the initial lexicon and that is compatible with the input PLD – i.e. it can yield, for each pairing of LF and PF interface conditions in the PLD, a derivation that satisfies the LF and PF interface condition; notably, the output lexicon is the smallest possible lexicon

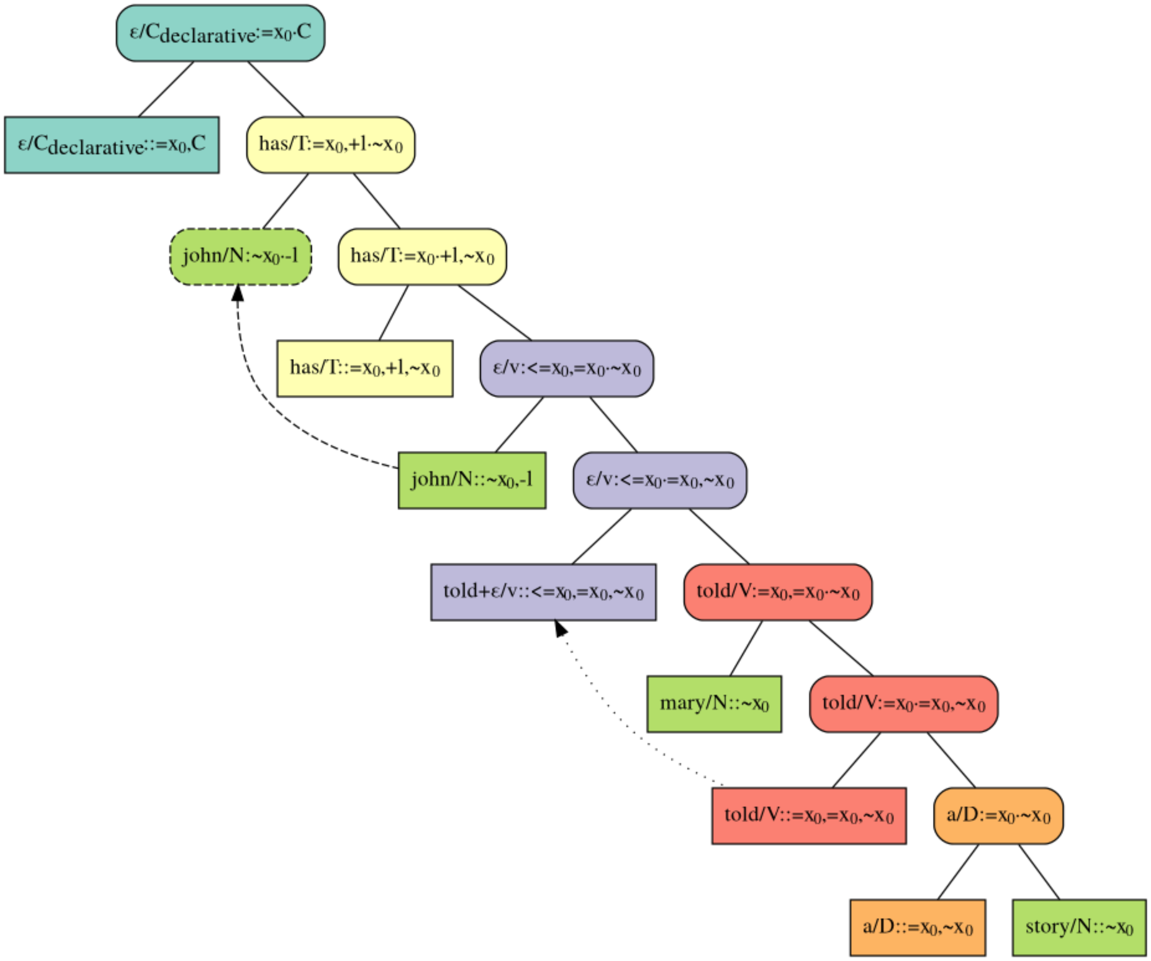


Figure 1-2: A derivation for the sentence *"John has told Mary a story."* that aligns with form prescribed by contemporary theories of minimalist syntax. This derivation satisfies interface conditions $I_{26}$ in Table-1.1 and may be yielded by the lexicon listed in Table 1.3. The leaf nodes are lexical items selected from the lexicon. The derivation is assembled in a bottom-up manner via repeated applications of the structure-building operation *merge*. The feature sequences displayed in non-leaf nodes have a dot, $\cdot$ , separating features that have already been consumed (on the left) from those that have not (on the right). Nodes with the same *head* have the same color. The dashed arrow denotes phrasal movement – i.e. "John" moves from the specifier position of the projection of the (covert) light verb $\epsilon_v$ to the specifier position of the tense marker "has." The dotted arrow denotes head movement – i.e. the lexical head "told" undergoes $V$-to-$v$ head-movement.

(with respect to the number of lexical entries and the total number of symbols appearing in the lexicon) that includes the initial lexicon and is compatible with the input PLD. To take an example of the problem that the procedure is tasked with solving, observe that the procedure can infer the lexicon listed in Table 1.2 from the PLD listed in Table 1.1 and an empty initial lexicon. This requires that the procedure identify both a set of finite sequences of syntactic feature and associations between members of this set and members of a set of phonological forms, with each such association coding for a lexical entry. (See Figure 1-3 for an illustration of this.) Although we might imagine solving this task ourselves by hand when the PLD is fairly small, the problem becomes increasingly difficult as the number and diversity of pairings of interface conditions appearing in the PLD increases. Let us now consider in further detail how the procedure solves this problem.

The procedure operates by first constructing an SMT-model of a minimalist grammar that is an extension of the SMT-model of a minimalist parser (introduced in §1.1), and then using an SMT-solver to solve this model and recover the inferred lexicon from the solution to the model. The procedure constructs a model of a (minimalist) grammar as follows. First, the procedure constructs an SMT-model of a minimalist lexicon that is (partially) constrained by the input lexicon so that the (inferred) lexicon recovered from a satisfiable interpretation of the lexicon model is a superset of the input lexicon. The procedure then processes the entries in the PLD one after another. When processing a pairing of interface conditions in the PLD, the procedure: (i) instantiates a model of a minimalist derivation, (ii) constrains it with SMT-formulae derived from the interface conditions, and finally (iii) adds additional axioms that serve to connect the model of the derivation to the model of the lexicon. Consequently, any derivation recovered from a satisfiable interpretation of the model satisfies in the interface conditions and can be yielded by the lexicon recovered from that interpretation of the model. In this way, the procedure constructs a model of a grammar that consists of the model of the lexicon, a derivation model for each of the entries in the PLD (constrained by the interface conditions listed in that entry), and axioms connecting each derivation model to the lexicon model. Finally, the procedure uses an SMT-solver to check whether or not there exists a satisfiable interpretation of the (constructed) SMT-model of the grammar – i.e. whether or not there exists a lexicon that is compatible with the input PLD. If a satisfiable interpretation of the model exists, the procedure uses the SMT-solver to identify the interpretation of the model from which the smallest lexicon (that is compatible with the PLD) may be recovered and then (automatically) recovers a lexicon from the (identified) model interpretation.

The procedure may be said to be *"solving for syntax"* in so far as it models a grammar as a system of (logical) equations made up of the model of the lexicon and the derivation models, with the input lexicon and interface conditions serving as "boundary conditions." In effect, the procedure is the modern, minimalist theory counterpart of earlier work by (Rayner et al., 1988) on using logic grammars to infer a lexicon, and just as the *"logic as grammar"* framework adapted a *"parsing as deduction"* style parser to the problem of inferring a lexicon by requiring the lexicon parse multiple strings at once, here too the procedure for inferring a minimalist grammar (developed in this thesis) is an extension of the minimalist parser (also developed in this thesis) that requires multiple pairings of interface conditions be parsed at the same time by an unspecified lexicon (with this requirement serving to constrain the model of the lexicon). Notably, the computational experiments presented in Chapter 3 of this thesis demonstrate that when the PLD is sufficiently large (e.g. including at least four to five entries), the inferred lexicon yields derivations that align with the prescriptions of contemporary theories of minimalist syntax for a diverse range of expressions including active and passive voiced declaratives, yes/no-questions, and wh-questions that involve intransitive, transitive and ditransitive verbs; notably, these derivations include empty lexical heads and involve various forms of syntactic movement including wh-raising, subject-raising, T-to-C head-movement and V-to-v head-movement. Additionally, the inferred lexicon can generate novel structures that are not yielded in satisfying any of the entries in the PLD.

Notably, this procedure takes the form of a computational model of child language learner as prescribed by (Berwick, 1985) and accords with the criterion for a model of language acquisition set out in (Chomsky, 1965). The initial state of the learner's knowledge is the input lexicon and the axiomatization of minimalist syntax (encoding UG) that is included in each model of a derivation. The target state of the learner's knowledge (i.e. the knowledge of language that the learner acquires) is the (inferred) lexicon output by the procedure. The procedure drives the state of the learner's knowledge from the initial state to the target state by consuming the PLD and incrementally constructing an SMT-model of a minimalist grammar, and then solving the (constructed) model using an SMT-solver and recovering the output lexicon from the solution. The procedure may also be adapted to incrementally consume the PLD in batches and output a lexicon after processing each batch, with each output lexicon a superset of the previously output lexicon, so that the procedure may be said to model a child language learner incrementally acquiring knowledge of language; we refer to this adaptation of the procedure as the *incremental acquisition procedure*, and the non-incremental variant of the procedure as the *instantaneous acquisition procedure* (so called because it models the learner "instantly" acquiring knowledge of language by processing all of the PLD at once). It is worth pointing out that the *incremental acquisition procedure* enables the learner to incrementally acquire a grammar from an arbitrarily large PLD as only a single batch of the PLD needs to be modeled at a time; indeed, the PLD may even be infinite in size if it is presented to the learner as a stream. The *incremental acquisition procedure* also enables the computationally tractable inference of larger grammars. The computational experiments presented in §3.3 demonstrate that the *incremental acquisition procedure* may be used to incrementally acquire (i.e. infer), from psychologically plausible input data consisting of a (small) finite sequence of simple sentences with at most one level of embedding, including sentences with embedded declaratives, questions and (restrictive) relative clauses, a lexicon that can yield derivations with $n$ levels of embedding for any $n \geq 0$ – thus the inferred lexicon can generate a countably infinite set of hierarchical structures that pair meaning with sound, demonstrating that the acquisition procedure can infer a lexicon that makes "infinite use of finite means."

Finally, let us consider the instantaneous and incremental acquisition procedures from the perspective of psychological plausibility.

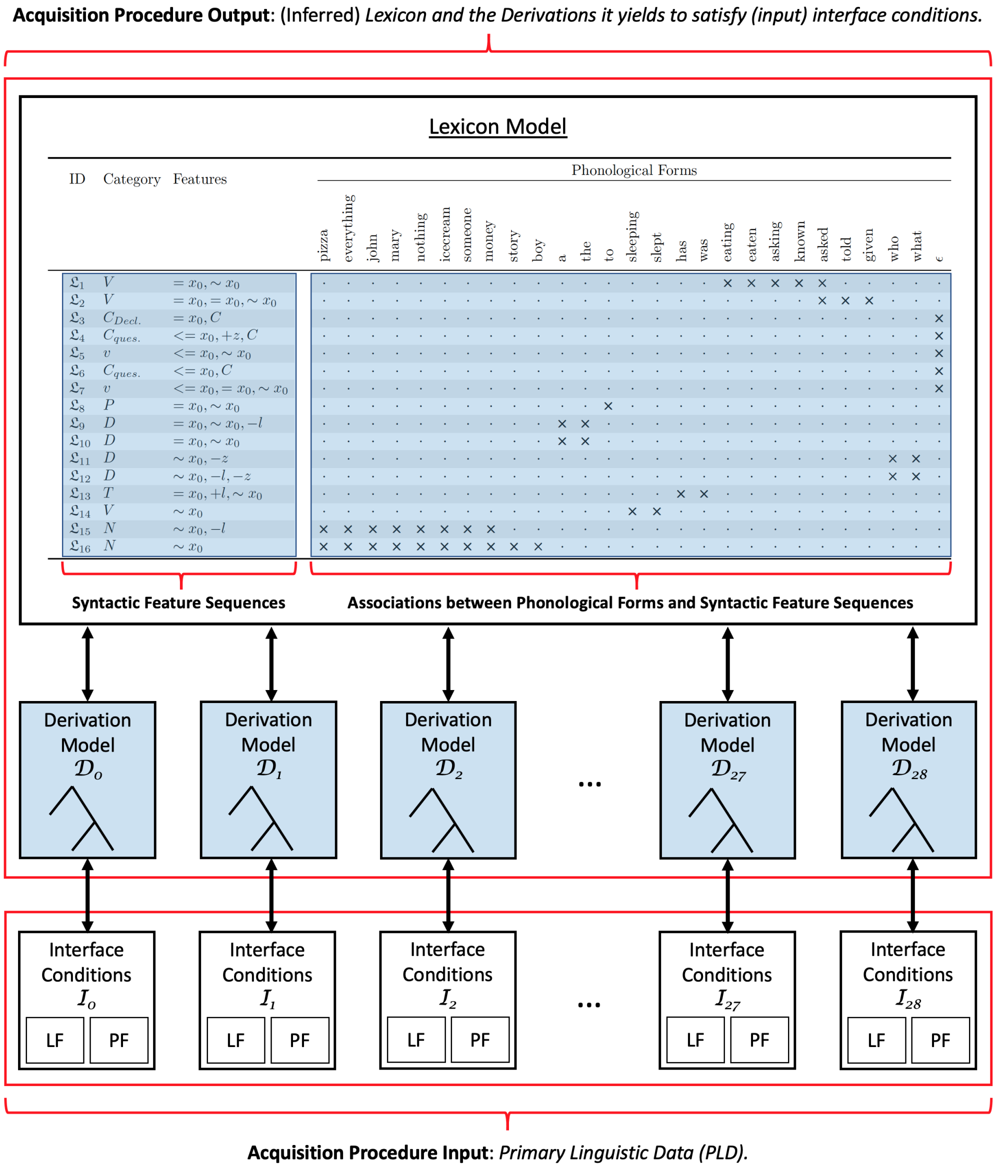
- The procedure infers a lexicon from positive evidence only – i.e. no *direct negative* evidence (i.e. evidence in the form of explicit corrections of mistakes) is presented to the system, nor does the system make use of *indirect* negative evidence (i.e. evidence derived from the observed absence of a particular production).

- The axiomatization of minimalist syntax underlying the model of the minimalist parser is a part of the initial state of the acquisition procedure (corresponding to the learner's innate knowledge of language – i.e UG). The procedure for acquisition accords with the *Strong Continuity Hypothesis*: since each SMT-model of a derivation constructed by the procedure includes the axiomatization of minimalist syntax (that encodes principles of UG), at each stage in the learner's acquisition trajectory, each member of the class of grammars (i.e. the space of satisfiable model interpretations) that the learner has narrowed down to must accord with the principles of UG.[9]

- The *instantaneous* acquisition procedure is robust to changes in the order in which the PLD is presented to the system.

- The *incremental* acquisition procedure only requires the learner have a small, finite (potentially zero) sized memory for remembering sentences from the PLD.

[9] See (Lust, 1999).



**Acquisition Procedure Output**: (Inferred) *Lexicon and the Derivations it yields to satisfy (input) interface conditions.*
Lexicon Model
Syntactic Feature Sequences
Associations between Phonological Forms and Syntactic Feature Sequences
Derivation Model $\mathcal{D}_0$
Derivation Model $\mathcal{D}_1$
Derivation Model $\mathcal{D}_2$
...
Derivation Model $\mathcal{D}_{27}$
Derivation Model $\mathcal{D}_{28}$
Interface Conditions $I_0$
LF
PF
Interface Conditions $I_1$
LF
PF
Interface Conditions $I_2$
LF
PF
...
Interface Conditions $I_{27}$
LF
PF
Interface Conditions $I_{28}$
LF
PF
**Acquisition Procedure Input**: *Primary Linguistic Data (PLD).*

| ID | Category | Features | Phonological Forms | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | pizza | everything | john | mary | nothing | icecream | someone | money | story | boy | a | the | to | sleeping | slept | has | was | eating | eaten | asking | known | asked | told | given | who | what | $\epsilon$ |
| $\mathfrak{L}_1$ | $V$ | $=x_0, \sim x_0$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | × | × | × | × | × | · | · | · | · | · |
| $\mathfrak{L}_2$ | $V$ | $=x_0, =x_0, \sim x_0$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | × | × | × | · | · | · |
| $\mathfrak{L}_3$ | $C_{Decl.}$ | $=x_0, C$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | × |
| $\mathfrak{L}_4$ | $C_{ques.}$ | $<=x_0, +z, C$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | × |
| $\mathfrak{L}_5$ | $v$ | $<=x_0, \sim x_0$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | × |
| $\mathfrak{L}_6$ | $C_{ques.}$ | $<=x_0, C$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | × |
| $\mathfrak{L}_7$ | $v$ | $<=x_0, =x_0, \sim x_0$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | × |
| $\mathfrak{L}_8$ | $P$ | $=x_0, \sim x_0$ | · | · | · | · | · | · | · | · | · | · | · | · | × | · | · | · | · | · | · | · | · | · | · | · | · | · | · |
| $\mathfrak{L}_9$ | $D$ | $=x_0, \sim x_0, -l$ | · | · | · | · | · | · | · | · | · | · | × | × | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · |
| $\mathfrak{L}_{10}$ | $D$ | $=x_0, \sim x_0$ | · | · | · | · | · | · | · | · | · | · | × | × | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · |
| $\mathfrak{L}_{11}$ | $D$ | $\sim x_0, -z$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | × | × | · |
| $\mathfrak{L}_{12}$ | $D$ | $\sim x_0, -l, -z$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | × | × | · |
| $\mathfrak{L}_{13}$ | $T$ | $=x_0, +l, \sim x_0$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | × | × | · | · | · | · | · | · | · | · | · | · |
| $\mathfrak{L}_{14}$ | $V$ | $\sim x_0$ | · | · | · | · | · | · | · | · | · | · | · | · | · | × | × | · | · | · | · | · | · | · | · | · | · | · | · |
| $\mathfrak{L}_{15}$ | $N$ | $\sim x_0, -l$ | × | × | × | × | × | × | × | × | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · |
| $\mathfrak{L}_{16}$ | $N$ | $\sim x_0$ | × | × | × | × | × | × | × | × | × | × | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · |

Figure 1-3: The *instantaneous acquisition procedure* can infer the lexicon listed in Table 1.3 from the Primary Linguistic Data (PLD) listed in Table 1.1 (and an empty initial lexicon). (See §3.2.3 of Ch. 3 for a detailed presentation of this computational experiment.) The *instantaneous acquisition procedure* constructs an SMT-model that consists of: (i) a lexicon model (an SMT-formula) that is empty, and (ii) a derivation model (an SMT-formula) for each pairing of LF and PF interface conditions in the PLD (see derivation models $D_0 - D_{28}$). Next, each pairing of LF and PF interface conditions, $I_i$, in the PLD is translated into an SMT-formula that constrains the derivation model, $D_i$, so that any satisfiable interpretation of $D_i$ encodes an MG derivation that satisfies $I_i$ – each such SMT-formula is added to the SMT-model. Then, each derivation model is connected to the lexicon model via an SMT-formula that requires that derivation be yielded by the lexicon, thereby constraining satisfiable interpretations of both the derivation model and lexicon model – again, each such SMT-formula is added to the SMT-model. Finally, optimization constraints, derived from Principles of Economy, are added to the SMT-model, thereby further constraining satisfiable interpretations of the derivation and lexicon models. After constructing the SMT-model, an SMT-solver is used to identify a satisfiable interpretation of the SMT-model (i.e. a solution to the system of SMT-formulae that make up the SMT-model), and the inferred lexicon and derivation trees are then automatically recovered from the identified interpretation of the SMT-model. By solving the SMT-model, the SMT-solver works out the contents of the boxes shaded light-blue – i.e.: (i) what the (unspecified) syntactic feature sequences in the lexicon are, and (ii) the associations between the syntactic feature sequences and the phonological forms (that are obtained by scanning the PF interface conditions in the PLD).

- Following the *"Semantic Bootstrapping Hypothesis"*, the learner is assumed to know "*who* did *what* to *whom*", and thus each sentence is annotated with predicate-argument structure and agreement relations (these relations are encoded within the LF interface conditions).[10]

- The *incremental acquisition procedure* integrates language acquisition and language processing. When processing the first batch of the PLD, if the procedure is not supplied with an initial lexicon, then the procedure is functionally equivalent to the instantaneous acquisition procedure; as more batches of the PLD are processed over successive runs of the incremental acquisition procedure, in the limit, when the learner can already parse the next PLD batch with the (up to then) acquired lexicon, the procedure in effect becomes the procedure for parsing each entry in the PLD batch. This transition from acquisition to parsing is possible because both center on the same underlying axiomatization of minimalist syntax (detailed in §2.3), with the difference between the two being the degree to which the lexicon model is hard-coded with the lexicon that the learner has acquired up until that point.

Chapter 4 explores extensions to the procedure that may serve to bring it into further accordance with conditions of psychological plausibility.

## 1.3 Summary of Key Ideas and Takeaways

- The models of language parsing and acquisition developed in this thesis are grounded in three fundamental ideas:

  (i) that *Merge is all you need* – i.e. the simplest combinatory operation is sufficiently expressive;

  (ii) that all syntactic relations are established in the simplest possible way – i.e. locally within a syntactic structure, via the operation *merge*;

  (iii) that the acquired lexicon must yield derivations that are subject to *Economy Conditions* – i.e. derivations must involve as few steps (i.e. *merge* operations) as possible while still satisfying interface conditions pursuant to the *Principle of Full Interpretation*.

  This thesis, put plainly, is an attempt to see how far these three ideas can take us – i.e. to use the models developed to evaluate the degree to which economically satisfying interface conditions uniquely determines the acquired grammar, and the extent to which said grammar aligns with minimalist theories of syntax.

- This thesis introduces novel procedures for language acquisition that are able to use a modern, high-performance SMT-solver to infer, from an empty initial lexicon and primary linguistic data consisting of a (small) finite sequence of ⟨sentence, skeletal-meaning⟩ pairs (i.e. pairings of LF and PF interface conditions) with at most one level of structural embedding, an MG lexicon that is compatible with the primary linguistic data and that can generate a countably infinite set of (interpretable) hierarchical structures,

[10] See (Lust, 2006, Pgs. 42-43) for a review of the Semantic Bootstrapping Hypothesis.

each of which pairs meaning with sound (as demonstrated in the computational experiments presented in Ch. 3). In particular, the computational experiment detailed in §3.2 demonstrates that by solving for the optimal lexicon that is compatible with the primary linguistic data, the system infers a grammar that aligns with contemporary theories of minimalist syntax in so far as: (a) the grammar produces the prescribed derivations for a variety of syntactic structures, utilizing syntactic movement (including head-movement) and covert lexical items as needed; (b) expressions with related interpretations are assigned derivations systematically related by structural transformations. In particular, the optimized lexicon that was inferred yields derivations for declaratives, yes/no-questions, and wh-questions in both the active and passive voice, and these derivations involve various forms of syntactic movement including wh-raising, subject-raising, T-to-C head-movement and V-to-v head-movement; additionally, the inferred lexicon includes lexical entries for *covert* complementizers and light-verbs. Furthermore, the computational experiment detailed in §3.3 demonstrates that the system can acquire, from a finite set of sentences with at most one level of embedding, a grammar that can yield an infinite number of distinct syntactic structures. That this idealistic model is able to recover most of the right syntax is remarkable; notably, the acquisition procedures does this without being provided a treebank of minimalist derivations that serve as examples of what the acquired lexicon should be able to yield, and to that end, the system constitutes a scheme for *unsupervised* inference of minimalist grammars.

- The lexicon model has a factored representation – i.e. it has the form of a two-dimensional association matrix, with a set of phonological forms on one axis, a set of lexical feature sequences on the other axis, with the entries in the matrix marking which lexical feature sequences are associated with which phonological forms. (See Table 1.3 on for an illustration of factored representation.) The lexicon's factored representation allows for it to be grown along the two dimensions of the matrix separately, with the dimension pertaining to lexical feature sequences controlling which syntactic structures the lexicon can yielded, and the dimension pertaining to phonological forms controlling the vocabulary of the learner. Specifically, the lexicon's factored representation enable us to define constraints, derived from optimization metrics grounded in Principles of Economy, that when added to the SMT-model of the grammar, serve to minimize the size of the lexicon with respect to the set of distinct lexical feature sequences without factoring in how often the different phonological forms appear in the primary linguistic data; these constraints ensure that, as the system processes the primary linguistic data, the system only adds new lexical feature sequences to the lexicon if the existing set of lexical feature sequences is insufficient for yielding a syntactic structure that can satisfy a pairing of LF and PF interface conditions. In particular, this means that if the system cannot parse an entry in the primary linguistic data, it will only add a new lexical feature sequence to the lexicon if adding a new association between a phonological form and an existing lexical feature sequence will not suffice. Finally, when constraints serving to minimize the optimization metrics – i.e. minimize the size of the lexicon and the size of the derivations it yields – are added to the (constructed) SMT-model, the *instantaneous acquisition procedure* infers a lexicon that yields derivations that accord (in form) with the prescriptions of contemporary theories of modern syntax.

- This thesis investigates whether and how it is possible to use an interactive theorem prover to aid in the study linguistic theory. To this end, the thesis develops procedures that center on an axiomatization of minimalist syntax that is expressed using a logic, SMT, and that use an SMT-solver (a kind of interactive theorem prover) for parsing MGs and (automatically) inferring MG lexicons respectively. This enables us to focus on developing the axioms and understanding how they are grounded in universal principles of language while leaving it to the SMT-solver to decide how best to work out the relevant deductions — in effect we are offloading the mechanical aspect of thinking about linguistic theory to the solver.

  (i) In the case of the procedure for parsing, we may focus on developing a minimal set of axioms (encoding UG) that are grounded in universal principles of language and that allow for the generation of derivations prescribed by contemporary theories of minimalist syntax while prohibiting the "over-generation" of derivations, leaving the automatic theorem prover to carry out the deductions involved in parsing without us having to stipulate the particular algorithms and implementation details of the parser.

  (ii) In the case of the procedure for acquisition, the automatic theorem prover enables us to separate out the questions of what knowledge of syntax the model of the child language learner acquires and how the model of the child language learner acquires it – i.e. it allows us setup computational experiments in which we focus on specifying the learner's initial state and the conditions that the learner's final state must satisfy (with respect to the primary linguistic data that the learner processes), and leave to the solver the questions of *how* the language-acquisition device goes from the initial state to the final state and *what* that final state is. In particular, the *interactive* nature of the theorem prover enables the procedure to incrementally construct the SMT-model of the grammar (by processing the primary linguistic data in batches) and incrementally growing the acquired lexicon (by solving the model after a batch of the primary linguistic data is processed).

  In this way, the system lets us explore the interaction of several simple principles – i.e. that derivations must satisfy interface conditions, that derivations are subject to economy conditions, and that all syntactic relations within a derivation are established by merge – in the context of parsing and acquisition. Furthermore, the models underlying the procedures for parsing and acquisition are amenable to modification and extension by way of adding or subtracting particular axioms, and one can construct and evaluate any hypothesis expressible as an SMT formula (really any decidable problem one can manage to express with SMT) and evaluate against the encoded linguistic theory, as discussed further in Ch. 4. Ultimately, the system may serve as a vehicle for better understanding how more can be done with less, which is a line of inquiry that lies at the heart of the Minimalist Program.

# Chapter 2

# Modeling a Minimalist Parser with Satisfiability Modulo Theories

This chapter presents a Satisfiability Modulo Theories (SMT) model of a minimalist parser that has been developed and implemented as a (working) computer program. The model takes as input a lexicon and a specification of LF and PF interface conditions (encoding word order, predicate-argument structure, and agreement relations), and outputs a minimalist derivation that can be yielded by the lexicon and that satisfies the specified interface conditions.

This chapter first develops an SMT-model of a minimalist derivation and an SMT-model minimalist lexicon, both of which based on the Minimalist Grammar (MG) formalism introduced in (Stabler, 1996), using a multi-sort first-order quantifier-free logic extended with the theory of equality and the theory of uninterpreted functions. The model of the derivation consists of several sorts, uninterpreted functions acting over these sorts, and a set of axioms (i.e. logical formulas constraining these functions) that an MG derivation must satisfy. Likewise, the model of the lexicon also consists of a number of sorts, uninterpreted functions acting over these sorts, and a set of axioms (constraining these functions) that an MG lexicon must satisfy. The model of the lexicon is connected to the model of the derivation via an uninterpreted function that maps projections in the model of the derivation to lexical entries in the model of the lexicon, so as to require that the derivation be yielded by the lexicon. When the parser is run, the specified interface conditions are (automatically) translated to first order logic formulas that further constrain the model of the derivation, and the model of the lexicon is "hardcoded" with the input lexicon – i.e. the model of the lexicon is further constrained so as to require that the input lexicon is the only lexicon that may be recovered from a satisfiable interpretation of the lexicon model. The SMT-model of the minimalist parser is then the conjunction of the SMT-formulae encoding (i) the derivation constrained by the specified interface conditions, (ii) the specified lexicon, and (iii) the mapping between projections in the derivation and lexical entries in the lexicon. Checking the model of the parser (i.e. determining whether or not the model has a satisfiable interpretation) is equivalent to solving the decision problem of whether there exists a derivation that both satisfies the specified interface conditions and is yielded by the input lexicon; if the answer is yes – i.e. the model has a satisfiable model interpretation – then such a derivation can be (automatically) recovered from the interpretation of the model, whereas if the answer is no, then the specified interface conditions cannot be parsed using the input lexicon.

This chapter then demonstrates (i) how an SMT-solver may be used to check the model of the parser and obtain a satisfiable interpretation of the model, and (ii) how to automatically recover a derivation from the model interpretation, thereby carrying out the task of parsing. We present examples of derivations, produced by our implementation of the parser, for sentences with a variety of syntactic structures, including wh-questions, passive constructions, and sentences with embedded clauses. Finally, we discuss how to configure the model of the parser so that it can handle out-of-vocabulary words, as well as the model's ability to parse partial specifications of interface conditions – e.g. the model can be used to parse LF interface conditions in the absence of PF interface conditions.

## 2.1 Overview

Modern natural language parsers are tasked with automatically mapping an expression, composed of a finite sequence of words, to a finite set of structures, such that each of these structures is associated with, and is intended to reflect the underlying syntax of, a distinct interpretation of the expression. Modern linguistic theory, specifically *Minimalist* theories of syntax, focuses on modeling the Human language Faculty (HLF), a computational system that is optimally designed for the problem of pairing sound with meaning; this is done by identifying syntactic structures, derived from a lexicon, that satisfy interface conditions imposed by the Conceptual-Intentional (CI) and Sensory-Motor (SM) systems, which process meaning (e.g. predicate-argument structure) and sound (e.g. word ordering) respectively.[1] (See Fig. 2-1.) This thesis asserts that the architecture of the HLF, as prescribed by contemporary theories of minimalist syntax (Adger, 2003; Radford, 1997; Hornstein et al., 2005), leads to a broader definition of the task of natural language parsing:

> *Given a lexicon and a specification of LF and PF interface conditions, the task of a natural language parser is to enumerate the set of syntactic structures that may be derived from the lexicon and that satisfy the specified interface conditions.*

This chapter presents a model of a minimalist parser that adheres to this more general definition of parsing, and that has been developed and implemented as a working computer program.[2] The implemented parser can, for example, take as input the MG lexicon listed in Table 2.1 and the interface conditions listed under entry $I_1$ in Table 2.4, and outputs the MG derivation in Fig. 2-2. The model centers on an axiomatization of minimalist syntax that closely follows the MG formalism (reviewed in §2.3) and is expressed declaratively using a logic, Satisfiability Modulo Theories (SMT).[3] We show how to further constrain

[1] From this perspective, contemporary NLP parsing may be re-framed as fully specifying the interface conditions imposed by SM (i.e. the expression to parse), and then returning a syntactic structure that is derived from the lexicon and that produces that expression; the task of recovering a logical form from the structure is typically passed over.

[2] This definition asserts that a parser should be able to process input for which *only* interface conditions imposed by CI are present, and still yield syntactic structures from which the surfaced expression may be recovered – this is effectively an inversion of the standard task of a natural language parser. Such a parser allows us to study the degree to which interface conditions imposed by the CI system determine the syntactic structures that may be produced; this is motivated by: (i) the Syntax-Semantics duality, which asserts that the syntactic structures associated with a natural language expression are closely aligned in form with logical form associated with the expression; (ii) the assertion that the universal properties and principles of language are almost entirely pertaining to the interface between the HLF and the CI system.

[3] An SMT-model is a logical formula expressed in first order logic with equality (optionally) extended with background theories such as the theory of uninterpreted functions or the theory of integers. Given an SMT model, the associated SMT problem is the decision problem of whether the formula is satisfiable. See (Barrett and Tinelli, 2018) for further reference.

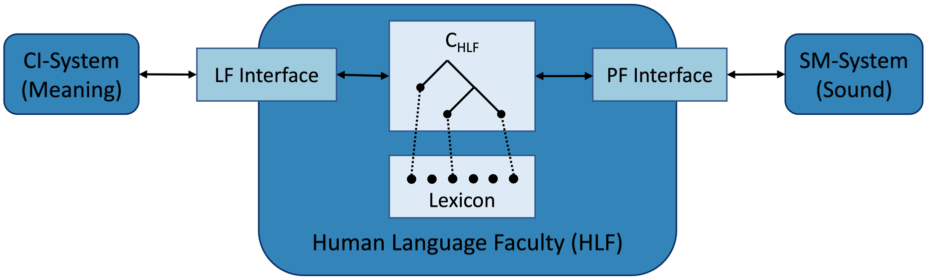


Figure 2-1: Minimalist theories of syntax assert that the Human Language Faculty (HLF) pairs meaning, which is processed by the Conceptual-Intensional (CI) System, with sound, which is processed by the Sensory-Motor (SM) System. The Computational System ($C_{HLF}$) derives a syntactic structure from the syntactic atoms in the Lexicon via the repeated application of the recursive binary function *merge*; after each step of the derivation (i.e. application of *merge*), (relevant) information is sent from $C_{HLF}$ to the LF and PF Interfaces. (Chomsky, 1995)

the model with additional SMT-formulae automatically derived from the specified interface conditions, thereby expressing the problem of parsing (specified interface conditions) as a decision problem (encoded as an SMT-model); this SMT-model is evaluated using a modern, high-performance SMT-solver[4] and the derivation constituting the output of the parser is automatically recovered from the interpretation of the model output by the SMT-solver.[5] (see §2.4)

Notably, the parser developed in this chapter is a modern adaptation of the *"parsing as deduction"* framework developed by (Pereira and Warren, 1983), with an axiomatization of minimalist grammars (MGs) in place of an axiomatization of context free grammars (CFGs), a pairing of LF and PF interface conditions replacing the input string, and deductions carried out by an SMT-solver instead of a Prolog engine.[6] The model of the parser may be viewed as a system of (logical) equations in which the lexicon and interface conditions (i.e. the inputs to the parser) are known quantities and the derivation is an unknown quantity that will be solved for, so that the approach taken here may be described as *"parsing via equation solving."*

## 2.2 Minimalist Grammars

This study models minimalist syntax using the Minimalist Grammar (MG) formalism, a well established formal model of syntax introduced in (Stabler, 1996) that is inspired by (Chomsky, 1995) and for which bottom-up chart-parsers and transition based parsers have been developed.[7]

This study opted to use MGs for two reasons. Firstly, the present study assumes the Mild Context-Sensitivity Hypothesis (MCSH), which claims that natural languages are Mildly Context-Sensitive,[8] and MGs are mildly context-sensitive grammars[9] and can model the

[4] SMT solvers are a type of automatic theorem prover that can solve an SMT problem; if a satisfiable interpretation of the associated model exists, the solver can enumerate them explicitly.

[5] Our model is restricted to the quantifier free subset of SMT so that model is a decidable decision problem.

[6] See also (Shieber et al., 1995).

[7] See (Harkema, 2001), (Niyogi and Berwick, 2005), (Stanojević, 2016), and (Torr et al., 2019).

[8] See (Joshi, 1985), (Vijay-Shanker et al., 1987), and (Joshi et al., 1990).

[9] See (Michaelis, 1998) and (Michaelis, 2001).

| Lexicon | |
|---|---|
| 1. $\epsilon/C_{Ques.}$ :: $<=x, +p, C$ | 18. *that* :: $=x, \sim y$ |
| 2. *has* :: $=x, +q, \sim x$ | 19. *he* :: $\sim y, -q$ |
| 3. *the* :: $=y, \sim y, -q$ | 20. *resigned* :: $\sim x$ |
| 4. *man* :: $\sim y$ | 21. *known* :: $=y, \sim x$ |
| 5. $\epsilon/v$ :: $<=x, =y, \sim x$ | 22. *everyone* :: $\sim y, -q, -p$ |
| 6. *eaten* :: $=y, \sim x$ | 23. *who* :: $=x, +p, \sim y$ |
| 7. *what* :: $\sim y, -p$ | 24. *loved* :: $=y, \sim x$ |
| 8. $\epsilon/v$ :: $<=x, \sim x$ | 25. $\epsilon/C_{Decl.}$ :: $=x, C$ |
| 9. $\epsilon/C_{Ques.}$ :: $<=x, C$ | 26. *knows* :: $=y, \sim x$ |
| 10. *was* :: $=x, +q, \sim x$ | 27. *john* :: $\sim y, -q$ |
| 11. *she* :: $\sim y, -q$ | 28. *given* :: $=y, \sim x$ |
| 12. *given* :: $=y, =y, \sim x$ | 29. $\epsilon/T$ :: $=x, +q, \sim x$ |
| 13. *money* :: $\sim y$ | 30. *money* :: $\sim y, -q, -p$ |
| 14. *will* :: $=x, +q, \sim x$ | 31. *that* :: $=x, +p, \sim y$ |
| 15. *who* :: $\sim y, -q, -p$ | 32. *stolen* :: $=y, \sim x$ |
| 16. *her* :: $\sim y$ | 33. *fears* :: $=y, \sim x$ |
| 17. *tell* :: $=y, =y, \sim x$ | 34. *money* :: $\sim y, -q$ |

Table 2.1: Each listed lexical item consists of: (i) a phonological form; (ii) an optionally specified categorical feature (e.g. entries 1 and 5); (iii) a sequence of syntactic features. $\epsilon$ denotes a covert phonological form. Selectional features are denoted by the prefixes $=$ and $\sim$ for selectors and selectees respectively, with the labels $x$ and $y$ used for the purpose of argument and non-argument selection; head-movement is triggered by the presence of a selector feature prefixed by a $<=$. Licensing features are denoted by the prefixes $+$ and $-$ for licensors and licensees respectively; the licensing label $p$ is involved in wh-movement and the formation of relative clauses, and the licensing label $q$ is involved in subject raising. The special feature $C$ marks the convergence of a derivation. The lexicon includes verbs with varying valencies, including intransitive verbs (e.g. entry 20), transitive verbs (e.g. entries 6, 17, 21, 24, 26, 28, 32, and 33), and ditransitive verbs (e.g. entries 12, and 17). It also includes auxiliary verbs (e.g. entries 2, 10, and 14), determiners (e.g. entry 3), and nominals (e.g. entries 4, 11, 13, 16, 19, 22, 27, 30, and 34).

cross-serial dependencies that arise in natural language.[10] Secondly, MGs can model the syntactic constraints that appear in contemporary syntax: (Rogers and Nordlinger, 1998) showed that the syntactic constraints that make up UG are expressible with Monadic Second-Order (MSO) logic, and (Graf, 2013) showed that any set of MSO-expressible constraints over MG derivation trees can be encoded within an MG lexicon.[11] Of particular relevance to modeling theories of minimalist syntax, MGs can model displacement, a basic fact of natural language that enables a phrase to be interpreted both in its final, surfaced position, as well as other positions within a syntactic structure (Chomsky, 2013b) – i.e. a single phrase satisfying multiple interface conditions often requires that it undergo syntactic movement to establish a discontinuous structure (i.e. a chain) with multiple local relations; per the *Principle of Last Resort*, movement is driven by morphological considerations – e.g. morphological agreement

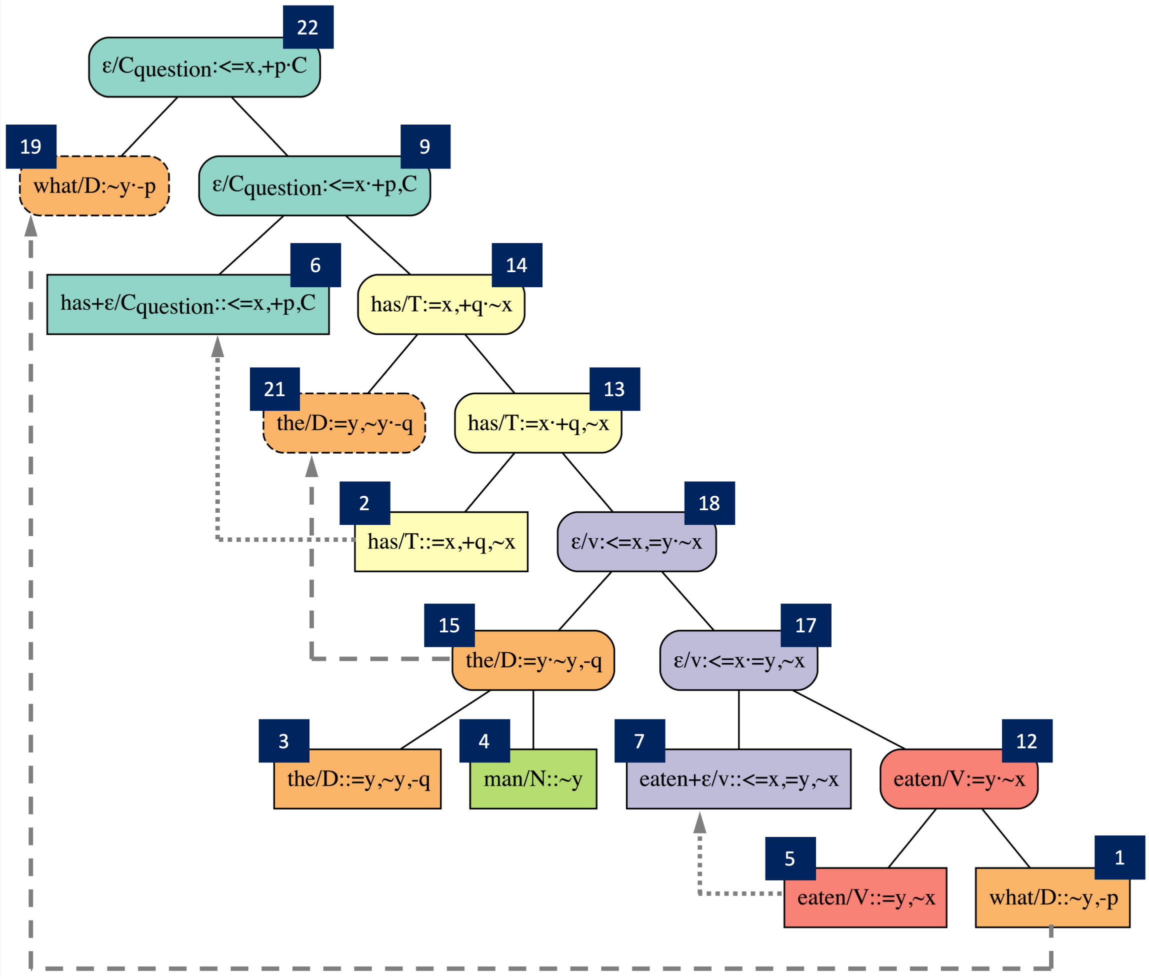


Figure 2-2: A minimalist derivation of the sentence *"What has the man eaten?"* that satisfies the LF and PF interface conditions listed under entry $I_1$ in Table 2.4. This derivation was recovered from the model interpretation presented in Table 2.6; each node in the derivation is labeled with the index of a row in the table. The depicted structure is a multi-dominance tree, with nodes $\{1, 5, 12, 7, 17, 3, 4, 15, 18, 2, 13, 14, 6, 9, 22\}$ making up the derivation tree from which this multi-dominance tree was derived. Lexical nodes are indicated by rectangular nodes, while derived nodes are indicated by rounded corners. Constituents with the same color have the same head. Dashed and dotted grey arrows indicate phrasal and head movement respectively; a dashed border indicates that a node is the target of phrasal-movement, with the lower (raised) structure implicitly copied to the target position.

(Chomsky, 1995).

Let us now review the algebraic formulation of MGs presented in (Stabler and Keenan, 2003). Formally, an MG, $G$, is defined by a five tuple: $(\Sigma, Sel, Lic, Lex, \mathbb{M})$. $\Sigma$ is a finite, non-empty vocabulary of phonological forms. A phonological form is either *overt* (i.e. pronounced) or *covert* (i.e. unpronounced), and we will let $\epsilon$ denote a covert phonological form. $Sel$ and $Lic$ are non-empty disjoint finite sets of feature labels for *selection* and *licensing*

[10] See (Stabler, 2004).

[11] See (Morawietz, 2008, Pgs. 29-42) for a review of MSO-logic (over trees).

respectively; with these feature labels the set of syntactic features, $F$, may be defined as the union of the following four disjoint sets of syntactic features, respectively distinguished by the symbols prefixing their members (i.e. $=$, $\sim$, $+$, and $-$):

$$\begin{aligned} \textit{selectors}&: \ \{{=}f|f \in \mathit{Sel}\} \qquad (2.1)\\ \textit{selectees}&: \ \{{\sim}f|f \in \mathit{Sel}\} \qquad (2.2)\\ \textit{licensors}&: \ \{{+}f|f \in \mathit{Lic}\} \qquad (2.3)\\ \textit{licensees}&: \ \{{-}f|f \in \mathit{Lic}\} \qquad (2.4) \end{aligned}$$

A $<$ or $>$ prefixed before a selector prefix – i.e. "$<=$" or "$>=$" – indicates that the selector triggers left or right head-movement respectively.

The set of chains is $H = \Sigma^* \times \mathit{Types} \times F^*$, with the set $\mathit{Types} = \{::, :\}$ used to designate whether a chain is *lexical* or *derived* (from *lexical* chains) respectively. The lexicon $\mathit{Lex}$ is defined as a non-empty finite set of lexical chains.[12] The set of expressions, $E = H^+$, may be recursively combined together to produce another expression via the binary function *Merge*, denoted by $\mathbb{M}$.

*Merge* has two disjoint subcases: (i) *external merge* (EM), which requires that both arguments of merge are disjoint from one another; (ii) *internal merge* (IM), which requires that one of the arguments is a constituent of the other. Let $s, t \in \Sigma^*$, $f \in \mathit{Sel}$, $g \in \mathit{Lic}$, $\gamma \in F^*$ and $\delta \in F^+$, and let $\alpha_1, ..., \alpha_k, \iota_1, ..., \iota_l \in H$ for $0 \leq k, l$; then EM is defined as the union of the three disjoint functions, $\{EM_1, EM_2, EM_3\}$, that employ feature *selection*:

$$\frac{[\mathrm{s} :: {=}f,\ \gamma] \qquad [\mathrm{t} \cdot {\sim}f],\ \iota_1...\iota_l}{[\mathrm{st} : \gamma],\ \iota_1...\iota_l}\ \mathrm{EM}_1 \tag{2.5}$$

$$\frac{[\mathrm{s} : {=}f,\ \gamma],\ \alpha_1...\alpha_k \qquad [\mathrm{t} \cdot {\sim}f],\ \iota_1...\iota_l}{[\mathrm{ts} : \gamma],\ \alpha_1...\alpha_k, \iota_1...\iota_l}\ \mathrm{EM}_2 \tag{2.6}$$

$$\frac{[\mathrm{s} \cdot {=}f,\ \gamma],\ \alpha_1...\alpha_k \qquad [\mathrm{t} \cdot {\sim}f,\ \delta],\ \iota_1...\iota_l}{[\mathrm{s} : \gamma],\ \alpha_1...\alpha_k,\ [\mathrm{t} : \delta],\ \iota_1...\iota_l}\ \mathrm{EM}_3 \tag{2.7}$$

The separation of the phonetic form and the syntactic features by the symbol $\cdot$ designates that the chain may be either *lexical* or *derived.* IM is defined as the union of the following two disjoint functions, $\{IM_1, IM_2\}$, that employ feature licensing:

$$\frac{[\mathrm{s} : {+}f,\ \gamma],\ \alpha_1...\alpha_{i-1},\ [\mathrm{t} : {-}f],\ \alpha_{i+1}...\alpha_k}{[\mathrm{ts} : \gamma],\ \alpha_1...\alpha_{i-1}, \alpha_{i+1}...\alpha_k}\ \mathrm{IM}_1 \tag{2.8}$$

$$\frac{[\mathrm{s} : {+}f,\ \gamma],\ \alpha_1...\alpha_{i-1},\ [\mathrm{t} : {-}f,\ \delta],\ \alpha_{i+1}...\alpha_k}{[\mathrm{s} : \gamma],\ \alpha_1...\alpha_{i-1},\ [\mathrm{t} : \delta],\ \alpha_{i+1}...\alpha_k}\ \mathrm{IM}_2 \tag{2.9}$$

Additionally, $IM_1$ and $IM_2$ are subject to the *Shortest Move Constraint* (SMC): when a licensor, $\alpha$, binds to a licensee, $\beta$, it must be the case that $\beta$ is the only licensee to which $\alpha$ can bind.[13]

[12] The syntactic features in the MG formalism are strictly uninterpretable, and *merge* deletes the pairs of features that check each other.

[13] The SMC ensures that the licensor will always select the closest licensee (with respect to hierarchical

An expression is *complete* if its only syntactic feature is the special symbol $C$. The language generated by $G$ consists of the complete expressions in $closure(Lex, \mathbb{M})$. A *derivation* is defined as a sequence of expressions produced by recursive application of $\mathbb{M}$ to a collection of lexical chains, and a *complete* derivation is one in which the concluding expression is *complete*. To parse a sentence, a set of lexical items is selected from the lexicon and combined together, via the recursive application of *merge*, into a single structure in which all of the selectional and licensing features have been consumed; if the ordering of the phonological forms in the resulting structure aligns with the order of the words in the sentence being parsed, then the structure is considered to be a valid parse of the sentence. See Fig. 2-2 for an example of an MG derivation derived from the MG lexicon listed in Table 2.1.

Having reviewed the MG formalism, we will now walk through several examples (of progressively increasing complexity) that illustrate the MG formalism. The figures associated with these examples were developed using an agenda-based parser for MGs[14] (extended with Head Movement) that we developed to verify the lexicon and derivations produced by the systems in this thesis – i.e. this parser serves as an external verification of the derivations output by the SMT-model of a minimalist parser developed in this chapter; see Fig. 2-3 for further details of this parser.

**Example 1** This example illustrates how the determiner phrase:

(1) the ball

is derived from the following MG lexicon:

(2) Example-Lexicon-1
  a. the :: $=NP$, $\sim DP$
  b. ball :: $\sim NP$

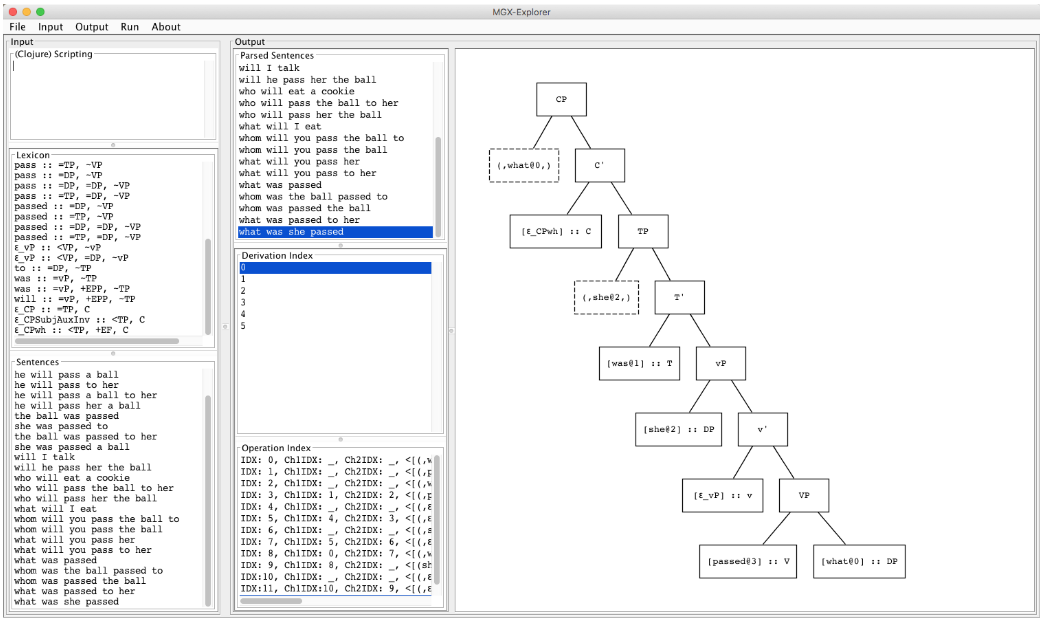
There are two lexical entries in (2), one corresponding to the word "the" and one corresponding to the word "ball"; each lexical entry consists of a word paired with a finite, ordered sequence of (syntactic) features, with the symbol "::" separating the phonological form on the left from the list of syntactic features on the right. There are two types of *selectional* features that appear in this lexicon: *selectors*, indicated by the prefix of =, and *selectees*, indicated by the prefix of ∼. The lexical item for "the" has two syntactic features: the first feature is $=NP$ and is referred to as *selector* $NP$; the second feature is $\sim DP$ and is referred to as *selectee* $DP$. The lexical item for "ball" has a single syntactic feature $\sim NP$ that is referred to as *selectee* $NP$. Let us now step through the derivation of (1):

$$[_{DP}\ [_{D}\ \text{the}]\ [_{NP}\ \text{ball}]]$$

After the lexical entries for "the" and "ball" are both projected from the lexicon, because they have appropriately matching *selectional* features (i.e. because the *selector* feature $=NP$ in the lexical item for "the" matches the *selectee* feature $\sim NP$ in the lexical item for "ball"),

distance), as at each point in the derivation, there can only be one possible licensee available to be licensed; this has the side-effect of making internal merge strictly deterministic (with respect to which licensee a licensor will license), so that a derivation can be determined entirely from knowledge of the order in which the various lexical heads (and projections thereof) *externally* merge with one another.

[14] See the work of (Harkema, 2001) and (Niyogi and Berwick, 2005).



Lexicon

```
pass :: =TP, ~VP
pass :: =DP, ~VP
pass :: =DP, =DP, ~VP
pass :: =TP, =DP, ~VP
passed :: =DP, ~VP
passed :: =TP, ~VP
passed :: =DP, =DP, ~VP
passed :: =TP, =DP, ~VP
ε_vP :: <VP, ~vP
ε_vP :: <VP, =DP, ~vP
to :: =DP, ~TP
was :: =vP, ~TP
was :: =vP, +EPP, ~TP
will :: =vP, +EPP, ~TP
ε_CP :: =TP, C
ε_CPSubjAuxInv :: <TP, C
ε_CPwh :: <TP, +EF, C
```

Sentences

```
he will pass a ball
he will pass to her
he will pass a ball to her
he will pass her a ball
the ball was passed
she was passed to
the ball was passed to her
she was passed a ball
will I talk
will he pass her the ball
who will eat a cookie
who will pass the ball to her
who will pass her the ball
what will I eat
whom will you pass the ball to
whom will you pass the ball
what will you pass her
what will you pass to her
what was passed
whom was the ball passed to
whom was passed the ball
what was passed to her
what was she passed
```

Parsed Sentences

```
will I talk
will he pass her the ball
who will eat a cookie
who will pass the ball to her
who will pass her the ball
what will I eat
whom will you pass the ball to
whom will you pass the ball
what will you pass her
what will you pass to her
what was passed
whom was the ball passed to
whom was passed the ball
what was passed to her
what was she passed
```

Derivation Index

```
0
1
2
3
4
5
```

Operation Index

```
IDX: 0, Ch1IDX: _, Ch2IDX: _, <[(,w
IDX: 1, Ch1IDX: _, Ch2IDX: _, <[(,p
IDX: 2, Ch1IDX: _, Ch2IDX: _, <[(,w
IDX: 3, Ch1IDX: 1, Ch2IDX: 2, <[(,p
IDX: 4, Ch1IDX: _, Ch2IDX: _, <[(,ε
IDX: 5, Ch1IDX: 4, Ch2IDX: 3, <[(,ε
IDX: 6, Ch1IDX: _, Ch2IDX: _, <[(,s
IDX: 7, Ch1IDX: 5, Ch2IDX: 6, <[(,ε
IDX: 8, Ch1IDX: 0, Ch2IDX: 7, <[(,w
IDX: 9, Ch1IDX: 8, Ch2IDX: _, <[(sh
IDX:10, Ch1IDX: _, Ch2IDX: _, <[(,ε
IDX:11, Ch1IDX:10, Ch2IDX: 9, <[(,ε
```

Figure 2-3: *A screenshot of the MG(X)-Explorer software developed for this study.* In order to study the derivations that may be generated from a given lexicon, we developed the *MG(X)-Explorer* software package. *MG(X)-Explorer* includes: (i) a chart parser for MGs (extended with head movement) based on (Harkema, 2001), (ii) a bottom-up generator for MGs (extended with head movement), and (iii) a graphical user interface allowing the user to visually render and step through the derivations generated by a given lexicon for a specified corpus of sentences. The parser and generator can each handle both SVO and SOV orderings. This study used *MG(X)-Explorer* to verify that a derivation recovered from an interpretation of the SMT-model of a (minimalist) parser can in fact be yielded by the input lexicon.

the two lexical items are merged[15] together, thereby deriving a structure for the determiner phrase "the ball". Note that the determiner "the" is considered to be the most significant constituent of the phrase "the ball" (in accordance with the $DP$-hypothesis) and is thus the *head* of the phrase "the ball"; accordingly, the lexical item for "the" projects (and thus heads the DP "the ball") as dictated by the presence of the *selector* type feature $=NP$, whereas the non-projecting lexical item (for "ball") is marked by the presence of the *selectee* type feature $\sim NP$.[16][17] The feature sequence of the derived structure is taken from whichever of the two constituents is the *head* of the derived structure – in this case, the feature sequence is taken from the lexical item for "the". See Figure-2-4 for the corresponding MG derivation tree that was generated from the lexicon listed in (2).

**Example 2** Consider the following sentence:

(3) She will pass the ball.

[15] Neither of the two lexical items for "the" and "ball" is a subset of the other, so this is an example of *external merge*; the sequence of syntactic features in a lexical item is consumed from left to right and once a feature is consumed it is deleted from the sequence of features.

[16] The lexical item that projects is said to *select* the non-projecting lexical item – in this case, the lexical item for "the" *selects* (via external merge) the lexical item for "ball" and thus the lexical item for "the" *projects* through the (external) merge operation.

[17] See (Adger, 2003, Pg. 92): *"Headedness: The item that projects is the item that selects."*

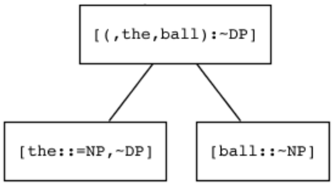


Figure 2-4: The MG derivation for Example-1. Within the MG formalism, given the lexicon listed in (2), the parser would assign (1) this derivation. For now the reader may safely interpret the 3-tuple structure of the phonological form by concatenating the three entries. (This 3-tuple corresponds to the specifier, head and complement entries traditional to XBar theory) The external merge operation checks and then deletes the selectional features; the structure resulting from merge consists of the bare phrase structure for "the ball" paired with the remnant feature $\sim NP$ from "the", which will enable the structure to be (externally) merged with another structure seeking a determiner phrase.

The transitive verb "pass" has two arguments: an *external argument*, "She", that corresponds to the subject of the sentence, and an *internal argument*, "the ball", that corresponds to the object of the sentence. Let us step through the following derivation of (3) and observe how this predicate-argument structure is modeled:

$$[_{CP}\ [_{C}\ \emptyset]\ [_{TP}\ [_{PRN}\ \text{She}]\ [_{T'}\ [_{T}\ \text{will}]\ [_{vP}\ t_i\ [_{v'}\ [_{v}\ \text{pass}+\emptyset]\ [_{VP}\ [_{V}\ t_j]\ [_{DP}\ \text{the ball}]]]]]]]$$

Let us examine this derivation. The lexical entry for the transitive verb "pass" is projected from the lexicon and merges with the determiner phrase "the ball" (which was built up in Example-1) to produce a verb phrase headed by "pass." The lexical entry for little-$v$ (i.e. a covert light verb) is then projected from the lexicon; it first merges with the verb phrase headed by "pass" and then triggers $V$-to-$v$ head-movement (denoted by the dotted movement arrow), before merging with the lexical entry for "she".[18] Next, the lexical entry for "will" is projected from the lexicon and merges with the phrase headed covert light verb. The subject "she" is then raised via phrasal-movement (i.e. *Internal Merge*), denoted by the solid movement arrow; this is referred to as *A-movement*. Finally, a $\emptyset$-complementizer (see $\epsilon_{CP}$ in (4) below) is projected from the lexicon and merges with the phrase headed by "will" to produce a complete derivation. The corresponding MG derivation tree (see Figure-2-5) is derived from the following lexicon:

(4) Example-Lexicon-2

a. she :: $\sim DP$, $-EPP$
b. will :: $=vP$, $+EPP$, $\sim TP$
c. pass :: $=DP$, $\sim VP$
d. the :: $=NP$, $\sim DP$
e. ball :: $\sim NP$
f. $\epsilon_{vP}$ :: $<=VP$, $=DP$, $\sim vP$
g. $\epsilon_{CP}$ :: $=TP$, C

[18] The structure built up thus far involving the two arguments, the light verb and the lexical verb is referred to as a VP-shell structure; see Example 3 for more details.

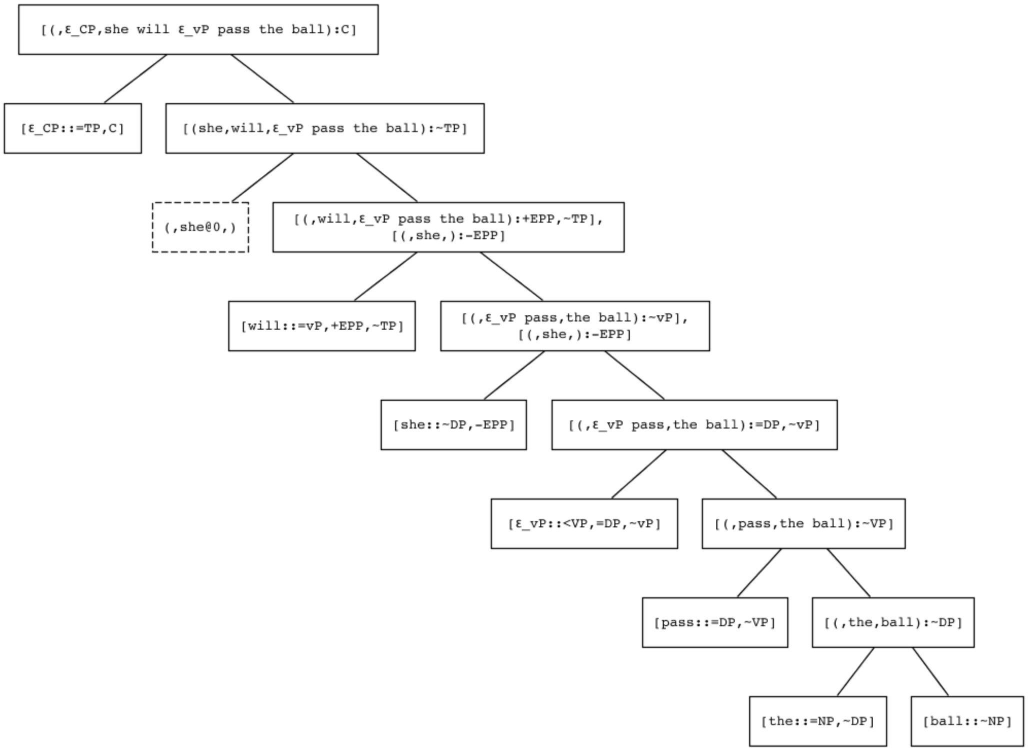


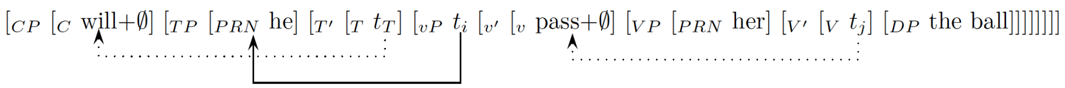
Figure 2-5: A derivation of the expression "she will pass the ball", generated from the lexicon listed in (4). Here we have an example of the transitive verb "pass." It has one external argument, "she", and one internal argument, "the ball". This example includes an instance of phrasal movement, in which the entire phrase "she" is moved, that is known as *A-movement* (i.e. Subject-Raising); the dashed box indicates the landing site of the moved phrase "she". Note that the phrasal movement is an instance of *internal merge* that driven by the need to check the licensing feature, $+EPP$, on the tense marker "will". This example also includes an instance of $V$*-to-*$v$ head movement in which just the head of the verb phrase (i.e. "pass") undergoes movement out of the its projection; note that little-$v$ (i.e. $\epsilon_{CP}$) is covert, as indicated by the $\epsilon$ phonetic form. Finally, in *MG(X)-Explorer*, which was used to produce this image, the special prefix <= that is used to indicate head-movement is shortened to "<".

Two remarks about this lexicon are in order: first, *internal merge* (i.e. syntactic movement) is triggered by licensing – i.e. the need for the licensor feature $+EPP$ in the lexical item for "will" to be checked by the licensee feature $-EPP$ in the lexical item for "she"; second, *head-movement* is triggered when a special type of *selector* feature, denoted by the prefix <=, is checked during an external merge operation – in this example $V$-to-$v$ *head-movement* is triggered by the feature $<=VP$.

**Example 3** This example illustrates a derivation of the interrogative:

(5) Will he pass her the ball?

with the derivation:


$[_{CP}\ [_{C}$ will+∅] $[_{TP}\ [_{PRN}$ he] $[_{T'}\ [_{T}\ t_T]\ [_{vP}\ t_i\ [_{v'}\ [_{v}$ pass+∅] $[_{VP}\ [_{PRN}$ her] $[_{V'}\ [_{V}\ t_j]\ [_{DP}$ the ball]]]]]]]

Let us examine this derivation. The ditransitive form of the predicate "pass" has a single external argument, "she" and *two* internal arguments – "her" and "the ball." The phrasal-structure involving a covert light-verb (i.e. $\epsilon_{vP}$) that merges with the (lexical) verb ("pass") is referred to as the (Double) VP-Shell structure.[19] (See Fig. 2-13 for details on how this VP-shell structure is able to consistently assign thematic roles to the external and internal arguments of the predicate across a wide variety of syntactic configurations.) Once the VP-shell structure has been constructed, "will" is projected from the lexicon and merges with it, after which *A-movement* is triggered, thus raising the subject "he" to its final position in the derivation (i.e. the specifier position in the projection of the lexical head "will"). Finally, a lexical entry for a ∅-complementizer (see $\epsilon_{CP-SubjAuxInv}$ in (6)) is projected from the lexicon and merges with the tense phrase, triggering subject-auxiliary verb inversion via *head-movement* (see the $<=TP$ feature in (6-h)). The associated MG derivation tree in Figure–2-6 may be derived from the following lexicon:

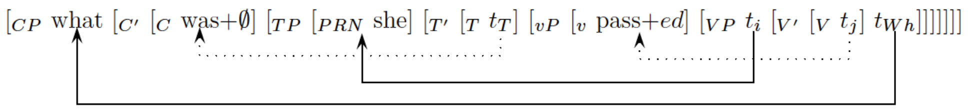
(6) Example-Lexicon-3

a. he :: $\sim DP$, $-EPP$
b. will :: $=vP$, $+EPP$, $\sim TP$
c. pass :: $=DP$, $=DP$, $\sim VP$
d. her :: $\sim DP$
e. the :: $=NP$, $\sim DP$
f. ball :: $\sim NP$
g. $\epsilon_{vP}$ :: $<=VP$, $=DP$, $\sim vP$
h. $\epsilon_{CP-SubjAuxInv}$ :: $<=TP$, C

**Example 4** In this example we will consider a passive-voice wh-question:

(7) What was she passed?

with the following derivation:


$[_{CP}$ what $[_{C'}\ [_{C}$ was+∅] $[_{TP}\ [_{PRN}$ she] $[_{T'}\ [_{T}\ t_T]\ [_{vP}\ [_{v}$ pass+*ed*] $[_{VP}\ t_i\ [_{V'}\ [_{V}\ t_j]\ t_{Wh}]]]]]]]$

The ditransitive verb ("passed") has two *internal* arguments, both of which undergo *phrasal-movement* once the VP-shell structure has been constructed: **(i)** after the VP-shell structure merges with the tense marker "will", the argument "she" undergoes *A-movement* and is raised to Spec-TP; **(ii)** once the Tense phrase is constructed in (i), a ∅-complementizer projects from the lexicon, merges with the Tense phrase, triggers subject-auxiliary verb inversion via *V-to-v head-movement*, and then finally triggers *Wh-movement* (i.e. raising to Spec-CP) of the argument "what" to form a Wh-question.[20] Note that the two syntactic dependencies

[19] For more details see (Hale and Keyser, 2002).
[20] Wh-fronting is triggered by the need to check an *edge feature (e.g. +EF)*.

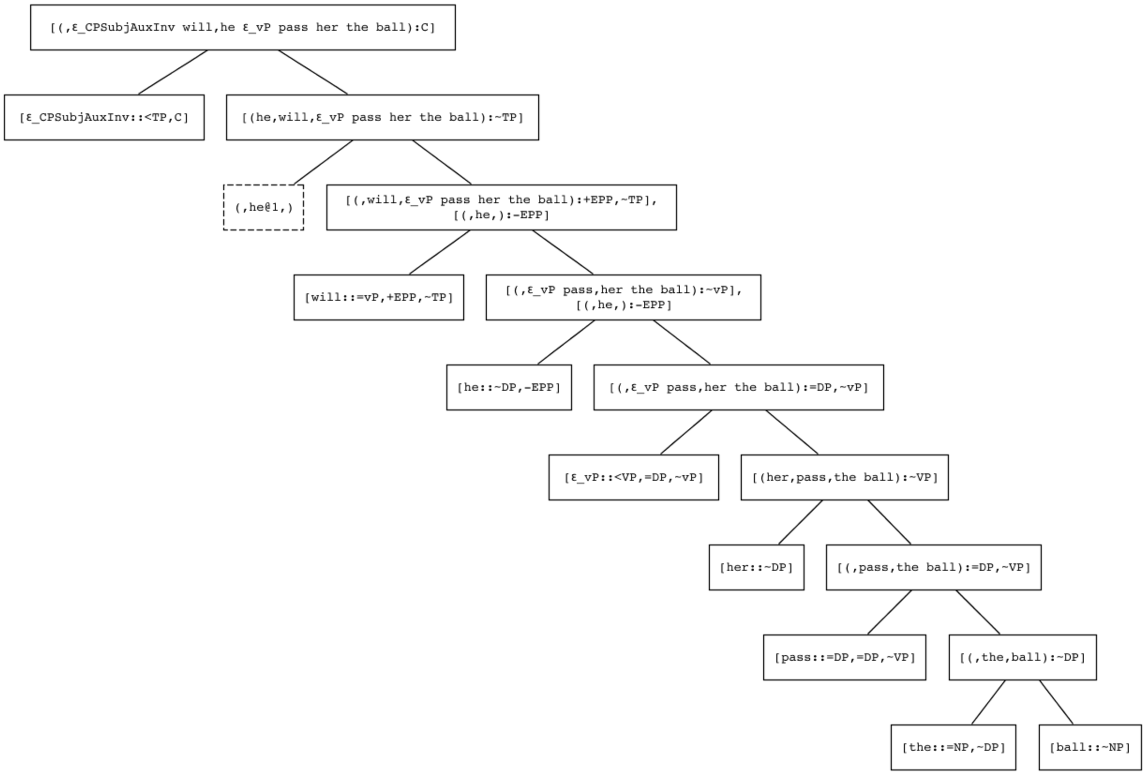

[(,ε_CPSubjAuxInv will,he ε_vP pass her the ball):C]
[ε_CPSubjAuxInv::<TP,C]
[(he,will,ε_vP pass her the ball):~TP]
(,he@1,)
[(,will,ε_vP pass her the ball):+EPP,~TP], [(,he,):-EPP]
[will::=vP,+EPP,~TP]
[(,ε_vP pass,her the ball):~vP], [(,he,):-EPP]
[he::~DP,-EPP]
[(,ε_vP pass,her the ball):=DP,~vP]
[ε_vP::<VP,=DP,~vP]
[(her,pass,the ball):~VP]
[her::~DP]
[(,pass,the ball):=DP,~VP]
[pass::=DP,=DP,~VP]
[(,the,ball):~DP]
[the::=NP,~DP]
[ball::~NP]

Figure 2-6: A derivation of the interrogative expression "Will he pass her the ball?" that was yielded by from the lexicon listed in (6). Here we have an example of a ditransitive instance of the verb "pass", which has one external argument, "she", and two internal arguments, "her" and "the ball"; these three arguments are merged into a predicate-argument complex known as the VP-shell structure (i.e. the light verb $\epsilon_{vP}$ and the "pass"), where they are assigned thematic roles, before possibly undergoing movement. As in Example 2, this derivation has an instance of both *A-movement* and *V-to-v head-movement*. However, this derivation also has an instance of *T-to-C* head-movement for subject auxiliary verb inversion. As in Fig. 2-5, the (selector feature) prefix that indicates head-movement has been shortened from <= to <.

captured by *phrasal-movement* are *nested* rather than *crossed*; that this is the case is due to the *Shortest Movement Constraint*. The associated MG derivation tree in Figure–2-7 may be derived from the following lexicon:

(8) Example-Lexicon-4

a. what :: $\sim DP$, $-EF$
b. was :: $=vP$, $+EPP$, $\sim TP$
c. she :: $\sim DP$, $-EPP$
d. passed :: $=DP$, $=DP$, $\sim VP$
e. $\epsilon_{vP}$ :: $<=VP$, $\sim vP$
f. $\epsilon_{CP-Wh}$ :: $<=TP$, $+EF$, C

Finally, let us consider whether an MG may be modeled with SMT. (Rogers and Nordlinger,

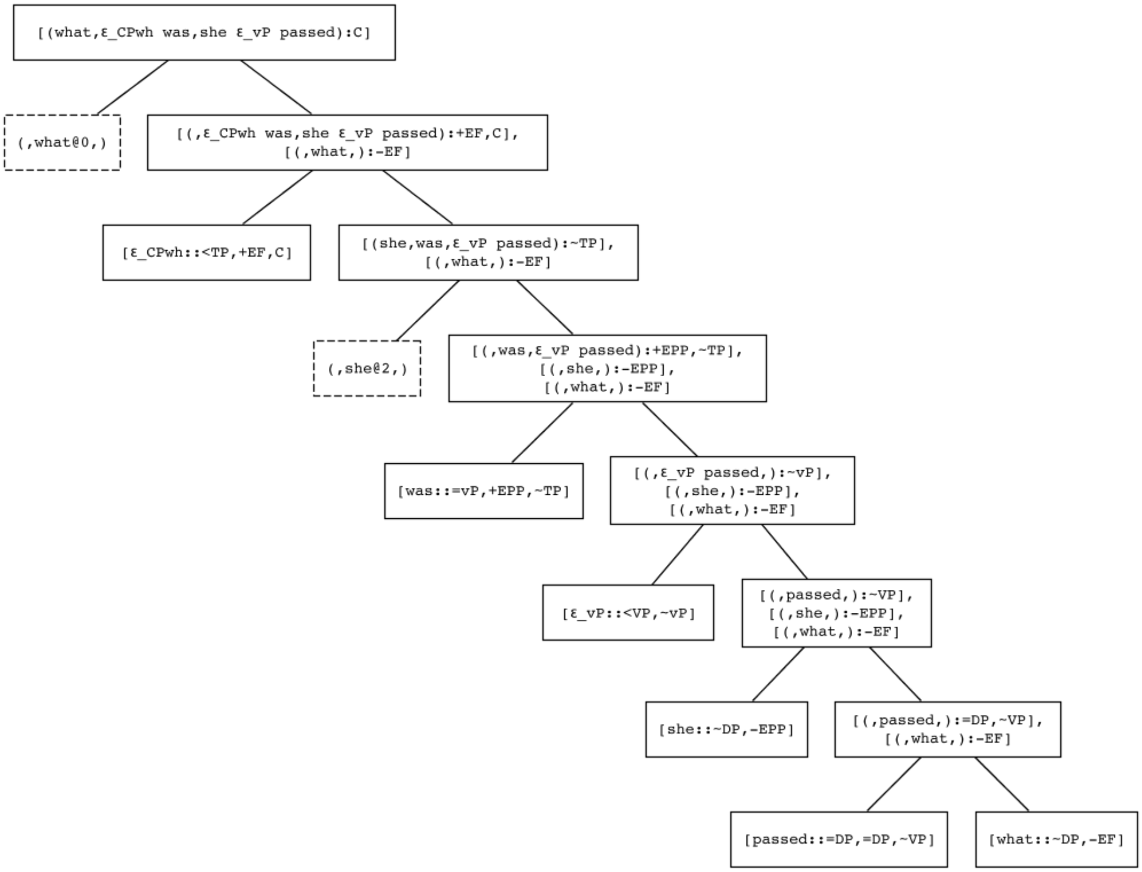

[(what,ε_CPwh was,she ε_vP passed):C]
(,what@0,)
[(,ε_CPwh was,she ε_vP passed):+EF,C], [(,what,):-EF]
[ε_CPwh::<TP,+EF,C]
[(she,was,ε_vP passed):~TP], [(,what,):-EF]
(,she@2,)
[(,was,ε_vP passed):+EPP,~TP], [(,she,):-EPP], [(,what,):-EF]
[was::=vP,+EPP,~TP]
[(,ε_vP passed,):~vP], [(,she,):-EPP], [(,what,):-EF]
[ε_vP::<VP,~vP]
[(,passed,):~VP], [(,she,):-EPP], [(,what,):-EF]
[she::~DP,-EPP]
[(,passed,):=DP,~VP], [(,what,):-EF]
[passed::=DP,=DP,~VP]
[what::~DP,-EF]

Figure 2-7: A derivation of the (passive-voice) Wh-question "What was she passed?" that was yielded by from the lexicon listed in (8). In this derivation, the ditransitive verb "pass" has no external argument but does have two internal arguments, "what" and "she"; these two arguments are merged into the VP-shell structure and assigned their respective thematic roles after which "she" undergoes *A-movement* (i.e. subject raising) and "what" undergoes *Wh-movement*. Note that *Wh-movement* is an instance of phrasal movement that is driven by the need to check the licensing feature, $+EF$, on the complementizer $\epsilon_{CP}$. This example is of particular interest as it involves two instances of phrasal movement that overlap in the course of the derivation.

1998) established that the axioms of GB theory are expressible with Monadic Second Order logic (MSO); subsequently, (Graf, 2013) produced an MSO axiomatization for MGs[21], and notes that over finite domains these constraints may be expressed with first order logic. As this study only considers models with finite domains, we therefore *can* develop a finite theory of MGs with an axiomatization based in part on the MSO axiomatization of MGs by (Graf, 2013).

[21] Constraints may be encoded in an MG lexicon if and only if they are MSO expressible. (Graf, 2013)

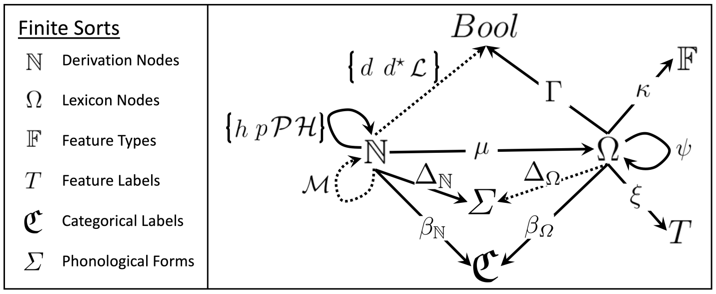


Figure 2-8: Model Formula Diagram for the SMT-Model of the Minimalist Parser. This diagram illustrates how the finite sorts and the uninterpreted functions that make up the model are connected. The nodes represent (finite) sorts (see Table 2.2) and the arrows represent uninterpreted functions (see Table 2.3), with solid arrows representing unary uninterpreted functions (i.e. $h$, $p$, $\mathcal{P}$, $\mathcal{H}$, $\Delta_{\mathbb{N}}$, and $\beta_{\mathbb{N}}$ for the derivation; $\psi$, $\Delta_{\Omega}$, $\beta_{\Omega}$, $\xi$, and $\kappa$ for the lexicon), and dotted arrows indicating a binary function (i.e. $\mathcal{M}$, $d$, and $d^{\star}$ for derivation model; $\mathcal{L}$ for the lexicon model). The derivation model is connected to the lexicon model via the "bus" function, $\mu$, that maps members of the derivation node sort, $\mathbb{N}$, to members of the lexicon node sort, $\Omega$; in particular, $\mu$ maps derivation node sequences (i.e. projections and chains) to lexical feature sequences.

## 2.3 Model Definition

This section defines an SMT-model of a parser for minimalist syntax that is expressed using a multi-sort, quantifier-free, first-order logic extended with the theory of equality and uninterpreted functions. The model centers on an axiomatization of (a fragment of) minimalist syntax that closely follows the MG formalism; whenever applicable, the model axioms are grounded in universal linguistic principles that govern minimalist syntax.[22]

This section will introduce and detail the finite sorts, uninterpreted functions, and axioms (i.e. SMT-formulae) that make up the definition of each of these component models. Summaries of the sorts and uninterpreted functions that make up the model are provided in Table 2.2 and Table 2.3 respectively[23]. Additionally, the reader may find it helpful to consult Fig. 2-8, which is a map of the model, showing how the various finite sorts and uninterpreted functions are connected and organized. Finally, the reader is strongly encouraged to carefully review the model architecture diagrams presented over the course of this section – i.e. Fig 2-14, Fig 2-10, Fig 2-11, and Fig 2-14; these diagrams collectively serve to ground the manifold definitions presented in this section by illustrating how they work together to model a minimalist lexicon and derivation.

### 2.3.1 The Lexicon Model

This subsection introduces and details an SMT-model of a minimalist lexicon. A minimalist lexicon is taken to consist of a finite set of lexical entries, each of which is modeled as a 3-tuple that consists of: (i) a finite sequence of syntactic features, which we will frequently refer to as a *"lexical feature sequences"*; (ii) a syntactic category; (iii) a phonological form. The SMT-

[22] This work is informed by earlier work axiomatizing minimalist grammars by (Graf, 2013) and the formalization of minimalist syntax by (Collins and Stabler, 2016).

[23] A full, documented presentation of the axioms can be found in the reference implementation of the model.

| Sort | Description |
|---|---|
| $\mathbb{N}$ | Nodes in a derivation – i.e. each node corresponds to a position in a projection in the derivation. |
| $\Omega$ | Nodes in a lexicon – i.e. each node corresponds to a location in a lexical entry in the lexicon. |
| $\mathbb{F}$ | Set of syntactic feature *labels* – e.g. $\{x, y, p, q, r\}$. |
| $T$ | Set of syntactic feature *types*: selector (=), selectee ($\sim$), licensor (+) , licensee ($-$), $C$. |
| $\mathfrak{C}$ | Set of categories; the functional categories are C, T, v, V and the lexical categories are P, D, N. |
| $\Sigma$ | Set of phonological forms – i.e. the union of the set of tokens in the expressions to be parsed. |
| *Bool* | The boolean values true and false. |

Table 2.2: Finite sorts and descriptions of the roles they play in the SMT-model of the minimalist parser. The sizes of the sorts are either determined from the model parameters supplied by the user or are derived from the specified interface conditions. (see §2.4.1)

model of the lexicon consists of sorts, uninterpreted functions, bounding parameters, and a set of axioms that, taken together, constitute a decision problem that is satisfiable by any minimalist lexicon that fits within the specified bounds. Looking ahead, §2.3.4 details how the model of the lexicon is connected to the model of the derivation, and §2.4.1 shows how a specified input lexicon can be translated into first order logical formulas that constrain what values the uninterpreted functions in the model can take on, thereby encoding the specified lexicon into the model.[24]

The remainder of this subsection is organized as follows. We will begin by detailing the form of the model, which follows the MG formalism closely, and situate it within contemporary theories of minimalist syntax. We will then introduce the (finite) sorts that model the component parts (e.g. features, lexical entries, phonological forms) from which the lexicon is constructed. Finally, we will introduce the uninterpreted functions that model the relations between the component parts that make up the lexicon, and the axioms that constrain what interpretations each of these functions may take on. A summary of the sorts and functions listed in this section is provided in Table 2.2 and Table 2.3 respectively; additionally, the reader may wish to review Figure-2-8 and reference it while reading this section.

### Model Form

Per minimalist theories of syntax, a lexicon is a set of atomic syntactic structures that encodes information required by the computational system ($C_{HLF}$) to assemble complex

[24] I.e. it is possible to hard-code the model values so that the interpretation of the lexicon model aligns exactly with the specified lexicon; in the next chapter we will see why it is sometimes desirable, in the context of language acquisition, not to constrain the lexicon model with a specified lexicon as the lexicon is itself what is being solved for.

| Function | Signature | Description |
|---|---|---|
| $h$ | $\mathbb{N} \rightarrow \mathbb{N}$ | Head of a node. |
| $p$ | $\mathbb{N} \rightarrow \mathbb{N}$ | Parent of a node (see the solid black arrows in Fig. 2-2). |
| $d$ | $\mathbb{N} \times \mathbb{N} \rightarrow Bool$ | Predicate encoding the *derivation* tree that holds if the former constituent (node) dominates the latter constituent (node). |
| $\mathcal{M}$ | $\mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ | Product of merging two nodes in the derivation. |
| $\mathcal{P}$ | $\mathbb{N} \rightarrow \mathbb{N}$ | Target that a node undergoing phrasal movement is raised to (see the dashed grey arrows in Fig. 2-2). |
| $\mathcal{H}$ | $\mathbb{N} \rightarrow \mathbb{N}$ | Destination of a node undergoing head-movement (e.g. the target of the dashed grey arrows in Fig. 2-2). |
| $\beta_{\mathbb{N}}$ | $\mathbb{N} \rightarrow \mathfrak{C}$ | Category associated with a node. |
| $d^{\star}$ | $\mathbb{N} \times \mathbb{N} \rightarrow Bool$ | Predicate encoding the *derived* tree that is true if the former constituent (node) dominates the latter constituent (node). |
| $\mathcal{L}$ | $\mathbb{N} \times \mathbb{N} \rightarrow Bool$ | Predicate encoding precedence relations between constituents in the derived tree; this predicate is true if, after all syntactic movement has taken place, the former constituent precedes the latter constituent (w.r.t. SVO linearization). |
| $\Delta_{\mathbb{N}}$ | $\mathbb{N} \rightarrow \Sigma$ | Phonological form associated with a derivation node (i.e. associates lexical heads with phonological forms). |
| $\mu$ | $\mathbb{N} \rightarrow \Omega$ | Maps derivation nodes to lexicon nodes. |
| $\psi$ | $\Omega \rightarrow \Omega$ | Maps a lexicon node to its successor. |
| $\kappa$ | $\Omega \rightarrow \mathbb{F}$ | Maps a lexicon node to its syntactic feature *label*. |
| $\xi$ | $\Omega \rightarrow T$ | Maps a lexicon node to its syntactic feature *type*. |
| $\Gamma$ | $\Omega \rightarrow Bool$ | Indicates if the argument triggers head-movement. |
| $\beta_{\Omega}$ | $\Omega \rightarrow \mathfrak{C}$ | Associates a lexicon node with a category. |
| $\Delta_{\Omega}$ | $\Sigma \times \Omega \rightarrow Bool$ | Associates a lexicon node with a phonological form. |

Table 2.3: Uninterpreted functions in the SMT-model of the derivation and the SMT-model of the lexicon (separated by a horizontal line), and descriptions of the role they play in the SMT-model of the minimalist parser.

syntactic structures that satisfy interface conditions.[25][26] The members of a lexicon, referred to as *lexical entries*, each consists of a matrix of features, with features defined as per (Adger and Svenonius, 2011):

> *A syntactically relevant property of a syntactic atom which is not shared by all syntactic atoms and which is not derivable from some other property is a feature.*

The features associated with a lexical entry encode the semantic, syntactic and phonetic properties of that lexical entry.[27] The lexical entries in the lexicon encode, with features, *all* of the information that is required to assemble a syntactic structure - this is per the **Inclusiveness Condition**[28], which stipulates that a syntactic structure is exclusively composed of elements of the atomic syntactic structures that enter into the derivation – i.e. the features associated with the atomic syntactic structures entering into a derivation are the only features that may appear in the derivation. Syntactically-relevant features[29] – i.e. those features that play a role in syntactic processes – may be divided into two categories: features that are both involved in syntactic processes and interpreted at the interfaces are referred to as *interpretable features*, with features interpretable at the LF or PF interface referred to as *LF* or *PF interpretable features* respectively; features that are only involved in syntax and not interpreted at the interfaces are referred to as *uninterpretable features.* E.g. categorical features[30] and phi-features of nouns are interpretable whereas case-marking features are uninterpretable.[31][32] The particular role syntactic features play in a derivation is prescribed by the particular theory of minimalist syntax used as well as the formalism employed, but ultimately they must serve to regulate the syntactic structures the lexicon can yield by controlling when merge can be applied, and what information is presented to the interfaces.

Having outlined some of the broader requirements that a lexicon must satisfy per contemporary theories of minimalist syntax, we will next consider how the MG formalism models

[25] See (Chomsky, 1995, Pg. 6): *"For each particular language, the cognitive system, we assume, consists of a computational system CS and a lexicon. The lexicon specifies the elements that CS selects and integrates to form linguistic expressions – (PF, LF) pairings, we assume. The lexicon should provide just the information that is required for CS, without redundancy and in some optimal form, excluding whatever is predictable by principles of UG or properties of the language in question."*

[26] See (Chomsky, 1995, Pg. 239): *"On the simplest assumptions, the lexical entry provides, once and for all, the information required for further computations – in particular for the operations of the phonological component (including morphology, we assume)."*

[27] See (Chomsky, 1995, Pg. 130): *"The lexicon is a set of lexical elements, each an articulated system of features. It must specify, for each such element, the phonetic, semantic, and syntactic properties that are idiosyncratic to it, but nothing more."*

[28] See the Inclusiveness Condition (Chomsky, 1995, Pg. 228 of §4.2.1.): *"A 'perfect language' should meet the condition of inclusiveness: any structure formed by the computation (in particular $\pi$ and $\lambda$) is constituted of elements already present in the lexical items selected for N; no new objects are added in the course of computation apart from rearrangements of lexical properties (in particular, no indices, bar-levels in the sense of X-Bar theory, etc. . . )."*

[29] See (Chomsky et al., 2019): *"But informational notions such as "topic" or "focus," like grammatical functions or thematic roles, are properties of configurations and their syntactic/discursive context, not of individual syntactic objects (Chomsky 1965; Hale & Keyser 1993); consequently, they should neither be represented in the lexicon, nor in the narrow syntactic derivation (cf. Uriagereka 2003; Fortuny 2008; López 2009; Gallego 2013a, 2016)"*

[30] E.g. C, T, v, V, D, and N. See (Chomsky, 2001).

[31] Here we separate phi-features, which encode gender, person, number, etc, from features that encode case. See (Chomsky, 1995)pg. 278.

[32] In English, only pronomials have their case explicitly visible.

a lexicon. The MG formalism models checking theory, in which merge imposes constraints on the relations between corresponding features in the feature matrices of the two merging syntactic structures; in the case of (c-)selection and licensing, the two features must check one another, which requires that the two features to have the same label.[33] In particular, the MG formalism provides syntactic features for both selection and licensing, which drive *external merge* and *internal merge* (i.e. syntactic movement) respectively,[34]. Additionally, the MG formalism has also been extended to include selectional features for triggering head-movement;[35] these syntactic features are strictly *uninterpretable*, and in effect serve as book-keeping variables that serve a syntactic purpose strictly within the derivation, but not exposed to the interfaces.[36]

Although the MG formalism is rather conservative in the types of features it includes, and the kind of feature matrix allowed (i.e. a finite sequence of features), the formalism is powerful enough that an MG lexicon can encode any MSO-expressible (syntactic) constraint (over the derivations trees the lexicon will yield),[37][38], and simple enough that an axiomatization of a minimalist can be written down. The MG formalism is thus a good starting point for modeling a minimalist lexicon, with an understanding that substantive improvements and refinements to the formalism are required to bring it into full alignment with contemporary minimalist syntax. To this end, the model of the lexicon developed in this thesis will closely follow the MG formalism, with several minor modifications that we will now go over. See Fig. 2-14 for a diagram illustrating the architecture of the lexicon model; see Fig 2-10 for a diagram illustrating in further detail how the model represents a single lexical feature sequence.

We will now consider the form of a lexical entry, and how these lexical entries are arranged to form a lexicon. First, all lexical entries in the lexicon have the form:

$$PF/CAT\ ::\ [SFS]$$

here $PF$ is a phonological form (e.g. “dog”), $CAT$ is a categorical variable (e.g. $N$) that is formally interpretable at the LF interface, and $SFS$ is a finite sequence of uninterpretable syntactic features, either selectional or licensing (i.e. just as prescribed by the MG formalism). Whereas a syntactic feature sequence may associate with more than one phonological

[33] The feature system adopted in this thesis is based on checking theory as presented in (Chomsky, 1995), and not the Probe/Goal system developed in (Chomsky, 2001); see (Fong and Ginsburg, 2019) for a presentation of a minimalist parser that faithfully implements the theory of phases.

[34] This is grounded in the **Principle of Last Resort** (Chomsky, 1995, Pgs. 200-201), which asserts that (syntactic) movement is driven triggered by the need to check an uninterpretable feature.

[35] See (Stabler, 2001) and (Stabler and Keenan, 2003).

[36] For details, see (Stabler, 1998, Pg. 11): *“In the first place, Chomsky (1995), Ura (1996), Collins (1997) and others have argued that natural languages can be defined more elegantly with more elaborate feature checking regimes. For example, Chomsky does not assume that syntactic features are disjoint from the interpreted and phonetic ones, suggesting that there are two types of features, the -interpretable features which are eliminated 'at LF', and the +interpretable ones which are not (Chomsky, 1995, pp. 278-279), where the -interpretable syntactic features may have phonological effects. It might also be desirable to allow syntactic operations to check many features in a single step. These schemes are presumably strictly more powerful than the ones proposed in the simple formalism above. In our formal grammars, the features are, in effect, all -interpretable, without phonological consequences, and each operation checks and deletes exactly two features.“*

[37] See (Graf, 2013).

[38] Per (Rogers and Nordlinger, 1998), most (if not all) syntactic constraints are MSO-expressible.

form, every syntactic feature sequence is associated with exactly one categorical variable.[39] Secondly, the lexicon will encode associations between phonological forms and syntactic feature sequences, which we will sometimes refer to as a "lexical feature sequences", by an auxiliary table that pairs the index of a lexical feature sequences with the indices of the phonological forms. This form of representation, which we will often refer to as a "factored representation" of the lexicon, has two benefits: (i) it allows the lexicon to more efficiently encode a larger number of lexical entries without having to maintain copies of lexical feature sequences or phonological forms; (ii) it enables the size of the lexicon to be *optimized* by minimizing (specifically) the number of lexical feature sequences present in the lexicon, without penalizing the number of phonological forms each lexical entry is associated with – this will be extensively detailed in Ch.3.[40]

Having outlined the form of the model of the lexicon, a strategy for developing it thus comes into view. First, we will employ a sort to model the set of lexical entries in the lexicon, and then impose additional structure upon this sort so that its elements may be organized into a finite set of finite sequences (of syntactic features), thereby allowing us to reference sequences of syntactic features. Then we will employ additional finite sorts and functions to represent: (i) the label and type of each syntactic feature (in each lexical feature sequence); (ii) the phonetic forms and categorical variable associated with each lexical feature sequence. Let us now see how this strategy plays out in practice.

### Model Sorts

The model includes several (finite) sorts, each of which describes one of the various types of "things" from which the lexicon is built up – i.e. the labels and types of syntactic features, the lexical feature sequence (i.e. the sequence of syntactic features that constitutes a "feature matrix"), phonetic features, and categorical features.

**The Lexicon Node Sort.** To begin, we will first introduce a sort to model the set of lexical entries in the lexicon, and then impose additional structure upon this sort so that its elements may be organized into a finite set of finite sequences.

Define $\Omega$, referred to as the *lexicon node sort*, to be a finite sort with two distinct members especially labeled: the terminal node, $\omega_{\varnothing}$, and the complete node, $\omega_C$. The complete node serves to code for the special syntactic feature $C$. The terminal node serves to indicate the end of a syntactic feature sequence, and is itself not coding for any particular syntactic feature.[41] With respect to the form of an MG lexicon, each member of $\Omega$, aside from $\omega_C$ and $\omega_{\varnothing}$, is associated (uniquely) with a syntactic feature in the lexicon; this subset, $\Omega - \{\omega_C, \omega_{\varnothing}\}$, is denoted $\Omega^{'}$.

Next we will impose additional substructure upon $\Omega$ to model sequences of features. A *lexicon node sequence* of length $k$ is defined to be a finite sequence of nodes, $x_1, \ldots, x_k \in \Omega$. The set of node sequences in $\Omega$ is denoted $\lambda$. Each member of $\lambda$ corresponds to a lexical feature sequence in the lexicon (with there being a one-to-one correspondence between the sequence of nodes in a lexicon node sequence, and the sequence of features in a lexical

[39] There are far fewer distinct categories than there are distinct phonological forms; furthermore, the number of categories in the model is fixed and independent of the input, whereas the number of phonological forms is not fixed and may grow depending on the input.

[40] As we will see later, there are upper-bounds on the number of phonological forms that any given lexical feature sequence can associate with.

[41] Looking ahead to §2.3.2, the terminal node serves a similar role as $\bot \in \mathbb{N}$, whereas the complete node serves a similar role as the $R_{\mathbb{N}} \in \mathbb{N}$.

feature sequence). Note that $\omega_\varnothing$ and $\omega_C$ do not appear in any of the lexicon node sequences. As a convention, the first node in a lexicon node sequence will sometimes stand in for the entire lexicon node sequence itself; this is necessary because the lexicon node sequences are not explicitly modeled themselves – i.e. there is no "lexicon node sequence sort" in the lexicon model). Instead the lexicon node sequences are implicit in the external indexing of the lexicon node sort (i.e. within the python program that will construct this model, the members of the lexicon node sort are organized as a list of lists).

**Sorts for Syntactic Features.** Each syntactic feature in an MG lexicon has a *type* and a *label*. The *type* of a feature is either *selector*, *selectee*, *licensor*, *licensee* or a special symbol that indicates convergence of the derivation – these types are denoted by the prefixed symbols $=$, $\sim$, $+$, $-$ and $C$ respectively.[42] In the context of parsing (as in this chapter), the *label* of a feature is specified by the (input) lexicon; in the context of acquisition (as in the next chapter), the *label* of a feature is one of a set of automatically enumerated feature labels that is available to the system.[43]

The *label* of a feature is either one that was specified by the user (e.g. in the case of an interface condition requiring that a particular feature appear at a particular point in a derivation) or from the set of automatically enumerated feature labels available to the system (intended for uninterpreted features).[44]

We will now define two sorts, the first of which encodes the *type* of a feature, and the second of which encodes the *label* of the feature. Define $T$, referred to as the *lexicon node type sort*, to be a finite sort consisting of the following members: $\{\tau_t, \tau_c, \tau_=, \tau_\sim, \tau_+, \tau_-, \tau_\varnothing\}$. Define $\mathbb{F}$, referred to as the *syntactic feature sort*, to be a finite sort that is divided into three disjoint subsets: the selectional feature subset, $\mathbb{F}_S$, the licensing feature subset, $\mathbb{F}_L$, and a singleton set consisting of an element, $\varnothing_\mathbb{F}$, that is referred to as the nil syntactic feature. The nil syntactic feature, $\varnothing_\mathbb{F}$, serves to designate that a lexicon node is not associated with a feature type, and plays a role similar to $\omega_\varnothing \in \Omega$; likewise, $\tau_\varnothing$ serves to designate that a lexicon node is not associated with a feature label, and plays a role similar to $\varnothing_\mathbb{F}$. Fig. 2-10 illustrates how these two sorts will be associated with the members of the lexicon node sort (using the model functions that we will introduce a little later on).

**A Sort for Categorical Features** Categories are interpretable properties (i.e. features) of atomic syntactic structures. The model includes a sort to code for categories, including both functional categories (i.e. $\{C, T, v, D, P\}$) and lexical categories (i.e. $\{N, V\}$).[45][46]

[42] There is also a special type of selector that triggers head-movement and is indicated by the prefix <= or >= for left and right head-movement respectively; see (Stabler and Keenan, 2003) for more details.

[43] The number of automatically enumerated feature labels available to the system is specified by the user as a model parameter. Typically more labels than are required are provided to the system as we do not know, apriori, how many such labels the system will require.

[44] The number of automatically enumerated feature labels available to the system is specified by the user. Typically more labels than are required are provided to the system as we do not know, apriori, how many such labels the system will require.

[45] For a discussion of functional vs. lexical categories, see (Adger, 2003, Pg. 165), (Lust, 2006, Pgs. 184-185), and (Chomsky, 1995, Pg. 54); note that contemporary theories of minimalist syntax at times disagree on whether prepositions (P) are functional or lexical categories – this thesis takes them to be functional categories, although whether P is a functional or lexical category does not substantively impact any of the results presented in this thesis.

[46] This study does not deal with adjectives or adverbs, hence the absence of a lexical category "A"; note that this does not preclude the model from being extended to handle adjectives and adverbs, and the choice to ignore them was made for the purposes of simplifying the definition of the model.

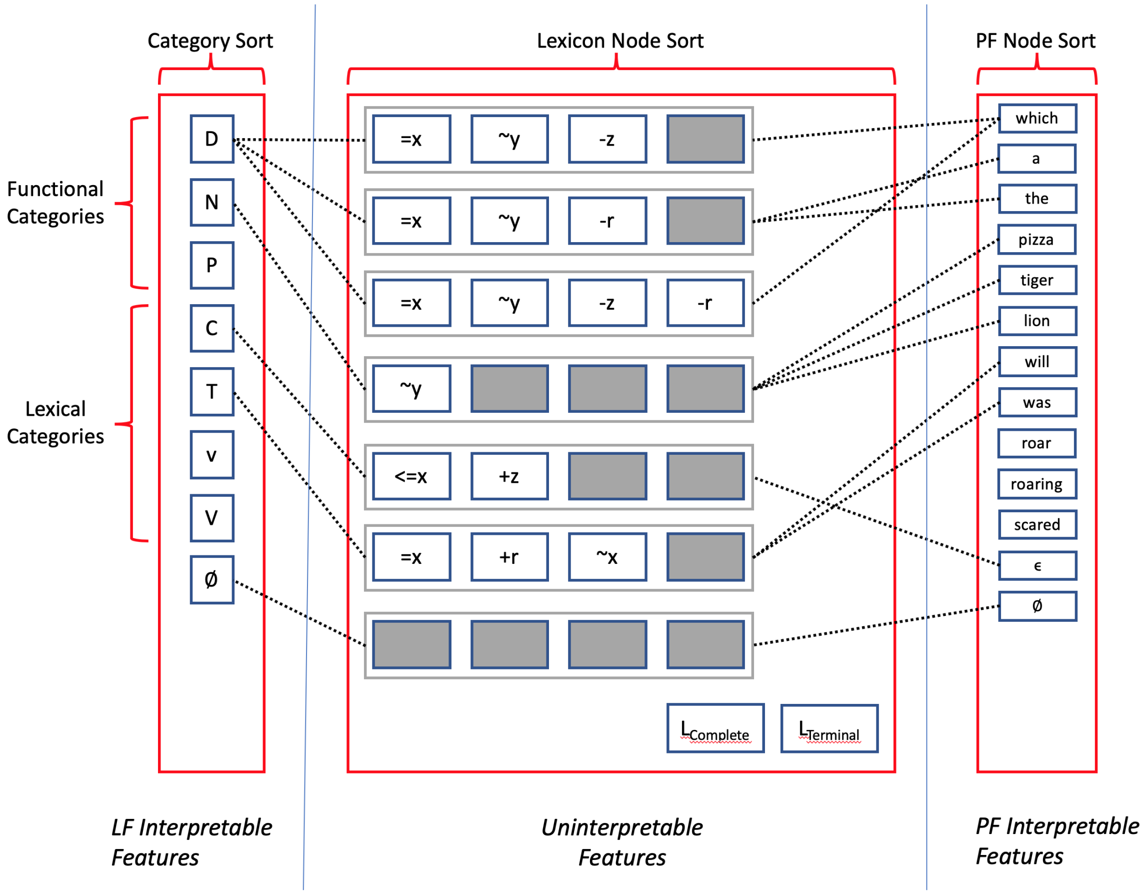


Figure 2-9: Lexicon Model Architecture Diagram. This diagram illustrates how several of the sorts that make up the lexicon model are connected. The left hand side column depicts the members of the category sort (i.e. $\mathfrak{C}$), with each member coding for a unique categorical feature; these features are pre-determined (formally) LF interpretable features that are referenced by some of the axioms for interface conditions presented later in §2.3.3. The right hand side column depicts the members of the PF node sort (i.e. $\Sigma$), with each member coding for a phonetic feature; the phonetic features are (formal) PF interpretable features. The middle column depicts the members of the lexicon node sort (i.e. $\Omega$) (displayed with their associated feature type and label) organized in rows (outlined in thin grey boxes), with each row coding for a lexical feature sequence that consists of uninterpretable (syntactic) features that must be checked and deleted over the course of the derivation; the filled grey boxes are "inactive" lexicon nodes that do not code for any syntactic feature, and the bottom two members (i.e. $L_{Complete}$ and $L_{Terminal}$) code for special members of the lexicon node sort that signify (respectively) (i) the special symbol "C" that marks the end of a derivation, and (ii) the end of a lexical feature sequence. Each lexical feature sequence is associated with exactly one categorical feature (via $\Delta_{\Omega}$), and one or more phonetic features, as indicated by the dotted lines. The "null" sort values in the category sort and PF node sort each serve to indicate "no assignment", and the bottom most syntactic feature sequence properly connects to each of them to indicate that as an inactive lexical feature sequence (i.e. an unused entry in the lexicon), it is thus not affiliated with any phonetic form or categorical feature.

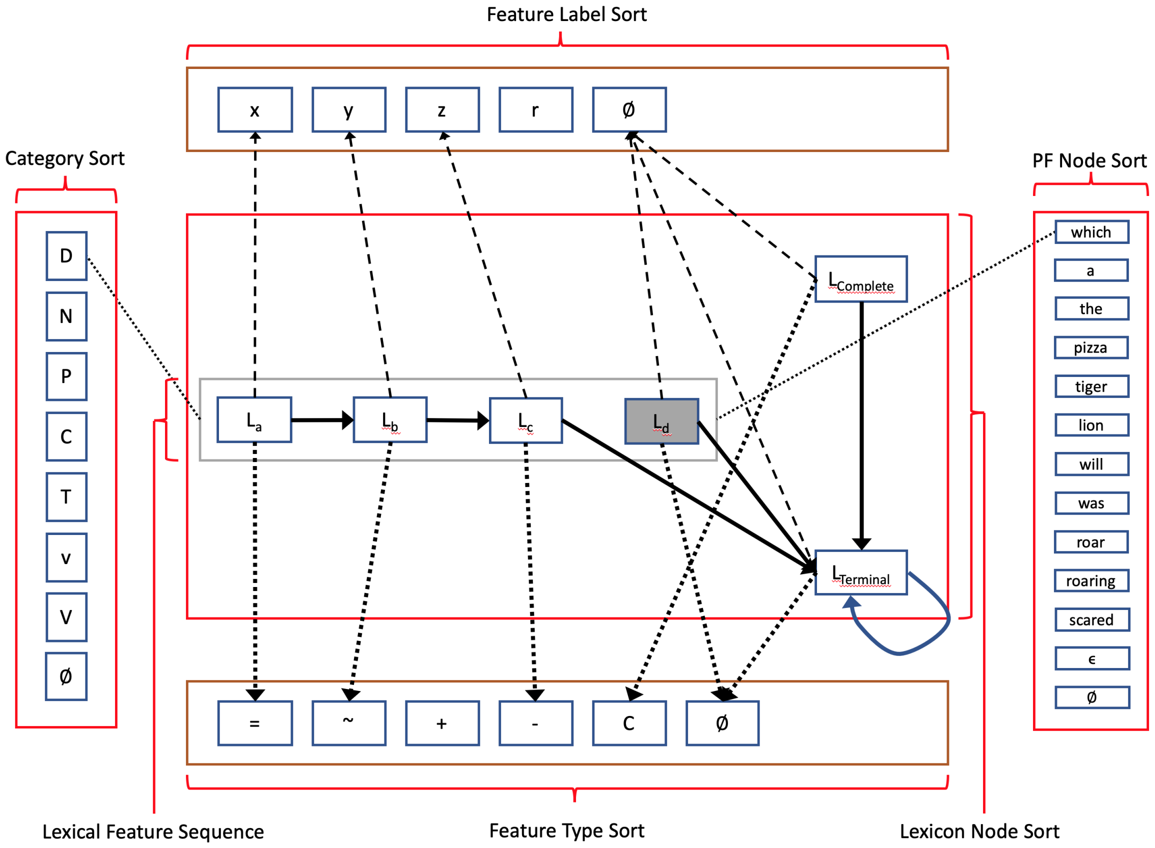


Figure 2-10: Lexical Feature Sequence Diagram. This diagram illustrates how several of the sorts and uninterpreted functions that make up the lexicon model are organized to represent a lexical feature sequence (specifically the top most lexical feature sequence displayed in Fig. 2-14). The solid arrows map each member of the lexicon node sort (i.e. $\Omega$) to its successor (via the map $\psi$). Here, the first three members of the lexical feature sequence appear in succession, followed by the $L_{Terminal}$ node, which serves to indicate the end of the sequence (the last entry in the lexical feature sequence is allocated but not used, as indicated by it being greyed out). The dashed arrows map (via $\xi$) members of the lexicon node sort that make up a lexical feature sequence to their respective feature labels (i.e. members of $\mathbb{F}$). Likewise, the dotted arrows map (via $\kappa$) entries in the lexical feature sequence to their respective feature type (i.e. members of $T$). Members of the lexicon node sort that are not associated with any feature in particular are mapped to the $\emptyset$ feature type and label respectively (e.g. $L_d$ and $L_{Terminal}$).

Define $\mathfrak{C}$, referred to as the *category sort*, to be a finite sort consisting of the following members:

$$\mathfrak{C} = \{\mathfrak{c}_{C_{Decl.}}, \mathfrak{c}_{C_{Ques.}}, \mathfrak{c}_T, \mathfrak{c}_v, \mathfrak{c}_V, \mathfrak{c}_P, \mathfrak{c}_D, \mathfrak{c}_N, \mathfrak{c}_\varnothing\} \tag{2.10}$$

Each member of $\mathfrak{C}$ is indexed by either the category marked in its subscript, or for $\varnothing$ in the case of the singleton member that serves to designate no category association.

**A Sort for Phonetic Features** Define $\Sigma$, referred to as the "PF node sort", to be a finite sort that is the union of the three disjoint subsets: $\Sigma_o$, which code for overt phonetic features, the singleton set, $\Sigma_c$, that contains the covert phonetic feature $\epsilon$, and the singleton set, $\{\varnothing_\Sigma\}$, that codes for the null node (standing to represent "no phonetic feature"). Each element of $\Sigma_o$ is indexed by the unique word it codes for; the set of words $\Sigma_o$ codes for must be supplied when the sort is initialized. Looking ahead, we will later go over this in more detail with a specific example in §2.4.1.

**Bounding the Model.** The system does not know, a-priori: (i) which lexical entries (or even how many) are in the lexicon; (ii) how many features each lexical entry has. Thus, upper-bounds must be supplied for each of these unknown quantities; given these upper bounds, it is then possible to compute a bound on the size of $\Omega$. Although these bounds can be determined by examining the lexicon that will be input into the parser (as will be done in §2.4.1), in the next chapter on acquisition, there will often be no (input) lexicon specified and these upper-bounds will have to be supplied by the user as model parameters.

We can compute the upper bound on the cardinality of $\Omega$ as follows. Let $n_L$ be the maximum number of lexical items, $q$ be the maximum length of a lexical feature sequence. Then the size of $|\Omega|$ is bounded above by:

$$|\Omega| \leq 2 + n_L q \tag{2.11}$$

Note that the 2 appearing in (2.11) is included to account for the null node and the complete node.

Next, we can compute an upper-bound on the cardinality of the set of selectional and licensing feature labels as follows. Let $s$ be the maximum number of selectional features, and let $l$ be the maximum number of licensing features. Then the size of $\mathbb{F}$ is bounded above by:

$$|\mathbb{F}| \leq |\{\varnothing_\mathbb{F}\} \cup \mathbb{F}_S \cup \mathbb{F}_L| = 1 + s + l \tag{2.12}$$

The labels of selectional and licensing features, if not explicitly supplied by the user as model parameters, are automatically generated by the system to be $x_1, x_2, \ldots, x_s$ and $y_1, y_2, \ldots, y_l$ respectively.

### Model Axioms

The model includes both uninterpreted functions that map the model sorts to one another, and model axioms that constrain interpretations of these functions; consequently, the axioms collectively constrain the space of satisfiable interpretations of the lexicon model as a whole. Fig. 2-14 and Fig. 2-10 illustrate the relations that the (uninterpreted) functions to be introduced will establish between the members of the lexicon model sorts; intuitively, the model axioms will serve to restrict what relations may be established. Looking ahead, the task of the SMT-solver will be to identify *satisfiable* interpretations of the uninterpreted functions that make up the lexicon model – i.e. valuations of the uninterpreted functions that accord with the axioms. See Table 2.6 for a detailed example of a satisfiable interpretation of the model of the minimalist parser (which this lexicon model is a component of). Finally, before proceeding, a remark concerning notation is in order – namely, that when "(Ax.**??**)" shows up at the end of a sentence, the reader should understand the sentence to be explaining the referenced axiom – this notation will be used throughout the remainder of this chapter.

**Axioms for Syntactic Features.** To begin, we will introduce two uninterpreted (unary) functions that associate members of the lexicon node sort with feature labels and feature types.

Given a member of the lexicon node sort $x \in \Omega$, $\kappa(x)$ is the *label* of the syntactic feature assigned to $x$, where $\kappa$ is defined to be an uninterpreted function with signature $\Omega \rightarrow \mathbb{F}$. Likewise, given $x \in \Omega$, $\xi(x)$ is the *type* of the syntactic feature assigned to $x$, where $\xi$ is defined to be an uninterpreted function with signature $\Omega \rightarrow T$. Several axioms constrain interpretations of $\kappa$ and $\xi$.[47]

$$\kappa(\omega_{\varnothing}) = \kappa(\omega_C) = \varnothing_{\mathbb{F}} \tag{2.13}$$

$$\xi(\omega_C) = \tau_c \tag{2.14}$$

$$\xi(\omega_{\varnothing}) = \tau_t \tag{2.15}$$

$$\bigwedge \left\{ \langle \xi(x), \tau_c, \tau_t \rangle \mid x \in \Omega^{'} \right\} \tag{2.16}$$

The terminal node does not have an associated syntactic feature and the complete node (which associates with the special feature $C$ in the MG formalism) associates with a special syntactic feature that has a type but not a label (Ax.2.13). The feature types $\tau_c$ and $\tau_t$ are restricted to the complete node and the terminal node respectively (Ax.2.14, Ax.2.15); other nodes (i.e. $\Omega^{'}$) may not associate with these two feature types (Ax.2.16).

Given $x \in \Omega$, $\Gamma(x)$ indicates whether or not a node triggers head-movement[48], where $\Gamma$ is defined to be an uninterpreted function with signature $\Omega \rightarrow$ *Boolean*. Several axioms constrain interpretations of $\Gamma$:

$$\Gamma(x) \rightarrow \xi(x) = \tau_{=} \tag{2.17}$$

$$\bigwedge \{ (\kappa(x) = \varnothing_{\mathbb{F}}) \leftrightarrow (\xi(x) \in \{\tau_{\varnothing}, \tau_c, \tau_t\}) \mid x_i \in s, s \in \lambda \} \tag{2.18}$$

$$\bigwedge \{ (\xi(x) \in \{\tau_{+}, \tau_{-}\}) \rightarrow \kappa(x) \neq f \mid x_i \in s, s \in \lambda, f \in \mathbb{F}_S \} \tag{2.19}$$

$$\bigwedge \{ (\xi(x) \in \{\tau_{=}, \tau_{\sim}\}) \rightarrow \kappa(x) \neq f \mid x_i \in s, s \in \lambda, f \in \mathbb{F}_L \} \tag{2.20}$$

Only a feature of type *selector* can trigger head movement (Ax.2.17). Nodes involved in licensing cannot have selectional features (Ax.2.19) and likewise nodes involved in selection cannot have licensing features (Ax.2.20). Finally, a node in a lexicon node sequence associates with nil feature if and only if the node is not involved in selection or licensing

[47] Note that for some of the axioms presented below, we will sometimes evaluate set-membership with respect to a finite set – in these cases it should be understood that such an expression can be replaced (and is within the reference implementation) with a disjunction of equality against each member of the finite set in question; see (Ax.2.18) for an example of an axiom in which this replacement is applicable.

[48] This predicate serves the purpose of allowing us to sub-type the selector feature type.

(Ax.2.18).

**Axioms for Lexical Feature Sequences.** Next, we will introduce an uninterpreted function that will impose sequencing relations on the nodes within each lexicon node sequence. Given $x \in \Omega$, $\psi(x)$ is the *successor* of $x$, where $\psi$ is defined to be an uninterpreted function with signature $\Omega \rightarrow \Omega$. Several axioms constrain interpretations of $\psi$:

$$\bigwedge \{\psi(x) \neq y \mid x \in s_i, y \in s_j, i \neq j, s_i, s_j \in \lambda^2\} \tag{2.21}$$

$$\bigwedge \{\psi(x_i) \in \{\omega_\varnothing, x_{i+1}, \omega_C\} \mid x_i \in s, s \in \lambda, i < k\} \tag{2.22}$$

$$\bigwedge \{\psi(x) \in \{\omega_\varnothing, \omega_C\} \mid x_i \in s, s \in \lambda, i = k\} \tag{2.23}$$

$$\psi(\omega_C) = \psi(\omega_\varnothing) = \omega_\varnothing \tag{2.24}$$

A node cannot succeed a node from another lexicon node sequence (Ax.2.21). Every node but the last in a node sequence has the next node in the sequence as its successor (Ax.2.22, Ax.2.23). The terminal node is the successor of both the complete node and the terminal node (Ax.2.24).

$$\bigwedge \{(y = \psi(x)) \rightarrow (\xi(y) \neq \tau_\varnothing) \mid x, y \in \langle \Omega^2 \rangle\} \tag{2.25}$$

$$\bigwedge \{(\xi(x) = \tau_\varnothing) \rightarrow (\psi(x) = \omega_\varnothing) \mid x \in \Omega\} \tag{2.26}$$

An *inactive* lexicon node is one that is not used in any derivation (and thus wouldn't appear in the extracted lexicon when the model is evaluated). Inactive nodes are not a successor to any node (Ax.2.25) and have the terminal node as a successor (Ax.2.26).

The following axioms are derived from the observation made in (Hunter and Dyer, 2013) that the feature sequence consists of a sequence of zero or more selectors and licensors followed by either the special symbol $C$ (and nothing more) or a selectee that is in turn followed by a sequence of zero or more licensees.

$$\bigwedge \{(\xi(x) \in \{\tau_=, \tau_+\}) \rightarrow \xi(\psi(x)) \neq \tau_- \mid x \in s, s \in \lambda\} \tag{2.27}$$

$$\bigwedge \{(\xi(x) \in \tau_\sim, \tau_-\}) \rightarrow \xi(\psi(x)) \in \{\tau_-, \tau_t\} \mid x \in s, s \in \lambda\} \tag{2.28}$$

The successor of a Selector or Licensor cannot be a Licensee (Ax.2.27); likewise, the successor of a Selectee or Licensee is either a Licensee or Terminal (Ax.2.28).

**Axioms for Pairing Feature Matrices with Phonological Forms.** Finally, we will introduce an uninterpreted function, specifically a binary predicate, that will associate each lexicon node sequence (each of which codes for a lexical feature sequence) with one or more members of the PF node sort (which codes for the phonetic features).[49]

Given a member of the lexicon node sort, $x \in \Omega$, and a member of the PF node sort, $y \in \Sigma$, that codes for a phonetic feature $w$, $\Delta_\Omega(x, y)$ is true if and only if the phonetic form $w$ is associated with $x$, where $\Delta_\Omega$ is defined to be an uninterpreted function with signature:

$$\Delta_\Omega : \Sigma \times \Omega \rightarrow \mathit{Boolean}$$

[49] A lexicon node sequence can associate with the "null" member of the PF node sort, $\varnothing_\Sigma$, to indicate no association with any phonetic feature.

The following axioms constrains interpretations of $\Delta_\Omega$:

$$\bigwedge \{\Delta_\Omega(x, \varnothing_\Sigma) \mid x_i \in s, s \in \lambda, i > 1\} \tag{2.29}$$

$$\bigwedge \{\Delta_\Omega(s[0], \varnothing_\Sigma) = (\xi(s[0]) = \tau_\varnothing) \mid s \in \lambda\} \tag{2.30}$$

$$\bigwedge \left\{ \Delta_\Omega(s[0], \varnothing_\Sigma) \neq \left( \bigvee_{x \in \{\Sigma - \varnothing_\Sigma\}} \Delta_\Omega(s[0], x) \right) \mid s \in \lambda \right\} \tag{2.31}$$

Per the (earlier established) convention that the first node in a lexicon node sequence will sometimes code for the entire lexicon node sequence, only the first node in a lexicon node sequence (referred to as a the *representative node*) may associate with a phonetic form; all nodes after the first node in a lexicon node sequence are not associated with any overt or covert phonetic form (Ax.2.29). The representative node of a lexicon node sequence is *active* if and only if it is not associated with $\varnothing_\Sigma$ (Ax.2.30), and every *active* representative node must associate with at least one phonetic feature (Ax.2.31) — i.e. a member of the PF node sort that is not $\varnothing_\Sigma$.

The final set of axioms presented here serve to bound the maximum number of lexical feature sequences that each unique phonetic feature can associate with. To this end, let $k^{overt}_{connect}$ be the maximum number of lexical feature sequences an overt phonetic feature can associate with, and let $k^{covert}_{connect}$ be the maximum number of lexical feature sequences with which a covert phonetic feature can associate. Then the following pair of axioms enforce these bounds for overt and covert nodes respectively:

$$\bigwedge \left\{ \left( k^{overt}_{connect} \geq \sum_{s \in \lambda} \Delta_\Omega(s[0], x) \right) \mid x \in \Sigma_o \right\} \tag{2.32}$$

$$\bigwedge \left\{ \left( k^{covert}_{connect} \geq \sum_{s \in \lambda} \Delta_\Omega(s[0], x) \right) \mid x \in \Sigma_c \right\} \tag{2.33}$$

Each of these two axioms (Ax.2.32 and Ax.2.33) includes a pseudo-boolean constraint that enforces the upper bound on the number of associations with phonetic features. These two axioms are optional and are included for reasons that will become apparent in the next chapter that pertains to acquisition.

This concludes the presentation of the model of the lexicon. Looking ahead to §2.4.1, it should be noted that it is possible to recover, from a satisfiable interpretation of the lexicon model, each of the lexical entries, and for each lexical entry, the sequence of syntactic features.

**Remark 1.** ***Example of an Axiom.*** *Let us walk through an example of an axiom to clarify any confusions introduced by notation. Consider axiom (2.28), replicated below:*

$$\bigwedge \{(\xi(x) \in \tau_{\sim}, \tau_{-}\}) \rightarrow \xi(\psi(x)) \in \{\tau_{-}, \tau_t\} \mid x_i \in s, s \in \lambda\}$$

*We begin by observing that the expression is a conjunction of a set of expressions formed by enumerating over each node in each lexicon node sequence. Note that the quantification is carried out not within the SMT-solver, but externally, in the python program that calls out to the SMT solver; this allows us to maintain an index over the nodes in $\Omega$ (i.e. the* lexicon node sort*) outside of the SMT-solver. The expressions in the conjunction are logical implications of the form:*

$$(\xi(x) \in \tau_{\sim}, \tau_{-}\}) \rightarrow \xi(\psi(x)) \in \{\tau_{-}, \tau_t\}$$

*Here $\xi(\psi(x))$ denotes the feature type of the lexical node of the successor of (the node) $x$, $\tau_{\sim}$ codes for the selectee feature type, $\tau_{-}$ codes for the licensee feature type, and $\tau_t$ codes for the "terminal" feature type (that serves to terminates a feature sequence). Additionally, the sub-expression:*

$$\xi(\psi(x)) \in \{\tau_{-}, \tau_t\}$$

*is written using set-membership notation as a notational convenience – i.e. the expression may be translated to:*

$$(\xi(\psi(x)) = \tau_{-}) \vee (\xi(\psi(x)) = \tau_t)\}$$

*Finally, we present the python code that corresponds to this axiom, so that the reader may see how the set-builder notation translates into a python comprehension:*

```python
s.add_conj(Implies(Or(lnT(x) == lts.Selectee,
                      lnT(x) == lts.Licensee),
                   Or(lnT(succ(x)) == lts.Licensee,
                      lnT(succ(x)) == lts.Terminal))
           for x in le.nodes)
```

### 2.3.2 The Derivation Model

This subsection introduces and details an SMT-model of a minimalist derivation using a quantifier free, multi-sort first order logic.[50] In particular, this subsection will models a class of hierarchical structures that are each assembled from a set of atomic syntactic structures (selected from the lexicon) via the recursive application of *merge*. To this end, we will specify additional sorts, functions and axioms that extend the theory developed in §2.3.1 so that in addition to modeling a minimalist lexicon, the theory also models a minimalist derivation. Looking ahead, the model developed in this subsection will next be extended in §2.3.2 so that a derivation may be constrained by a specified set of interface conditions, which are translated into structural conditions (i.e. SMT-formulae) that constrain satisfiable interpretations of the model of the derivation. After that, §2.3.4 will connect the model of the derivation to the model of the lexicon via shared symbols, so that a satisfiable interpretation of the model of the derivation must be one that is yielded by the lexicon retrieved from the (satisfiable) interpretation of the lexicon model, thereby enabling the model to serve as an MG parser.

This subsection is organized as follows: We will begin by detailing the form of the model, which follows the MG formalism closely, and situate it within contemporary theories of minimalist syntax. We will then introduce the (finite) sorts that model the component parts (e.g. lexical heads, projections and chains) from which the derivation is constructed; these sorts collectively make up the model's *domain of discourse*. Finally, we will introduce the uninterpreted functions that model the relations between the component parts that make up the derivation, and the axioms that constrain what interpretations each of these functions may take on - e.g. these axioms will encode *bare phrase structure* and restrictions on *syntactic movement*.

A summary of the sorts and functions listed in this section is provided in Table 2.2 and Table 2.3 respectively; additionally, the reader may wish to review Fig. 2-8 and reference it while reading this section. Importantly, the reader should make sure to understand how the derivation presented in Fig. 2-2 connects to the derivation model architecture diagram presented in Fig. 2-11, as that is the key to developing an intuitive understanding of how the axioms come together to form a theory of minimalist derivations.

#### Model Form

To begin, we will first outline the theoretical construct of a minimalist derivation in light of contemporary theories of minimalist syntax, and then turn to considering aspects of the MG formalism that make it an appealing vehicle for modeling a minimalist derivation.

All theories of language that align with the Minimalist Program share several defining design characteristics. The language faculty has (at a minimum):

(i) A finite, discrete lexicon (a model of which was developed in §2.3.1) consisting of a set of lexical entries – i.e. atomic syntactic structures – each of which encodes a feature matrix that includes features interpretable at the PF interface (e.g. a phonological form), features interpretable at the LF interface (e.g. a category), and uninterpretable features.[51]

[50] The axioms in the theory must be quantifier free as the SMT-solvers cannot guarantee decidability for problems involving universal quantifiers, so we will work with explicit quantification.

[51] In a minimalist theory of language, all variability in the design of a particular natural language is found in the lexicon – this notion is formalized by the *Borer-Chomsky Conjecture*. See (Baker, 2008).

(ii) A recursive, binary structure building operation, *merge*, that combines two syntactic structures together to derive a new syntactic structure; a syntactic structure built up via *merge* is a *derivation*, and a derivation corresponding to a complete grammatical expression is a *complete derivation*.

This subsection will develop a model of a derivation that is constructed by *merge*. *Merge* has two distinct cases, *external merge* and *internal merge*, that are logically disjoint:[52]

(i) In the case of *external merge*, the two input structures are disjoint (i.e. neither structure is a substructure of the other). Each of the two arguments of *external merge* is either (i) projected from the lexicon and referred to as a *lexical structure*, or (ii) derived by an earlier *merge* operation and referred to as a *derived structure*. This thesis assumes *checking theory*, so that *external merge* applies only if the features in the feature matrices of the two structures match appropriately; the new structure (produced by external merge) will derive its feature matrix from whichever of the two constituents projects, and which of the two constituents projects is determined by their respective features.

(ii) In the case of *internal merge*, one of the two structures is a proper substructure of the other, with the substructure that contains the other serving as the *head* of the newly derived structure (that is produced by internal merge).[53] Internal merge is subject to the principle of minimal computation in that a licensor will identify and raise the (matching) licensee that is closest with respect to hierarchical distance. Internal merge forms a discontinuous object - i.e. a *chain* - with each link (formed by internal merge) in the chain linking the source (i.e. trace) position to the target (i.e. raised) position.[54]

When two syntactic structures are merged together, the resulting structure is assigned a *label*, and this assignment holds over the remaining course of the derivation.[55] The *label* of a syntactic structure must be computed only from the properties of the constituent structures, and is typically taken to be the *label* of the projecting constituent structure.[56] This

[52] See (Collins and Stabler, 2016, Pg. 47-48): *"Given any two distinct syntactic objects A, B, Merge$(A, B) = \{A, B\}$. Merge takes two syntactic objects and combines them into a single syntactic object. [...] lexical item tokens are syntactic objects, so Merge can combine them. This is the basic structure building operation of syntax. [...] We make no distinction between external Merge and internal Merge. They are not two separate operations (see, e.g. (Chomsky, 2005, Pg. 12)) Rather, external Merge corresponds to the case of Merge where $A, B \in W$ (a workspace). Internal Merge corresponds to the case of Merge where $A \in W$, and A contains B."* See also (Chomsky, 1995, Pg. 243).

[53] *Internal Merge* is how Minimalist theories of language capture the notion of (syntactic) movement of phrases.

[54] See (Chomsky et al., 2019): *"IM thus turns Y into a discontinuous object (or chain), which can be understood as a sequence of occurrences of Y in K.3."* See also (Chomsky, 1995, Pg. 250): *"The operation Move forms the chain $CH = (\alpha, t(\alpha))$, $t(\alpha)$ the trace of $\alpha$. Assume further that CH meets several other conditions (C-Command, Last Resort, and others), to be spelled out more carefully as we proceed."*

[55] See (Chomsky, 1995, Pgs. 243-244): *"We assume further that the label of K is determined derivationally (fixed once and for all as K is formed), rather than being derived representationally at some later stage of the derivation (say, LF). This is of course, not a logical necessity; Martian could be different. Rather, it is an assumption about how* human *language works, one that fits well with the general thesis that the computational processes are strictly derivational, guided by output conditions only in that the properties available for computational purposes are those interpreted at the interface."*

[56] This thesis assumes, as in (Adger, 2003, Pg. 66) and (Radford, 2016, Pg. 20 & Pg. 28), that the label is the category associated with the head of the constituent that projects. More generally, there is a notion of a *labeling algorithm* that is tasked with computing the label for a syntactic structure – e.g. see (Hornstein and Pietroski, 2009).

is a consequence of minimalist theories of syntax incorporating the theory of *bare phrase structure* (BPS), which requires that:[57] (i) no special (theoretical) treatment be given to minimal or maximal projections, which are relegated to being conventions of convenience for discussion;[58] (ii) no allowance be made for notions of "bar levels" to distinguish between intermediate projections falling between minimal and maximal projections of a lexical head; (iii) no distinction be made between lexical items and the heads they project.[59][60]

There are also a number of principles of minimalist syntax that restrict the derivations that a lexicon may yield. Keeping in line with the goals of the Minimalist Program, these principles center on simplifying the requirements made of $C_{HLF}$, namely by: (i) restricting the information that $C_{HLF}$ has access to and that it may present to the LF and PF interfaces; (ii) constraining the topology of the structures that *merge* may produce; (iii) requiring that derivations be derived bottom-up and that they not involve any steps or stages that are not strictly required to satisfy interface conditions. The most important and well established principles are worth mentioning explicitly. The ***No Tampering Condition*** requires that merge does not alter the arguments (i.e. the arguments are left unchanged).[61] The ***Extension Condition*** requires that merge yields a new syntactic structure that strictly contains its two arguments and doesn't contain anything other than these two arguments.[62] The ***Inclusiveness Condition*** requires that interface levels are presented with nothing more than lexical features that originate in the lexical items that enter the into derivation, and that the derivation contains nothing beyond lexical items and syntactic structures derived (via merge) from these lexical items; in particular, lexical features may be deleted (e.g. during feature checking), but no features may appear in the derivation beyond those found in the lexical items (e.g. there are no "bar levels" or "traces").[63][64] The ***Principle of Full Interpretation*** requires that there are no superfluous features within LF or PF representations

[57] BPS is a theory of phrase structure that succeeded and was informed by *X-Bar theory*, a staple of earlier theories within the Principles and Parameters framework.

[58] See (Chomsky, 1995, Pgs. 242-243): *"There are no such entities as $XP$ ($X^{max}$) or $X^{min}$ in the structures formed by $C_{HL}$, though we continue to use the informal notations for expository purposes, along with $X'$ (X-bar) for any other category. A category that does not project any further is a maximal projection $XP$, and one that is not a projection at all is a minimal projection $X^{min}$; any other is an $X'$, invisible at the interface and for computation."*

[59] See (Chomsky, 1995, pg. 228): *"We thus have the outlines of a bare phrase structure theory that derives fairly strictly from naturalist minimalist principles. The bare theory departs from conventional assumptions in several respects: in particular, categories are elementary constructions from properties of lexical items, satisfying the inclusiveness condition; there are no bar levels and no distinction between lexical items and "heads" projected from them. A consequence is that an item can be both an $X^0$ and an $XP$."*

[60] See (Chomsky, 1995, Pg. 245): *"... phrase structure representation is 'bare,' excluding anything beyond lexical features and objects constructed from them..."*

[61] Note that the No Tampering Condition implies a "copy theory of movement", so that a raised constituent is not replaced with a trace (which accords with the prescriptions of the theory of bare phrase structure). See (Chomsky, 2005, Pg. 13) for further discussion of this point. See also (Radford, 2016, Pg. 348).

[62] See (Chomsky, 1995, Pg. 190).

[63] See (Chomsky, 1995, Pg. 225): *"A perfect language should meet the condition of inclusiveness: any structure formed by the computation (in particular, $\pi$ and $\lambda$) is constituted of elements already present in the lexical items selected for $N$; no new objects are added in the course of computation apart from rearrangements of lexical properties (in particular, no indices, bar levels in the sense of X-bar theory, see note 7)."*

[64] See (Chomsky, 2001, Pg. 2-3): *"On such grounds, we try to eliminate levels apart from the interface levels, and to maintain a bare phrase structure theory and the* Inclusiveness Condition, *which bars introduction of new elements (features) in the course of computation: indices, traces, syntactic categories or bar levels, and so on."*

and that *every* LF or PF representation produced must be interpretable.[65][66][67][68] ***Economy of Derivation*** stipulates that a derivation not involve any superfluous steps (i.e. instances of merge) (this will be covered in greater detail later in this thesis in §3.2.1).[69] Finally, the ***Principle of Last Resort*** requires that internal merge is driven by features (and feature checking).[70]

Having touched on how derivations are formulated as per (earlier) minimalist theories of syntax, let us now make several observations about the MG formalism that will inform the approach we will take to developing an SMT-model of a minimalist derivation.

- Every MG derivation tree is uniquely associated with a multi-dominance tree (i.e. the (bare) phrase structures that linguists are familiar with).[71] The multi-dominance tree may be obtained from the derivation tree by appending, for each instance of internal merge in the derivation tree, a node at the target of the movement, thereby establishing a dominance relation over the source of movement; note that whereas the derivation tree *is not* a strictly binary-branching tree (since there are unary branches indicating the presence of *internal merge*) the associated multi-domimance tree *is* strictly binary-branching, since every non-lexical node in the multi-dominance tree is the product of either *external merge* or *internal merge* and has two constituent (i.e. child) nodes.[72] We assume, for both the derivation tree and the multi-dominance tree, that every

[65] See (Chomsky, 1986, Pg. 98.) *"there is a principle of full interpretation (FI) that requires that every element of PF and LF, taken to be the interface of syntax (in the broad sense) with systems of language use, must receive an appropriate interpretation – must be licensed in the sense indicated. None can simply be disregarded."*

[66] See (Chomsky, 1995, Pg. 199): *"$\pi$ is a PF representation and $\lambda$ an LF representation, each consisting of "legitimate objects" that can receive an interpretation (perhaps as gibberish). If a generated representation consists entirely of such objects, we say that it satisfies the condition of Full Interpretation (FI)."*

[67] See (Chomsky, 1995, Pg. 151): *"The analogous principle for representations would stipulate that, just as there can be no superfluous steps in derivations, so there can be no superfluous symbols in representations. This is the intuitive content of the notion of Full Interpretation (FI), which holds that an element can appear in a representation only if it is properly 'licensed.' "*

[68] The Principle of Full Interpretation (FI) is a Principle of Economy of Representation – see (Chomsky, 1995, Pg. 220): *"Let us now look more closely at the economy principles. These apply to both representations and derivations. With regard to the former, we may take the economy principle to be nothing other than FI: every symbol must receive an 'external' interpretation by language-independent rules."*

[69] See (Collins, 2001, Pg. 40): *"Consider an operation OP applying in a derivation D leading to the representations (PF, LF) (phonetic form and logical form). Economy considerations suggest that OP be as small as possible, and be applied in a way that minimizes search. Given a series of operations that form a derivation D, economy conditions suggest that the length or cost of the derivation must be minimized in some way. Lastly, economy considerations suggest that the representations formed in the course of a derivation should be as simple as possible, consisting of a minimal number of syntactic objects, each of which is interpretable (at LF or PF)."*

[70] See (Chomsky, 1995, Pg. 228): *"A core property of $C_{HL}$ is feature checking, the operation that drives movement under the Last Resort condition."*

[71] See the discussion of "augmented derivation trees" in (Graf, 2013, Pgs. 12-24): *"An augmented derivation tree provides a record of the steps taken during the derivation and their relative order: its leafs are annotated with the LIs a given tree is constructed from, and interior nodes are labeled with the name of the operation that takes place at this point and the name of the two features being checked. The daughters of an interior node are the elements that are combined by the operation the node represents. Note that an augmented derivation tree is actually a strictly binary branching multi-dominance tree because a given subtree might be involved in both Merge and Move."*

[72] This is closely related to the two-step approach of first lifting information implicitly encoded within a derivation tree (i.e. the information encoded in the structure of the multi-dominance tree) to make the information explicit, and then reconstructing the (derived) phrase structure tree that linguists are more familiar with. See (Graf, 2013, Pgs. 32-50) for a review of the two-step approach of lifting an MG derivation to its associated the multi-dominance tree and then reconstructing the "derived tree"; see also (Kobele

node has a “head”; furthermore, since the two nodes that merge cannot have the same head, we can identify which of two constituents that merge together projects based on the head of the product of merge. Notably, the derivation tree and the multi-dominance tree *do not* explicitly encode the precedence relations between the lexical heads entering into the derivation. The derivation tree may be obtained from the multi-dominance tree by deleting all instances of movement.[73] The derived tree can be obtained from the multi-dominance tree by deleting all nodes in the multi-dominance tree that were raised to a higher position (because they were the source of movement). *The multi-dominance tree is in effect a super-position of the derivation tree and the derived tree – it is the multi-dominance tree associated with an MG derivation that will serve as the domain of discourse in the SMT-model of a minimalist derivation that we will develop.* Throughout this thesis, we will work with these multi-dominance trees, and refer to the associated derivation tree and derived tree with the understanding that they are components of a multi-dominance tree.

- Each lexical item appearing in a derivation has a (bottom-up) trajectory through the associated multi-dominance tree:

  (i) the lexical item first undergoes a sequence of (zero or more) projections, driven by either external merge (driven by a selector feature) or internal merge (driven by a licensor feature);[74]

  (ii) the lexical item is then either the end of the derivation (marked by the presence of the special symbol $C$) or is selected by some other lexical head (driven by the presence of a selectee feature);

  (iii) finally the lexical item is optionally raised (via internal merge) one or more times to form a chain (with each movement operation forming a link in the chain).

  There are two key points to take away from this observation. First, every node in the multi-dominance tree is associated with one of the lexical items (i.e. leaf nodes) in the derivation – i.e. the lexical item that is the head of that node – and the nodes associated with a lexical head may be organized as a sequence in the order in which they appear in the multi-dominance tree (starting from the bottom); we will refer to such a sequence as a “derivation node sequence” and observe that the multi-dominance tree associated with an MG derivation is a structural arrangement of derivation node sequences.[75][76] Second, given a multi-dominance tree associated with an MG derivation, it is possible to recover the multiset of lexical items from which

et al., 2007). See (Morawietz, 2008, Pgs. 131-182) for a presentation of the two-step approach for multiple context-free grammars (MCFGs), noting that MGs may be translated into MCFGs (per (Michaelis et al., 2000)).

[73] A node in the multi-dominance tree is the product of internal merge (i.e. movement) if the head of one constituent is dominated by the root node of the other constituent.

[74] Note that we can distinguish between whether the projection is driven by external or internal merge based on whether the head of the non-projecting constituent is dominated by the projecting constituent.

[75] See (Stabler, 2013, Pg. 612): *“A successful MG parse is a kind of structurally conditioned check of lexical requirements, implemented as the traversal of a graph representing the lexicon.”*

[76] The notion of a “derivation node sequence” is inspired by the closely related notion of “slices” (of a derivation tree) that is employed in (Graf, 2013) Specifically, see (Graf, 2013, Pg. 28): *“Just like phrase structure trees can be decomposed into subtrees containing exactly one LI and all its projections, derivation trees can be decomposed into subderivations that consist of an LI and all the interior nodes that check one of the LI’s positive polarity features (i.e. selector and licensor features). These subderivations are called slices.”* See also (Graf, 2013, Pg. 29-30): *“Intuitively, slices are the derivation equivalent of phrasal projection. Just*

the multi-domimance tree is derived (modulo the labels of the syntactic features); to see this, note that each node in a *derivation node sequence* is associated with exactly one type of syntactic feature – i.e. selector, selectee, licensor, licensee, or the special symbol $C$ – and that the feature type of a node may be determined by the location of that node within the multi-domimance tree, so that so that given a derivation node sequence associated with a particular lexical entry, we can recover the sequence of syntactic feature types present in that lexical entry from the derivation node sequence. (See Fig. 2-11 for a presentation of the derivation node sequences that are assembled to form the derivation presented in Fig 2-2.) *We can therefore construct an SMT-model of a minimalist derivation by modeling the derivation node sequences that make up the associated multi-dominance tree and restricting the topology of the multi-domimance tree by using the model axioms to constrain how the derivation node sequences may be assembled together.*

*We can therefore construct an SMT-model of a minimalist derivation by modeling the derivation node sequences that make up the associated multi-dominance tree and using the model axioms to restrict the topology of the multi-domimance tree by constraining how the derivation node sequences may be assembled together.*

- Although the axioms for external and internal merge (listed in §2.2) combine both the building of the hierarchical structure of a derivation tree and computing the linear ordering of the derived string, in fact the rules for linearizing an MG derivation can be factored apart from the rules for assembling an MG derivation. This is possible because we can first assemble an MG derivation tree without taking into consideration the linear ordering of words, and then transform a derivation tree into a derived tree, at which point the linear-ordering of words is accounted for. In particular, given a minimalist derivation tree, the ordering of nodes in the associated derived tree can be determined by observing that: (i) a head precedes its complement, and (ii) a head is preceded by its specifier. Since the complement and specifier of a head are *strictly* hierarchical relations, they may be determined entirely from the hierarchical structure of the multi-dominance tree that pertains to the derived tree (i.e. the sub-structure of the multi-domimance tree that corresponds to the derived tree). *As a consequence, the model axioms that constrain how the derivation node sequences may be assembled to form a multi-dominance tree (i.e. associated with a MG derivation tree) can be separated from the axioms involved in establishing a linear precedence over the leaf nodes of the derived tree (obtained from the multi-dominance tree)*; the former set of axioms will be presented here in §2.3.2, and the latter set of axioms will be presented later in §2.3.4.

These observations lead us to developing an SMT-model of a minimalist derivation as outlined below. The model will include a finite sort (i.e. the domain of discourse) and a set of (unary and binary) uninterpreted functions that will be used to model the multi-dominance

*like every node in a phrase structure tree is either an LI or one of its projections, every node in a derivation belongs to the slice of some LI. The slice of an LI is readily determined: its leaf is the LI itself, and for every positive polarity feature an interior node of the appropriate type is added on top of the slice – Merge for selector features, Move for licensor features. Hence every Minimalist lexicon can be converted into a set of slices. These slices can be recombined to yield derivation trees, although not all combinations may obey the requirements of the feature calculus."* Note that derivation node sequences involve both positive and negative polarity features, along with the special symbol $C$, whereas "slices" involve only the positive polarity features.

tree associated with an MG derivation tree. Members of the model sort will be organized into derivation node sequences. The uninterpreted functions will serve to establish relations between the nodes in the multi-dominance tree – e.g. the parent of a node, the sister of a node, the head of a node, relations between the source and target of (syntactic) movement, and relations between the source and target of head-movement. The model axioms will translate the Principles of Minimalist Syntax discussed earlier into constraints that govern how the derivation node sequences may be (legitimately) assembled together within an MG derivation; in particular, the model includes axioms for bare phrase structure, syntactic movement (including head-movement), and extended functional projections. Notably, this model is concerned only with the *hierarchical structure* of the multi-domimance tree associated with a minimalist derivation; later in this chapter, §2.3.3 will introduce axioms requiring the precedence relations in the derived tree (which is a substructure of the multi-dominance tree being modeled) align with the linear ordering of the phonological forms prescribed in the specified PF interface conditions; §2.3.4 will then introduce an uninterpreted function that maps derivation node sequences in the derivation model to lexicon node sequences in the lexicon model, so as to restrict which derivation node sequences may appear in a derivation (i.e. only those corresponding to a feature sequence found in a lexical entry) as well as how the derivation node sequences may be combined together (which is dependent on the feature labels found in the lexical entry associated with a derivation node sequence). Note that the model is in effect a static representation of a complete derivation – i.e. the model itself is not derivational in so far as the model does not have the notion of workspaces or stages.

Having outlined the approach to modeling a minimalist derivation, let us now proceed develop the model in detail.

### Model Sorts

**The Derivation Node Sort.** The domain of discourse is defined to be a finite sort, $\mathbb{N}$, a subset of which corresponds to the set of nodes that form the multi-dominance tree associated with a minimalist derivation; members of $\mathbb{N}$ will sometimes be referred to as "nodes." *(Note again that throughout this section, whenever we reference a derivation, unless we explicitly refer to the derivation tree or to the derived tree, it should be understand that we are referring to the multi-dominance tree associated with that derivation.)* $\mathbb{N}$ is the union of the following disjoint subsets: a singleton set, the only member of which is denoted $\perp$; the set of *lexical nodes*, $\mathbb{L}_\mathbb{N}$; the set of *intermediate nodes*, $\mathbb{I}_\mathbb{N}$; a singleton set, the only member of which is denoted $R_\mathbb{N}$. The set of lexical nodes, $\mathbb{L}_\mathbb{N}$, can only serve as leaf nodes in the derivation – i.e. these nodes are referred to as lexical nodes because they correspond to the lexical items that are assembled together via *merge* to form the derivation. $\mathbb{L}_\mathbb{N}$ is divided into two disjoint sets: the set of *overt* lexical nodes, ${\mathbb{L}_\mathbb{N}}^+$, and the set of *covert* lexical nodes, ${\mathbb{L}_\mathbb{N}}^-$. The root node, $R_\mathbb{N}$, must correspond to the root node in the derivation, and the intermediate nodes, $\mathbb{I}_\mathbb{N}$, are nodes that can only appear within the derivation between the lexical nodes and the root node. The set of *derived nodes*, defined as $D = \mathbb{I}_\mathbb{N} \cup \{R_\mathbb{N}\}$, may appear as non-leaf nodes in the derivation. Finally, the *bottom node*, $\perp$, serves as a *"null"* value for functions and predicates acting over $\mathbb{N}$: whenever an uninterpretable function is applied to an argument outside of its domain (which may be a subset of $\mathbb{N}$), the output of the function must be the *bottom node*; in the case of predicates, the output will instead be False. Only a subset of $\mathbb{N}$ will be used to form a derivation, with each member of this subset uniquely corresponding to a node in the associated multi-dominance tree.[77]

**Derivation Node Sequences.** Having introduced the theory's domain (i.e. the *derivation node sort*) and its relevant subsets, we will next organize the members of the derivation node sort to reflect the structure imposed by: (i) the *projection* of lexical heads in accordance with the theory of *bare phrase structure*, and (ii) the chains established by *movement* of projections. The nodes in $(\mathbb{L}_{\mathbb{N}} \cup \mathbb{I}_{\mathbb{N}})$ can be grouped by their head. The nodes in one such group, all having the same head, can be organized as a sequence of nodes, starting with the lexical node; this sequence of nodes consists of sequence of projections (via external merge) that starts from the given lexical head and ends in a maximal projection (which may very well be a minimal projection as well), followed by the sequence of nodes that forms the *chain* associated with the maximal projection of the given lexical head.[78] The nodes that form the chain are ordered unambiguously by the c-command relation that must hold between the origin and target of raising.[79] Each node in the sequence beyond the starting node is either the parent of or was raised from (via movement) the node prior to it in the sequence; in the former case, the prior node projects to its parent node (i.e. within the derivation), while in the latter case the prior node is *c-commanded* by the position in the derivation to which it will be raised. We will now formalize these points and incorporate them in to the model.

A *derivation node sequence* of length $k$ is defined to be a finite sequence of nodes, $x_1, \ldots, x_k \in \mathbb{N}$ with $x_1 \in \mathbb{L}_{\mathbb{N}}$ and $x_i \in \mathbb{I}_{\mathbb{N}}$ for $1 < i$. If $x_1 \in \mathbb{L}_{\mathbb{N}}{}^{+}$ then the derivation node sequence is referred to as an *overt* derivation node sequence, whereas if $x_1 \in \mathbb{L}_{\mathbb{N}}{}^{-}$ then the derivation node sequence is referred to as a *covert* derivation node sequence. The set of derivation node sequences (in the derivation model) is denoted $\mathbb{S}$, the set of overt derivation node sequences is denoted $S^{+}$, and the set of covert derivation node sequences is denoted $S^{-}$.[80] Each member of the set $S^{+}$ is associated with an index (i.e. a natural number) that corresponds to the position in the sentence of the lexical head associated with that member;[81] associating derivation node sequences with the position of the lexical head, rather than the associated phonetic feature, is necessary so that the model axioms can refer separately to different tokens in a sentence that are associated with the same word.[82][83]

A derivation node sequence of length $k$ is organized as two consecutive subsequences of nodes. The first subsequence of derivation nodes begins with a lexical head, and is optionally followed by one or more nodes corresponding to the successive projections of the lexical head; the first node in this subsequence corresponds to the minimal projection of the lexical head, and the last node in this subsequence corresponding to the maximal projection of the lexical

[77] In particular, note that only subsets of $\mathbb{L}_{\mathbb{N}}$ and $\mathbb{I}_{\mathbb{N}}$ may appear in the multi-dominance tree associated with the derivation; which particular subsets of these sets appear will be determined the SMT-solver in accordance with the model axioms.

[78] Note that a chain may have zero links in the case that the maximal projection of the lexical head is never raised.

[79] See (Chomsky, 1995, Pgs. 251-252).

[80] N.b. the derivation nodes $\perp$ and $R_{\mathbb{N}}$ do not appear in any of the derivation node sequences.

[81] N.b. these indices serve no other purpose that to distinguish one position from another, and do not serve any other function with respect to syntax.

[82] This approach is based on that of (Collins and Stabler, 2016, Pg. 45): *"In order to allow structures in which a given lexical item occurs twice in a structure, the lexical items in our structures are indexed with integers. [...] For example, in the sentence 'The dog saw the other dog', there are two tokens of the single lexical item dog, tokens with different numerical indices. The integer in the lexical item token plays no other role in the syntactic computation. For example, the integer will not be used in 'counting.' "*

[83] Whenever it is clear from context, for purposes of brevity and clarity, we will directly refer to particular overt node-sequences in a derivation model via the phonetic feature (i.e. word) with which it is associated.

head. The second subsequence of derivation nodes (immediately following the first) consists of zero or more members, with successive members corresponding to successive links in the chain established by the raising of the maximal projection of the lexical head. The length of the two subsequences together is at most $k$, with any remaining nodes (after the two subsequences) considered to be *inactive* in so far as they will not correspond to any node in the derivation.[84] *In this way, a derivation node sequence captures the trajectory that a lexical head takes through a derivation, and serves as a data structure that can be used to represent both projections and chains in a common way.*

To summarize, organizing the derivation node sort as a set of derivation node sequences is advantageous in that it enables the model axioms to be written using the indexing scheme of node sequences – e.g. the model axioms can quantify over all of the node sequences in the derivation; see Figure-2-11 for an illustration of this.[85] Additionally, this organization break symmetries (in the model) in so far as restricting which members of the derivation node sort can take on which role within a derivation – this was found to be essential to improving run time performance enough to ensure that the SMT solver is able to check and optimize the model in a tractable amount of time (i.e. overnight). *Looking ahead, additional model axioms will be introduced in §2.3.4 that will serve to associate derivation node sequences with lexicon node sequences, so that the sequence of syntactic features associated with a lexical head will align with the trajectory the lexical head takes through a derivation.*

### Model Axioms

Having defined the underlying sort, $\mathbb{N}$, let us now turn to the task of defining functions (operating over $\mathbb{N}$) and axioms that will collectively model *bare phrase structure*, *extended functional projections*, and *syntactic movement.*

[84] In particular, observe that not every lexical or derived node in the domain (of discourse) need be used in the construction of a derivation – e.g. the theory may be instantiated with a very large number of intermediate nodes as we do not know ahead of time how many nodes are required to model the instances of phrasal movement that occur in the derivation; likewise, the number of instances of covert lexical items (i.e. empty categories) that will be required for the derivation is not known apriori.

[85] In general, since we are using first order logic, any sets of sets to be quantified over must be indexed explicitly, external to the solver; consequently, we will index the sorts for derivation nodes and lexicon nodes in multiple ways, so that the model axioms may be written in a more transparent and comprehensible manner.

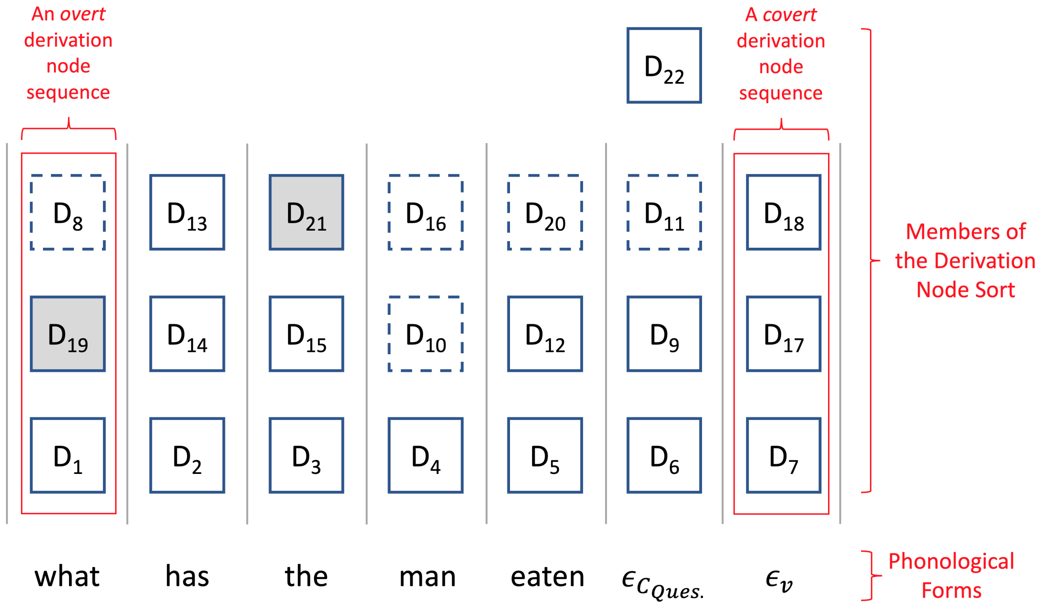


Figure 2-11: An illustration of how the members of the derivation node sort are organized into overt and covert node-sequences (displayed as columns, starting from the bottom and going up) that are associated with overt and covert phonological forms respectively; Figure 2-2 illustrates how the derivation node sequences depicted here are arranged into a derivation. The greyed out boxes are inactive derivation nodes that do not participate in the derivation – i.e. they do not appear in the derivation in Figure 2-2. Active nodes (depicted as white boxes) that are in the same column have the same head. The boxes with dashed boundaries are part of a chain, whereas the boxes with solid boundaries are projections. The root node is $D_{22}$, and as $D_{22}$ has the same head as $D_9$ and $D_6$, it is depicted here atop the covert node-sequence associated with covert phonological form $\epsilon_{C_{Ques.}}$. Note that the root node is not a part of any node sequence, and is treated as a special case in the axioms. The node-sequence structure provides an indexing scheme that enables us to write axioms that constrain the uninterpreted functions that operate over the derivation node sort – i.e. the derivation node sequence associated with a particular phonological form provides an indexing over the derivation node sort that lets us reference the projection (and potentially chain) associated with that phonological form.

**Notation**

1. Given $X \subseteq \mathbb{N}$ and $f : X \rightarrow \mathbb{N}$,

   - the **support** of $f$, denoted $\operatorname{supp} f$, is defined to be $\{x|f(x) \neq \perp, x \in X\}$;
   - the **kernel** of $f$, denoted $\ker f$, is defined to be $\{x|f(x) = \perp, x \in X\}$;
   - the **image** of $f$, denoted $\operatorname{im} f$, is defined to be $\{y|f(x) = y, x \in X, y \in \mathbb{N}\}$.

   Although many of the axioms in the model will make reference to $\operatorname{supp} h$, they will be limited to determining whether a given member of $\mathbb{N}$ is a member of $\operatorname{supp} h$, which can be checked by evaluating whether $h(x) \neq \perp$, thereby allowing us to avoid quantifying over the possible values of $\operatorname{supp} h$ which is not allowed in a quantifier-free first-order theory of logic.

2. Given $X \subseteq \mathbb{N}$ and a function $f : X \rightarrow Boolean$,

   - the **support** of $f$ is: $\operatorname{supp} f = \{x|f(x), x \in X\}$;
   - the **kernel** of $f$ is: $\ker f = \{x|\neg f(x), x \in X\}$.

3. Given sets $X_1, X_2, ..., X_p$, let $X^{'} = X_1 \times X_2 \times ... \times X_p$; then the set operation $\langle X^{'} \rangle$ is defined to be $\{(x_1, x_2, ..., x_p)|x_i \neq x_j, 1 \leq i < j \leq p, (x_1, x_2, ..., x_p) \in X^{'}\}$.

**Axioms for Bare Phrase Structure.** We will now introduce and detail several uninterpreted functions and axioms that collectively model the theory of **Bare Phrase Structure** (BPS) as presented in (Chomsky, 1995).

To begin, each node in the derivation is associated with a (lexical) *head* - i.e. one of the leaf nodes in the derivation; this is per the **Headedness Principle**, according to which *"every nonterminal node in a syntactic structure is a projection of a head word."*[86] The head of a node $x \in N$ is $h(x)$, where $h$ is defined to be an uninterpreted function with signature:

$$h : \mathbb{N} \longrightarrow \mathbb{N} \tag{2.34}$$

Several axioms constrain interpretations of $h$:

$$\operatorname{supp} h \subseteq (\mathbb{L}_{\mathbb{N}} \cup D) \tag{2.35}$$

$$\operatorname{im} h \subseteq \{\perp\} \cup (\mathbb{L}_{\mathbb{N}} \cap \operatorname{supp} h) \tag{2.36}$$

$$\bigwedge \{(h(x) = \perp) \oplus (h(x) = x) \mid x \in \mathbb{L}_{\mathbb{N}}\} \tag{2.37}$$

$$\bigwedge \{(h(x_i) = h(x_0)) \vee (h(x_i) = \perp) \mid x_i \in s, s \in \mathbb{S}\} \tag{2.38}$$

As every node in a derivation has a head, if a node $x \in \mathbb{N}$ does not play a role in the derivation then $h(x) = \perp$. (Note that the bottom node, $\perp$, does not have a head in the derivation – i.e. $h(\perp) = \perp$ – as the bottom node does not correspond to a phrase in the derivation.) A member of $\mathbb{N}$ need not be used (i.e. play a role in the derivation), implying that $\operatorname{supp} h \subseteq \mathbb{N}$ (Ax.2.35). Each leaf node in a derivation is associated with a phonological form, and thus the head must be a leaf node that plays a role in the derivation (Ax.2.36). If a lexical node plays a role in the derivation then it is its own head (Ax.2.37). All nodes

[86] This definition is taken from (Radford, 2009, Pg. 43).

in a node sequence that play a role in the derivation must have the same head, namely the head of the beginning of the node sequence (Ax.2.38).[87]

Each node in the derivation also has a *parent*.[88] The parent of a node $x \in \mathbb{N}$ is $p(x)$, where $p$ is defined to be an uninterpreted function with signature:

$$p : \mathbb{N} \longrightarrow \mathbb{N} \tag{2.39}$$

Two nodes with the same parent are said to have been *merged* to form the parent node; later, we will introduce a function that directly captures this relation between each parent node and its two children in the derivation. Several axioms constrain interpretations of $p$:

$$\bigwedge \{p(x) \neq x \mid x \in (\mathbb{L}_{\mathbb{N}} \cup D)\} \tag{2.40}$$

$$\operatorname{supp} p \subseteq (\mathbb{L}_{\mathbb{N}} \cup \mathbb{I}_{\mathbb{N}}) \cap \operatorname{supp} h \tag{2.41}$$

$$\operatorname{im} p \subseteq \{\bot\} \cup (D \cap \operatorname{supp} h) \tag{2.42}$$

$$\bigwedge \{\langle \bot, p(x), R_{\mathbb{N}} \rangle \rightarrow (p(p(x)) \neq \bot) \mid x \in (\mathbb{L}_{\mathbb{N}} \cup \mathbb{I}_{\mathbb{N}})\} \tag{2.43}$$

$$\bigwedge \{(h(x) \neq \bot) \rightarrow (p(x) \neq \bot) \mid x \in (\mathbb{L}_{\mathbb{N}} \cup \mathbb{I}_{\mathbb{N}})\} \tag{2.44}$$

A node in a derivation cannot be its own parent; thus the function $p$ has no fixed points other than $\bot$ (Ax.2.40). The root node, $R_{\mathbb{N}}$, does not have a parent (Ax.2.41). The parent of a node in a derivation cannot be a lexical node (Ax.2.42). All intermediate nodes with children have a parent (Ax.2.43 and Ax.2.44).

$$\bigwedge \left\{ (p(x) \neq \bot) \rightarrow \left( 2 = \sum_{y \in (\mathbb{L}_{\mathbb{N}} \cup \mathbb{I}_{\mathbb{N}})} (p(x) = p(y)) \right) \mid x \in (\mathbb{L}_{\mathbb{N}} \cup \mathbb{I}_{\mathbb{N}}) \right\} \tag{2.45}$$

$$\bigwedge \left\{ (p(x) = p(y)) \rightarrow ((h(p(x)) = h(x)) \vee (h(p(y)) = h(y))) \mid x, y \in \langle (\mathbb{L}_{\mathbb{N}} \cup \mathbb{I}_{\mathbb{N}})^2 \rangle \right\} \tag{2.46}$$

If a node $x$ in a derivation is the parent of another node $y$, per the **Binary Branching Hypothesis**,[89] $x$ must be the parent of exactly two distinct nodes in the derivation (Ax.2.45), and exactly one of the two children projects (Ax.2.46).

A parent-child relation implies dominance (otherwise known as "containment"), and the transitive closure of this binary predicate establishes a tree structure (i.e. the *derivation tree*). Accordingly, dominance relations between nodes in the derivation tree are modeled by an additional uninterpreted binary predicate that operates over $\mathbb{N}$: given $x, y \in \mathbb{N}$, $d(x, y)$ denotes that $x$ dominates $y$ (i.e. successive application of $p$ to $y$ will produce $x$), where $d$ is defined to be an uninterpreted function with signature:

$$d : \mathbb{N} \times \mathbb{N} \rightarrow \mathit{Boolean} \tag{2.47}$$

[87] Given a node $x \in \mathbb{N}$, $h(x) = \bot$ if and only if $x$ does not appear in the derivation – for this reason, many axioms in the derivation model will use $h(x) \neq \bot$ as a proxy for whether the node appears in the derivation.

[88] The parent of the root node of the derivation is the node $\bot$; this indicates that the root node does not have a parent within the derivation.

[89] See (Radford, 2009, Pg. 375).

Several axioms constrain interpretations of $d$:

$$\operatorname{supp} d \subseteq (D \cap \operatorname{supp} h) \times ((\mathbb{L}_{\mathbb{N}} \cup \mathbb{I}_{\mathbb{N}}) \cap \operatorname{supp} h) \tag{2.48}$$

$$d \text{ is irreflexive, anti-symmetric and transitive} \tag{2.49}$$

$$\bigwedge \left\{ d(x,y) \rightarrow (x = p(y)) \oplus (d(x, p(y))) \mid x, y \in \langle (\mathbb{L}_{\mathbb{N}} \cup D)^2 \rangle \right\} \tag{2.50}$$

$$\bigwedge \{ p(x) \neq \bot \rightarrow d(p(x), x) \mid x \in \mathbb{N} \} \tag{2.51}$$

Lexical nodes (i.e. leaf nodes) cannot dominate any other node and the root node cannot be dominated by any other node (Ax.2.48). The $d$ is a (binary) predicate that is irreflexive, transitive and asymmetric relation over $(\operatorname{supp} h \times \operatorname{supp} h)$, it is therefore a strict partial ordering over $\operatorname{supp} h$ (Ax.2.49). Parenthood implies domination (Ax.2.51), and domination is either a direct consequence of a parent relation or is derived from the parent relation – i.e. $d$ is the transitive closure of $p$ over $\operatorname{supp} h$ (Ax.2.50). *The domination predicate will be used in axioms related to phrasal movement so as to impose the requirement that the target position of phrasal movement must c-command the trace position.*

As mentioned earlier, two distinct nodes with the same parent node (that is not the bottom node $\bot$) are said to merge to form the parent node; to this end, the model includes an uninterpreted binary function, $\mathcal{M}$, that captures this relation between sister nodes that will be merged together, and is constrained by axioms that model the recursive structure building operation, *Merge*.[90][91] Given two nodes $x, y \in \mathbb{N}$, the phrase resulting from merging $x$ and $y$ is $\mathcal{M}(x,y)$, where $\mathcal{M}$ is defined to be an uninterpreted function with signature:

$$\mathcal{M} : \mathbb{N} \times \mathbb{N} \longrightarrow \mathbb{N} \tag{2.52}$$

Several axioms constrain interpretations of $\mathcal{M}$:

$$\operatorname{supp} \mathcal{M} \subseteq ((\mathbb{L}_{\mathbb{N}} \cup \mathbb{I}_{\mathbb{N}}) \cap \operatorname{supp} h)^2 \tag{2.53}$$

$$\operatorname{im}(\mathcal{M}) \subseteq \{\bot\} \cup (D \cap \operatorname{supp} h) \tag{2.54}$$

$$\mathcal{M} \text{ is a symmetric relation} \tag{2.55}$$

$$\bigwedge \left\{ (\mathcal{M}(x,y) \neq x) \wedge (\mathcal{M}(x,y) \neq y) \mid x, y \in \langle (\mathbb{L}_{\mathbb{N}} \cup \mathbb{I}_{\mathbb{N}})^2 \rangle \right\} \tag{2.56}$$

$$\bigwedge \left\{ \mathcal{M}(x,y) = \text{If}\,(p(x) = p(y), p(x), \bot) \;\mid x, y \in \langle (\mathbb{L}_{\mathbb{N}} \cup \mathbb{I}_{\mathbb{N}})^2 \rangle \right\} \tag{2.57}$$

$$\bigwedge \{ \mathcal{M}(x_i, x_j) = \bot \mid (x_i, x_j) \in (s \times s), s \in \mathbb{S}, i \neq j \} \tag{2.58}$$

Every node in the derivation besides the root node can be an argument of *Merge* (Ax.2.53). All non-lexical nodes in the derivation are produced by *Merge* (Ax.2.54). The output of *Merge* doesn't change if we permute its two arguments (Ax.2.55). If neither of the two arguments to *Merge* is $\bot$, then (in accordance with the ***Extension Condition***) the output of *Merge* cannot be one of the two arguments (Ax.2.56). Both arguments to merge have the same parent (Ax.2.57). In accordance with the property of *Universal Endocentricity*, a node with cannot merge with another node within the same projection (Ax.2.58).[92]

[90] See (Chomsky, 1995, Pg. 243) and (Collins and Stabler, 2016).

[91] The function $\mathcal{M}$ is not strictly necessary as it can be derived from the function $p$ and $\mathcal{P}$. However it has been included as it allows for axioms that will be presented downstream to be expressed more concisely.

[92] See (Chomsky et al., 2019, Pg. 247): *"The assumption of universal endocentricity carried over to the Bare Phrase Structure model of Chomsky 1995, where MERGE(X, Y) is taken to yield a labeled object* $\{L, \{X, Y\}, L \in \{X, Y\}\}$*."*

**Axioms for Categories and Extended Projections.** Categories are interpretable properties of lexical items that can project. The category that a lexical head and its projections may associated with is constrained by the particular structural configurations that the projections of that lexical head enter into (within a derivation); in particular, per Grimshaw's *Extended Functional Projections*, the structural configurations that a lexical head may enter into is constrained by two extended projections for the functional hierarchies $C > T > v > V$ and $P > D > N$.[93][94][95] Each node in the derivation is associated with a member of the *category* sort, $\mathfrak{C}$. Given a node $x \in \mathbb{N}$, the category associated with the head of $x$ is $\beta_{\mathbb{N}}(x) \in \mathfrak{C}$, where $\beta_{\mathbb{N}}$ is defined as an uninterpreted function with signature:

$$\beta_{\mathbb{N}} : \mathbb{N} \rightarrow \mathfrak{C} \tag{2.59}$$

Several axioms constrain interpretations of $\beta_{\mathbb{N}}$:

$$\bigwedge \{\beta_{\mathbb{N}}(x) = \beta_{\mathbb{N}}(\mathfrak{H}(x)) \mid x \in D\} \tag{2.60}$$

$$\bigwedge \{(\beta_{\mathbb{N}}(x) = \mathfrak{c}_{\varnothing}) = (h(x) = \bot) \mid x \in \mathbb{N}\} \tag{2.61}$$

$$(\beta_{\mathbb{N}}(R_{\mathbb{N}}) = \mathfrak{c}_{C_{Decl.}}) \vee (\beta_{\mathbb{N}}(R_{\mathbb{N}}) = \mathfrak{c}_{C_{Ques.}}) \tag{2.62}$$

The category associated with a given node in the derivation is the category associated with the head of that node (Ax.2.60). If a member of $\mathbb{N}$ does not play a role in the derivation, then that member is associated with the "null" category, $\mathfrak{c}_{\varnothing}$ (Ax.2.61). The root node is associated either with $\mathfrak{c}_{C_{Decl.}}$ in the case of a declarative, or with $\mathfrak{c}_{C_{Ques.}}$ in the case of an interrogative (Ax.2.62). Additionally, axioms encoding the (aforementioned) two extended projections are supplied, thus constraining what structural configurations the lexical heads may be arranged in within a valid MG derivation. To this end, we will first define two helper functions (entirely for convenience and brevity of exposition):

$$isComp(x, y) = (\mathcal{M}(x, y) \neq \bot) \wedge (h(p(x)) = h(x)) \tag{2.63}$$

$$projHier(\alpha, \beta) = \bigwedge \{(isComp(x, y) \wedge (\beta_{\mathbb{N}}(x) = \alpha)) \rightarrow (\beta_{\mathbb{N}}(y) = \beta) \mid (x, y) \in \langle \mathbb{L}_{\mathbb{N}} \times \mathbb{N} \rangle\} \tag{2.64}$$

Then the extended projections are encoded by the following two axioms:

$$projHier(\mathfrak{c}_{C_{Decl.}}, \mathfrak{c}_T) \wedge projHier(\mathfrak{c}_{C_{Ques.}}, \mathfrak{c}_T) \wedge projHier(\mathfrak{c}_T, \mathfrak{c}_v) \wedge projHier(\mathfrak{c}_v, \mathfrak{c}_V) \tag{2.65}$$

$$projHier(\mathfrak{c}_P, \mathfrak{c}_D) \wedge projHier(\mathfrak{c}_D, \mathfrak{c}_N) \tag{2.66}$$

Additionally, the functional heads associated with categories $C$, $T$, $v$, and $P$ must take on a complement:

$$\bigwedge \{(\beta_{\mathbb{N}}(x) = y) \rightarrow (h(p(x)) = h(x)) \mid x \in \mathbb{L}_{\mathbb{N}}, y \in \{\mathfrak{c}_C, \mathfrak{c}_T, \mathfrak{c}_v, \mathfrak{c}_P\}\} \tag{2.67}$$

[93] See (Grimshaw, 2005), (Adger, 2003, Pg. 333), and (Adger and Svenonius, 2011).

[94] See (Lust, 2006, Pg. 198) for a discussion of the "Functional Projection Hypothesis" .

[95] N.b. Categories and Extended Projections are an addition made to further constrain the space of grammars; they are not part of the original MG formalism and the associated axioms may be omitted if one wishes to be in strict agreement with the MG formalism.

and the lexical heads of merged structures must have distinct categories:

$$\bigwedge \left\{ (\beta_{\mathbb{N}}(x) = \beta_{\mathbb{N}}(y)) \rightarrow (\mathcal{M}(x,y) = \bot) \mid x, y \in \langle \mathbb{N}^2 \rangle \right\} \tag{2.68}$$

**Axioms for Syntactic Movement.** We will next introduce model axioms for (syntactic) movement, a fundamental feature of language that enables a word or phrase to be displaced from one position in a sentence (where it may receive a feature pertaining to its interpretation) to another. To this end, we will introduce an uninterpreted unary function, $\mathcal{P}$, that models the chains produced by the movement of phrases within the derivation, and an uninterpreted unary function, $\mathcal{H}$, that models head movement.

To begin, we will consider the raising of maximal projections. Given a node $x \in \mathbb{N}$, the phrase at $x$ is raised to $\mathcal{P}(x) \in \mathbb{N}$, where $\mathcal{P}$ is defined to be an uninterpreted function with signature

$$\mathcal{P} : \mathbb{N} \rightarrow \mathbb{N} \tag{2.69}$$

If the phrase at $x$ does not undergo raising then $\mathcal{P}(x) = \bot$. Several axioms constrain interpretations of $\mathcal{P}$:

$$\bigwedge \{\mathcal{P}(x) \neq x \mid x \in (\mathbb{L}_{\mathbb{N}} \cup D)\} \tag{2.70}$$

$$\operatorname{supp} \mathcal{P} \subseteq (\mathbb{L}_{\mathbb{N}} \cup \mathbb{I}_{\mathbb{N}}) \cap \operatorname{supp} h \tag{2.71}$$

$$\operatorname{im}(\mathcal{P}) \subseteq \{\bot\} \cup (\mathbb{I}_{\mathbb{N}} \cap \operatorname{supp} h) \tag{2.72}$$

$$\bigwedge \{\mathcal{P}(x) = y \rightarrow (h(x) \neq h(p(x))) \wedge (h(y) \neq h(p(y))) \mid x, y \in \langle (\mathbb{L}_{\mathbb{N}} \cup D) \times \mathbb{I}_{\mathbb{N}} \rangle\} \tag{2.73}$$

$$\bigwedge \{\mathcal{P}(x) = y \rightarrow d(p(y), p(x)) \mid x, y \in \langle (\mathbb{L}_{\mathbb{N}} \cup \mathbb{I}_{\mathbb{N}}) \times \mathbb{I}_{\mathbb{N}} \rangle\} \tag{2.74}$$

A phrase cannot move to itself and thus $\mathcal{P}$ has no fixed points other than $\bot$ (Ax.2.70). The phrase with the root node as its maximal projection cannot undergo movement (Ax.2.71). A phrase can only move to an intermediate node (Ax.2.72). Both the source and target of raising do not project (Ax.2.73). The target of phrasal movement c-commands the source (Ax.2.74).[96]

Next, we will consider the movement of minimal projections via head- movement, which is constrained by the ***Head-Movement Constraint*** as given in (Hale and Keyser, 1993, Pg. 55): *"A head X may only move to a head Y if Y properly governs X."*[97][98] Given a node $x \in \mathbb{N}$, the head of the phrase at $x$ moves to incorporate with the node at $\mathcal{H}(x) \in \mathbb{N}$, where $\mathcal{H}$ is defined to be an uninterpreted function with signature:

$$\mathcal{H} : \mathbb{N} \rightarrow \mathbb{N} \tag{2.75}$$

If the head at $x$ does not undergo movement then $\mathcal{H}(x) = \bot$. Several axioms constrain

[96] See (Chomsky, 1995, Pg. 253) for the "C-command condition."

[97] For further reference on the head-movement constraint, see (Baker, 1988) and (Stabler, 2001).

[98] N.b. Head movement is applied to the *derived tree* – this is because head-movement is meant to operate at the PF-interface rather than in the course of the derivation itself.

interpretations of $\mathcal{H}$:

$$\bigwedge\{\mathcal{H}(x) \neq x \mid x \in (\mathbb{L}_{\mathbb{N}} \cup D)\} \tag{2.76}$$

$$\operatorname{supp} \mathcal{H} \subseteq \mathbb{L}_{\mathbb{N}} \cap \operatorname{supp} h \tag{2.77}$$

$$\operatorname{im}(\mathcal{H}) \subseteq \{\bot\} \cup (\mathbb{L}_{\mathbb{N}} \cap \operatorname{supp} h) \tag{2.78}$$

$$\bigwedge\{(\mathcal{H}(x) = y) \rightarrow (h(p(y)) = y) \mid x, y \in \langle \mathbb{L}_{\mathbb{N}} \times \mathbb{L}_{\mathbb{N}} \rangle\} \tag{2.79}$$

$$\bigwedge\{(\mathcal{H}(x) = y) \rightarrow (\mathcal{P}(x) = \bot) \mid x, y \in \langle \mathbb{L}_{\mathbb{N}} \times \mathbb{L}_{\mathbb{N}} \rangle\} \tag{2.80}$$

$$\bigwedge\{(\mathcal{H}(x) = y) \rightarrow d(p(y), x) \mid x, y \in \langle \mathbb{L}_{\mathbb{N}} \times \mathbb{L}_{\mathbb{N}} \rangle\} \tag{2.81}$$

$$\bigwedge\{((\mathcal{H}(x) = y) \wedge (\mathcal{M}(y, z) \neq \bot)) \rightarrow (h(z) = x) \mid x, y, z \in \langle \mathbb{L}_{\mathbb{N}} \times \mathbb{L}_{\mathbb{N}} \times \mathbb{I}_{\mathbb{N}} \rangle\} \tag{2.82}$$

Only lexical nodes can undergo head-movement (Ax.2.77), and only lexical nodes may move to incorporate with other lexical nodes (Ax.2.78), so that $\mathcal{H}$ has no fixed points other than $\bot$ (Ax.2.76). The target location of head-movement (i.e. the receiving lexical head) projects (Ax.2.79) and the parent of the target lexical head dominates the source lexical head (Ax.2.81). A node that undergoes head-movement cannot also undergo phrasal movement (Ax.2.80). Head-movement is a local operation: a lexical head $x$ can only be raised to merge (via head-movement) with a lexical head $y$ if $y$ merges with the maximal projection of $x$ (Ax.2.82).

Finally, we will present axioms that further bound the model by limiting the number of instances of movement that may occur in a derivation – these axioms are entirely optional and are included for the sake of performance considerations (with respect to the runtime of the SMT-solver).

$$k_p \geq \sum_{x \in (\mathbb{L}_{\mathbb{N}} \cup D)} (\mathcal{P}(x) \neq \bot) \tag{2.83}$$

$$k_h \geq \sum_{x \in \mathbb{L}_{\mathbb{N}}} (\mathcal{P}(x) \neq \bot) \tag{2.84}$$

$$\bigwedge\{(\mathcal{H}(x) = y) \rightarrow (\mathcal{H}(y) = \bot) \mid x, y \in \langle \mathbb{L}_{\mathbb{N}} \times \mathbb{L}_{\mathbb{N}} \rangle\} \tag{2.85}$$

(Ax.2.83) sets an upper bound, $k_p$, on the number of instances of phrasal movement that occur in a derivation. (Ax.2.84) imposes an upper bound, $k_h$, on the number of instances of head-movement that occur in a derivation. (Ax.2.85) serves to restrict successive head-movement.

**Axioms for Bounding Derivation Node Sequences.** Finally, let us consider a number of axioms that serve to establish boundary conditions and otherwise establish the general structure of the derivation node sequences; additionally, these axioms have a substantive impact on the (runtime) performance of the SMT-solver when checking the model of the minimalist parser.

$$\bigwedge \{(h(x_i) = \bot) \rightarrow (h(x_j) = \bot) \mid (x_i, x_j) \in (s \times s), s \in \mathbb{S}, i < j\} \tag{2.86}$$

$$\bigwedge \{(p(x) = x_{i+1}) \vee (\mathcal{P}(x) = x_{i+1}) \vee (h(x_{i+1}) = \bot) \mid x_i \in s, s \in \mathbb{S}, i < k\} \tag{2.87}$$

$$\bigwedge \{(\mathcal{P}(x_i) = x_{i+1}) \vee (\mathcal{P}(x_i) = \bot) \mid x_i \in s, s \in \mathbb{S}, i < k\} \tag{2.88}$$

$$\bigwedge \{p(x_i) \neq x_j \mid (x_i, x_j) \in (s \times s), s \in \mathbb{S}, i + 1 \neq j\} \tag{2.89}$$

$$\bigwedge \{\mathcal{P}(x_k) = \bot \mid s \in \mathbb{S}\} \tag{2.90}$$

Within a derivation node sequence, the nodes that appear in the derivation form a contiguous sub-sequence that must start with the first node in the derivation node sequence (Ax.2.86). A node sequence is a sequence consisting of projections or links in the chain established by successive instances of movement (of a maximal projection): given a node sequence of length $k$, the parent of a node, $x$, must appear immediately after $x$ within the sequence; likewise, the target node of phrasal movement must appear immediately after the source node of movement (Ax.2.88, Ax.2.87 and Ax.2.89). Phrasal movement is prohibited from appearing at the end of a node sequence (Ax.2.90).

### 2.3.3 Constraining the Derivation Model with Interface Conditions

A derivation outputs syntactic objects that are interfaces to the parts of the mind that process meaning and sound – i.e. the Conceptual-Intentional system (CI) and the Sensory-Motor system (SM);[99] in this way, a derivation can be said to “link” sound and meaning together.[100] A Logical Form (LF) is a representation produced by the derivation that serves as an interface between the derivation and CI. LF includes information pertaining to case-marking, quantifier-scope, predicate argument structure[101] and other (syntactic) information that plays a role in interpreting the meaning of a sentence. A Phonetic Form (PF) is a representation produced by the derivation that serves as an interface between the derivation and SM. PF includes information pertaining to word-ordering, pronunciation, and the application of morphological rules. In accordance with the Principle of Full Interpretation, derivations must produce LF and PF interfaces that are interpretable by the CM and SM systems respectively – i.e. the interfaces provide relevant information and instructions to these systems.

It is sometimes desirable to constrain what interpretations the derivation may have, which in turn requires constraining the logical forms and phonetic forms produced by the derivation by way of stipulating conditions that the interfaces must satisfy – i.e. *interface*

[99] The Sensory-Motor system is sometimes referred to as the *Articulatory-Perceptual* system (A-P).

[100] See (Chomsky, 1995, Pg. 2): *“A more specific assumption is that the cognitive system interacts with just two such ’external’ systems: the articulatory-perceptual system A-P and the conceptual-intentional system C-I. Accordingly, there are two interface levels, Phonetic Form (PF) at the A-P interface and Logical Form at the C-I interface. This ’double interface’ property is one way to express the traditional description of language as sound with a meaning traceable at least back to Aristotle.”* For further discussion see (Adger, 2003, Pg. 145) and (Radford, 2016, Pg. 25-26).

[101] Argument structure here is defined as in (Hale and Keyser, 2002, Pg. 1): *“We use the term argument structure to refer to the syntactic configuration projected by nuclear items. While a lexical entry is more than this, of course, argument structure in the sense intended here is nothing other than this. Argument structure is determined by properties of lexical items, in particular, by the syntactic configurations in which they must appear. There are just two syntactic relations, complement and specifier, defined so as to preclude iteration and to permit only binary branching.“*

*conditions.* (See Table 2.4 for examples of paired LF and PF interface conditions) The constraints (expressed as SMT-formulae) that are derived from LF and PF interface conditions only reference components of the derivation model, and do not reference the lexicon model; this is because the MG formalism lacks interpretable features (i.e. features that are interpretable by the CI or SM systems). Instead derivations are to be interpreted by examining the structural configurations entered into by the lexical heads participating in the derivation.[102] The interface conditions translate into requirements on what syntactic relations are established within a derivation; since all syntactic relations are (local) structural relations established via *merge*, the interface conditions are in effect structural constraints over the derivation. See Fig. 2-12 for a detailed illustration of how interface conditions are expressed as structural constraints over a derivation.

A full specification of both LF and PF interface conditions can constrain the space of derivations that the lexicon may yield more than is required – i.e. a partial specification of the interface conditions would have sufficed to guarantee that the lexicon yields the same derivation as it would if the interface conditions were fully stipulated. Importantly, this property can be employed to our benefit: the model can be used in various ways by partially constraining the interfaces in various ways – e.g. when parsing, the PF interface conditions are available (as the surface form constitutes the expression to be parsed) but the LF interface conditions may not be available, whereas in the externalization of thoughts, it may be that the LF interface conditions (encoding meaning) are specified, but the PF interface conditions are not specified. The model of the minimalist parser supports several types of basic LF and PF interface conditions from which a partial specification of interface conditions can be composed; We will now introduce the SMT-formulae derived from these basic LF and PF interface conditions.

### Axioms for LF Interface Conditions

The axioms encoding LF interface conditions model constrain the derivation so as to restrict interpretations thereof at the LF interface, in turn restricting what logical forms may be associated with that derivation. We will now introduce the constraints associated with each of the four types of LF interface conditions that the system supports.

**Predicate Argument Structure.** The axioms pertaining to predicate argument structure serve to establish local syntactic relations (via merge) between a predicate (i.e. a lexical verb) and an argument (i.e. a phrase). The system supports three types of arguments: external arguments, internal arguments that serve as direct objects, and internal arguments that serve as indirect objects. The head of a phrase that is an argument of a predicate is restricted to associating with the following categories:

$$\mathfrak{C}_{arg} = \{P, D, N, C_{Declarative}, C_{Question}\} \tag{2.91}$$

The predicate is located within a (double) VP-shell structure that consists of two projections: a lower projection, the head of which is the predicate and is associated with the category $V$, and a higher projection, the head of which is a light-verb and is associated with the category

[102] See (Adger and Svenonius, 2011, :) *"In sum, some syntactic features might have interpretations at the SM interface, though there are no uncontroversial examples of this. In the case of linearization, the most likely examples involve different options in the spell-out of chains and of heads."*

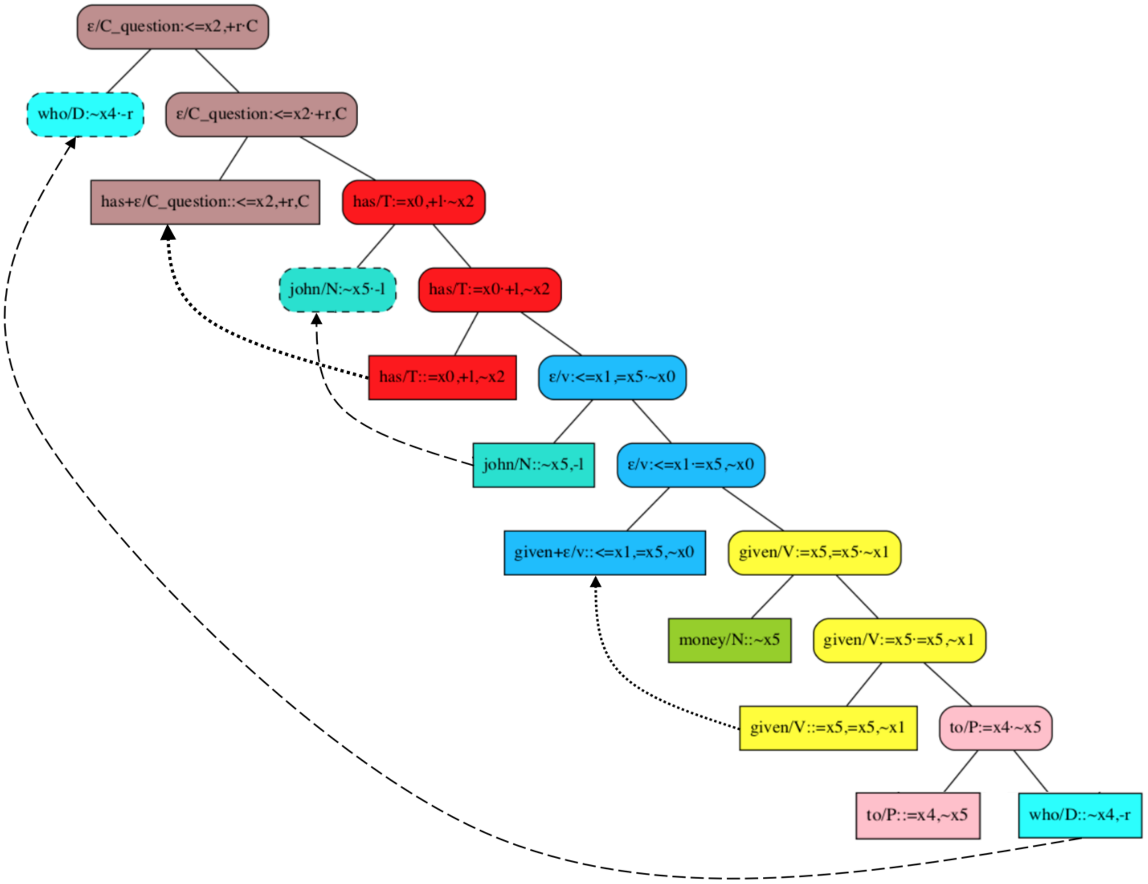


Figure 2-12: An MG derivation for the (interrogative) sentence *"Who has John given money to?"* that aligns with the prescriptions of contemporary theories of syntax; see $I_{13}$ in Table-3.2 for the interface conditions satisfied by this derivation. The feature sequences displayed in non-leaf nodes have a dot, $\cdot$, separating features that have already been consumed (on the left) from those that have not (on the right). The dashed arrows denote phrasal movement. The dotted arrows denote head movement. Nodes with the same *head* have the same color. The derivation is assembled in a bottom-up manner via merge: *"given"* merges with *"to who"* (formed by first merging *"to"* and *"who"*) and then with *"money"*, thus establishing (via locality) predicate-argument relations; the resulting structure merges with a functional head – i.e. a covert light verb – before undergoing $V$-to-$v$ head-movement and then merging with the argument *"john"* in accordance with the VP-Internal Subject Hypothesis (Hale and Keyser, 2002); the resulting structure then merges with the auxiliary verb *"has"*, after which the argument *"john"* undergoes subject-raising from the VP-shell by (internally) merging with *"has"*, thus establishing an agreement relation between *"john"* and *"has"*; next, the head of *"has"* undergoes $T$-to-$C$ head-movement to merge with the covert complementizer, $\epsilon/C_{question}$, which indicates that the sentence is an interrogative; finally, *"who"* undergoes wh-fronting by (internally) merging with $\epsilon/C_{question}$. Wh-fronting (of *"who"*) and Subject-raising (of *"john"*), instances of A'-movement and A-movement respectively, are triggered by different licensor features, the former by $+r$ and the latter by $+l$. Note that This figure was taken from an earlier presentation of the work in this thesis by (Indurkhya, 2020).

$v$.[103] Following the theory of argument structure presented in (Hale and Keyser, 1993) and further developed in (Hale and Keyser, 2002), predicate-argument structure is determined by the particular structural configuration entered into by the lexical heads associated with the predicate and its arguments, and not by the thematic role associated with the arguments[104] – i.e an external argument merges with the light-verb per the VP Internal Subject Hypothesis (VPISH),[105] and internal arguments merge with the lexical verb. (See Figure 2-13 for details of how the different types of arguments are located within the double-VP shell structure.)

We will begin by defining a number of utility macros (i.e. functions) that will be employed in the axioms that we will later define. Given node sequence $s$, $s[i]$ refers to the $i^{th}$ element of the sequence. The node sequence associated with a word at position $i$ is denoted $S_i^+$. Let $W$ be a set of words indexed by position in the sentence - i.e. $w_i \in W$ is the $i^{th}$ word in the sentence. The set of overt lexical nodes associated with $W$ is denoted:

$$\mathfrak{A}(W) = \{S_i^+[0] | w_i \in W\} \tag{2.92}$$

Given a set $Y$ of overt lexical nodes involved in the argument phrase (as in formula 2.92), and a derivation node $x$, the following function checks both that the head of $x$ is contained in $Y$, and that $x$ dominates each member $Y$ with respect to either the derivation tree or the derived tree:

$$\mathbb{H}(x, Y) = \left( \bigvee_{y \in Y} (h(x) = y) \right) \wedge \left( \bigwedge_{y \in Y} ((x = y) \vee d(x, y) \vee d^\star(x, y)) \right) \tag{2.93}$$

This function serves to establish that $x$ is the (raised maximal) projection of the head of the (argument) phrase.

We will now introduce constraints that establish a local relation (via merge) between a phrase that serves as an *external argument* of a verb phrase (VP) that serves as the predicate. The phrase serving as the external argument phrase is translated into a set of overt lexical nodes, denoted $A$; importantly, information pertaining to the (linear) ordering of the overt phonological forms associated with the nodes in $A$ is not included. The VP-shell structure is made up of the node-sequence for the projection of the light verb, $s_v$, and the derivation node associated with the maximal projection of the lexical verb, `p`. Given a derivation node `a`, the following formula holds true if: (i) there is a local relation established (via merge) between `p` and `a`, and (ii) `a` is the (raised) maximal projection of the head of a

[103] See (Radford, 2009, Pg. 408): "The VP-shell analysis maintains that verb phrases can be split into (at least) two separate projections, a lower one headed by a lexical verb, and a higher one headed by a light verb."

[104] See (Hale and Keyser, 1993): *"Our basic answer to question (21a) — why there are so few thematic roles — is that, in an important sense, there are no thematic roles. Instead there are just the relations determined by the categories and their projections, and these are limited by the small inventory of lexical categories and by Unambiguous Projection. While we might assign a particular thematic label — say, "agent" — to the NP in (22), its grammatical status is determined entirely by the relation(s) it bears in the relational structure projected by the lexical head V."*

[105] See (Radford, 2009, Pg. 408): "VPISH in the hypothesis that subjects originate internally within the verb phrase."

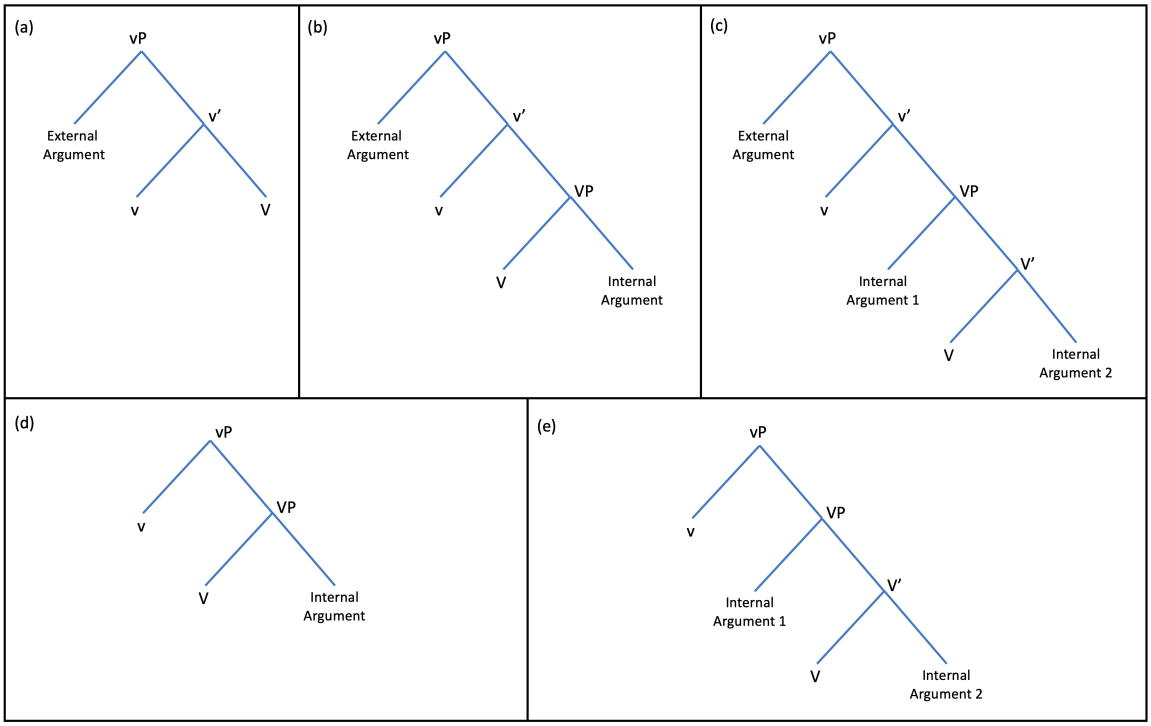


Figure 2-13: Configurations of the Double VP-shell Structure that encode argument structure for various types of predicates: (a) intransitive verb; (b) active-voice transitive verb; (c) active-voice ditransitive verb; (d) passive-voice transitive verb; (e) passive-voice ditransitive verb. The Double VP-shell Structure is made up of the projection of two lexical heads – i.e. the projection of a (lexical) verb (i.e. $V$) that takes up to two internal arguments, and the lexical projection of a (typically covert) light verb (i.e. the functional head $v$) that takes up a single external argument. Arguments are interpreted as associating with thematic roles based on their hierarchical position within the Double-VP shell structure; this association aligns with the *Uniformity of θ-Assignment Hypothesis* (UTAH). Specifically, an external argument is associated with the thematic role of *agent*, a internal argument serving as the direct object is associated with the thematic role of *theme*, and a internal argument serving as the indirect object is associated with the thematic role of *goal*. (See (Adger, 2003, Pgs. 138-139) for a review of UTAH and thematic roles.) Per the *VP Internal Subject Hypothesis*, one of the arguments will be raised to Spec-TP and thereby become the (structural) subject of the sentence.

phrase containing $A$.

$$\begin{aligned} Arg_e(\mathtt{p}, \mathtt{a}, A, s_v) = & \,(\beta_{\mathbb{N}}(s_v[0]) = v) \\ & \wedge (\beta_{\mathbb{N}}(\mathtt{p}) = V) \\ & \wedge (\mathtt{a} \in \mathfrak{C}_{arg}) \\ & \wedge \mathbb{H}(\mathtt{a}, A) \\ & \wedge (\mathcal{M}(s_v[0], \mathtt{p}) = s_v[1]) \\ & \wedge (\mathcal{M}(s_v[1], \mathtt{a}) = s_v[2]) \end{aligned} \tag{2.94}$$

the first two terms of the conjunction – i.e. $(\beta_{\mathbb{N}}(s_v[0]) = v)$ and $(\mathtt{a} \in \mathfrak{C}_{arg})$ – ensure that the predicate and the light verb are associated with categories $V$ and $v$ respectively; the third term – i.e. $(\beta_{\mathbb{N}}(\mathtt{p}) = V)$ – ensures that the argument has an appropriate category for arguments (see 2.91); the fourth term – i.e. $\mathbb{H}(\mathtt{a}, A)$ – ensures that the argument node is the (raised) maximal projection of the head of the argument phrase – i.e. the top-most-node of the argument phrase structure is what merges with the predicate; the fifth term – i.e. $(\mathcal{M}(s_v[0], \mathtt{p}) = s_v[1])$ – merges the little-v and V to form the double-VP shell structure, ensuring that the light verb projects and that $\mathtt{p}$ is the maximal projection of the lexical verb and is the complement of the light verb; the sixth term – i.e. $(\mathcal{M}(s_v[1], \mathtt{a}) = s_v[2])$ – ensures the argument phrase merges into the specifier position of the projection of the light verb.

We can now explicitly quantify over the different possible local relations between nodes in predicate associated derivation node sequences ($\mathtt{p}_{idxs}$ indexes this set of derivation node sequences) and nodes in argument associated derivation node sequences ($\mathtt{a}_{idxs}$ indexes this set of derivation node sequences) that can be established, checking to see if (2.94) holds in any of these cases:

$$\bigvee_{\substack{i_{\mathtt{a}} \in \mathtt{a}_{idxs} \\ i_{\mathtt{p}} \in \mathtt{p}_{idxs} \\ i_{\mathtt{a}} \neq i_{\mathtt{p}}}} \left( \bigvee \{ Arg_e(\mathtt{p}, \mathtt{a}, \mathfrak{A}(W_{arg}), s_v) \mid \mathtt{a} \in S^+_{i_{\mathtt{a}}}, \mathtt{p} \in S^+_{i_{\mathtt{p}}}, s_v \in S^- \} \right) \tag{2.95}$$

This formula has two levels of disjunctions, intentionally factored apart (for purposes of exposition) so that one can point out what each is iterating over: the outer disjunction is quantifying over the possible assignments of heads of the argument phrase and predicate; the inner disjunction is quantifying over the possible assignments of indices in the associated node sequences (so as to pick particular derivation nodes within each node sequence).[106]

Next, we will introduce constraints pertaining to a phrase serving as an internal argument that corresponds to the direct object, establishing a local relation with the projection of the lexical verb serving as the predicate; these constraints are similar to those for the external argument, except that internal arguments merge with the projection of the lexical verb, and thus there is no need to directly reference the node sequence associated with the projection of the light verb $s_v$.

$$\begin{aligned} Arg_{i1}(\mathtt{p}, \mathtt{a}, A) = {} & (\beta_{\mathbb{N}}(\mathtt{p}) = V) \\ & \wedge (\mathtt{a} \in \mathfrak{C}_{arg}) \\ & \wedge (h(p(\mathtt{p})) = h(\mathtt{p})) \\ & \wedge \mathbb{H}(\mathtt{a}, A) \\ & \wedge (\mathcal{M}(\mathtt{a}, \mathtt{p}) \neq \bot) \end{aligned} \tag{2.96}$$

The first term in the conjunction – i.e. $(\beta_{\mathbb{N}}(\mathtt{p}) = V)$ – ensures that the predicate-node (i.e. the node associated with the lexical verb) has category $V$; the second term – i.e. $(\mathtt{a} \in \mathfrak{C}_{arg})$ – ensures that the argument node has an appropriate argument category (see 2.91); the third term – i.e. $(h(p(\mathtt{p})) = h(\mathtt{p}))$ – ensures that the predicate projects; the fourth term – i.e. $\mathbb{H}(\mathtt{a}, A)$ – ensures that the argument node is the (raised) maximal projection of the head of the argument phrase. the fifth term – i.e. $(\mathcal{M}(\mathtt{a}, \mathtt{p}) \neq \bot)$ – ensures that the argument and the predicate will merge. It is then possible, as it was in the case of an external argument, to explicitly quantify over the different possible local relations between predicate associated nodes and argument associated nodes that can be established, checking to see if (2.96) holds

[106] This is an example of explicit quantification.

in any of these case:

$$\bigvee_{\substack{i_{\mathtt{a}} \in \mathtt{a}_{idxs} \\ i_{\mathtt{p}} \in \mathtt{p}_{idxs} \\ i_{\mathtt{a}} \neq i_{\mathtt{p}}}} \left( \bigvee \{ Arg_{i1}(\mathtt{p}, \mathtt{a}, \mathfrak{A}(W_{arg})) \,|\, \mathtt{a} \in S^{+}_{i_{\mathtt{a}}}, \mathtt{p} \in S^{+}_{i_{\mathtt{p}}}[:2] \} \right) \tag{2.97}$$

Finally we will introduce constraints that pertain to a phrase serving as an internal argument that corresponds to the indirect object, establishing a local relation with the lexical verb serving as the predicate:

$$\begin{aligned} Arg_{i2}(\mathtt{p}, \mathtt{a}, A) = (&\beta_{\mathbb{N}}(\mathtt{p}) = V) \\ &\wedge (\mathtt{a} \in \mathfrak{C}_{arg}) \\ &\wedge (h(p(\mathtt{p})) = h(\mathtt{p})) \\ &\wedge \mathbb{H}(\mathtt{a}, A) \\ &\wedge (\mathcal{M}(\mathtt{a}, \mathtt{p}) \neq \bot) \end{aligned} \tag{2.98}$$

As before, we can explicitly quantify over the different possible local relations between predicate associated nodes and argument associated nodes that can be established, checking to see if (2.98) holds in any of these cases:

$$\bigvee_{\substack{i_{\mathtt{a}} \in \mathtt{a}_{idxs} \\ i_{\mathtt{p}} \in \mathtt{p}_{idxs} \\ i_{\mathtt{a}} \neq i_{\mathtt{p}}}} \left( \bigvee \{ Arg_{i2}(\mathtt{p}, \mathtt{a}, \mathfrak{A}(W_{arg})) \,|\, \mathtt{a} \in S^{+}_{i_{\mathtt{a}}}, \mathtt{p} \in S^{+}_{i_{\mathtt{p}}}[:2] \} \right) \tag{2.99}$$

We do not differentiate between whether a node coding for an internal argument serves as a direct object or an indirect object here in the axioms as this distinction should be determined via selectional features and the position of the argument within the VP-shell structure. All three types of arguments can be headed by an empty argument node by changing "$\mathtt{a} \in S^{+}_{i_{\mathtt{a}}}$" to "$\mathtt{a} \in s_{arg}, s_{arg} \in \left(S^{-} \cup \{S^{+}_{i_{\mathtt{a}}}\}\right)$" (thereby including the covert node sequences $S^{-}$) in formulas (2.95), (2.97), and (2.99) respectively. E.g. in the case of the internal argument taking an embedded sentence headed by a covert complementizer (e.g. *"Bob told Mary the dog ran away"*) or an embedded complement with inifinitival-"to" (e.g. *"Bob told Mary to eat pizza"*), formula (2.99) may be modified to:

$$\bigvee_{\substack{i_{\mathtt{a}} \in \mathtt{a}_{idxs} \\ i_{\mathtt{p}} \in \mathtt{p}_{idxs} \\ i_{\mathtt{a}} \neq i_{\mathtt{p}}}} \left( \bigvee \{ Arg_{i2}(\mathtt{p}, \mathtt{a}, \mathfrak{A}(W_{arg})) \,|\, \mathtt{a} \in s_{arg}, s_{arg} \in \left(S^{-} \cup \{S^{+}_{i_{\mathtt{a}}}\}\right), \mathtt{p} \in S^{+}_{i_{\mathtt{p}}}[:2] \} \right) \tag{2.100}$$

**Agreement.** Any of the three types of arguments can be raised to structural subject position (i.e. Spec-T) and thereby establish an agreement relation between the (structural) subject (i.e. a phrase serving as an argument that originated in the VP-shell) and a tense-marker (e.g. an auxiliary verb).[107] The following formula is true only if there is a locality relation established via *merge* between the subject (denoted by "$\mathtt{s}$" below) and the tense-

[107] See (Adger, 2003, Pgs. 167-170) for a review of agreement within minimalist syntax in the context of checking theory.

marker (denoted by "$\mathtt{t}$" below) it agrees with:

$$\begin{aligned} Agr_{Subj}(\mathtt{t}, \mathtt{s}, A) = (h(p(\mathtt{t})) &= h(\mathtt{t})) \\ &\wedge (\mathcal{M}(\mathtt{s}, \mathtt{t}) \neq \bot) \\ &\wedge \mathbb{H}(\mathtt{s}, A) \end{aligned} \tag{2.101}$$

The first two terms in the conjunction serves to ensure that when the subject and the tense-marker it agrees with merge together, and that the tense-marker projects through the merge operation; the third term serves to ensure that it is the (raised) maximal projection of the head of the subject (phrase) that merges with the tense-marker. Observe that, as compared with axioms (2.95), (2.97), and (2.99) that establish predicate-argument structure, axiom (2.99) does not restrict what category the subject or the tense-marker must associate with. It is then possible, as it was in the case of the external argument, to explicitly quantify over the different possible (local) relations between nodes in tense-marker associated derivation node sequences ($\mathtt{t}_{idxs}$ indexes over this set of derivation node sequences) and subject associated nodes ($\mathtt{s}_{idxs}$ indexes over this set of derivation node sequences) that can be established, checking to see if the formula above holds in any case:

$$\bigvee_{\substack{i_{\mathtt{s}} \in \mathtt{s}_{idxs} \\ i_{\mathtt{t}} \in \mathtt{t}_{idxs} \\ i_{\mathtt{s}} \neq i_{\mathtt{t}}}} \Big( \bigvee \{ Agr_{Subj}(\mathtt{t}, \mathtt{s}, \mathfrak{A}(W_{subj})) \,|\, \mathtt{s} \in S^{+}_{i_{\mathtt{s}}}, \mathtt{t} \in S^{+}_{i_{\mathtt{t}}} \} \Big) \tag{2.102}$$

Axiom (2.102) is more general than axioms (2.95), (2.97), and (2.99) in so far as it quantifies over all of the nodes in the node-sequences for the subject (phrase) and the tense-marker; as in the case of the axioms for establishing predicate-argument structure, axiom (2.102) can be modified to allow the subject (phrase) to be headed by an empty lexical item by changing "$\mathtt{s} \in S^{+}_{i_{\mathtt{s}}}$" to "$\mathtt{s} \in s_{arg}, s_{arg} \in \big(S^{-} \cup \{S^{+}_{i_{\mathtt{s}}}\}\big)$."

**Sentence Type.** End-of-sentence punctuation marks each sentence as declarative or interrogative: this annotation is translated into a constraint that determines which one of the two pre-specified categories, $C_{Declarative}$ or $C_{Question}$, is associated with the root node of the derivation (of the sentence). Consequently, the system distinguishes between derivations for declaratives and interrogatives sentences by inspecting the category associated with the head of the matrix clause. The system assumes that a declarative or interrogative is headed by a covert lexical head.[108][109] Define a variable $C_X \in \mathfrak{C}$ and assign it to either $\mathfrak{c}_{C_{Decl.}}$ or $\mathfrak{c}_{C_{Ques.}}$ depending on whether the sentence is a declarative or interrogative respectively; then the following axiom requires that the lexical head that projects to become the root node has

[108] See (Adger, 2003, Pg. 333): *"We proposed that C is the place where the interpretable-clause-type feature is found. This feature determines whether a CP is interpreted as a question or as a declarative statement. This feature is also responsible for valuing an uninterpretable-clause-type feature on T: when this feature is strong, as in the case of Q in English, then T-to-C movement takes place, so that auxiliary inverts over the subject."*

[109] See the *Interrogative Condition* stipulated in (Radford, 2009, Pg. 161) for languages such as English: *"A clause is interpreted as a non-echoic question if (and only if) it is a CP with an interrogative specifier (i.e. a specifier containing an interrogative word)."*

category $C_X$.

$$\bigwedge_{x \in \mathbb{L}_{\mathbb{N}}^{-}} ((h(R_{\mathbb{N}}) = h(x)) \wedge (h(x) = x) \wedge (\beta_{\mathbb{N}}(x) = C_X)) \tag{2.103}$$

**Categorical Constraints.** Given the constraint that the word $w_i$ at position $i$ has category $c_i$, let $S_i^+$ be the overt node sequence associated with $w_i$, and let $x$ be the first node in $S_i^+$; then the model is constrained by adding the following axiom:

$$\beta_{\mathbb{N}}(S_i^+) = c_i \tag{2.104}$$

Looking ahead, the ability to designate the category that a token in an expression in is associated with will play an important role in the acquisition models presented in Ch.3.

### Axioms for PF Interface Conditions

The axioms encoding PF interface conditions model the effects of externalizing the derivation to the SM system, central to which is the linearization of the overt lexical items, which must match the (linear) surface ordering prescribed by conditions on the PF interface.[110] These axioms may be broken into three groups: (i) axioms for relating the derivation tree to the derived tree (both of which are substructures of the associated multi-dominance tree); (ii) axioms that encode the linear ordering, prescribed by the supplied PF interface conditions, over the lexical heads; (iii) axioms that require that the linear ordering of overt lexical heads established by the derived tree matches the linear ordering prescribed by the supplied PF interface conditions. We will now introduce and detail each of these groups of axioms in turn.

Computing the linear ordering relations between the lexical heads entering into a derivation requires that the effects of syntactic movement be taken into account; this is accomplished by modeling the *derived tree*, in which all constituents are raised to their final position.[111][112][113] Recall that the axioms previously introduced encode a derivation tree, and in particular provide a predicate, $d$, that imposes a strict partial ordering over supp($h$). In order to derive the derived tree from the derivation tree, we will introduce a new predicate – i.e. an uninterpreted function – that will serve to impose a strict partial ordering over supp($h$) that encodes the derived tree:

$$d^{\star} : \mathbb{N} \times \mathbb{N} \rightarrow Boolean \tag{2.105}$$

[110] See (Chomsky, 2013b, Pg. 36): *"Order and other arrangements are a peripheral part of language, related solely to externalization at the SM interface, where of course they are necessary."*

[111] For a more formal treatment of how the derived tree is connected to the derivation tree, see (Kobele, 2011) and (Stabler, 2013, Appendix B).

[112] The decision to introduce and detail the axioms for the derived tree and the derivation tree separately aligns with an observation made in (Stabler, 2013, Pg. 613): *"[...] it is natural to regard MGs as folding together two different sorts of principles: those that define the derivation tree, and those that map the derivation to its pronounced and interpreted form. One insight that comes from this perspective is that both of those steps, the definition of derivations (like the tree on the right, above), and the mappings to derived structures (like the tree on the left), are very simple, finite state computations.“*

[113] N.B. the modeling of the surface structure here diverges from contemporary minimalist syntax in that there is no operation "Spellout" that is applied at each stage of the derivation (i.e. in between merge operations); rather, the surface structure is constructed in parallel to the derivation tree, and may be thought of as one single large PF interface. See (Collins and Stabler, 2016, Pg. 75) for further discussion.

Given $x, y \in \mathbb{N}$, $d^\star(x, y)$ denotes that $x$ dominates $y$ via one of either $p$ or $\mathcal{P}$, after all phrasal movement has taken place. In contrast to $d$, $d^\star$ models dominance relations encoded in the derived tree and is referred to as *derived-tree-dominance* (as this dominance relation is with respect to the derived tree in which all constituents have moved to their final destination). Several axioms that will constrain interpretations of $d^\star$:

$$d^\star \text{ is irreflexive, anti-symmetric and transitive over } (\operatorname{supp} h)^2 \tag{2.106}$$

$$\operatorname{supp} d^\star \subseteq (D \cap \operatorname{supp} h) \times ((\mathbb{L}_{\mathbb{N}} \cup \mathbb{I}_{\mathbb{N}}) \cap \operatorname{supp} h) \tag{2.107}$$

$$\bigwedge \{(p(x) \neq \bot) \rightarrow d^\star(p(x), x) \oplus d^\star(\mathcal{P}(x), x) \mid x \in \mathbb{N}\} \tag{2.108}$$

$$\bigwedge \{\mathcal{P}(x) \neq \bot \rightarrow d^\star(\mathcal{P}(x), x) \mid x \in \mathbb{N}\} \tag{2.109}$$

$$\bigwedge \{\neg d(x_i, x_j) \wedge \neg d^\star(x_i, x_j) \mid (x_i, x_j) \in (s \times s), s \in \mathbb{S}, i \leq j\} \tag{2.110}$$

The $d^\star$ is a (binary) predicate that is irreflexive, transitive and asymmetric relation over $(\operatorname{supp} h \times \operatorname{supp} h)$, it is therefore a strict partial ordering over $\operatorname{supp} h$ (Ax.2.106). Lexical nodes cannot derived-tree-dominate a derived node (Ax.2.107). Axiom (2.108) requires that every active node in the derived tree dominated (within the derived tree) either by its parent or by the target-location of movement; axiom (2.109) requires that phrasal movement implies derived-tree-dominance. Within a derivation node sequence, a node cannot derived-tree-dominate another node that appears prior to it within the same node sequence (Ax.2.110).

The linear ordering (over the words in the sentence) prescribed by the supplied PF interface conditions are modeled using a precedence relation – i.e. an uninterpreted binary predicate – over pairs of (leaf) nodes in the *derived* tree:[114]

$$\mathcal{L} : \mathbb{N} \times \mathbb{N} \rightarrow \mathit{Bool} \tag{2.111}$$

Given two lexical (derivation) nodes $x, y \in L$, $\mathcal{L}(x, y)$ denotes that the lexical head $x$ precedes the lexical head $y$ with respect to the (surfaced) linear ordering. The interpretation of $\mathcal{L}$ is constrained by several axioms:

$$\operatorname{supp} \mathcal{L} \subseteq (\mathbb{L}_{\mathbb{N}} \cap \operatorname{supp} h) \times (\mathbb{L}_{\mathbb{N}} \cap \operatorname{supp} h) \tag{2.112}$$

$$\bigwedge \{\mathcal{L}(x, y) = (i < j) \mid x_i, x_j \in ({\mathbb{L}_{\mathbb{N}}}^+ \times {\mathbb{L}_{\mathbb{N}}}^+)\} \tag{2.113}$$

These axioms impose a particular sequential ordering, derived from the ordered sequence of phonological forms found in the PF interface conditions, over the leaf nodes in the derived tree. The first axiom, (2.112), establishes that the precedence operation pertains to the ordering of lexical heads (both overt and covert); the second axiom, (2.113), establishes the pairwise ordering relations between the *overt* lexical heads.

Since each overt node-sequence in a derivation model is indexed by the position of the word (in the sentence) that it codes for[115], the PF interface conditions determine what phonetic feature each overt node sequence is associated with. To this end, we will now

[114] The focus is on the derived tree because we want to determine precedence relations between constituents in the derivation *after* syntactic movement has taken place.

[115] I.e. the representative node in the node sequence codes for a lexical head that is in turn associated with a particular word in the sentence.

introduce an uninterpreted function that will model the association between node sequences and phonological forms. Let $W$ be a set of words indexed by position in the sentence – i.e. $w_i \in W$ is the $i^{th}$ word in the sentence. Recall that the (overt) derivation node sequence associated with $w_i$ is denoted $S_i^+$, and that $S_i^+[0]$ refers to the first element of the derivation node sequence. Define $\Delta_{\mathbb{N}}$ to be an uninterpreted function with signature:

$$\Delta_{\mathbb{N}} : \mathbb{N} \to \Sigma \tag{2.114}$$

Then the following axioms constrain interpretations of $\Delta_{\mathbb{N}}$:

$$\bigwedge \{\Delta_{\mathbb{N}}(S_i[0]) = \Sigma_{w_i} \mid w_i \in W\} \tag{2.115}$$

$$\bigwedge \left\{\Delta_{\mathbb{N}}(s[0]) \neq \Sigma_x \mid s \in S^-, \Sigma_x \in \Sigma_o\right\} \tag{2.116}$$

$$\Delta_{\mathbb{N}}(\bot) = \varnothing_{\Sigma} \tag{2.117}$$

Axiom (2.115) constrains $\Delta_{\mathbb{N}}$ so as to associate each word $w_i \in W$ with an overt derivation node sequence; by convention, this association is made between a phonetic feature and the *first* member of the derivation node sequence (which codes for a lexical head). Axiom (2.116) entails that a covert derivation node sequence cannot associate with an overt phonetic feature.[116] Axiom (2.117) establishes a boundary condition requiring that the "null" node of the derivation node sort associate with the "null" node of the PF node sort.

Finally, we will introduce axioms that require that the ordering of the (overt) leaf nodes in the derived tree agree with the ordering over the words in the sentence (per the specified the PF interface conditions). The following axioms require that a specifier precedes its head and that a head precedes its complement (which accords with the SVO ordering of English); in this way, linear ordering is derived from the structure of the derivation.

$$\begin{aligned}
&\forall x, y \in \langle (\mathbb{L}_{\mathbb{N}} \cup \mathbb{I}_{\mathbb{N}})^2 \rangle \\
&\quad \{(\mathcal{M}(x, y) \neq \bot) \\
&\quad\quad \wedge (h(x) = h(p(x))) \\
&\quad\quad \wedge (\mathcal{P}(y) = \mathcal{H}(h(x)) = \mathcal{H}(h(y)) = \bot)\} \\
&\quad\quad \to \text{If}\left(\bigvee_{z \in \mathbb{L}_{\mathbb{N}}} (x = z), \mathcal{L}(h(x), h(y)), \mathcal{L}(h(y), h(x))\right)
\end{aligned} \tag{2.118}$$

Axiom (2.118) establishes, for every two non-root nodes $x$ and $y$ in the derivation, a conditional; the antecedent in the conditional is true if $x$ and $y$ merge together, $x$ projects, $y$ is not raised, and neither $x$ nor $y$ will undergo head-movement – if the antecedent holds, then the consequent being true implies that if the $x$ is a lexical head, then $x$ precedes $y$ (corresponding to $EM_1$), and $y$ precedes $x$ otherwise (corresponding to $EM_2$).

$$\begin{aligned}
&\forall x, y, z \in \langle (\mathbb{L}_{\mathbb{N}} \cup D) \times \mathbb{I}_{\mathbb{N}} \times (\mathbb{L}_{\mathbb{N}} \cup \mathbb{I}_{\mathbb{N}}) \rangle \\
&\quad\quad \{d^\star(x, y) \wedge d^\star(y, z) \wedge (\mathcal{H}(h(x)) = \mathcal{H}(h(y)) = \mathcal{H}(h(z)) = \bot)\} \\
&\quad\quad\quad \to (\mathcal{L}(h(x), h(y)) = \mathcal{L}(h(x), h(z))) \wedge (\mathcal{L}(h(y), h(x)) = \mathcal{L}(h(z), h(x)))
\end{aligned} \tag{2.119}$$

[116] A covert derivation node sequence need not associate with a covert phonological form as not every covert derivation node sequence actively participates in the derivation.

Axiom (2.119) has three quantifiers $x$, $y$ and $z$, with $x$ quantifying over the nodes in the derivation, $y$ quantifying over the nodes in the derivation that are neither one of the lexical nodes nor the root node, and $z$ quantifying over the nodes in the derivation that are not the root node. The antecedent of the conditional is true if none of $x$, $y$, or $z$ undergo head-movement, and, with respect to the *derived tree*, $x$ contains $y$ and $y$ contains $z$, which implies that $x$ is distinct from the surfaced phrase containing $y$ and $z$. The consequent of the conditional being true implies that the lexical heads $y$ and $z$ must both either precede or succeed the lexical head $x$, which accords with $y$ and $z$ being part of a single phrase that either comes before or after $x$.

$$\begin{aligned}&\forall x, y \in \langle (\mathbb{L}_{\mathbb{N}} \cup \mathbb{I}_{\mathbb{N}})^2 \rangle \\ &\quad \{(d^{\star}(p(x), x) \wedge d^{\star}(p(x), y)) \\ &\quad\ \wedge (\neg d(x, y) \wedge \neg d(y, x) \wedge (p(x) \neq p(y))) \\ &\quad\ \wedge (\mathcal{H}(h(y)) = \perp)\} \\ &\quad\ \rightarrow \mathcal{L}(h(x), h(y))\end{aligned} \tag{2.120}$$

Axiom (2.120) requires both that: (i) a lexical head $x$ precedes every node $y$ strictly contained (with respect to the derived tree) by the maximal projection of the head of the *complement* of $x$, and (ii) if the maximal projection of a lexical head $x$ merges into the specifier position of the projection for a lexical head $v$, then $x$ precedes every node $y$ strictly contained (with respect to the derived tree) by the sister of $x$ (i.e. a projection of $v$). To see this, observe that the axiom explicitly quantifies over distinct pairs of non-root nodes $x$ and $y$ in the derivation – the antecedent is true if: (i) the parent of $x$ strictly contains (with respect to the derived tree) both $x$ and $y$; (ii) $x$ and $y$ do not merge and do not dominate one another with respect to the derivation tree; (iii) $y$ does not undergo head movement (although $x$ might). (In effect, the antecedent requires that $x$ c-command $y$.)

$$\begin{aligned}&\forall x^{'}, x, y \in \langle (\mathbb{L}_{\mathbb{N}} \cup \mathbb{I}_{\mathbb{N}})^3 \rangle \\ &\quad \{(d^{\star}(p(x), x) \wedge d^{\star}(p(x), y) \wedge d^{\star}(x, x^{'})) \\ &\quad\ \wedge (\neg d(x, y) \wedge \neg d(y, x) \wedge (p(x) \neq p(y)) \\ &\quad\ \wedge (\mathcal{H}(h(y)) = \perp)\} \\ &\quad\ \rightarrow \mathcal{L}(h(x^{'}), h(y))\end{aligned} \tag{2.121}$$

Axiom (2.122) is related to (2.120) in that it asserts that under the conditions of (2.122), if there is a third node $x^{'}$ that is strictly contained (with respect to the derived tree) by $x$, then $x^{'}$ will precede $y$ – this is motivated by the observation that the entire phrase contained by $x$ with respect to the derived tree (which does not contain $y$) precedes $y$.

$$\bigwedge \{(\mathcal{H}(x) = y) \rightarrow \mathcal{L}(x, y) \mid x, y \in \langle \mathbb{L}_{\mathbb{N}} \times \mathbb{L}_{\mathbb{N}} \rangle\} \tag{2.122}$$

$$\bigwedge \{(\mathcal{H}(x) = y) \rightarrow (\neg(\mathcal{L}(x, z) \wedge \mathcal{L}(z, y)) \wedge \neg(\mathcal{L}(y, z) \wedge \mathcal{L}(z, x))) \mid x, y, z \in \langle \mathbb{L}_{\mathbb{N}}^3 \rangle\} \tag{2.123}$$

Finally, axioms (2.122) and (2.123) pertain to head movement. Supposing lexical head $x$ is raised to $y$ (via head-movement), then: axiom (2.122) establishes that all head-movement is suffixing – i.e. the lexical head $y$ must precede the lexical head $x$ – and axiom (2.123) establishes that there can be no third lexical head $z$ (distinct from $x$ and $y$) such that $z$

comes between $x$ and $y$ (with respect to the linear ordering of the lexical heads).

### 2.3.4 Connecting the Derivation Model to the Lexicon Model

The trajectory of a lexical head through a derivation as it first (optionally) projects and then (optionally) undergoes raising is captured via the *node-sequence* data structure; relatedly, the sequence of interactions within a derivation that a lexical item undergoes is dictated by the *lexical (syntactic) feature sequence* component of the associated lexical head. This raises the possibility of developing a mapping between derivation nodes and lexicon nodes that maps each *(derivation) node sequence* associated with a lexical head in the derivation to a *lexical (syntactic) feature sequence* associated with a lexical entry in the lexicon, thereby *connecting* the derivation model to the lexicon model. To this end, the model includes an (uninterpreted) function that connects the model of the derivation to the model of the lexicon and axioms constraining this function so as to establish the aforementioned mapping.[117] See Fig. 2-14 for an illustration of how the members of a derivation node sequence are connected to members of a lexicon node sequence. We will now develop this notion formally.

To begin, we will introduce an uninterpreted function that maps nodes in the derivation to nodes in the lexicon:

$$\mu : \mathbb{N} \to \Omega \tag{2.124}$$

This function will serve to map node sequences in the derivation with node sequences in the lexicon. Several axioms constrain interpretations of $\mu$:

$$\bigwedge \{(x \in \ker h) \to (\mu(x) = \omega_{\varnothing}) \mid x \in \mathbb{N}\} \tag{2.125}$$

$$\bigwedge \{\Delta_{\Omega}(\mu(x), \Delta_{\mathbb{N}}(x)) \mid x \in (\mathbb{L}_{\mathbb{N}} \cup D)\} \tag{2.126}$$

$$\bigwedge \left\{\Gamma(\mu(y)) = \bigvee \{\mathcal{H}(x) = y \mid x \in \mathbb{L}_{\mathbb{N}}\} \mid y \in \mathbb{L}_{\mathbb{N}}\right\} \tag{2.127}$$

$$\bigwedge \{(h(x) \neq h(p(x)) \neq \bot) \to (\psi(\mu(x)) = \text{If}\,(\mathcal{P}(x) \neq \bot, \mu(\mathcal{P}(x)), \omega_{\varnothing})) \mid x \in (\mathbb{L}_{\mathbb{N}} \cup D)\} \tag{2.128}$$

$$\bigwedge \{(\mu(x) = \omega_C) \to (x = R_{\mathbb{N}}) \mid x \in (\mathbb{L}_{\mathbb{N}} \cup D)\} \tag{2.129}$$

$$\xi(\mu(R_{\mathbb{N}})) = \text{If}\,(h(R_{\mathbb{N}}) = \bot, \tau_{\varnothing}, \tau_c) \tag{2.130}$$

$$\bigwedge \{\beta_{\mathbb{N}}(x) = \beta_{\Omega}(\mu(x)) \mid x \in \mathbb{L}_{\mathbb{N}}\} \tag{2.131}$$

Members of the derivation node sort that do not play a role in the derivation map to the terminal lexicon node, $\omega_{\varnothing}$ (Ax.2.125). A derivation node sequence maps to a lexicon node sequence only if both node sequences associate with the same phonological form – i.e. given a node $x$ in the derivation, the lexicon node that $x$ maps to (via $\mu$) must associate with the phonological form that $x$ associates with (Ax.2.126). The special selectional feature for head movement, <=, appears in a lexical entry if and only if the associated lexical head is the target of head movement (Ax.2.127). Non-projecting derivation nodes either undergo movement or terminate their projection chain (Ax.2.128). Only the root node can map to

[117] Note that a direct mapping between the derivation nodes and lexicon nodes comports with modern incarnations of minimalist syntax that allow dispensing with the concept of a numeration – i.e. a subset of the lexicon that will be merged together to form a derivation is not required to be explicitly selected prior to the derivation. See (Chomsky et al., 2019) for a discussion of this point.

the complete lexicon node (Ax.2.129), and the root node maps to the complete lexicon node if and only if the derivation converges (Ax.2.130). A lexical node in the derivation has the same category as the lexicon node it maps to (Ax.2.131).

$$\bigwedge \{\mathcal{P}(x) = y \rightarrow \xi(\mu(y)) = \tau_{-} \mid (x, y) \in \langle (\mathbb{L}_{\mathbb{N}} \cup D) \times (\mathbb{L}_{\mathbb{N}} \cup D)\rangle\} \tag{2.132}$$

$$\bigwedge \{\xi(\mu(x)) \neq \tau_{+} \wedge \xi(\mu(x)) \neq \tau_{-} \mid x \in \mathbb{L}_{\mathbb{N}}\} \tag{2.133}$$

$$\bigwedge \{\mathcal{M}(x, y) \neq \bot \rightarrow \kappa(\mu(x)) = \kappa(\mu(y)) \mid (x, y) \in \langle (\mathbb{L}_{\mathbb{N}} \cup D) \times (\mathbb{L}_{\mathbb{N}} \cup D)\rangle\} \tag{2.134}$$

$$\begin{aligned} \forall x, y \in & \langle (\mathbb{L}_{\mathbb{N}} \cup D) \times (\mathbb{L}_{\mathbb{N}} \cup D)\rangle \\ & ((\mathcal{M}(x, y) \neq \bot) \wedge (h(x) = h(p(x)))) \\ & \rightarrow ((\xi(\mu(x)) = \tau_{=}) \wedge (\xi(\mu(y)) = \tau_{\sim})) \vee ((\xi(\mu(x)) = \tau_{+}) \wedge (\xi(\mu(y)) = \tau_{-})) \end{aligned} \tag{2.135}$$

$$\begin{aligned} \forall x, y \in & \langle (\mathbb{L}_{\mathbb{N}} \cup D) \times (\mathbb{L}_{\mathbb{N}} \cup D)\rangle \\ & ((\mathcal{M}(x, y) \neq \bot) \wedge (h(x) = h(p(x)))) \rightarrow \psi(\mu(x)) = \mu(\mathcal{M}(x, y)) \end{aligned} \tag{2.136}$$

If a node in the derivation is the target location of phrasal movement then the lexicon node it maps to must code for a licensee feature (Ax.2.132). Lexical nodes in the derivation cannot map to lexicon nodes involved in licensing. (Ax.2.133) When two nodes in the derivation are merged together, they must both associate with the same (non-nil) syntactic feature label (Ax.2.134); either the argument that projects is a selector while the other argument is a selectee or the argument that projects is a licensor while the other argument is a licensee (Ax.2.135). Finally, given two derivation nodes, $x_1$, and $x_2$, that are the arguments of merge, if $x_1$ projects, then $p(x_1) = \mathcal{M}(x_1, x_2)$ and the functions $\mu$, $p$ and $\psi$ form a commutative diagram – i.e. $\psi(\mu(x_1)) = \mu(p(x_1))$ (Ax.2.136).

## 2.4 Parsing

This section introduces a procedure for parsing MGs (listed on Pg. 86) that uses the SMT-model of a minimalist parser (defined in §2.3), in conjunction with an SMT-solver. The input to the parser consists of: (i) a specification of LF and PF interface conditions to be satisfied (see Table 2.4 for examples of such specifications); (ii) a valuation of model parameters that serves to bound the (finite) model of the parser (e.g. see Table 2.5); (iii) a minimalist lexicon consisting of a set of lexical entries (e.g. see Table 2.1). The output of the parser is a minimalist derivation that may be yielded by the specified lexicon and that satisfies the specified LF and PF interface conditions. We used the Z3 SMT-solver (v. 4.8.6), a state-of-the-art high-performance SMT-solver with widespread application that can both (i) evaluate (check) whether an SMT-model has a satisfiable interpretation – i.e. a satisfiable assignment of values to the free variables in the formula that makes up the decision problem – and (ii) identify a satisfiable interpretation of the model in the case that it has one.[118]

This section is organized as follows. We will begin by detailing (in §2.4.1) how to construct an SMT-model that encodes the (decidable) decision problem of whether the specified minimalist lexicon can yield a (minimalist) derivation that satisfies the specified interface conditions. We will then detail (in §2.4.2) how this decision problem may be evaluated by using an SMT-solver to check the model, and in the case that there is a satisfiable interpretation of the model (implying that the decision problem may be answered in the affirmative),

[118] In the case of the SMT-model of the parser, identifying a satisfiable interpretation of the model requires identifying an interpretation of each uninterpreted function that accords with the model axioms.

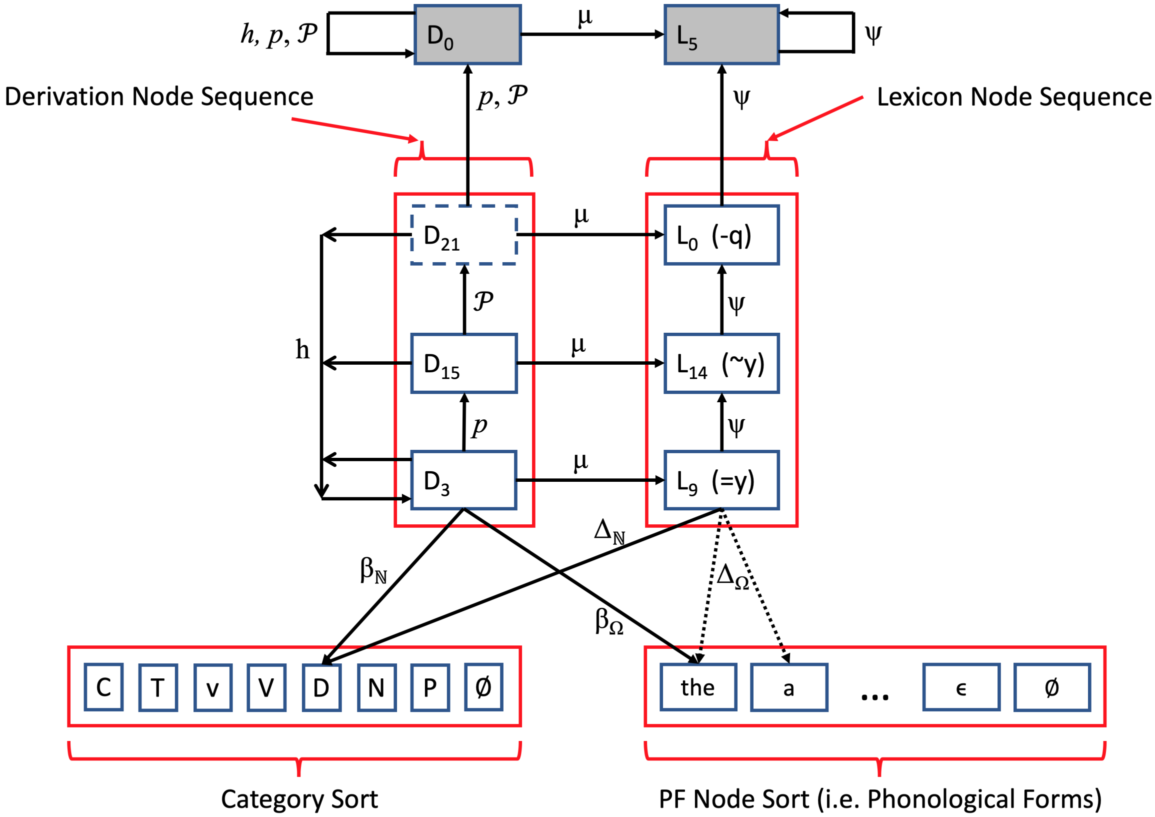


Figure 2-14: Connecting a Derivation Model to the Lexicon Model. This diagram illustrates how the members of the derivation node sequence originating at node $D_3$ in the derivation presented in Fig. 2-2 are mapped (via the uninterpreted function $\mu$) to the members of the lexicon node sequence corresponding to the lexical entry $[the/D ::= y, {\sim}y, -q]$ based on the model interpretation presented in Table 2.6. The mapping ensures that the trajectory the lexical head takes through the derivation (i.e. via projection and raising) is dictated by the lexical feature sequence associated with the corresponding lexical entry. The head (i.e. $h$) of each member of the derivation node sequence is the first derivation node in the sequence (i.e. the lexical head $D_3$). The first member in each sequence is associated with a member of the category sort (i.e. $D$); likewise, the first member of the derivation node sequence is associated with a phonological form (i.e. "the") via the map $\Delta_{\mathbb{N}}$. However the first member of the lexicon node sequence can associate with more than one phonological form via the map $\Delta_{\Omega}$ (in this case both "the" and "a"); this mapping is visually distinguished (from other mappings) in this illustration via the use of a dotted arrow. Each member of the lexicon node sequence is connected to its successor via the uninterpreted function $\psi$. Each member of the derivation node sequence is connected to the next via either the function $p$ or the function $\mathcal{P}$. The last members of the derivation node sequence and the lexicon node sequence – i.e. $D_{21}$ and $L_0$ respectively – map to the "nullary" nodes for the derivation node sort (i.e. $D_0$) and lexicon node sort (i.e. $L_5$) respectively. Note that $D_0 = \bot$ and $L_5 = \omega_{\varnothing}$. Importantly, each derivation node sequence in the derivation model is mapped, via the function $\mu$, to one of the lexicon node sequences (encoding a syntactic feature sequence) in the lexicon model.

how a minimalist derivation – i.e. the output of the parser – may be automatically recovered from the interpretation of the model, thereby carrying out the task of parsing. Finally, we will evaluate the parser (in §2.4.3) by using it to parse each entry in Table 2.4 using the lexicon listed in Table 2.1 and produce the respective derivations listed in Table **??**. Along the way, we will illustrate how each step of the *parsing procedure* works using a running example – i.e. we will show how the *parsing procedure* can take as input an entry, $I_1$, listed in Table 2.4 and the lexicon listed in Table 2.1 and output the corresponding derivation (for $I_1$) listed in Table **??**. (See Fig. 2-2 for a presentation of this derivation)

### 2.4.1 Constructing the Model of the Parser

Following Step-2 of the *parsing procedure*, the model is constructed by first initializing a lexicon model from the specified minimalist lexicon (see Step-3a), then initializing a derivation model and restricting interpretations of it by via introduction of constraints derived from the specified LF and PF interface conditions (see Step-2b), and then finally connecting the lexicon model to the derivation model (see Step-2c). The SMT-model constructed by the procedure employs a *quantifier-free*, multi-sort, first-order logic extended with the theory of uninterpreted functions, and the associated decision problem is thereby *decidable*; consequently, given enough time, the SMT-solver will eventually determine whether or not the model has a satisfiable interpretation. We will now detail each of these three steps in turn.

#### Constructing the Lexicon Model

The procedure processes the supplied lexicon and produces an SMT-formula that models the specified lexicon; this is accomplished in two stages.

In the first stage, the procedure processes the specified (input) lexicon and (automatically) determines the size (i.e. the cardinality) of each of three sorts that are associated with the lexicon model: the *lexicon node sort*, ($\Omega$), the *syntactic feature label sort* ($\mathbb{F}$), and the sort of phonological forms ( $\Sigma$). The size of the lexicon node sort, $\Omega$, is computed using the upper bound listed in equation (2.11), which is a function of product of the number of lexical entries in the lexicon and the maximum number of syntactic features in any one lexical entry in the specified lexicon; in the case of our running example, the lexicon has a total of 34 lexical entries, with a lexical entry having at most 3 syntactic features (e.g. see lexical entries 2, 3, 10, 12, 14, 15, 17, 22, 23, 29, 30, and 31 in Table 2.1), giving:

$$|\Omega| = 2 + (34)(3) = 104 \tag{2.137}$$

The sets of selectional and licensing feature labels in the lexicon – in the case of our running example, $\mathbb{F}_S = \{x, y\}$ and $\mathbb{F}_L = \{p, q\}$ respectively – are used to compute the size of the syntactic feature label sort, $\mathbb{F}$, per equation (2.12):

$$|\mathbb{F}| = |\mathbb{F}_S| + |\mathbb{F}_L| + 1 = 2 + 3 + 1 = 6 \tag{2.138}$$

(Note that there are 2 distinct selectional feature labels, 2 distinct licensing feature labels, and 1 null syntactic feature, $\varnothing_{\mathbb{F}}$.) The set of phonological forms that appears in the lexicon is used to construct the sort of phonological forms (i.e. $\Sigma$); in the case of our running example, there are a total of 23 distinct overt phonological forms, one covert phonological

| $I_i$ | Interface Conditions | |
|---|---|---|
| $I_1$ | PF: | what has the man/N eaten/V? |
| | LF: | $\theta_{\text{eaten}}[s\colon \text{the man}, o\colon \text{what}]$, $\mathcal{A}_{\text{has}}[s\colon \text{the man}]$ |
| $I_2$ | PF: | was she/N given/V money/N? |
| | LF: | $\theta_{\text{given}}[o\colon \text{money}, i\colon \text{she}]$, $\mathcal{A}_{\text{was}}[s\colon \text{she}]$ |
| $I_3$ | PF: | who will tell/V her/N that he/N has resigned/V? |
| | LF: | $\theta_{\text{tell}}[s\colon \text{who}, o\colon \text{that he has resigned}, i\colon \text{her}]$, $\mathcal{A}_{\text{will}}[s\colon \text{who}]$, $\theta_{\text{resigned}}[s\colon \text{he}]$, $\mathcal{A}_{\text{has}}[s\colon \text{he}]$ |
| $I_4$ | PF: | she/N has known/V everyone/N who was loved/V. |
| | LF: | $\theta_{\text{known}}[s\colon \text{she}, o\colon \text{everyone who was loved}]$, $\mathcal{A}_{\text{has}}[s\colon \text{she}]$, $\theta_{\text{loved}}[o\colon \text{everyone}]$, $\mathcal{A}_{\text{was}}[s\colon \text{everyone}]$ |
| $I_5$ | PF: | she/N knows/V that john/N has given/V money/N. |
| | LF: | $\theta_{\text{knows}}[s\colon \text{she}, o\colon \text{that john has given money}]$, $\theta_{\text{given}}[s\colon \text{john}, o\colon \text{money}]$, $\mathcal{A}_{\text{has}}[s\colon \text{john}]$ |
| $I_6$ | PF: | john/N has given/V money/N that was stolen/V. |
| | LF: | $\theta_{\text{given}}[s\colon \text{john}, o\colon \text{money that was stolen}]$, $\mathcal{A}_{\text{has}}[s\colon \text{john}]$, $\theta_{\text{stolen}}[o\colon \text{money}]$, $\mathcal{A}_{\text{was}}[s\colon \text{money}]$ |
| $I_7$ | PF: | john/N fears/V everyone/N who knows/V her/N. |
| | LF: | $\theta_{\text{fears}}[s\colon \text{john}, o\colon \text{everyone who knows her}]$, $\theta_{\text{knows}}[s\colon \text{everyone}, o\colon \text{her}]$ |
| $I_8$ | PF: | john/N fears/V that money/N was stolen/V. |
| | LF: | $\theta_{\text{fears}}[s\colon \text{john}, o\colon \text{that money was stolen}]$, $\theta_{\text{stolen}}[o\colon \text{money}]$, $\mathcal{A}_{\text{was}}[s\colon \text{money}]$ |

Table 2.4: Parser Input: Corpus of Paired (LF and PF) Interface Conditions. PF Interface conditions provide surface order data, and some phonological forms are associated with a specified category (indicated by a slash followed by the category). LF Interface Conditions include relations for agreement (denoted by $\mathcal{A}$) and predicate-argument structure (denoted by $\theta$, with the predicate indicated in the suffix); the LF interface conditions are entirely hierarchical/structural in the constraints they impose – i.e. the values filling the slots consist of *sets* of tokens, not sequences of tokens. A predicate is associated with one or more arguments: an external argument is preceded by "s:", an internal argument serving as a direct object is preceded by "o:", and an internal argument serving as an indirect object is preceded by an "i:". Entries with an embedded clause (i.e. $I_3$, $I_4$, $I_5$, $I_6$, $I_7$, and $I_8$) have LF interface conditions stipulated for each clause. The type of the sentence – i.e. either declarative or interrogative – is also marked in each sentence, indicated by the end-of-sentence punctuation.

## Parsing Procedure

1. The input supplied to the procedure consists of:
   - (a) a lexicon, $l$;
   - (b) a pair of (LF and PF) interface conditions, $I$, that will be parsed;
   - (c) a valuation of model parameters;
   - (d) an *empty* SMT-solver stack, $S$, with each entry on the stack an SMT-formula. (Note that to “check the model” is to use the SMT-solver to check the conjunction of the terms on the solver’s stack.)

2. Construct the SMT-model of the parser.
   - (a) Construct the model of the lexicon and constrain it with the supplied (input) lexicon:
     - i. initialize a lexicon model (i.e. an SMT formula), $m_l$, from the supplied model parameters;
     - ii. push $m_l$ onto the stack;
     - iii. construct an SMT-formula, $c_l$, that restricts interpretations of the lexicon model to align with the supplied (input) lexicon;
     - iv. push $c_l$ onto the stack.
   - (b) Construct the model of the derivation and constrain it so as to satisfy the interface conditions:
     - i. initialize a derivation model (i.e. an SMT formula), $m_d$, from model parameters and interface conditions $I$;
     - ii. push $m_d$ onto the stack;
     - iii. translate the interface conditions $I$ into an SMT–formula, $m_I$, that constrains the derivation model $m_d$;
     - iv. push $m_I$ onto the stack;
   - (c) Connect the model of the derivation to the model of the lexicon:
     - i. construct an SMT-formula, $m_b$, that connects, via shared symbols, the derivation model, $m_d$, to the lexicon model, $m_l$;
     - ii. push $m_b$ onto the stack;

3. Use an SMT-solver to check the SMT-model of the parser for a satisfiable interpretation; if a satisfiable interpretation of the model is found, recover a derivation from the interpretation.
   - (a) (optionally) optimize the model of the parser with respect to economy considerations by adding axioms (2.144) and (2.145) onto the stack;
   - (b) check the model of the parser using the SMT-solver, and if the model is found to be satisfiable, recover the identified (satisfiable) model interpretation;
   - (c) if a satisfiable interpretation of the model (i.e. a solution to the model) is identified:
     - i. recover the interpretation from the solver;
     - ii. (automatically) recover a derivation from the interpretation.

4. The output of the procedure is:
   - (a) if a satisfiable interpretation of the model was found, the output is a derivation, $d$, that can be yielded by the supplied (input) lexicon and that satisfies the specified interface conditions;
   - (b) if no satisfiable interpretation of the model was found, then the supplied (input) lexicon cannot yield a derivation that both comports with the supplied model parameters and that also satisfies the interface conditions.

form ($\epsilon$), and one "null" phonological form ($\varnothing_{\Sigma}$) so that:

$$|\Sigma| = 23 + 1 + 1 = 25 \tag{2.139}$$

Having determined the size of these three sorts, the uninterpreted functions and axioms that make up the lexicon model (as defined in §2.3.1) can be instantiated; the procedure then pushes the SMT-formula that makes up the lexicon model onto the SMT-solver's stack (of formulas).[119]

In the second stage, the procedure translates each lexical item listed in the lexicon into an SMT-formula that constrains the lexicon model – i.e. the conjunction of these formulae is appended to the lexicon model so that the specified lexicon is hte only lexicon that may be recovered from a satisfiable interpretation of the model. Intuitively, the lexicon model is being "hard-coded" with the values from the specified lexicon – i.e. the procedure is, in effect, stipulating what the interpretation of the model must be, and the solver's only task (when checking the model) is to ensure that the interpretation is consistent with the model axioms. Hard-coding the interpretation of a lexicon node sequence (in the lexicon model) involves: (i) valuing the feature for each node in the lexical node sequence, and (ii) connecting the phonological forms to the lexical node sequence. Let us see how the interpretation of a given a lexicon node sequence $[L_i, L_{i+1}, L_{i+2}, L_{i+3}]$ is constrained to match lexical entry 23 in Table 2.1 (i.e. who::=$x$, $+p$, $\sim y$) via the introduction of two constraints (i.e. SMT-formulae). The first constraint (2.140) assigns both the feature label and the feature type of each of the three syntactic features in the lexical entry to the three (successive) nodes in the lexicon node sequence:

$$\begin{aligned}&((\kappa(L_i) = \mathbb{F}_x) \wedge (\xi(L_i) = \tau_{=}))\\ &\quad \wedge ((\kappa(L_{i+1}) = \mathbb{F}_p) \wedge (\xi(L_{i+1}) = \tau_{+}))\\ &\quad \wedge ((\kappa(L_{i+2}) = \mathbb{F}_y) \wedge (\xi(L_{i+2}) = \tau_{\sim}))\end{aligned} \tag{2.140}$$

The second constraint (2.141) associates the head of the lexicon node sequence (here $L_i$) with and only with the member of the sort of phonological forms that corresponds to the phonological form present in lexical entry (i.e. "who"):

$$\{\Delta_{\Omega}(L_i, \Sigma_k) = (\Sigma_k = \Sigma_{who})|\ \forall \Sigma_k \in \Sigma\} \tag{2.141}$$

The procedure then pushes the conjunction of these two SMT-formulae onto the SMT-solver's stack.

### Constructing the Derivation Model

The parser then processes the supplied (input) lexicon and interface conditions, producing an SMT-formula that models a derivation that satisfies the supplied interface conditions. This happens in two stages.

In the first stage, a derivation model (as defined in §2.3.2) is instantiated from the model-parameters, which are used to determine the size of the derivation node sort. Some model-parameters are (automatically) derived from either the supplied interface conditions (e.g. the number of overt lexical heads is the number of words in the tokenized sentence

[119] The Z3 SMT-solver maintains a stack of formulas that may be pushed to and popped from; at any point the SMT-solver can check (evaluate) the conjunction of all of the terms then on the stack.

listed in the PF interface condition), the supplied lexicon (e.g. the maximum number of syntactic features associated with a lexical entry); the other model parameters must be supplied by the user - e.g.: the number of covert lexical heads; the maximum number of instances of head movement; the maximum number of instances of phrasal movement; the maximum length of a node-sequence (see Table 2.5).[120] The SMT-formulae that make up the derivation model are pushed onto the SMT-solver's stack.

| ID | Model Parameter | Description |
|---|---|---|
| (a) | Max. Num. Empty Lex. Items | Upper bound on the number of empty (i.e. unpronounced) lexical heads that may participate in a derivation. |
| (b) | Max. Num. Phrasal Movements | Upper bound on the number of (phrasal) movements (i.e. internal merge operations) that occur in a derivation. |
| (c) | Max. Num. Head Movements | Upper bound on the number of head movement operations that occur in a derivation. |

Table 2.5: Model Parameters for the Parsing Procedure. These parameters are all finite and serve to bound the model of the minimalist derivation. In the case of parsing the pairing of LF and PF interface conditions $I_1$ through $I_8$ in Table 2.1, parameters (a), (b) and (c) have values 6, 6 and 4 respectively.

In the second stage, each of the supplied interface conditions are translated into an SMT-formula that constrains the derivation model (as detailed in §2.3); the conjunction of these formulae is pushed onto the SMT-solver's stack. Our running example (i.e. entry $I_1$ in Table 2.4) illustrates the four types of LF interface conditions that our model currently supports, and how they are expressed as SMT-formulae that constrain the model:

- **Predicate-Argument Structure.** Each predicate-argument relation is translated into a local relation established by merging two constituents in the derivation[121], one of which is a projection of the lexical head serving as the predicate, and the other of which is the earliest node in the derivation (with respect to a bottom-up ordering of the nodes in the associated multi-dominance tree) that contains the set of specified overt lexical heads that make up the argument phrase[122] – e.g. in the derivation of $I_1$ in Fig. 2-2, the argument "what" is an internal argument of the predicate "eaten" because the lexical head for "what" ($D_1$) establishes a local relation (via EM) with the lexical head for "eaten" ($D_5$);
- **Agreement.** An agreement relation is likewise translated into a locality condition that must hold between two constituents – e.g. in the derivation for $I_1$ as presented in

[120] Alternatively, some of the model parameters supplied by the user can instead be determined via heuristics – e.g. the maximum number of instances of head-movement and phrasal movement operations permissible is bounded below by the number of predicates stipulated in the LF interface conditions (see §3.3 for further discussion).

[121] This is in accordance with Hale and Keyser's theory of argument structure (Hale and Keyser, 1993; Hale and Keyser, 2002), which requires that the lexical heads associated with a predicate and its arguments must enter into particular structural configurations within a derivation.

[122] This containment may either be with respect to the derivation tree via $d$ (in the case that the argument phrase has not undergone movement) or with respect to the *derived* tree via $d^\star$ (in the case that the argument phrase has undergone movement).

Fig. 2-2, the phrase {"the", "man"} establishes a locality relation with “has” because the constituent $D_{21}$, which contains via $d^\star$ (i.e. with respect to the derived tree, as the phrase “the man” undergoes movement) the lexical heads $D_3$ and $D_4$, merges with $D_{13}$, which is a projection of the lexical head $D_2$;

- **Sentence Type.** The end-of-sentence punctuation that marks each sentence as declarative or interrogative is translated into a constraint that stipulates which one of the two pre-specified categories, $C_{Decl.}$ or $C_{Ques.}$, is associated with the head of the entire derivation (i.e. the lexical head that projects to the root node) – e.g. since $I_1$ is an interrogative, the derivation for $I_1$ in Fig. 2-2 has the lexical head $D_6$ marked with the category $C_{Ques.}$;
- **Categorical Labels.** The two derivation nodes that are the heads of the overt lexical entries associated with the phonological forms “man” and “eaten” (i.e. $D_4$ and $D_5$ in Fig. 2-2) must map (via $\beta_{\mathbb{N}}$) to the categorical variables $N$ and $V$ respectively.[123]

Turning to the PF interface, the procedure associates the $i^{th}$ overt lexical (derivation) node with the $i^{th}$ word in the surface form, $w_i$:

$$\bigwedge_{x_i \in \mathbb{L}_{\mathbb{N}}{}^{+}} (\Delta_{\mathbb{N}}(x_i) = \Sigma_{w_i}) \tag{2.142}$$

Note that we do not have to provide conditions for either of the interfaces, and this is appropriate in several contexts: (i) when parsing, it is likely the case that LF interface conditions will not be provided[124]; (ii) when generating expressions from logical forms, the LF interface conditions may be provided and the PF interface conditions are not.

#### Connecting the Derivation Model to the Lexicon Model

Finally, the model of the derivation is connected to the SMT-formula associated with the lexicon by adding the axioms associated with the function $\mu$ as outlined in §2.3.4; if the model were to be checked prior to this step, it is a possibility that the recovered derivation (from a satisfiable interpretations of the model) satisfies the interface conditions and yet is not yielded by the supplied lexicon. This concludes the construction of the model of the parser.

### 2.4.2 Checking the Model and Recovering a Minimalist Derivation

The model of the minimalist parser constructed in Step 2 of the procedure constitutes a (decidable) decision problem encoding the problem of whether there exists a derivation yielded by the lexicon that satisfies the interface conditions. The parsing procedure uses the SMT-solver to check the model – i.e. evaluate (using the solver's decision procedure) whether the associated decision problem is satisfiable – and determine whether the model has a satisfiable interpretation. If the model of the minimalist parser has a satisfiable interpretation, then the SMT-solver identifies what it is, and *the output of the parser is obtained by automatically recovering a minimalist derivation from the identified satisfiable*

[123] Although categorical features are marked under PF interface conditions in Table 2.4 for convenience of exposition, this study considers them to be LF interface conditions; this also holds for the end-of-sentence punctuation that is used to denote whether the sentence is a declarative or an interrogative expression.

[124] Recovering the LF interface conditions is likely to motivate parsing the expression to begin with.

*interpretation of the model*; the recovered (minimalist) derivation can be yielded by the supplied (input) lexicon and also satisfies the interface conditions specified in the input. Turning to our running example of parsing entry $I_1$, see Table 2.6 and Table **??** for a summary of the satisfiable interpretation (of the model) yielded by the SMT-solver, and see Fig. 2-2 for the minimalist derivation (automatically) recovered from this interpretation.

The procedure recovers the derivation from the interpretation of the model by executing a depth-first traversal over the associated multi-dominance tree that starts at the root node, and reconstructs the derivation in a bottom-up manner, taking advantage of the fact that within the MG formalism, internal merge operates deterministically due to the Shortest Movement Condition, and thus it is the instances of external merge that are of interest. At each step in the traversal, the following (recursive) visitor procedure is called on the visited node:

- if the visited node $x \in \mathbb{N}$ is a *lexical node* (i.e. leaf node in the derivation), then the procedure returns the lexical item associated with this node; this lexical item may be recovered from the lexicon node $\mu(x)$ as detailed in §2.3.1.

- if the visited node $x \in \mathbb{N}$ is not a lexical node, then it must be a product of *merge*, and has two distinct children nodes, $c_1, c_2 \in \mathbb{N}$. Without loss of generality, let us say $c_1$ projects and $c_2$ does not; then $x$ is a product of *internal merge* if there is some node $y \in \mathbb{N}$ such that $\mathcal{P}(y) = c_2$ (i.e. indicating that $c_2$ is raised from position dominated by $c_1$), and is a product of *external merge* otherwise.

  - if $x$ is the product of *external merge*, then the procedure returns $\{visit(c_1), visit(c_2)\}$, first visiting $c_2$ and then visiting $c_1$.
  - if $x$ is the product of *internal merge*, then the procedure returns $visit(c_1)$.

Turning again to our running example, the recovery of a minimalist derivation from the model interpretation listed in Table 2.6 is illustrated by tracing the following sequence of derivation nodes in the derivation presented in Figure 2-2:

$$\left[D^I_{22}, D^E_9, D^I_{14}, D^E_{13}, D^E_{18}, D^E_{15}, D^L_4, D^L_3, D^E_{17}, D^E_{12}, D^L_1, D^L_5, D^L_7, D^L_2, D^L_6\right] \tag{2.143}$$

A superscript of $L$ indicates that the node is a lexical node, whereas a superscript of $E$ or $I$ indicates that the node is the product of *external merge* or *internal merge* respectively.

In the case that there are multiple distinct model interpretations that are satisfiable, since the model is finite, the set of distinct recoverable derivations is finite and can therefore be enumerated as follows: after checking the model and recovering a derivation, push onto the solver's stack a constraint that prohibits a solution that yields the derivation just recovered; repeat this process until the solver reports back *unsatisfiable*.

### Optimization and Principles of Economy

The model of the parser may be extended by appending additional formulas to the model, thereby further constraining the space of satisfiable model interpretations and thus enabling the use of the model to search for derivations that have specific additional properties. For example, the model may be extended to identify derivations that are *optimal* with respect to a specified metric, such as the number of EM or IM operations, by adding (to the model)

| Node | Label | $\beta_{\mathbb{N}}$ | $h$ | $p$ | $\mathcal{P}$ | $\mathcal{H}$ | $\mu$ | $(\psi \circ \mu)$ | $\Delta_{\mathbb{N}}$ |
|---|---|---|---|---|---|---|---|---|---|
| $D_0$ | | | $D_0$ | $D_0$ | $D_0$ | $D_0$ | $L_5$ | $L_5$ | |
| $D_1$ | $what/D :: {\sim}y, -p$ | D | $D_1$ | $D_{12}$ | $D_{19}$ | $D_0$ | $L_{37}$ | $L_3$ | what |
| $D_2$ | $has/T ::= x, +q, {\sim}x$ | T | $D_2$ | $D_{14}$ | $D_0$ | $D_6$ | $L_{32}$ | $L_{36}$ | has |
| $D_3$ | $the/D ::= y, {\sim}y, -q$ | D | $D_3$ | $D_{15}$ | $D_0$ | $D_0$ | $L_9$ | $L_{14}$ | the |
| $D_4$ | $man/N :: {\sim}y$ | N | $D_4$ | $D_{15}$ | $D_0$ | $D_0$ | $L_8$ | $L_5$ | man |
| $D_5$ | $eaten/V ::= y, {\sim}x$ | V | $D_5$ | $D_{12}$ | $D_0$ | $D_7$ | $L_6$ | $L_{33}$ | eaten |
| $D_6$ | $\epsilon/C_{ques.} ::<= x, +p, C$ | $C_{ques.}$ | $D_6$ | $D_9$ | $D_0$ | $D_0$ | $L_{23}$ | $L_7$ | $\epsilon$ |
| $D_7$ | $\epsilon/v ::<= x, = y, {\sim}x$ | v | $D_7$ | $D_{17}$ | $D_0$ | $D_0$ | $L_{17}$ | $L_4$ | $\epsilon$ |
| $D_8$ | | | $D_0$ | $D_0$ | $D_0$ | $D_0$ | $L_5$ | $L_5$ | |
| $D_9$ | $\epsilon/C_{ques.} :<= x \cdot +p, C$ | $C_{ques.}$ | $D_6$ | $D_{22}$ | $D_0$ | $D_0$ | $L_7$ | $L_{27}$ | |
| $D_{10}$ | | | $D_0$ | $D_0$ | $D_0$ | $D_0$ | $L_5$ | $L_5$ | |
| $D_{11}$ | | | $D_0$ | $D_0$ | $D_0$ | $D_0$ | $L_5$ | $L_5$ | |
| $D_{12}$ | $eaten/V := y \cdot {\sim}x$ | V | $D_5$ | $D_{17}$ | $D_0$ | $D_0$ | $L_{33}$ | $L_5$ | |
| $D_{13}$ | $has/T := x, +q \cdot {\sim}x$ | T | $D_2$ | $D_9$ | $D_0$ | $D_0$ | $L_{24}$ | $L_5$ | |
| $D_{14}$ | $has/T := x \cdot +q, {\sim}x$ | T | $D_2$ | $D_{13}$ | $D_0$ | $D_0$ | $L_{36}$ | $L_{24}$ | |
| $D_{15}$ | $the/D := y \cdot {\sim}y, -q$ | D | $D_3$ | $D_{18}$ | $D_{21}$ | $D_0$ | $L_{14}$ | $L_0$ | |
| $D_{16}$ | | | $D_0$ | $D_0$ | $D_0$ | $D_0$ | $L_5$ | $L_5$ | |
| $D_{17}$ | $\epsilon/v :<= x \cdot = y, {\sim}x$ | v | $D_7$ | $D_{18}$ | $D_0$ | $D_0$ | $L_4$ | $L_{35}$ | |
| $D_{18}$ | $\epsilon/v :<= x, = y \cdot {\sim}x$ | v | $D_7$ | $D_{14}$ | $D_0$ | $D_0$ | $L_{35}$ | $L_5$ | |
| $D_{19}$ | $what/D : {\sim}y \cdot -p$ | D | $D_1$ | $D_{22}$ | $D_0$ | $D_0$ | $L_3$ | $L_5$ | |
| $D_{20}$ | | | $D_0$ | $D_0$ | $D_0$ | $D_0$ | $L_5$ | $L_5$ | |
| $D_{21}$ | $the/D := y, {\sim}y \cdot -q$ | D | $D_3$ | $D_{13}$ | $D_0$ | $D_0$ | $L_0$ | $L_5$ | |
| $D_{22}$ | $\epsilon/C_{ques.} :<= x, +p \cdot C$ | $C_{ques.}$ | $D_6$ | $D_0$ | $D_0$ | $D_0$ | $L_{27}$ | $L_5$ | |

Table 2.6: Model interpretation for the derivation of the sentence: *"What has the man eaten?"* (see $I_1$ in Table 2.4). Values are supplied for several of the uninterpreted functions (e.g. $h(D_{15}) = D_3$ and $p(D_9) = D_{22}$) that make up: (i) the derivation formula (i.e. $h$ (head), $p$ (parent), $\mathcal{P}$ (phrasal movement), $\mathcal{H}$ (head movement), $\Delta_{\mathbb{N}}$ and $\beta_{\mathbb{N}}$), and (ii) the lexicon formula (i.e. $\psi$ (successor) and $\mu$ (bus)). Each node $D_i$ is a member of the finite sort $\mathbb{N}$, and the label for each node was recovered from the model interpretation. Not all of the members of the sort $\mathbb{N}$ are used in the derivation; the head and parent of a node that is not used in the derivation, e.g. $\{D_0, D_8, D_{11}, D_{16}, D_{20}\}$, is (the bottom node) $D_0$, a node reserved for uninterpreted functions to map nodes that don't play a role in the derivation.

pseudo-boolean equations that encode these metrics[125], e.g.:

$$k_{EM} \geq \sum_{x_i, x_j \in \mathbb{N}, i<j} (1, d(\mathcal{M}(x_i, x_j), x_i) \wedge d(\mathcal{M}(x_i, x_j), x_j)) \tag{2.144}$$

$$k_{IM} \geq \sum_{x_i, x_j \in \mathbb{N}, i<j} (1, (\mathcal{P}(x_i) = x_j) \wedge d(p(x_j), x_i)) \tag{2.145}$$

Axioms (2.144) and (2.145) are true if and only if the total number of EM and IM operations in the derivation is less than or equal to the constants $k_{EM}$ and $k_{IM}$ respectively; this is how the number of EM and IM operations in each derivation (listed in Table **??**) is computed. Given upper bounds on the number of EM and IM operations in a derivation (i.e. at most $|\mathbb{N}|$), lower bounds on the number of EM and IM operations can be identified via binary

[125] These SMT formulae utilize the background theory of integers that is built into the Z3 SMT-solver.

search[126]; in this way, the solver can search for model interpretations that are *optimal* with respect to a specified metric, enabling investigations pertaining to Principles of Economy (Collins, 2001). This is further considered in §3.1 of the next chapter.

**Parsing Expressions with Out-of-Vocabulary (Phonological) Forms**

The model may be applied to more general scenarios then described thus far by removing some of the model axioms. For example, the model may be extended to parse an expression with a phonological form that is Out-of-Vocabulary (OOV) (i.e. the form not is not listed in the lexicon). This is accomplished by first adding the new OOV form to the PF sort ($\Sigma$) and then removing any axiom that restricts which lexical entry (feature sequence) in the lexicon the OOV phonological form is connected to, so that the OOV form can freely associate with any lexical feature sequence in the lexicon model.

In the case that none of the lexical feature sequences in the lexicon suffice, the addition of a lexical entry with a new lexical feature sequence (to the lexicon) may be sufficient to parse the supplied (input) interface conditions. This (missing) lexical feature sequence may be inferred by modifying the construction of the lexicon model (in Step 3a) such that an additional lexical entries is instantiated but not constrained to take on any particular interpretation (i.e. a blank-slate entry); then the solver will identify a satisfiable solution to this (modified) model from which may be recovered a derivation derived (in part) from an entirely new lexical item associated with the OOV. This process can be applied repeatedly to incrementally grow the lexicon and thereby increase the coverage of a given minimalist grammar; this is explored in depth in §3.3 of the next chapter.

### 2.4.3 Evaluating the Parser

The parser was evaluated as follows. First, we assembled the corpus of pairings of LF and PF interface conditions listed in Table 2.4. This corpus, intended to span a diverse range of sentence types, includes:

- interrogatives without any embedded sentences – e.g. see $I_1$ for a Wh-question, and see $I_2$ for a yes/no-question;
- expressions with a covert tense marker in the matrix clause (e.g. $I_5$), the embedded clause (e.g. $I_8$), or both the matrix and embedded clause (as in $I_7$);
- Wh-question with an embedded complementizer phrase ($I_3$);
- declaratives with an embedded relative clause[127] in which the antecedent is either the subject of the embedded predicate (as in $I_6$ or $I_4$) or the object of the embedded predicate (as in $I_7$);
- expressions with verbs of varying valency – e.g.: $I_3$ includes the intransitive verb *"resigned"*, $I_1$ includes the transitive verb *"eaten"*, and $I_2$ includes the ditransitive form of verb *"given"*;

[126] I.e. determining whether a particular value for the number of EM or IM operations is licit can be determined by pushing the appropriate pseudo-boolean constraint (for that value) onto the solver's stack, checking the model for satisfiability, and then popping the stack; once a lower bound is identified, a derivation respecting this bound may be automatically recovered (as outlined in §**??**) by first pushing the relevant pseudo-boolean constraint onto the solver stack before checking the model

[127] These derivations accord with the wh-movement analysis of relative clauses – see §3.3.2 for further discussion.

|  | $D_0$ | $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ | $D_6$ | $D_7$ | $D_8$ | $D_9$ | $D_{10}$ | $D_{11}$ | $D_{12}$ | $D_{13}$ | $D_{14}$ | $D_{15}$ | $D_{16}$ | $D_{17}$ | $D_{18}$ | $D_{19}$ | $D_{20}$ | $D_{21}$ | $D_{22}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $D_0$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · |
| $D_1$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · |
| $D_2$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · |
| $D_3$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · |
| $D_4$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · |
| $D_5$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · |
| $D_6$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · |
| $D_7$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · |
| $D_8$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · |
| $D_9$ | · | ○ | ⊕ | ⊕ | ⊕ | ⊕ | ⊕ | ⊕ | · | · | · | · | ⊕ | ⊕ | ⊕ | ⊕ | · | ⊕ | ⊕ | · | · | ⊕ | · |
| $D_{10}$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · |
| $D_{11}$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · |
| $D_{12}$ | · | ○ | · | · | · | ⊕ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · |
| $D_{13}$ | · | ○ | ⊕ | ⊕ | ⊕ | ⊕ | · | ⊕ | · | · | · | · | ⊕ | · | ⊕ | ⊕ | · | ⊕ | ⊕ | · | · | ⊕ | · |
| $D_{14}$ | · | ○ | ⊕ | ○ | ○ | ⊕ | · | ⊕ | · | · | · | · | ⊕ | · | · | ○ | · | ⊕ | ⊕ | · | · | · | · |
| $D_{15}$ | · | · | · | ⊕ | ⊕ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · |
| $D_{16}$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · |
| $D_{17}$ | · | ○ | · | · | · | ⊕ | · | ⊕ | · | · | · | · | ⊕ | · | · | · | · | · | · | · | · | · | · |
| $D_{18}$ | · | ○ | · | ○ | ○ | ⊕ | · | ⊕ | · | · | · | · | ⊕ | · | · | ○ | · | ⊕ | · | · | · | · | · |
| $D_{19}$ | · | + | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · |
| $D_{20}$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · |
| $D_{21}$ | · | · | · | + | + | · | · | · | · | · | · | · | · | · | · | + | · | · | · | · | · | · | · |
| $D_{22}$ | · | ⊕ | ⊕ | ⊕ | ⊕ | ⊕ | ⊕ | ⊕ | · | ⊕ | · | · | ⊕ | ⊕ | ⊕ | ⊕ | · | ⊕ | ⊕ | ⊕ | · | ⊕ | · |

Table 2.7: Model interpretation of the binary uninterpreted functions $d$ and $d^\star$ for the derivation of the sentence: *"What has the man eaten?"* (see $I_1$ in Table 2.4). An entry in the table for row $D_i$ and column $D_j$ may be interpreted as follows: ○ indicates that node $D_i$ dominates $D_j$ with respect to the derivation tree but not the derived tree, + indicates that node $D_i$ dominates $D_j$ with respect to the derived tree but not the derivation tree, ⊕ indicates that $D_i$ dominates $D_j$ with respect to both the derivation tree and the derived tree, and · indicates that $D_i$ does not dominate $D_j$ (with respect to either the derivation tree or the derived tree). E.g. $D_{21}$ dominates $D_{15}$ with respect to the derived tree but not the derivation tree: notice in Table 2.6 that $\mathcal{P}(D_{15}) = D_{21}$, but there is no $k \in [0, 22]$ such that $p(D_k) = D_{21}$. In contrast, $D_{18}$ dominates $D_{17}$ with respect to the derivation tree but not the derived tree: notice in Table 2.6 that $p(D_{17}) = D_{18}$ but there is no $k \in [0, 22]$ such that $\mathcal{P}(D_k) = D_{18}$. The root node of the derivation, $D_{22}$, dominates all other nodes in the derivation with respect to both the derivation tree and the derived tree. Finally, rows $D_1, D_2, \ldots, D_8$, being leaf nodes in the derivation, do not dominate any other nodes in the derivation (as indicated by those rows being entirely filled with dots).

- declaratives with embedded complementizer phrases in which the embedded clause is either in the active voice (as in $I_5$) or in the passive voice (as in $I_8$).

Next, we assembled a minimalist lexicon (listed in Table 2.1) that can, for each entry in the corpus, generate a derivation prescribed by contemporary theories of minimalist syntax. Then, we parsed each entry in the corpus using the assembled lexicon and list the derivation recovered by the parser for each entry in Table **??**); for each entry, the derivation recovered from the parser matched the prescribed derivation. See Fig. 2-2, Fig. 2-15, Fig. 2-16, Fig. 2-

17, Fig. 2-18, Fig. 2-19, Fig. 2-20, and Fig. 2-21 for illustrations of the derivations recovered from parsing interface conditions $I_1$ through $I_8$ respectively; see Table 2.8 for the statistics of each of these derivations. The derivations associated with these sentences (as prescribed by contemporary theories of minimalist syntax) involve a plethora of syntactic phenomenon; the running example of parsing $I_1$ in Table 2.4 yielded a derivation (see Fig. 2-2) that illustrates several of these phenomenon – e.g.:

- There are two instances of syntactic movement of maximal projections: wh-movement (i.e. an instance of A'-movement) from $D_1$ to $D_{19}$, and subject-raising (i.e. an instance of A-movement) of a phrase $D_{15}$ to $D_{21}$; these two instances of syntactic movement are nested and accord with the *minimal link condition*.
- there are two instances of syntactic movement of minimal projections (i.e. head-movement): aux-raising via T-to-C head-movement of $D_2$ to $D_6$, and V-to-v head-movement of $D_5$ to $D_7$.
- There are two empty lexical items involved in the derivation, with heads $D_7$ and $D_6$.
- A double-VP-shell structure for the transitive verb "eaten" is formed by $D_5$ and $D_7$.

Finally, we carried out two variations of this evaluation: firstly, we modified the model of the parser by removing model axioms pertaining to enforcing LF interface conditions, and were able to recover the same derivations as before; secondly, we modified the model of the parser by removing model axioms pertaining to the enforcement of PF interface conditions (i.e. enforcing the ordering of surface forms), and again were able to recover the same derivations as before. Although the result of these two variations of the evaluation process are not surprising, as removing model axioms only serves to increase the space of satisfiable interpretations of the model, they serve to illustrate that the parser can be used without the interface conditions being fully specified, and in particular that the parser can be applied to process only a specification of LF interface conditions, which may be of utility in natural language generation.

| $I_i$ | # Words | # Empty Lex. Items | #EM | #IM | $\beta_{\mathbb{N}}$ | $d$ | $h$ | $\mathcal{H}$ | $\mathcal{M}$ | $\mathcal{P}$ | $d^{\star}$ | $p$ | $\Delta_{\mathbb{N}}$ | $\mathcal{L}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $I_1$ | 5 | 2 | 6 | 2 | 3108 | 56976 | 213316 | 35192 | 14808 | 4192 | 74116 | 61242 | 58 | 34341 |
| $I_2$ | 4 | 2 | 5 | 1 | 2322 | 37074 | 134077 | 21880 | 10172 | 3104 | 47416 | 39268 | 50 | 21546 |
| $I_3$ | 8 | 3 | 9 | 3 | 15170 | 605160 | 2573452 | 439130 | 123380 | 21364 | 839452 | 691071 | 118 | 415490 |
| $I_4$ | 7 | 3 | 9 | 3 | 13123 | 501622 | 2119000 | 361000 | 103798 | 18800 | 693386 | 570994 | 110 | 342035 |
| $I_5$ | 7 | 4 | 10 | 2 | 13304 | 501716 | 2119084 | 361000 | 103930 | 18800 | 693480 | 570991 | 110 | 342035 |
| $I_6$ | 7 | 3 | 8 | 3 | 9971 | 331474 | 1377216 | 233672 | 71192 | 14164 | 454250 | 374310 | 98 | 222195 |
| $I_7$ | 6 | 5 | 10 | 3 | 11671 | 410752 | 1721626 | 292736 | 86596 | 16400 | 565470 | 465729 | 102 | 277832 |
| $I_8$ | 6 | 4 | 9 | 2 | 11511 | 410710 | 1721581 | 292736 | 86480 | 16400 | 565428 | 465738 | 102 | 277832 |

Table 2.8: Statistics for the derivation yielded by the parser when processing each of the pairings of LF and PF interface conditions listed in Table 2.4 using the lexicon listed in Table 2.1. The statistics for each derivation includes: the number of words in each parsed sentence (i.e. overt lexical heads participating in the derivation); the number of empty lexical items participating in the derivation; the number of instances of external and internal merge respectively; supplied; the number of instances of various uninterpreted functions appearing in the SMT-model of the derivation that was constructed in the process of parsing the derivation.

## 2.5 Summary

This chapter presented an SMT-model of a parser for Minimalist Grammars that has been implemented as a working computer program, and then demonstrated that this model may be used in conjunction with an SMT-solver to parse a specification of LF and PF interface conditions using a supplied MG lexicon. The model is (conceptually) simple in design and serves to illustrate how an SMT-model of a (minimalist) parser can be constructed. Central to the architecture of the model was the insight that the multi-dominance tree lifted from a derivation tree can be viewed as an assembly of derivation node sequences, each of which is mapped to a lexicon node sequence in the lexicon model, so that the trajectory a lexical item takes through a derivation aligns with the syntactic feature sequence in the associated lexical entry. Additionally, by observing that the multi-dominance tree is in effect a superposition of both the associated derivation tree and the derived tree, we were able to separate the model axioms that establish syntactic relations over the hierarchical structure of the derivation[128] (i.e. the axioms in §2.3.2) from the axioms that establish the linear ordering prescribed by the specified PF interface conditions over the lexical heads with overt phonological forms in the derivation (i.e. the axioms listed §2.3.3); in this way, rules for linear-ordering, which is performed by the Sensor-Motor (SM) system, were factored apart from the structure dependent rules that build a derivation.

Notably, the model may be used to search for derivations that satisfy a *partial* specification of the lexicon and interface conditions, thereby enabling alternative uses of the parser such as:

(i) parsing a given pairing of LF and PF interface conditions with only a partially specified lexicon, as in the case of a child encountering a new verb, and recovering the (novel) lexical entry used by the derivation output by the parser;

(ii) parsing with a specification of a lexicon and LF interface conditions, but without any PF interface conditions supplied, as in the case of a child externalizing a thought, and recovering the PF interface conditions satisfied by the derivation output by the parser;

(iii) parsing with a specification lexicon and PF interface conditions, but without any LF interface conditions supplied, as in the case of a child parsing speech, and recovering the LF interface conditions satisfied by the derivation output by the parser.

In this way, the model enables the study of the extent to which the lexicon and the interface conditions constrain the space of possible derivations that a parser must search through. More generally, a declaratively specified, logical model of a grammar enables one to focus on developing the logical axioms that make up the model and ensuring they are faithfully grounded in universal principles of language, while delegating to the SMT-solver the task of carrying out the appropriate deductions that constitute the model-checking process; importantly, in the case that a parse cannot be solved (i.e. there is no satisfiable interpretation of the model), the SMT-solver can identify which model axioms are in conflict (e.g. by identifying minimum unsatisfiable cores), so that we might gain insight in how the universal principles of language underlying these axioms might contradict one another.

Finally, we note that there is much that can be done to improve and extend the model (e.g. by adding or subtracting model axioms) so as to come into closer alignment with the

[128] All syntactic operations depend only on hierarchical structure; see (Radford, 2016, Pg. 155) for further discussion of this point.

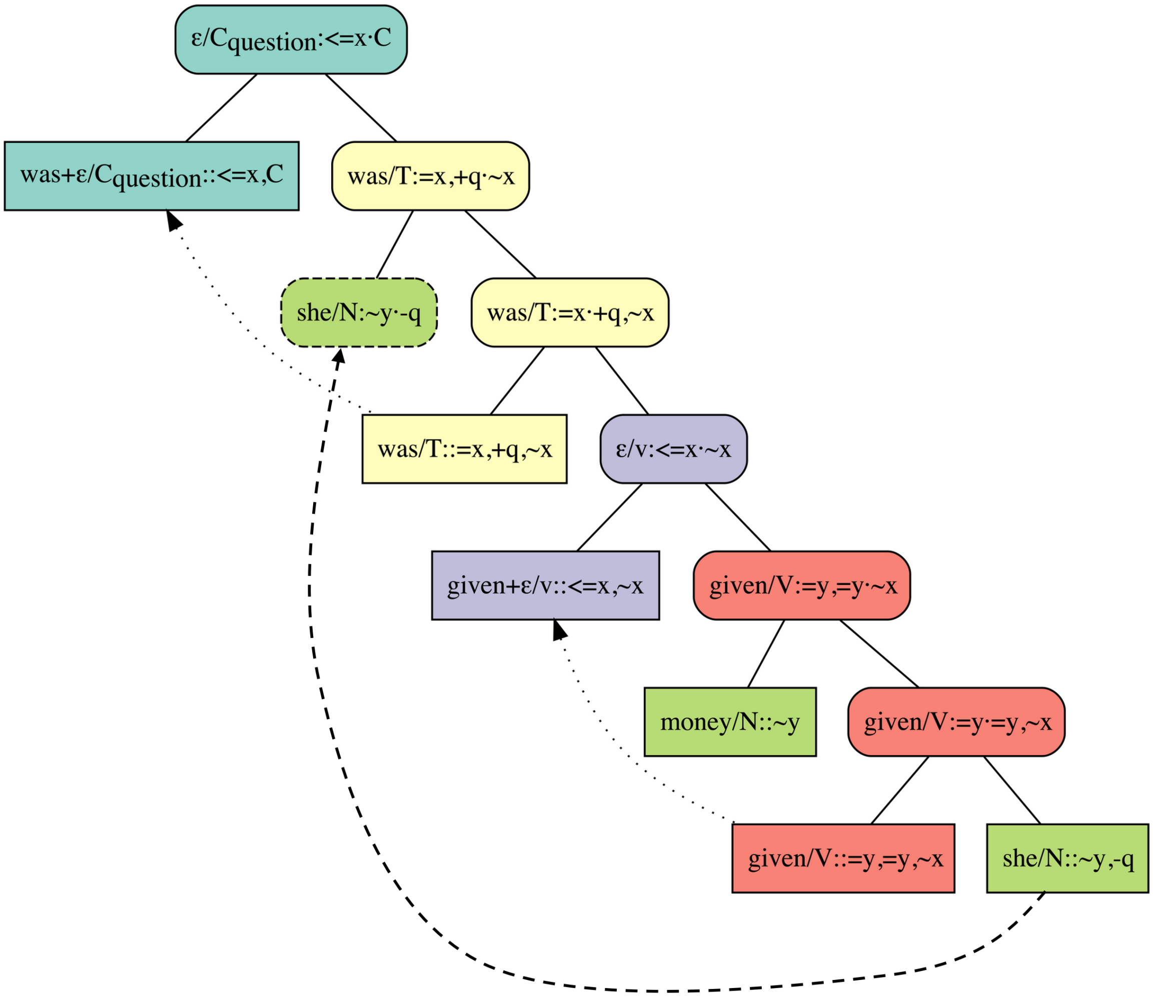


Figure 2-15: A derivation for *"Was she given money?"* that was output by the parser when processing interface conditions $I_2$ (listed in Table 2.4) using the lexicon presented in Table 2.1. The depicted structure is a multi-dominance tree encodes both the derivation tree (from which this multi-dominance tree was derived) as well as the derived tree. Lexical nodes are indicated by rectangular nodes, while derived nodes are indicated by rounded corners. Constituents with the same color have the same head. Dashed and dotted arrows indicate phrasal and head movement respectively; a dashed border indicates that a node is the target of phrasal-movement, with the lower (raised) structure implicitly copied to the target position.

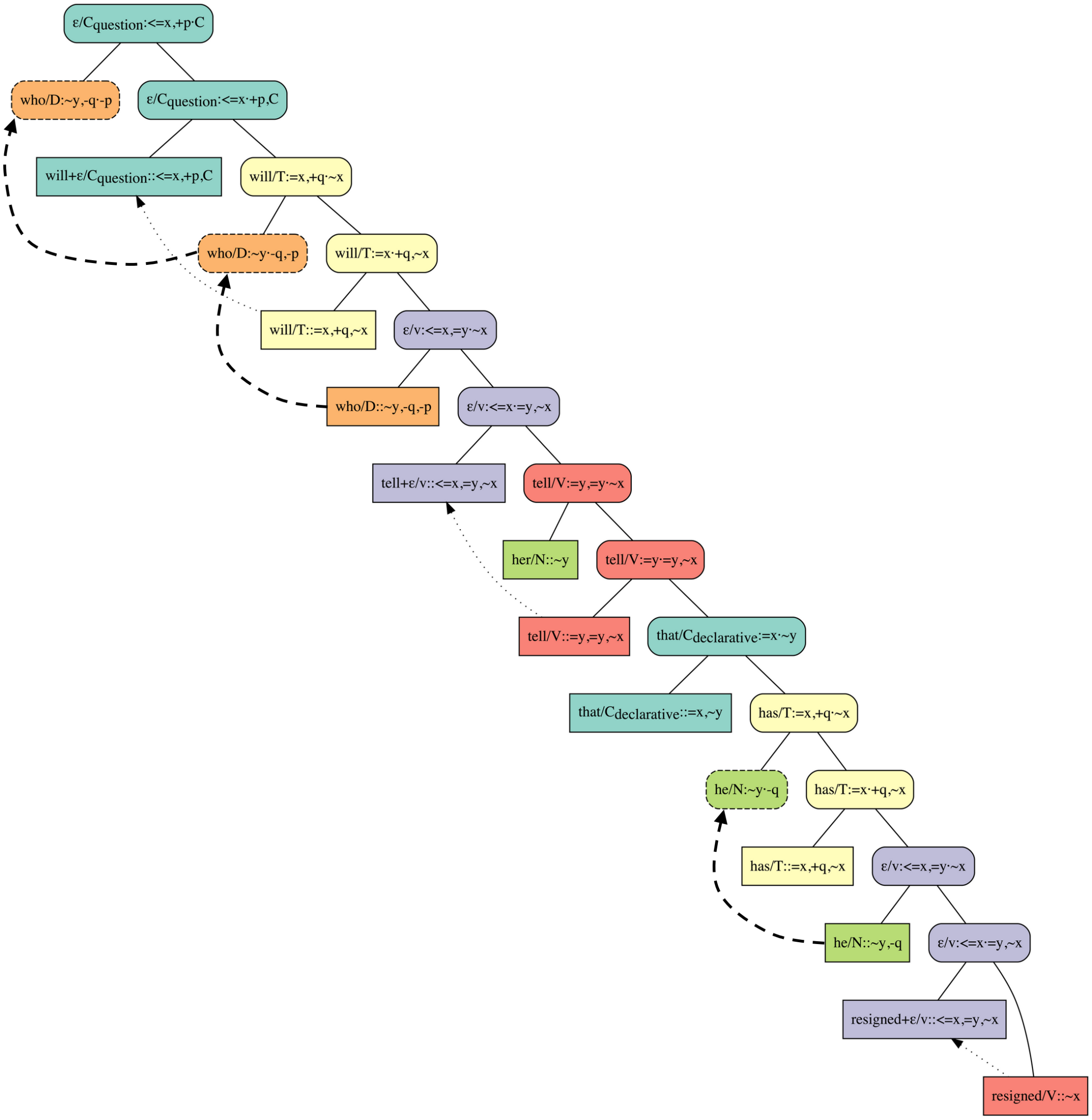


Figure 2-16: A derivation for *"Who will tell her that he has resigned?"* that was output by the parser when processing interface conditions $I_3$ (listed in Table 2.4) using the lexicon presented in Table 2.1. The depicted structure is a multi-dominance tree encodes both the derivation tree (from which this multi-dominance tree was derived) as well as the derived tree. Lexical nodes are indicated by rectangular nodes, while derived nodes are indicated by rounded corners. Constituents with the same color have the same head. Dashed and dotted arrows indicate phrasal and head movement respectively; a dashed border indicates that a node is the target of phrasal-movement, with the lower (raised) structure implicitly copied to the target position.

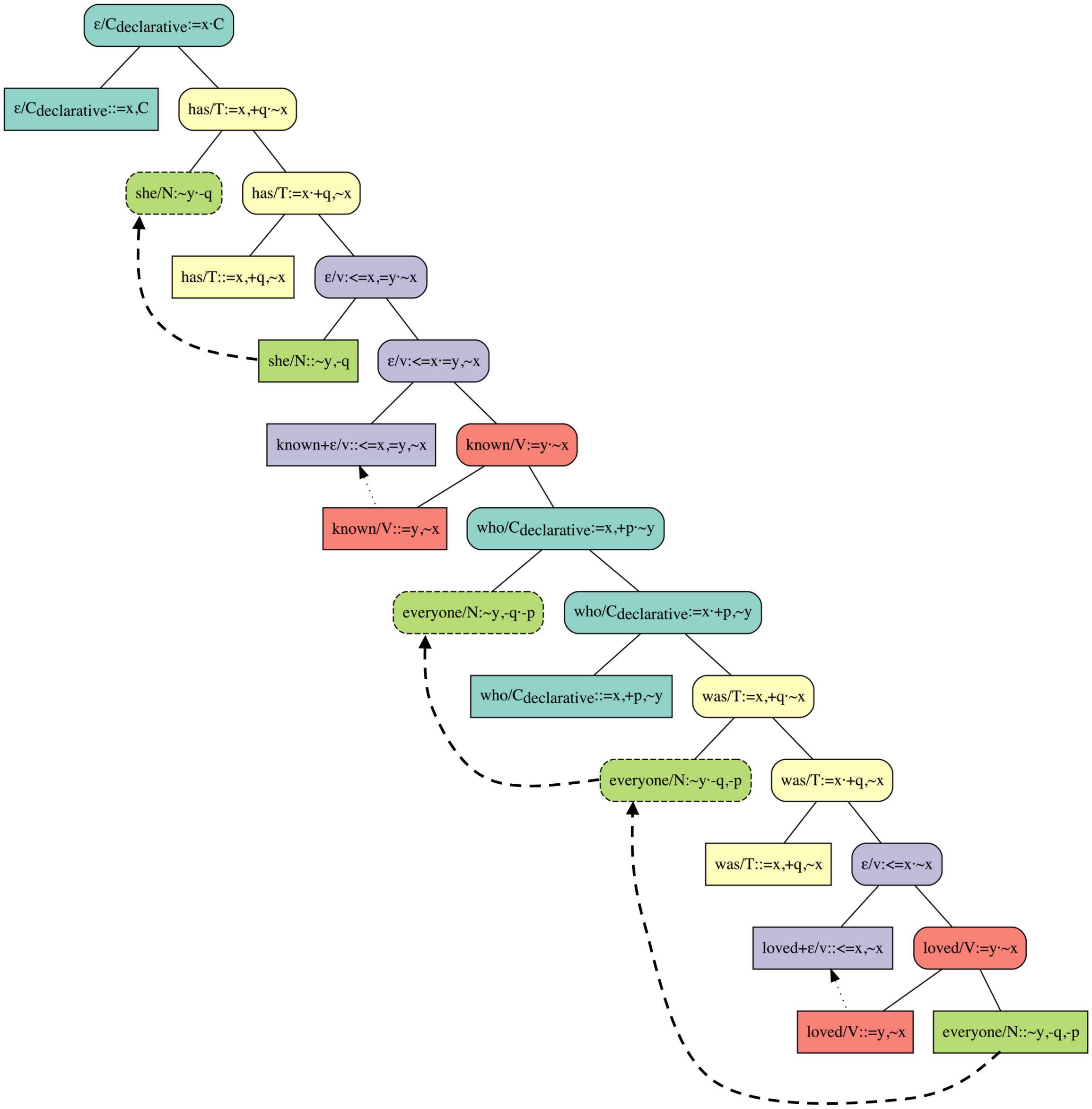


Figure 2-17: A derivation for *"She has known everyone who was loved."* that was output by the parser when processing interface conditions $I_4$ (listed in Table 2.4) using the lexicon presented in Table 2.1. The depicted structure is a multi-dominance tree encodes both the derivation tree (from which this multi-dominance tree was derived) as well as the derived tree. Lexical nodes are indicated by rectangular nodes, while derived nodes are indicated by rounded corners. Constituents with the same color have the same head. Dashed and dotted arrows indicate phrasal and head movement respectively; a dashed border indicates that a node is the target of phrasal-movement, with the lower (raised) structure implicitly copied to the target position.

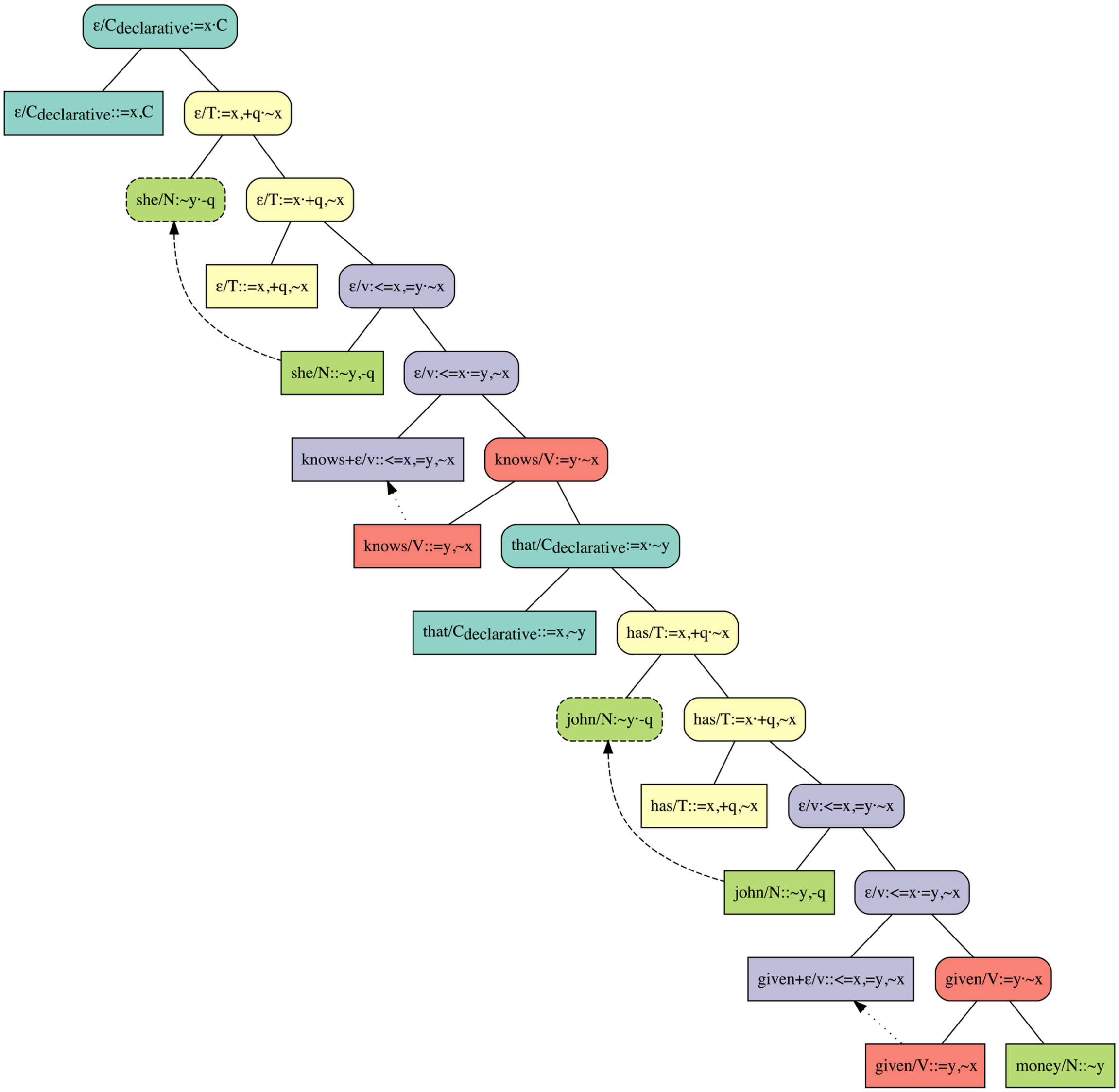


Figure 2-18: A derivation for *"She knows that John has given money."* that was output by the parser when processing interface conditions $I_5$ (listed in Table 2.4) using the lexicon presented in Table 2.1. The depicted structure is a multi-dominance tree encodes both the derivation tree (from which this multi-dominance tree was derived) as well as the derived tree. Lexical nodes are indicated by rectangular nodes, while derived nodes are indicated by rounded corners. Constituents with the same color have the same head. Dashed and dotted arrows indicate phrasal and head movement respectively; a dashed border indicates that a node is the target of phrasal-movement, with the lower (raised) structure implicitly copied to the target position.

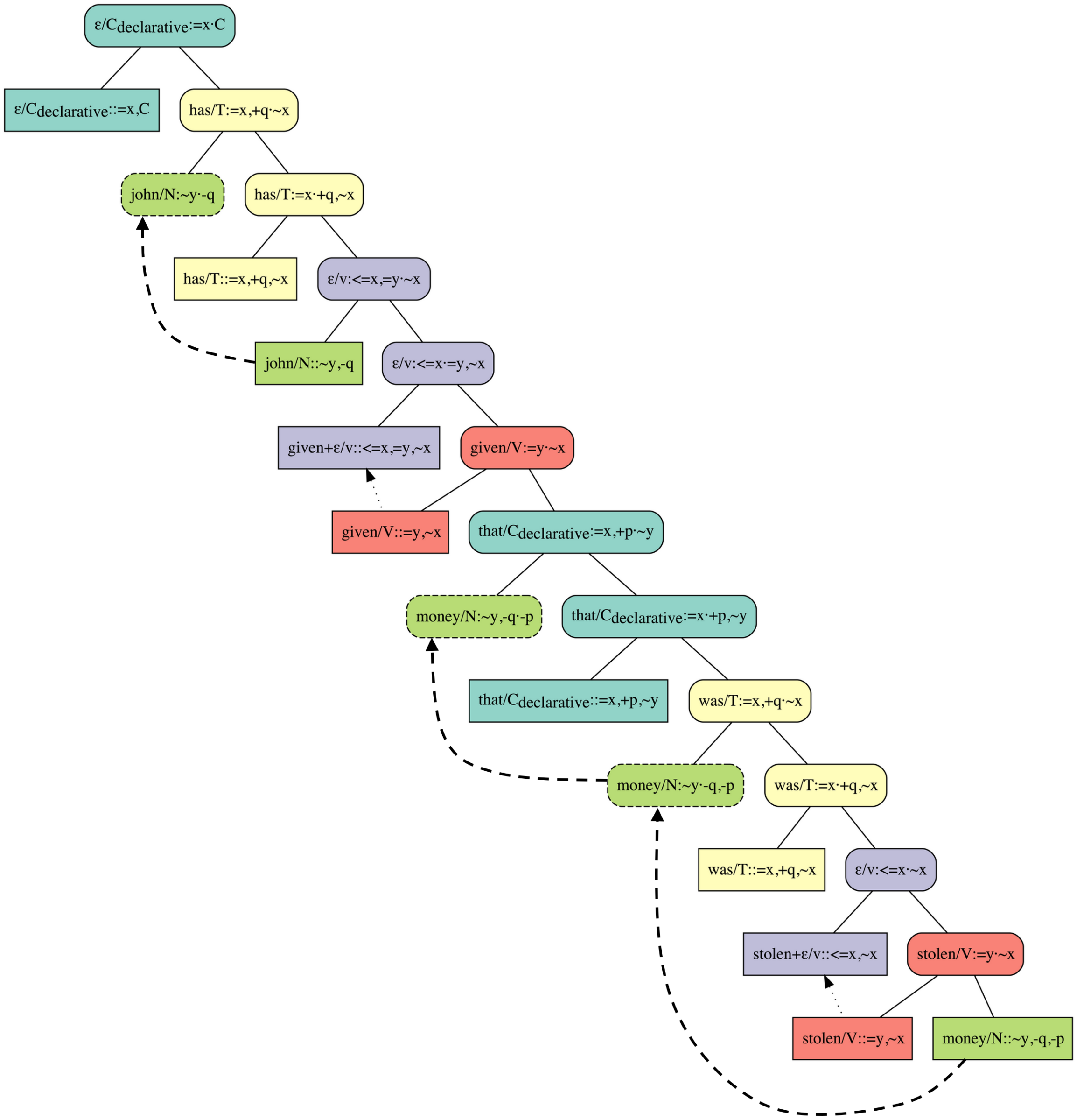


Figure 2-19: A derivation for *"John has given money that was stolen."* that was output by the parser when processing interface conditions $I_6$ (listed in Table 2.4) using the lexicon presented in Table 2.1. The depicted structure is a multi-dominance tree encodes both the derivation tree (from which this multi-dominance tree was derived) as well as the derived tree. Lexical nodes are indicated by rectangular nodes, while derived nodes are indicated by rounded corners. Constituents with the same color have the same head. Dashed and dotted arrows indicate phrasal and head movement respectively; a dashed border indicates that a node is the target of phrasal-movement, with the lower (raised) structure implicitly copied to the target position.

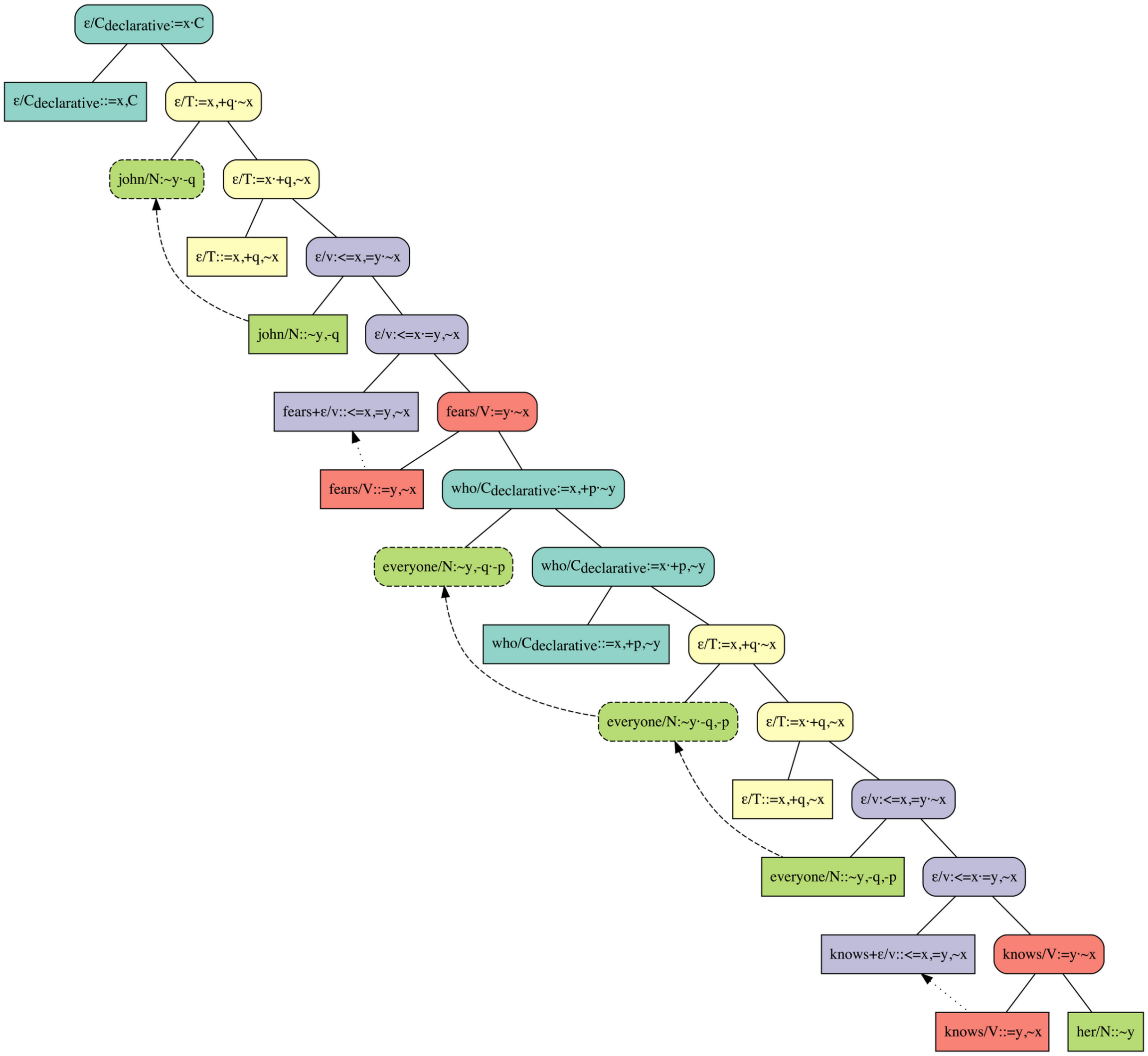


Figure 2-20: A derivation for *"John fears everyone who knows her."* that was output by the parser when processing interface conditions $I_7$ (listed in Table 2.4) using the lexicon presented in Table 2.1. The depicted structure is a multi-dominance tree encodes both the derivation tree (from which this multi-dominance tree was derived) as well as the derived tree. Lexical nodes are indicated by rectangular nodes, while derived nodes are indicated by rounded corners. Constituents with the same color have the same head. Dashed and dotted arrows indicate phrasal and head movement respectively; a dashed border indicates that a node is the target of phrasal-movement, with the lower (raised) structure implicitly copied to the target position.

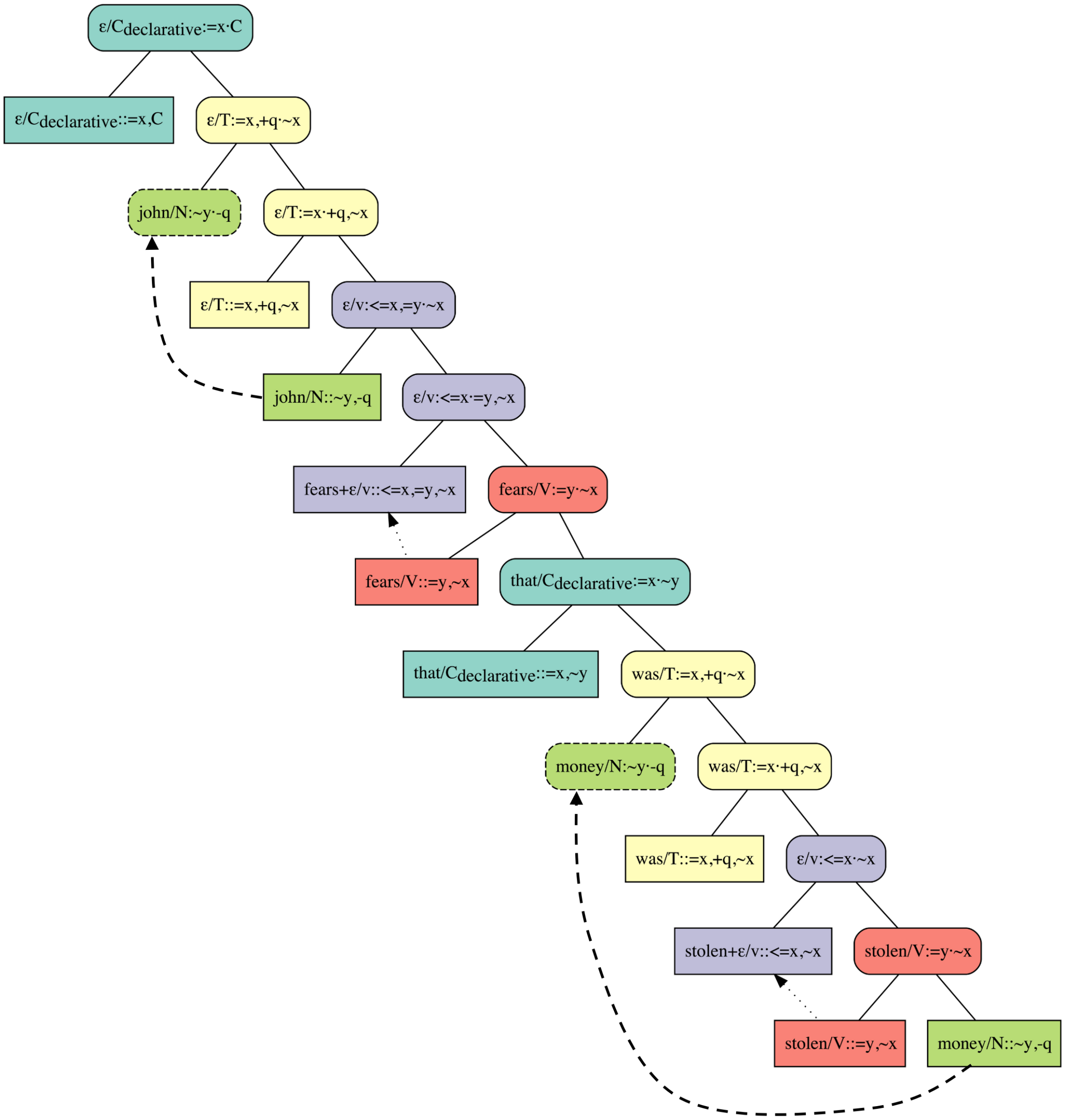


Figure 2-21: A derivation for *"John fears that money was stolen."* that was output by the parser when processing interface conditions $I_8$ (listed in Table 2.4) using the lexicon presented in Table 2.1. The depicted structure is a multi-dominance tree encodes both the derivation tree (from which this multi-dominance tree was derived) as well as the derived tree. Lexical nodes are indicated by rectangular nodes, while derived nodes are indicated by rounded corners. Constituents with the same color have the same head. Dashed and dotted arrows indicate phrasal and head movement respectively; a dashed border indicates that a node is the target of phrasal-movement, with the lower (raised) structure implicitly copied to the target position.

prescriptions of minimalist theories of syntax – e.g.: further developing the structure of a lexical entry's feature matrix; modeling advances in the theory pertaining to phase-based derivations that are driven by probe-goal machinery (which replaces checking theory); incorporating *parameters* within the model axioms so as to model differences in the (apparent) syntax of different languages; handling adjunctions within derivations. (See Ch. 4 for a discussion of such extensions.) Although the model only covers a small subset of syntax within the framework of earlier minimalist theories of syntax, it suffices to serve as the central component of the acquisition models that will be developed in the next chapter.

# Chapter 3

# Inferring Minimalist Grammars with an SMT-Solver

This chapter introduces a system for a inferring minimalist grammar from a small finite sequence of (sentence, skeletal "meaning") pairings. The system works by constructing an SMT-model of a language-acquisition device and using an SMT-solver to solve this model and recover a minimalist lexicon that can yield, for each sentence in the input sequence, a derivation that accords with the prescribed meaning for that sentence. This system extends the model of the minimalist parser developed in the previous chapter, and has the form of a computational model of a child language learner that accords with the criterion for a language-acquisition device set out in (Chomsky, 1965). The computational experiments detailed in this chapter show that this system is able to infer, from psychologically plausible input data – i.e. a small finite sequence of simple sentences with at most one level of embedding – a grammar that aligns with contemporary theories of minimalist syntax and that can yield an infinite number of structures encoding novel pairings of sound and meaning.

The sections of this chapter are organized as follows. First, §3.1 presents an overview of the system, outlining the problem that the system must solve, how the system solves this problem by "Solving for Syntax", how the system takes the form of a child language learner, and conditions of psychological plausibility that the system adheres to. Then, the two acquisition procedures that the system may employ – i.e. an instantaneous acquisition procedure and an incremental acquisition procedure – are detailed in §3.2 and §3.3 respectively, along with computational experiments demonstrating their capacity to infer grammars that align with contemporary theories of minimalist syntax.

## 3.1 Overview

### 3.1.1 The System Acquires Knowledge of Language

To begin, let us specify the task that the system must solve in terms of inputs and outputs – i.e. what knowledge of language the system acquires, and the input from which said knowledge is acquired.

The system takes as input: (i) an initial lexicon (which may be empty) that stands in for knowledge the child language learner may have already acquired; (ii) a finite sequence of pairings of LF and PF interface conditions (i.e. (sentence, skeletal "meaning") pairs) that stands in for the *Primary Linguistic Data* (PLD) a child language learner is exposed to.[1] Each individual pairing of LF and PF interface conditions in the PLD has the same form as the input to the model of the parser, as detailed in §2.

The system outputs a minimalist lexicon (i.e. a minimalist grammar) consisting of a set of lexical entries, each of which is a distinct pairing of a phonological form with a feature sequence (which includes an associated categorical feature), and that is a superset of the supplied initial lexicon. The (output) lexicon is able to yield, for each pair of interface conditions in the PLD, a derivation that satisfies the stipulated interface conditions – i.e. the lexicon is compatible with the PLD and may be used by the model of the minimalist parser to parse each entry in the PLD. Furthermore, the output lexicon is one that the system determines to be *optimal* with respect to metrics grounded in economy considerations; namely, the system outputs the smallest possible lexicon (i.e. the system minimizes the number of distinct lexical feature sequences and the total number of syntactic features appearing over all of the lexical feature sequences) that has the capacity to yield a set of derivations that satisfy the interface conditions presented in the PLD and that collectively use as few merge operations as possible. (See §3.2.1 for more details).

The computational experiments detailed in this chapter show that the system, when initialized with an empty lexicon, is able to:

1. Process an input PLD (listed in Table 3.2 on Pg. 126 and in Table 3.11 on Pg. 173) consisting of (deliberately chosen to be simple) active and passive voice sentences with at most one degree of embedding that are composed of declaratives, yes/no-questions and wh-questions, embedded sentences and (restrictive) relative clauses, and involve intransitive, transitive and ditransitive verbs.

2. Infer, from this input PLD, an (output) lexicon (listed in Table 3.9 on Pg. 163) that:

   (i) yields for each entry in the PLD a derivation that satisfies the interface conditions encoded in that entry and that accords with the derivation prescribed by contemporary theories of minimalist syntax (see §3.2.3 for examples);

   (ii) generalizes beyond the supplied PLD to yield novel structures (i.e. derivations) pairing sound and meaning – i.e. these structures were not yielded by the grammar to satisfy any of the entries in the PLD;

   (iii) limits the production of structures that may be considered "overgenerations" – see §3.2.3 for details;

   (iv) yields, for any $n > 0$, (grammatical) expressions with $n$-levels of embedding.

Importantly, these computational experiments demonstrate that the system can acquire, from a finite set of simple sentences with at most one level of embedding, a lexicon that has the capacity to generate an infinite number of structures pairing meaning and sound; the acquired lexicon, taken together with the model of the minimalist parser, approximates a *descriptively adequate* theory of language. Thus, the system may be said to solve (in part) the *Projection Problem.*[2]

[1] The Primary Linguistic Data (PLD) is: *"The actual original finite language data to which children are exposed, and from which they must map to knowledge of a specific language; a combination of sound and extra-linguistic experience."* (Lust, 2006, Pg. 32)

[2] See (Baker, 1979) for a discussion of the projection problem.

### 3.1.2 The System Solves for Syntax

The system outlined in this chapter is a grammar induction scheme that derives a system of (quantifier-free first order logic) equations from the input – i.e. a PLD and an initial lexicon – and then solves this system of equations to recover the output – i.e. a lexicon that is compatible with the PLD and that is a superset of the initial lexicon. Let us now examine a sketch of how the system does this.

First, let us review how the model of the parser (developed in Ch. **??**) operated at a conceptual level, such that it could be thought of as "parsing via equation solving." The parser took as input a lexicon and a pairing of LF and PF interface conditions. The input lexicon was used to instantiate a lexicon model (i.e. an SMT-formula) and "hardcode" its interpretation. Next, a derivation model (also an SMT-formula) was constructed, with bounds on the size of the model derived by inspecting the interface conditions; this derivation model was then "connected" to the lexicon model by way of an uninterpreted function mapping members of the derivation node sort to members of the lexicon node sort, and axioms that constrained this mapping. The interface conditions were then (automatically) translated into additional axioms that served to constrain interpretations of the derivation model. Finally, an SMT-solver was used to "solve" this system of equations (i.e. a conjunction of SMT-formulae) and the output of the parser – i.e. the derivation yielded by the input lexicon to satisfy the supplied interface conditions – was automatically recovered from the interpretation of the derivation model. To summarize, the lexicon and interface conditions were known quantities, and the derivation was an unknown quantity that was solved for, so that the approach may be viewed as a modern adaptation of the "parsing as deduction" framework (Pereira and Warren, 1983), with an axiomatization of minimalist grammars (MGs) in place of an axiomatization of context free grammars (CFGs), pairings of LF and PF interface conditions replacing strings, and an SMT-solver used instead of Prolog.

The system outlined in this chapter, which is in effect a scheme for grammar induction, is a modern adaptation of earlier work by (Rayner et al., 1988) on using logic grammars to infer a lexicon. Let us assume for the current discussion that the system will take an empty (initial) lexicon, so that the system is tasked with inferring the output lexicon. The system can obtain the set of phonological forms appearing in the lexicon (i.e. the vocabulary) by scanning the PF interface conditions in the PLD; the system must then infer: (i) the lexical feature sequences that appear in the lexicon, and (ii) how each lexical feature sequence associates with each phonological form. The system will do this in a manner similar to that of the case of parsing – namely, the system will construct a system of equations that is constrained by known quantities, and solve this system of equations to obtain valuations of the unknown quantities. However, there are two important differences between the aforementioned conception of "parsing as equation solving", and the system for grammar induction outlined in this chapter: (i) whereas in the case of parsing the lexicon was a known quantity, *the lexicon is now an unknown quantity*;[3] (ii) whereas in the case of parsing a single pairing of LF and PF interface conditions was provided as a known quantity, *there are now multiple pairings of LF and PF interface conditions provided as known quantities (i.e. the input PLD).* How then can the lexicon – an unknown quantity – be inferred? The approach taken by the system is inspired by the following insight made by (Rayner et al., 1988):

> *"The basic idea is as follows: a logic grammar can be viewed as the definition of a*

[3] Specifically, the lexical feature sequences are unknown as are the associations between the phonological forms and the lexical feature sequences.

> *relation between a string and a parse-tree. You can run it two ways: finding the parse-trees that correspond to a given string (parsing), or finding the strings that correspond to a given parse-tree (generating). However, if we view the lexicon as part of this relation, we get new possibilities. More specifically, we can compute the lexicons that correspond to a given string; this can in a natural way be viewed as a formalization of "lexicon learning from example sentences". In terms of the "explanation-based learning" paradigm, this makes the associated parse-tree the "explanation." ... This is the simplest variant of the idea: assume that there is one entry per word, and represent the lexicon as an association-list with one entry for each word. Each sentence now constrains the possible values of these entries to be ones which allow it to be parsed; the hope is that a conjunction of a suitably large number of such constraints will be enough to determine the lexicon uniquely."*

Here (Rayner et al., 1988) points out how a "parsing as deduction" style parser can be adapted to the problem of inferring a lexicon by requiring that the lexicon in effect have to parse multiple strings at once.

Let us now examine a rough sketch of how this insight can be adapted in a modern setting – i.e. in the context of minimalist syntax, replacing the axiomatization of CFGs with an axiomatization of MGs and using modern machinery – i.e. the Z3 SMT-solver in place of Prolog: In the case of the grammar induction system outlined here, the system can thus proceed by:

(i) instantiating a single lexicon model;

(ii) instantiating, for each pairing of LF and PF interface conditions in the PLD, a derivation model that is constrained to satisfy that pair of interface conditions;

(iii) connecting each individual derivation model to the lexicon model as was done in constructing the model of the parser;

(iv) adding to the model constraints derived from optimization metrics that aim to minimize the size of both the lexicon and the all of the derivations connected to the lexicon.

The result of this process is an SMT-model of a grammar for which an *optimal* interpretable (model) solution may be identified using an SMT-solver. Importantly, the model is constrained such that the obtained lexicon must be able to yield, for each pairing of interface conditions in the PLD, a derivation that satisfies that pairing of interface conditions. See Fig. 3-1 for an illustration of the SMT-model that the system will construct.

Experience working with the system outlined in this chapter has shown that:

- Typically, a grammar consonant with minimalist syntax cannot be inferred from just one PLD entry by itself; rather, the experiments described later in this chapter show that constraining the lexicon so that it must be compatible with several entries from the PLD together is usually sufficiently constrain the lexicon model.

- There may exist more than one lexicon that is compatible with the input PLD, and the question arises of how the system will choose between them. This is resolved by the addition of constraints derived from optimization metrics, that enable to the SMT-solver to identify a solution to the SMT-model that is *optimal* (with respect to the metrics); this help to both narrow the space of lexicons under consideration and also ensure that the system does not produce trivial and uninteresting lexicons – e.g. the system could identify, for each entry in the PLD, a lexicon that is compatible with only that entry, and then concatenate these lexicons together, resulting in a large lexicon that does not generalize (beyond the PLD) to produce novel structures that pair meaning and sound.

To summarize, the system constructs an SMT-model that has a satisfiable model interpretation if and only if there exists a lexicon that is compatible with both the model parameters and the (input) PLD; if such a lexicon does exist, it can be (automatically) recovered from a satisfiable interpretation of the model. As the constructed SMT-model is finite, and expressed with quantifier free first-order logic formulae, the associated decision problem – i.e. whether there exists a lexicon that is compatible with both the model parameters and the (input) PLD – is compatible with the PLD is *decidable*, which ensures that (given enough time) the SMT-solver will be able to identify a satisfiable interpretation of the model if one exists. Colloquially, we may say that the system is *"Solving for Syntax."*

### 3.1.3 The System Models a Child Language Learner

The system takes the form of a computational model of a child language learner, with the SMT-model corresponding to the learner's state of knowledge, and the process of constructing the SMT-model aligning with the advancing of the learner's state. Following (Berwick, 1985), the system has three components:

1. The *initial state* of the learner's knowledge, $S_0$, consists of:
   (a) a lexicon model that is (partially) constrained by the (input) initial lexicon (which may be empty) so that a lexicon recovered from a satisfiable interpretation of the lexicon model must include the initial lexicon as a subset;
   (b) a set of constraints, $C_{UG}$, expressed as a system of quantifier-free first-order logic formulae, that encode principles of Universal Grammar (UG) and that serve as axioms for deducing the class of minimalist lexicons.[4]
2. The *target state* of the learner's knowledge is the fully constructed SMT-model. The system uses an SMT-solver to check (i.e. solve) this model and identify a satisfiable interpretation of the model; the (inferred) output lexicon is then (automatically) recovered from the (identified) satisfiable model interpretation.
3. The acquisition process that drives the system from the initial to the final state. This process consists of:
   (a) The *input data* that the system processes is the PLD, which consists of a sequence of $n$ entries, denoted $I_0, I_1, I_2, ..., I_{n-1}$, with each entry a pairing of LF and PF interface conditions.
   (b) *Model parameters* that bound the size of the SMT-model that the system constructs[5] – e.g.: the maximum number of distinct lexical entries in the lexicon; the maximum number of syntactic features an entry in the lexicon may have; the

[4] This is the same axiomatization of minimalist syntax underlying the model of the minimalist parser developed in the prior chapter

[5] N.b. these parameters are not *linguistic parameters* that are prescribed by a theory of syntax.

maximum number of distinct selectional and licensing feature labels appearing in the lexicon respectively; the maximum number of lexical items in the lexicon that involving overt and covert phonological forms respectively.[6]

(c) An *acquisition procedure* made up of two functions:

   i. A function, $Q$, that takes as input a state, $S_i$, and an entry in the PLD, $I_i$, and outputs the successor state, $S_{i+1}$. More specifically, $Q$ produces $S_{i+1}$ by: first instantiating a new derivation model, $D_i$ (which includes the axioms in $C_{UG}$); then constraining this derivation model by adding to $D_i$ axioms derived from the LF and PF interface conditions listed in $I_i$; and then finally adding to $S_i$ (i.e. the SMT-model that the system has constructed up until this point) both $D_i$ and an uninterpreted function connecting $D_i$ to the lexicon model in $S_i$, thus resulting in the production of $S_{i+1}$.

   ii. A function, $R$, that maps a state $S_i$ to a set of MG lexicons, $G_i$, with the property that for each entry $I_j$ in the PLD, each lexicon $L \in G_i$ can yield a derivation $p_j^L$ that satisfies the LF and PF interface conditions listed in $I_j$.[7] More specifically, $R$ produces $G_i$ by first adding to $S_i$ axioms derived from optimization metrics and then using an SMT-solver to the set of enumerate distinct satisfiable model interpretations of $S_i$, which constitutes $G_i$.

The acquisition procedure consumes the PLD one entry at a time, using $Q$ to drive the initial state, $S_0$, to the final state, $S_n$; the function $R$ is then applied to $S_n$ to produce a set of MG lexicons, $G_n$, that constitutes the output of the inference procedure. The acquisition procedure comes in two flavors:

(i) the "instantaneous" acquisition procedure, introduced in §3.2, models a child language learner that first consume all of the PLD and only then learns a (learned) grammar – this is of course an idealization of a language learner, as noted in (Chomsky, 1965, Pg. 202);[8]

(ii) the "incremental" acquisition procedure, introduced in §3.3, is an extension of the instantaneous acquisition procedure that models a child language learner that incrementally consumes the PLD (in batches) and gradually builds up (i.e. learns) a lexicon.

See Fig. 3-2 for an illustration of both the instantaneous and incremental acquisition procedures. The instantaneous and incremental acquisition procedures have differing strong points. The advantage of the instantaneous acquisition procedure is that the order in which the PLD is consumed does not alter the set of lexicons that the system will output a member of, whereas for the incremental acquisition procedure the order in which the PLD is consumed may very well impact the set of lexicons the system will output a member of. The advantage of the incremental acquisition procedure is that the learner needs to hold in its memory only the batch of the PLD that is being processed, and not the entire PLD as in the case of the instantaneous acquisition procedure; this is because at any point in the

[6] The valuations of these parameters may be relaxed (increased) so as to widen the space of candidate model interpretations the SMT-solver will search over in trying to identify one that is satisfiable; however, the relaxation of these model parameters comes at the expense of potentially increasing the runtime of the SMT-solver.

[7] In the case of the initial state, $S_0$, since there are no constraints yet imposed by the input, $R(S_0)$ will map to the set of all minimalist lexicons.

[8] See (Lust, 2006, Pg. 62) for a discussion of the "Instantaneous Hypothesis" of child language acquisition.

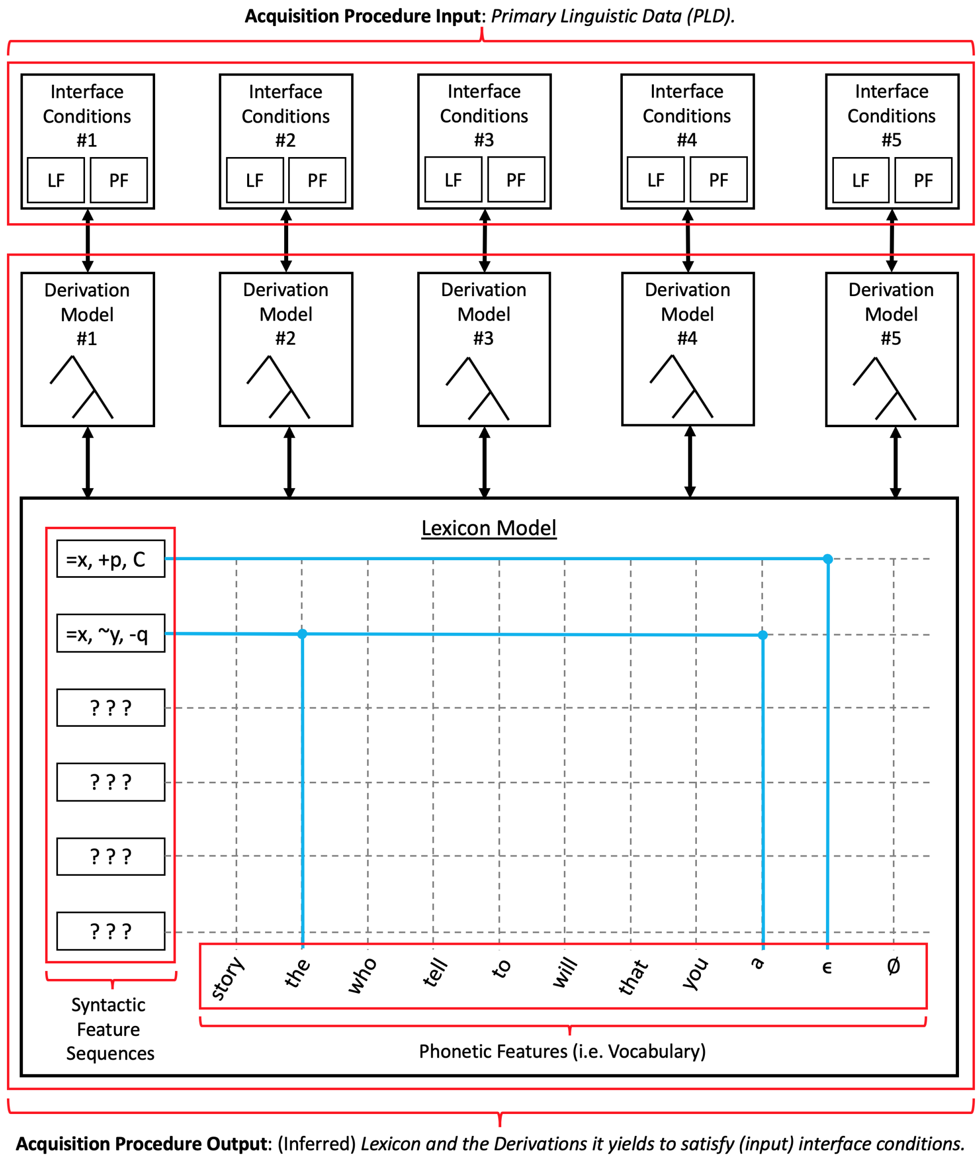


Figure 3-1: The acquisition procedure constructs an SMT-model that consists of: (i) a derivation model (an SMT-formula) for each pairing of LF and PF interface conditions in the PLD; (ii) a lexicon model (an SMT-formula) that is (optionally) partially specified. Each pairing of interface conditions in the Primary Linguistic Data (PLD) is translated into an SMT-formula that constrains the associated derivation model so that any satisfiable interpretation of the derivation model encodes a derivation (tree) that satisfies the associated interface conditions. Each derivation model is connected to the lexicon model via an SMT-formula that requires that the derivation be yielded by the lexicon, thereby constraining satisfiable interpretations of both the derivation model and lexicon model. Finally, optimization constraints derived from economy considerations further constrain satisfiable interpretations of the derivation and lexicon models. After constructing the SMT-model, an SMT-solver is used to identify a satisfiable interpretation of the SMT-model (i.e. a solution to the system of SMT-formulae that make up the SMT-model), and the inferred lexicon and derivation trees are then automatically recovered from the identified interpretation of the SMT-model. Solving the SMT-model requires that the SMT-solver work out both: (i) what the (unspecified) lexical feature sequences (denoted by the "???") in the lexicon are, and (ii) the associations (denoted by the blue lines) between the lexical feature sequences and the phonological forms (obtained by scanning the PF interface conditions in the PLD).

acquisition trajectory, the lexicon that the learner has acquired up until that point is both compatible with all of the PLD processed up to that point, and is also guaranteed to be a subset of the next lexicon that the learner will acquire, so that the learner need only concern themselves with adding to the lexicon new lexical items that ensure compatibility with the batch of the PLD being processed at that time.

Importantly, the system outlined in this chapter aligns with the conditions set out in (Chomsky, 1965, Pgs. 30-31) that a "language-acquisition device" (i.e. a model of language acquisition) that aims for explanatory adequacy should fulfill:[9][10]

> "A child who is capable of language learning must have
>
> (i) a technique for representing input signals
>
> (ii) a way of representing structural information about these signals
>
> (iii) some initial delimitation of a class of possible hypotheses about language structure
>
> (iv) a method for determining what each such hypothesis implies with respect to each sentence
>
> (v) a method for selecting one of the (presumably, infinitely many) hypothesis that are allowed by (iii) and are compatible with the given primary linguistic data
>
> ... assume tentatively that the primary linguistic data consist of signals classified as sentences and nonsentences, and a partial and tentative pairing of signals with structural descriptions. A language-acquisition device that meets conditions (i)-(iv) is capable of utilizing such primary linguistic data as the empirical basis of language learning. This device must search through the set of possible hypotheses $G_1, G_2, \dots$, which are available to it by virtue of condition (iii), and must select grammars that are compatible with the primary linguistic data, represented in terms of (i) and (ii). It is possible to test compatibility by virtue of the fact that the device meets condition (iv). The device would then select one of these potential grammars by the evaluation measure guaranteed by (v). The selected grammar now provides the device with a method for interpreting an arbitrary sentence, by virtue of (ii) and (iv)." (Pgs. 30-32 of (Chomsky, 1965))

Let us now consider how the various parts of the system adhere to the criterion specified in the Aspects model of language acquisition:

- condition (i) corresponds to the PLD;
- condition (ii) corresponds to the derivation models, which encode "structural descriptions" of the entries in the PLD;
- condition (iii) corresponds to $C_{UG}$ – i.e. the axioms encoding principles of UG;
- condition (iv) corresponds to the lexicon model, which encodes a grammar, and the function $Q$, which translates interface conditions into the constraints that a derivation must satisfy;
- condition (v) corresponds to the function $R$, which is able to identify the optimal lexicon that is compatible with the PLD.

To summarize, the system outlined in this chapter is a computational model of a child language learner that fulfills the conditions for a language-acquisition device set out in (Chomsky, 1965).

[9] See (Lust, 2006, Pg.54) for further review of the Aspects' acquisition model.

[10] See (Rizzi, 2017) for a review of "explanatory adequacy."

### 3.1.4 Psychological Fidelity

The system satisfies several conditions of psychological plausibility that are briefly considered below; later in Ch. 4, we will revisit these conditions and consider how the system may be extended to further align with the ultimate goal of explanatory adequacy.

- The system acquires a grammar from psychologically plausible input consisting of a (*small*) finite sequence of simple sentences (i.e. the PLD) that have at most one level of embedding; these sentences were selected primarily for the purpose of demonstrating the capabilities of this system, which is ultimately an idealized model of language acquisition. Furthermore, following the "Semantic Bootstrapping Hypothesis"[11], the system assumes that when processing a sentence, the learner knows "*who* did *what* to *whom*?"; thus each sentence is annotated with its predicate-argument structure[12] and with agreement relations (these annotations are encoded within the LF interface conditions).[13][14]

- The system learns from positive evidence only – i.e. no *direct negative* evidence is presented to the system, nor does the system make use of *indirect* negative evidence.[15]

- The system comports with the *Strong Continuity Hypothesis*[16] – i.e. because the system is setup to always respect the axioms in $C_{UG}$ that encode principles of UG, at each stage in the learner's acquisition trajectory, each member of the class of grammars that the learner has narrowed down to must be fully in accordance with the principles of UG.

- The *instantaneous* acquisition procedure is robust to changes in the order in which the PLD is presented to the system.

- In the case of the *incremental* acquisition procedure, the learner requires only a small, finite (potentially zero) sized memory for remembering sentences from the PLD.

- The acquisition system is an extension of a language processing system, namely the model of a minimalist parser introduced in Ch. 2. The parser and the acquisition system are built out of the same model components – i.e. the same lexicon model, the

[11] See (Lust, 2006, Pgs. 42-43) for a discussion of the Semantic Bootstrapping Hypothesis; see also (Grimshaw, 1981), (Pinker, 1984) and (Bloom, 1994).

[12] Predicate-argument structure is sometimes also referred to as "thematic-role structure"; see (Jackendoff, 1983) for further discussion.

[13] N.b. the annotations encoding predicate-argument structure do not contain any information about word ordering.

[14] In the case of sentences involving embedding, the predicate-argument structure does (implicitly) encode some hierarchical information in so far as an embedded phrase is marked as the argument of a predicate; note however, that even when presented only with simple sentences that do not involve any embedding, as in the case of the PLD listed in Table 3.2 (on Pg. 173), the acquisition system converges on a grammar that aligns with contemporary theories of syntax.

[15] Negative evidence comes in two forms: *direct negative evidence* takes the form of an explicit correction (e.g. by an adult) of the child when the child makes a mistake; with *indirect negative evidence*, the observed absence of a particular form when it might otherwise be reasonably expected or circumstance serves as evidence that the form cannot be produced. Empirical investigations indicate that *direct negative evidence* is rarely if ever provided to the child, and is typically rebuffed by the child; *indirect negative evidence* has also been shown to not be required by child language learners. See (Newport et al., 1977) and (Hirsh-Pasek et al., 1984); also see (Marcus, 1993), (Lust, 2006) and (Yang, 2015) for further discussion.

[16] See (Lust, 1999).

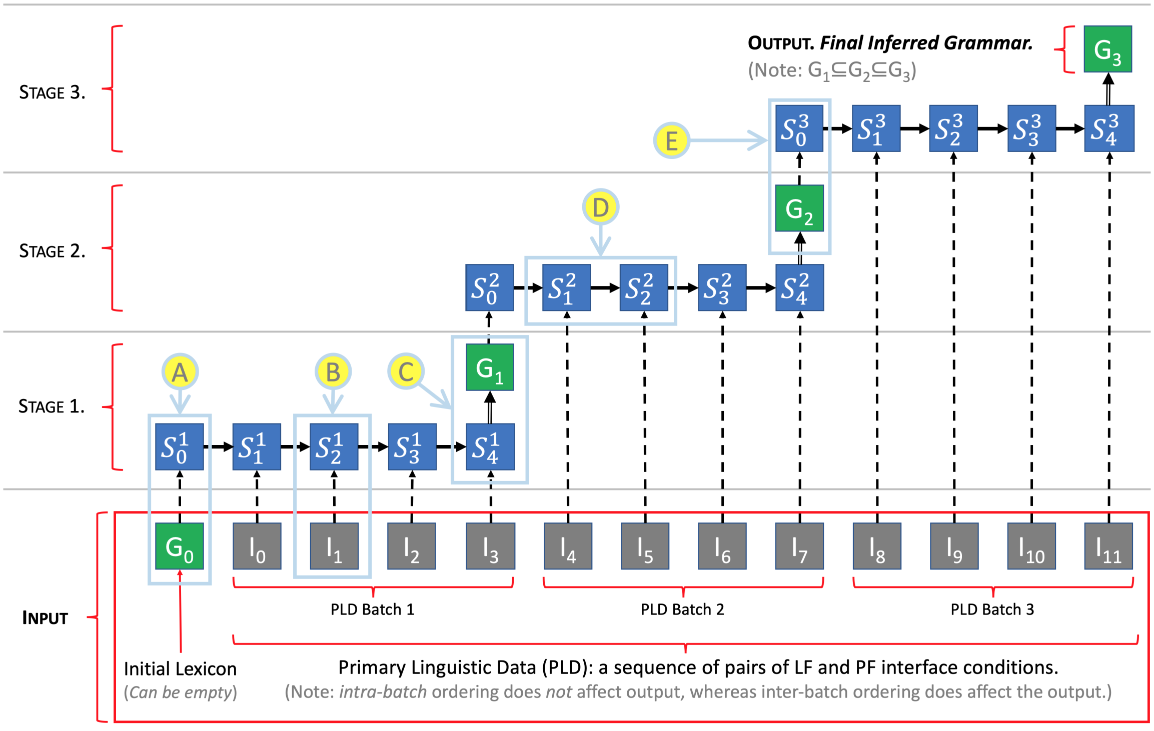

STAGE 3.
OUTPUT. Final Inferred Grammar.
(Note: G1⊆G2⊆G3)
G3
E
S03 S13 S23 S33 S43
STAGE 2.
D
G2
S02 S12 S22 S32 S42
STAGE 1.
A
B
C
G1
S01 S11 S21 S31 S41
INPUT
G0
I0 I1 I2 I3 I4 I5 I6 I7 I8 I9 I10 I11
PLD Batch 1
PLD Batch 2
PLD Batch 3
Initial Lexicon
(Can be empty)
Primary Linguistic Data (PLD): a sequence of pairs of LF and PF interface conditions.
(Note: intra-batch ordering does not affect output, whereas inter-batch ordering does affect the output.)

Figure 3-2: The Primary Linguistic Data (PLD) is a finite sequence of interface conditions that is presented to the learner in batches. The acquisition trajectory, over which the learner consumes the PLD, is divided into stages, with each stage corresponding to a batch of the PLD. At each point in the acquisition trajectory, the state of the learner is equivalent to the SMT-formulae on the SMT-solver's stack (of formulae), with state $S_i^j$ corresponding to the state of the learner in stage $j$ after consuming the first $i$ members of the $j^{th}$ batch. The initial state of the learner, $S_0^1$, consists of the lexicon model (an SMT-formula that may be partially constrained by an optionally specified initial lexicon) and the derivation model axioms that encode UG (see **(A)**). As the learner consumes the input (i.e. the PLD), their state evolves (driven by the function $Q$ as depicted in **(D)**), with each consumed pairing of LF and PF interface conditions resulting in the construction of: a new derivation model, an SMT-formula that connects the derivation model to the lexicon model, and an SMT-formula that constrains the derivation model, requiring satisfiable interpretations thereof to accord with the specified interface conditions; this construction is then added to the solver stack (see **(B)**). When the SMT-solver checks the model, as in **(C)**, an inferred lexicon is automatically recovered from the identified satisfiable interpretation of the model; this is the actualization of the function $R$. Each inferred lexicon is a superset of earlier inferred lexicons, and the inferred lexicon from one stage is then used as the "initial lexicon" for the succeeding stage, as in **(E)**. The first stage of the acquisition procedure is detailed in the presentation of the *instantaneous acquisition procedure*, so called because it simulates the learner consuming the PLD all at once; this procedure is succeeded by the *incremental acquisition procedure*, which is able to handle both the first stage, and then successive stages, simulating the learner incrementally growing a lexicon.

same derivation model, and the same set of UG axioms – and the lexicon output by the acquisition system may be used by the parser.

### 3.1.5 Summary of Key Insights and Ideas

- The computational experiments presented in this chapter demonstrate that modern high-performance SMT-solvers can be applied to the problem of inferring a minimalist grammar from a small sequence of (sentence, skeletal-meaning) pairs (i.e. pairings of LF and PF interface conditions); notably, the system does this without being provided a treebank of minimalist derivations that serve as examples of what the acquired lexicon should be able to yield, and to that end, the system constitutes a scheme for *unsupervised* inference of minimalist grammars.

- The computational experiment detailed in §3.2 demonstrates that by solving for the optimal lexicon, the system infers a grammar that aligns with contemporary theories of minimalist syntax in so far as: (a) the grammar produces the prescribed derivations for a variety of syntactic structures, utilizing syntactic movement (including head-movement) and covert lexical items as needed; (b) expressions with related interpretations are assigned derivations systematically related by structural transformations. Furthermore, the computational experiment detailed in §3.3 demonstrates that the system can acquire, from a finite set of sentences with at most one level of embedding, a grammar that can yield an infinite number of distinct syntactic structures. That this idealistic model is able to recover most of the right syntax is remarkable.

- The axiomatization of minimalist syntax underlying the model of the minimalist parser developed in the prior chapter is a part of the initial state of knowledge (corresponding to UG) that the model of the learner at the beginning of the acquisition process. By design, the system separates out the questions of what knowledge the model of the child language learner acquires and how the model of the child language learner acquires it, allowing us to specify the learner's initial state and the conditions that the learner's final state must satisfy (with respect to the PLD that the learner processes) – and leave to the SMT-solver the questions of how the language-acquisition device goes from the initial state to the final state and what that final state is. This enables us to focus on understanding which model axioms have what implication in an experiment working or not, and leave the task of carrying out the appropriate deductions to the computer. In this way, the system lets us explore the interaction of several simple principles – i.e. that derivations must satisfy interface conditions, that derivations are subject to economy conditions, and that all syntactic relations within a derivation are established by merge – and see how far they can take us. Ultimately, the system may serve as a vehicle for better understanding how more can be done with less, which is a line of inquiry that lies at the heart of the Minimalist Program.

- The lexicon model has a factored representation – i.e. it has the form of a two-dimensional association matrix, with a set of phonological forms on one axis, a set of lexical feature sequences on the other axis, and the entries in the matrix marking which lexical feature sequences are associated with which phonological forms. (See Table 3.3 on Pg. 127 for an illustration of factored representation.) The lexicon's factored representation allows for it to be grown along the two dimensions of the matrix separately, with the dimension pertaining to lexical feature sequences controlling which syntactic structures the lexicon can yielded, and the dimension pertaining to phonological forms controlling the vocabulary of the learner. Specifically, the lexicon's factored representation enables optimization metrics to be defined that minimize the size of the lexicon with respect to the set of distinct lexical feature sequences without factoring in how often the different phonological forms appear in the PLD; this ensures that, as the system processes the PLD, the system only adds new lexical feature sequences to the lexicon if the existing set of lexical feature sequences is insufficient for yielding a syntactic structure that can satisfy a pairing of LF and PF interface conditions. In particular, this means that if the system cannot parse an entry in the PLD, it will only add a new lexical feature sequence to the lexicon if adding a new association between a phonological form and an existing lexical feature sequence will not suffice.

## 3.2 An Instantaneous Model of Acquisition

This section introduces a computational model of language acquisition and then uses it to infer a minimalist grammar (presented in Table 3.3) that is compatible with the primary linguistic data (PLD) (presented in Table 3.2), optimal with respect to economy considerations, and closely aligns with contemporary theories of minimalist syntax.

The acquisition procedure, implemented as a working computer program, takes the form of a child language learner (following (Berwick, 1985)) and is in accordance with the criterion for a language acquisition device set out in *Aspects* (Chomsky, 1965):

1. The procedure takes as input the Primary Linguistic Data (PLD), which consists of a finite sequence of pairings of LF and PF interface conditions; this constitutes *"a technique for representing input signals"*, thereby satisfying criterion (i) of the Aspects model.

2. The learner processes each entry in the PLD, creating a new derivation model and constraining it with the addition of axioms derived from the interface conditions; this constitutes *"a way of representing structural information about these signals"*, thereby satisfying criterion (ii) of the Aspects model.

3. The learner then constructs an acquisition model by first initializing an empty lexicon model, and then connecting each (constrained) derivation model to the lexicon model;[17] the (constructed) acquisition model serves as an *"initial delimitation of a class of possible hypothesis about language structure"*, thereby satisfying criterion (iii) of the Aspects model.

4. The learner can solve the (constructed) acquisition model (i.e. check the SMT-model using the SMT-solver) to infer a lexicon that, for each entry of interface conditions in the PLD, can generate a derivation that satisfies constraints derived from those interface conditions; this constitutes *"a method for determining what each such hypothesis implies with respect to each sentence."*, thereby satisfying criterion (iv) of the Aspect's model.

[17] In connecting a derivation model (constrained by interface conditions) to the lexicon model, the system incorporates the model of parsing presented in Ch. 2.

5. The learner infers an optimal grammar by further constraining the acquisition model with the addition of model axioms derived from Principles of Economy, and then checking the acquisition model (using the SMT-solver); this constitutes *"a method for selecting one of the (presumably infinitely many) hypothesis that are allowed by (iii) and are compatible with the given primary linguistic data"*, thereby satisfying criterion (v) of the Aspects model.

The procedure is referred to as the "instantaneous" acquisition procedure because the lexicon model and each connected derivation model must be solved simultaneously (using an SMT-solver), in effect simulating a learner that has to consume the entire PLD before inferring a lexicon; thus the procedure is guaranteed to infer the optimal grammar (that is compatible with that PLD), so long as such a lexicon exists and the constructed (finite) model is large enough to represent it, no matter the order in which the PLD is presented to the learner.

The remainder of this section is organized as follows. To begin, we will introduce several optimization metrics that encode Principles of Economy (see §3.2.1). We will then introduce a procedure for inferring grammars that are optimal with respect to these optimization metrics (see §3.2.2). Finally, we will present a computational experiment that uses the acquisition model to infer an optimal grammar from a small, finite set of sentences paired with simplified representations of meaning (i.e. the PLD presented in Table 3.2), and then analyze the inferred grammar, showing that it aligns with the prescriptions of contemporary theories of minimalist syntax (see §3.2.3).

### 3.2.1 Economy Considerations

Economy Conditions, a well-established principle of minimalist theories of syntax, inform us that the derivations yielded must be optimal in some sense – e.g. a derivation is prohibited from involving superfluous movement operations or any application of *merge* not required for the derivation to converge. (Chomsky, 1995, Pg. 266) Concurrently, the Aspects model of language acquisition requires that an evaluation function be supplied that will identify *one* of many possible grammars that is compatible with the PLD. The approach taken here is to develop an evaluation function that selects for grammars that are *optimal* in the sense prescribed by economy considerations. To this end, the acquisition model developed in this chapter includes a number of *metrics* that each *measure* some quantitative property of the acquisition model that is pertinent to economy considerations; these metrics enable the system, by way of instructing the SMT-solver in the course of model-checking, to optimize the acquisition model so as to minimize or maximize the value measured by a specified metric. In this way, the inferred grammar will be the grammar that is *optimal* with respect to the supplied metrics, and the system is thus brought into accordance with Principles of Economy.

Let us now consider how economy conditions inform what aspects of grammar to optimize, and in turn what metrics will be needed. (Collins, 2001) broadly summarizes the implications of the condition as follows:

> *"Consider an operation OP applying in a derivation D leading to the representations (PF, LF) (phonetic form and logical form). Economy considerations suggest that OP be as small as possible, and be applied in a way that minimizes search. Given a series of operations that form a derivation D, economy conditions suggest that the length or cost of the derivation must be minimized in some way. Lastly, economy considerations suggest that the representations formed in*

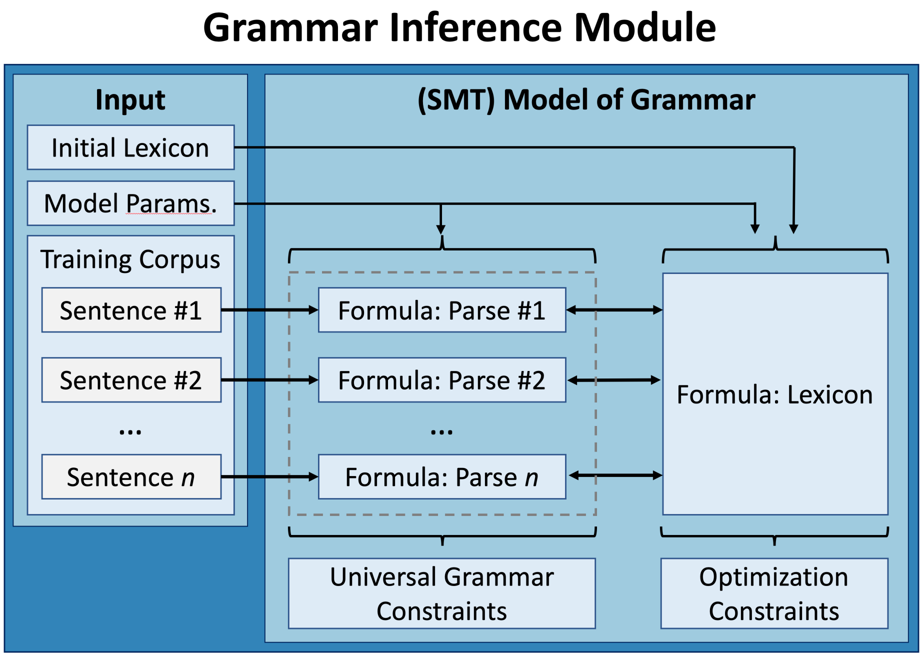


Figure 3-3: Diagram showing the Grammar Inference Module. For each sentence in the training corpus, there is an SMT formula encoding a minimalist parse, and each of these formulas is connected (via shared symbols) to the SMT formula encoding a minimalist lexicon. The model parameters limit the size of the acquisition model. The initial lexicon is optional. The Universal Grammar axioms constrain each SMT-formula for a minimalist parse. The optimization constraints ensure that the system finds the smallest minimalist lexicon.

> *the course of a derivation should be as simple as possible, consisting of a minimal number of syntactic objects, each of which is interpretable (at LF or PF)."* (Collins, 2001, Pg. 40)

This informs us that the metrics the system has should be able to minimize various properties of the derivation model. How about the lexicon model? Consider what would happen if we didn't optimize the lexicon model in any way while optimizing the derivation model – e.g. there are no upperbounds on how many selectional or licensing feature labels are available, or how many distinct lexical entries are in the lexicon, how many features a lexical entry may have; then the system could construct a separate lexicon for each derivation, and the inferred lexicon could simply be the set union of those separate lexicons. Yet, the lexicon should not include more than is necessary to yield the prescribe derivations. For this reason, the system also includes metrics that measure the size of the lexicon.

Each metric is encoded as an SMT-formula that takes the form of a pseudo-Boolean equation: on one side is a value that the metric can take on, and on the other side is a calculation of the metric using the uninterpreted functions and sorts that make up the lexicon and derivation models; the relation between the two sides may be an equality or an inequality depending on how the metric is going to be optimized. This SMT-formula, when added to the acquisition model, will force the SMT-solver to solve for a grammar that satisfies the constraint on what values the metric may take, thereby enabling the inference of *optimal* grammars. Let us now examine some of the metrics available to the system[18], and consider how optimizing the acquisition model with respect to a metric can enforce compliance with an economy condition:

1. The number of *active* lexical feature sequences[19] in the lexicon model (i.e. lexical feature sequences used by some derivation model) is:

$$\sum_{s \in \lambda} (A(s_0) \wedge B(s_0) \wedge C(s_0)) \tag{3.1}$$

   where $\lambda$ is the set of lexical node sequences in the lexicon model, $s_0$ is the starting node (i.e. the first node) in the lexical feature sequence $s$, and:

$$\begin{aligned} A(t) &= (\xi(t) = \tau_{\varnothing}) & (3.2)\\ B(t) &= \bigvee_{p \in \Sigma} (\Delta_{\Omega}(t) = p) & (3.3)\\ C(t) &= \bigvee_{\substack{d_i \in D \\ x \in \mathbb{L}_{\mathbb{N}_{d_i}}}} ((\mu_i(x) = t) \wedge (h_i(x) = x)) & (3.4) \end{aligned}$$

   The conjunction $A(s_0) \wedge B(s_0) \wedge C(s_0)$ serves to check whether the lexical feature sequence $s$ is active: $A(s_0)$ checks that the starting node in a lexical feature sequence is active; $B(s_0)$ checks that the starting node must map to one of the (non-null) phonological forms; $C(s_0)$ checks that the lexical feature sequence participates in at least one derivation by checking whether one of the derivations $d_i$ has a lexical head that maps to the starting node via (the uninterpreted function) $\mu_i$. Metric (3.1) is bounded above by a number of model parameters that collectively determine the maximum number of lexical feature sequences that the lexicon may have.

   The number of syntactic features (both selectional and licensing) over all *active* lexical feature sequences in the lexicon model is:

$$\sum_{\substack{s \in \lambda \\ x \in s}} (A(s_0) \wedge B(x)) \tag{3.5}$$

   where $A$ and $B$ are defined as in metrics (3.2) and (3.3) respectively. Metric (3.5) is bounded above by the product of the maximum number of lexical entries the lexicon may have and the maximum number of syntactic features a lexical entry may have.

   Optimizing with respect to the metrics (3.1) and (3.5) reduces the total size of the lexicon.

2. Given a derivation $d$, the total number of merge operations in the derivation is the sum of the number external and internal merge operations:

$$\sum_{x \in \mathbb{N}_d} (h_{d_i}(x) = x) \tag{3.6}$$

[18] The reader may want to review the definitions of the sorts (see ) and uninterpreted functions listed in Table 2.2 and Table 2.3 respectively.

[19] Recall from §2.3.1 that the lexical entries in a lexicon are made up of (phonological form, lexical feature sequence) pairs.

The number of (merge) operations in a derivation is bounded above by the product of two parameters of the derivation: (i) the maximum node-sequence length; (ii) the maximum number of node-sequences (both overt and covert). Given this, the total number of merge operations across a set of derivations $D$ is:

$$\sum_{\substack{d_i \in D \\ x \in \mathbb{N}_{d_i}}} (h_{d_i}(x) = x) \tag{3.7}$$

Metric (3.7) is bounded above by the sum of the upperbounds on number of merge operations for each derivation in $D$.

Optimizing a derivation model with respect to the metric (3.5) serves to make the derivation as economical as possible in that it minimizes the total number of structure building operations required for the derivation to converge (this metric was also employed by the model of the parser in the prior chapter); optimizing with respect to the metric (3.7) thus has the effect of simultaneously making every derivation connected to the lexicon as economical as possible by minimizing the total number of operations occurring over all of the derivations (e.g. by reducing instances of movement and empty lexical heads).

3. The number of *distinct* selectional features in the lexicon is:

$$\sum_{y \in \mathbb{F}_S} \left( \bigvee_{\substack{s \in \lambda \\ x \in s}} (\kappa(x) = y) \wedge (\xi(x) = \tau_{\varnothing}) \right) \tag{3.8}$$

Metric (3.8) is bounded above by the cardinality of the set of selectional feature labels specified in the model parameters, and bounded below by one, as every derivation requires at least one application of external merge to converge, and thus at least one selectional feature must be available.

Likewise, the number of *distinct* licensing features in the lexicon is:

$$\sum_{y \in \mathbb{F}_L} \left( \bigvee_{\substack{s \in \lambda \\ x \in s}} (\kappa(x) = y) \wedge (\xi(x) = \tau_{\varnothing}) \right) \tag{3.9}$$

Metric (3.9) is bounded above by the cardinality of the set of licensing feature labels specified in the model parameters, and bounded below by zero, as a derivation does not necessarily require any instances of internal merge to occur.

Minimizing the number of distinct selectional/licensing features labels reduces the number of distinct symbols (i.e. feature labels) the lexicon has and, together with the minimization of lexical feature sequences, aims to minimize the number of bits needed to represent the lexicon (from an information theoretic viewpoint).

Whereas the number of distinct selectional features may be one and the ability to yield desired structures will not be hampered because every selectors will be able to select any selectee, this is not the case with licensing features, because of the Shortest Movement Condition; notably, the minimum number of licensing features is particularly interesting because it sets a limit on how many long distance dependencies may all be simultaneously pairwise crossing within a derivation.

The system can optimize the acquisition model with respect to several of these metrics by ordering and combining them together to produce a lexicographically ordered metric. The order in which the metric are optimized by the system plays a critical role in the total effect they have. To see this, let us consider the particular combination of metrics that will be used in the instantaneous model of acquisition developed in this section. The system will first optimize the acquisition model with respect to (3.1), then (3.5), followed by (3.7) and finally (3.8) and (3.9). The order of optimization metrics applied locks in the size of the lexicon by minimizing the number of lexical feature sequences and the number of features in the lexicon; only after the size of the lexicon is determined does the system proceed to minimize the number of features involved in each derivation, in effect simultaneously optimizing all of the derivations connected to the already size-constrained lexicon model. Note that the combination of first two optimization metrics, (3.1) and (3.5), is not impacted by the number of syntactic structures the lexicon must yield, but rather by the set of syntactic structures the lexicon must be able to yield; it follows that if the first two metrics are optimized, then minimizing the number of derivation nodes will result not in a smaller lexicon. See Table 3.7 and Table 3.6 for examples of unoptimized and optimized lexicons (that were inferred from the same PLD) respectively, and see Figure 3-4b and Figure 3-4a for examples of unoptimized and optimized derivations (for the same entry in the PLD) that were yielded by the aforementioned lexicons. *(Note that all of the derivations presented in both this section and §3.3* do *accord with the ordering of phonological forms listed in the associated PLD entry – i.e. if they are redrawn with Specifier–Head-Complement linearization, then the correct SVO ordering becomes apparent; the figures were automatically rendered using Graphviz so that the arrows depicting syntactic movement would not overlap with the projections in the derivation.)*

### 3.2.2 Inferring an Optimal Grammar via Model Checking

We will now introduce an acquisition procedure (listed on Pg. 123) for inferring minimalist grammars that are optimal with respect to supplied optimization metrics; we will then review how this procedure takes the form of a child language learner that comports with the criterion set out in the Aspect's model of language acquisition.

Let us now consider details of how this procedure is implemented, and how it adheres to the form of a child language learner, as presented in §3.1.3, focusing in particular on how steps 3 and 4 of the procedure embody the functions $P$ and $Q$ respectively.

Step 3 plays the role of the function $Q$, consuming the input from entry $I_i$ in the PLD, and advancing the state $S_i$ to the state $S_{i+1}$; by repeated application of $Q$, the learner processes the entire PLD, driving the state of the learner from the initial state, $S_0$, to the final state, $S_n$. Each state $S_i$ is a system of first-order, quantifier-free logical formula that constitutes an SMT-model, encoding the (decidable) decision problem of whether there is a minimalist lexicon that is compatible with the PLD consumed up until that point – i.e. a minimalist lexicon that can yield a derivation for each pair of interface conditions consumed thus far from the PLD[20]; furthermore, the set of satisfiable interpretations of $S_i$ constitutes the class of possible hypothesis of language structures from condition (iii) of the Aspects

[20] All logical formulas in this study, being used to encode finite models over bounded domains, are first-order and quantifier-free; this has the benefit that these formulas are decidable.

| ID | Model Parameter | Description |
|---|---|---|
| (a) | Max. Num. Empty Lex. Items | Upper bound on the number of empty (i.e. unpronounced) lexical heads that may participate in a derivation. |
| (b) | Max. Num. Phrasal Movements | Upper bound on the number of (phrasal) movements (i.e. internal merge operations) that occur in a derivation. |
| (c) | Max. Num. Head Movements | Upper bound on the number of head movement operations that occur in a derivation. |
| (d) | Max. Num. Feats. per Lex. Item | Upper bound on the number of features in a lexical item. |
| (e) | Num. Overt Lex. Items per PF | Number of lexical feature sequences allocated, for each overt phonological form in the PLD, in the lexicon model; these lexical feature sequences may only associate with overt phonological forms. |
| (f) | Num. Covert Lex. Items in Lexicon | Number of lexical feature sequences allocated in the lexicon model for association with the covert phonological form; note that these lexical feature sequences can only associate with the covert phonological form. |
| (g) | Max. Num. Overt PF Connections | Upper bound on the number of lexical feature sequences an overt phonological form may associate with. |
| (h) | Max. Num. Covert PF Connections | Upper bound on the number of lexical feature sequences the covert phonological form may associate with. |
| (i) | Selectional Feature Labels | Finite set of labels for selectional features in the lexicon model. |
| (j) | Licensing Feature Labels | Finite set of labels for selectional features in the lexicon model. |

Table 3.1: Model Parameters for the Acquisition Procedure. These parameters are all finite and serve to bound the acquisition model. Parameters (a-d) pertain to the derivation models and parameters (d-j) pertain to the lexicon models.

## Instantaneous Acquisition Procedure

1. The input to the procedure consists of:
   (a) a queue of pairs of interface conditions, referred to as the PLD, with $n > 0$ entries;
   (b) a valuation of model parameters;
   (c) an *empty* SMT-solver stack, $S$, with each entry on the stack an SMT-formula, and the conjunction of the entries on the stack referred to as "the acquisition model." (Note that to "check the acquisition model" is to use the SMT-solver to check the conjunction of the terms on the solver's stack.)

2. The initial state of the learner, prior to consuming the PLD, is an empty lexicon:
   (a) initialize a lexicon model (i.e. an SMT formula), $m_l$ from the supplied model parameters and the PLD;
   (b) push $m_l$ onto the stack;
   (c) (optional) check the acquisition model.

3. The learner processes the PLD until it is empty, incrementally constraining the lexicon model:
   (a) pop an entry $I_i$ off of the queue;
   (b) initialize a derivation model (i.e. an SMT formula), $m_d^i$, from model parameters and interface conditions $I_i$;
   (c) push $m_d^i$ onto the stack;
   (d) translate $I_i$ into an SMT–formula, $m_I^i$, that constrains the derivation model $m_d^i$;
   (e) push $m_I^i$ onto the stack;
   (f) construct an SMT-formula, $m_b^i$, that connects, via an uninterpreted function, the derivation model, $m_d^i$, to the lexicon model, $m_l$;
   (g) push $m_b^i$ onto the stack;
   (h) (optional) check the acquisition model.

4. The learner selects a grammar by optimizing the model:
   (a) optimize the acquisition model using metric (3.1);
   (b) optimize the acquisition model using metric (3.5);
   (c) optimize the acquisition model using metric (3.7);
   (d) optimize the acquisition model using metric (3.8);
   (e) optimize the acquisition model using metric (3.9);
   (f) check the acquisition model using the SMT-solver, and if the acquisition model is found to be satisfiable, recover the identified (satisfiable) model interpretation (i.e. a solution to the acquisition model).

5. The output of the procedure is the final state of the learner:
   (a) for each entry $I_i$ in the PLD, a derivation, $d_i$, that satisfies the conditions imposed by $I_i$;
   (b) the inferred minimalist lexicon that can yield each $d_i$;
   (c) (Optional) the recovered model interpretation;
   (d) the solver stack holds: $m_l$; $m_d^i$ and $m_I^i$ for $1 \leq i \leq n$; constraints associated with each optimization metric.

model. The initial state, $S_0$, consists of an empty and unconstrained lexicon model, the size of which is determined by the valuation of (relevant) model parameters supplied in the input (see step 2).[21] The final state, $S_n$, consists of the SMT-formulae on the solver-stack after running step 3 – i.e. the lexicon model, the derivation models, and the formulae connecting the derivation models to the lexicon model. The procedure advances from one state to the next by consuming an entry (i.e. a pair of LF and PF interface conditions) from the PLD and then pushing onto the solver stack SMT-formulae that: instantiate a new model of a (minimalist) derivation (see step 3b); require that this derivation satisfy the interface conditions (see step 3d; and require that this derivation be produced by lexical entries drawn from the lexicon (see step 3f). this process serves to increasingly constrain the lexicon model by requiring that any (satisfiable) interpretation thereof must be compatible with the PLD that the learner has processed up until that point.

Step 4 plays the role of the function $R$, extracting an optimal grammar from a given state $S_i$ using an SMT-solver.[22] This is accomplished by first optimizing the acquisition model to obtain optimal metric values (see steps 4a-4e), and then using the SMT-solver to infer a lexicon, $G_i = R(S_i)$, by solving the conjunction of the formulae in $S_i$ and additional formulae that enforce that the optimal metric values determined (in steps 4a-4e) are respected by any solution to the acquisition model; this conjunction is summarized in 5d. Optimization (i.e. minimization) of the acquisition model with respect to a metric (i.e. in steps 4a-4e) is carried out by a sub-procedure that is outline here:

1. The input to the sub-procedure is: a metric, $m$; an upper bound for the metric, $m_{max}$; a lower bound for the metric, $m_{min}$; the acquisition model (i.e. the conjunction of terms on the solver's stack) that is to be optimized.

2. For each $k \in [m_{min}, m_{max}]$:

   (a) the sub-procedure constructs an SMT-formula $p_{m,k} \rightarrow (m \leq k)$, where $p_{m,k}$ is a boolean variable that serves to track whether the pseudo-boolean equation $m \leq k$ holds;

   (b) the constructed formula is added to the acquisition model by pushing it onto the solver stack.[23]

3. Then the optimal (i.e. smallest) value that the metric can take on is identified by searching over the range $[m_{min}, m_{max}]$, evaluating each potential metric value $k$ by checking the acquisition model with the additional assertion that $p_{m,k} = True$.[24]

[21] Additionally the sort of phonological forms (i.e. $\Sigma$) is established by scanning the PLD and obtaining the vocabulary as the union of the sets of overt phonological forms appearing in each PF interface condition listed in the PLD.

[22] This thesis uses Z3 SMT-solver (De Moura and Bjørner, 2008), a high-performance solver for Satisfiability Modulo Theories that can solve multi-sort quantifier-free first-order logic formulas that may combine symbols from a set of additional logics defined by a number of background theories such as the theory of uninterpreted functions with equality. Z3 centers on an implementation of the DPLL-algorithm (Davis et al., 1962; Nieuwenhuis et al., 2006) (which integrates back-tracking search with propagation of boolean constraints) that incorporates Conflict-Driven Clause Learning (CDCL) (Marques Silva and Sakallah, 1996; Marques-Silva and Sakallah, 1999).

[23] Adding this formula to the acquisition model does not force the pseudo-boolean equation on the right-hand side of the implication to take on a value less than or equal to $k$, as $p_{m,k}$ is a free variable; whether the acquisition model can be interpreted satisfiably when the metric takes on a value less than or equal to $k$ can be determined by checking the acquisition model while asserting that $p_{m,k}$ is true.

[24] The search can be done with either a linear search or a binary search; the reference implementation includes both.

4. The optimization sub-procedure concludes by pushing on to the solver's stack the formula ($p_{m,k_{optimal}} = True$), which forces any interpretation of the acquisition model to be such that when measured, the metric $m$ has the value $k_{optimal}$.

The value of each optimization metric can be computed for an interpreted model. When we run the solver to find a solution with metric value less than or equal to $k$, if the solver finds a solution with a metric value less than $k$, then we can advance our search using this information – e.g. if the found solution has a metric value of $k - 2$, then there is no need for us to run the solver looking for a solution with metric value of $k - 1$.

Having defined the inference procedure, some comments and observations are in order.

- The psychological plausibility condition pertaining to finite memory is only partially respected — i.e although the learner does need to hold all derivation models in head to solve at once, and the grammar learned only takes up a finite amount of memory, much smaller than size of PLD. Note that the PLD queue must have a finite number of items, as the entire PLD must be consumed before the inference procedure can output the inferred grammar; looking ahead, this requirement will be lifted when we turn to the *incremental acquisition procedure* that will be introduced in §3.3.
- The order in which the PLD corpus is processed by the procedure does not matter for two reasons: firstly, when the solver checks the acquisition model, it evaluates the *conjunction* of all formulae on the (solver's) stack at that point, and thus the order of terms doesn't impact the space of satisfiable interpretations (of the acquisition model); secondly, the optimization metrics are symmetric under permutation of the entries in the PLD (by commutativity of addition).
- The inferred set of lexicons $G_i$ is not enumerated explicitly – rather, it exists implicitly in the acquisition model produced by the SMT-solver (i.e. the solution to the system of logical formulas), and members of this set may be filtered, searched and sampled by querying this model using the SMT-solver.

### 3.2.3 Learning a Grammar of Matrix Clause Constructions

We will now present a computational experiment in which the acquisition procedure introduced in §3.2.2 is used to infer the minimalist lexicon listed in Table. 3.3 from the PLD listed in Table 3.2. We will first walk through the application of the acquisition procedure, and then evaluate and analyze the acquired (target) grammar (i.e. the inferred lexicon).

**Using the Instantaneous Acquisition Procedure to Infer an Optimal Grammar.**

Let us now step through the application of the acquisition procedure and see how the particulars play out in the context of this computational experiment.

1. The input to the procedure consists of the PLD and the supplied valuation of model parameters listed in Table 3.5. The PLD has a total of 29 entries (involving a total of 26 distinct phonological forms), and the expressions appearing in the PLD were selected so as to span a diverse range of (core) syntactic structures that do not include any embedded clauses. These expressions are comprised of: predicates of varying valency (i.e. intransitive, transitive and ditransitives verbs); arguments that are either

| ID | PF Interface Conditions | LF Interface Conditions |
|---|---|---|
| $I_0$ | who has eaten/V icecream/N? | $\theta_{\text{eaten}}[s\colon \text{who}, o\colon \text{icecream}]$, $Agr_{\text{has}}[s\colon \text{who}]$ |
| $I_1$ | icecream/N was eaten/V. | $\theta_{\text{eaten}}[o\colon \text{icecream}]$, $Agr_{\text{was}}[s\colon \text{icecream}]$ |
| $I_2$ | who was eating/V icecream/N? | $\theta_{\text{eating}}[s\colon \text{who}, o\colon \text{icecream}]$, $Agr_{\text{was}}[s\colon \text{who}]$ |
| $I_3$ | was pizza/N eaten/V? | $\theta_{\text{eaten}}[o\colon \text{pizza}]$, $Agr_{\text{was}}[s\colon \text{pizza}]$ |
| $I_4$ | what has john/N eaten/V? | $\theta_{\text{eaten}}[s\colon \text{john}, o\colon \text{what}]$, $Agr_{\text{has}}[s\colon \text{john}]$ |
| $I_5$ | has mary/N eaten/V pizza/N? | $\theta_{\text{eaten}}[s\colon \text{mary}, o\colon \text{pizza}]$, $Agr_{\text{has}}[s\colon \text{mary}]$ |
| $I_6$ | was john/N eating/V pizza/N? | $\theta_{\text{eating}}[s\colon \text{john}, o\colon \text{pizza}]$, $Agr_{\text{was}}[s\colon \text{john}]$ |
| $I_7$ | what was mary/N eating/V? | $\theta_{\text{eating}}[s\colon \text{mary}, o\colon \text{what}]$, $Agr_{\text{was}}[s\colon \text{mary}]$ |
| $I_8$ | what was eaten/V? | $\theta_{\text{eaten}}[o\colon \text{what}]$, $Agr_{\text{was}}[s\colon \text{what}]$ |
| $I_9$ | was mary/N given/V pizza/N? | $\theta_{\text{given}}[o\colon \text{pizza}, i\colon \text{mary}]$, $Agr_{\text{was}}[s\colon \text{mary}]$ |
| $I_{10}$ | what has mary/N given/V john/N? | $\theta_{\text{given}}[s\colon \text{mary}, o\colon \text{what}, i\colon \text{john}]$, $Agr_{\text{has}}[s\colon \text{mary}]$ |
| $I_{11}$ | mary/N has given/V john/N money/N. | $\theta_{\text{given}}[s\colon \text{mary}, o\colon \text{money}, i\colon \text{john}]$, $Agr_{\text{has}}[s\colon \text{mary}]$ |
| $I_{12}$ | who was money/N given/V to/P? | $\theta_{\text{given}}[o\colon \text{money}, i\colon \text{to who}]$, $Agr_{\text{was}}[s\colon \text{money}]$ |
| $I_{13}$ | who has john/N given/V money/N to/P? | $\theta_{\text{given}}[s\colon \text{john}, o\colon \text{money}, i\colon \text{to who}]$, $Agr_{\text{has}}[s\colon \text{john}]$ |
| $I_{14}$ | was the boy/N sleeping/V? | $\theta_{\text{sleeping}}[s\colon \text{the boy}]$, $Agr_{\text{was}}[s\colon \text{the boy}]$ |
| $I_{15}$ | the boy/N has slept/V. | $\theta_{\text{slept}}[s\colon \text{the boy}]$, $Agr_{\text{has}}[s\colon \text{the boy}]$ |
| $I_{16}$ | john/N was told/V nothing/N. | $\theta_{\text{told}}[o\colon \text{nothing}, i\colon \text{john}]$, $Agr_{\text{was}}[s\colon \text{john}]$ |
| $I_{17}$ | someone/N has known/V everything/N. | $\theta_{\text{known}}[s\colon \text{someone}, o\colon \text{everything}]$, $Agr_{\text{has}}[s\colon \text{someone}]$ |
| $I_{18}$ | who was asking/V nothing/N? | $\theta_{\text{asking}}[s\colon \text{who}, o\colon \text{nothing}]$, $Agr_{\text{was}}[s\colon \text{who}]$ |
| $I_{19}$ | nothing/N was asked/V. | $\theta_{\text{asked}}[o\colon \text{nothing}]$, $Agr_{\text{was}}[s\colon \text{nothing}]$ |
| $I_{20}$ | everything/N was known/V. | $\theta_{\text{known}}[o\colon \text{everything}]$, $Agr_{\text{was}}[s\colon \text{everything}]$ |
| $I_{21}$ | who was everything/N told/V to? | $\theta_{\text{told}}[o\colon \text{everything}, i\colon \text{to who}]$, $Agr_{\text{was}}[s\colon \text{everything}]$ |
| $I_{22}$ | john/N has asked/V someone/N everything/N. | $\theta_{\text{asked}}[s\colon \text{john}, o\colon \text{everything}, i\colon \text{someone}]$, $Agr_{\text{has}}[s\colon \text{john}]$ |
| $I_{23}$ | what was someone/N asked/V? | $\theta_{\text{asked}}[o\colon \text{what}, i\colon \text{someone}]$, $Agr_{\text{was}}[s\colon \text{someone}]$ |
| $I_{24}$ | who has told/V someone/N the story/N? | $\theta_{\text{told}}[s\colon \text{who}, o\colon \text{the story}, i\colon \text{someone}]$, $Agr_{\text{has}}[s\colon \text{who}]$ |
| $I_{25}$ | a boy/N was eating/V the pizza/N. | $\theta_{\text{eating}}[s\colon \text{a boy}, o\colon \text{the pizza}]$, $Agr_{\text{was}}[s\colon \text{a boy}]$ |
| $I_{26}$ | john/N has told/V mary/N a story/N. | $\theta_{\text{told}}[s\colon \text{john}, o\colon \text{a story}, i\colon \text{mary}]$, $Agr_{\text{has}}[s\colon \text{john}]$ |
| $I_{27}$ | the story/N was told/V to a boy/N. | $\theta_{\text{told}}[o\colon \text{the story}, i\colon \text{to a boy}]$, $Agr_{\text{was}}[s\colon \text{the story}]$ |
| $I_{28}$ | what was john/N asking/V? | $\theta_{\text{asking}}[s\colon \text{john}, o\colon \text{what}]$, $Agr_{\text{was}}[s\colon \text{john}]$ |

Table 3.2: Primary Linguistic Data (PLD). The learner is presented with a sequence of pairs of PF and LF interface conditions – i.e. a sequence of sentences annotated with syntactic relations. The PLD is an input to the acquisition procedure. The PF interface conditions consist of a tokenized sentence, with some tokens having their category prespecified (indicated by a suffix of a slash followed by the category). The LF interface conditions consists of: (i) locality constraints that include agreement (*Agr*) and predicate-argument structure (i.e. a $\theta$ grid), with the predicate indicated in the suffix and the subject, object and indirect object components marked by "s:", "o:" and "i:" respectively; (ii) the type of the sentence – i.e. either declarative or interrogative – is also annotated on each sentence, indicated by the end-of-sentence punctuation. The LF interface conditions are entirely hierarchical/structural in the constraints they impose – i.e. the values filling the slots consist of *sets* of tokens, not sequences of tokens.

simple uncountable nouns or more complex determiner phrases; various types of active and passive voiced constructions including declaratives, yes/no questions (formed by subj.-aux. inversion), and wh-questions. See Table 3.8 for further classification of the entries in the PLD.

2. The initial state of the learner is a lexicon model that is not constrained by having to be able to produce particular derivations to satisfy stipulated interface conditions – i.e. the initial lexicon is a blank slate for the learner. At this point in the acquisition

| ID | Category | Features | Phonological Forms | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | pizza | everything | john | mary | nothing | icecream | someone | money | story | boy | a | the | to | sleeping | slept | has | was | eating | eaten | asking | known | asked | told | given | who | what | $\epsilon$ |
| $\mathfrak{L}_1$ | $V$ | $= x_0, \sim x_0$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | × | × | × | × | × | · | · | · | · | · |
| $\mathfrak{L}_2$ | $V$ | $= x_0, = x_0, \sim x_0$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | × | × | × | · | · | · |
| $\mathfrak{L}_3$ | $C_{Decl.}$ | $= x_0, C$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | × |
| $\mathfrak{L}_4$ | $C_{ques.}$ | $<= x_0, +z, C$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | × |
| $\mathfrak{L}_5$ | $v$ | $<= x_0, \sim x_0$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | × |
| $\mathfrak{L}_6$ | $C_{ques.}$ | $<= x_0, C$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | × |
| $\mathfrak{L}_7$ | $v$ | $<= x_0, = x_0, \sim x_0$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | × |
| $\mathfrak{L}_8$ | $P$ | $= x_0, \sim x_0$ | · | · | · | · | · | · | · | · | · | · | · | · | × | · | · | · | · | · | · | · | · | · | · | · | · | · | · |
| $\mathfrak{L}_9$ | $D$ | $= x_0, \sim x_0, -l$ | · | · | · | · | · | · | · | · | · | · | × | × | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · |
| $\mathfrak{L}_{10}$ | $D$ | $= x_0, \sim x_0$ | · | · | · | · | · | · | · | · | · | · | × | × | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · |
| $\mathfrak{L}_{11}$ | $D$ | $\sim x_0, -z$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | × | × | · |
| $\mathfrak{L}_{12}$ | $D$ | $\sim x_0, -l, -z$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | × | × | · |
| $\mathfrak{L}_{13}$ | $T$ | $= x_0, +l, \sim x_0$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | × | × | · | · | · | · | · | · | · | · | · | · |
| $\mathfrak{L}_{14}$ | $V$ | $\sim x_0$ | · | · | · | · | · | · | · | · | · | · | · | · | · | × | × | · | · | · | · | · | · | · | · | · | · | · | · |
| $\mathfrak{L}_{15}$ | $N$ | $\sim x_0, -l$ | × | × | × | × | × | × | × | × | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · |
| $\mathfrak{L}_{16}$ | $N$ | $\sim x_0$ | × | × | × | × | × | × | × | × | × | × | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · |

Table 3.3: Factored View of the Optimized Inferred Lexicon. This is a *factored* view of the optimized inferred lexicon (listed in Table 3.6) that was the output of the acquisition procedure applied to the primary linguistic data listed in Table 3.2 using the valuation of model parameters listed in Table 3.5. The 27 columns each code for a distinct phonological form; there are 26 overt phonological forms, and one covert phonological form (i.e. $\epsilon$). The 16 rows each code for a distinct lexical feature sequence. An entry in the table indicates that the pairing of the associated phonological form and lexical feature sequence is an entry in the lexicon. Since every entry in the lexicon can be uniquely factored apart into a pairing of a phonological form and a lexical feature sequence, there is a one-to-one mapping between the entries in this table and the entries in the lexicon. The rows and columns have been seriated (using the hamming distance metric) so as to visually group together similar entries.

trajectory, the class of possible hypothesis – i.e. the set of satisfiable interpretations of the acquisition model constructed thus far – is the set of all minimalist lexicons that can accord with the bounds established by the supplied valuation of model parameters and thus can be encoded by the lexicon model. The size of the lexicon model initialized in Step 2 of the procedure – i.e. the size of the lexicon node sort $\Omega$ - is computed as follows. First, after the lexicon model is initialized, all phonological forms are scraped from the PLD so as to determine the number of distinct overt phonological forms, in this case 26. The size of the lexicon model is then computed using the following model parameters[25]:

- Model parameter (g) – i.e. “Maximum Number of Overt PF Connections” – which has a value of 1; as there are a total of 26 distinct phonological forms, $(26 \times 1)$ lexical feature sequences will be allocated for association with overt phonological forms.
- Model parameter (h) – i.e. “Maximum Number of Covert PF Connections” – which has a value of 6; as there is only one covert phonological form, $\epsilon$, a total

[25] This computation enables the size of the lexicon model to be automatically determined as a function of the PLD – c.f. simply stipulating the maximum number of lexical entries in the lexicon, which needs to be adjusted for different PLD.

| ID | Category | Features | Input Sentence | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | $I_{27}$ | $I_{16}$ | $I_9$ | $I_3$ | $\left\{\begin{matrix} I_1, \\ I_{19}, \\ I_{20} \end{matrix}\right\}$ | $I_8$ | $\left\{\begin{matrix} I_0, \\ I_2, \\ I_{18} \end{matrix}\right\}$ | $I_{24}$ | $I_{26}$ | $\left\{\begin{matrix} I_{11}, \\ I_{22} \end{matrix}\right\}$ | $I_{10}$ | $I_{13}$ | $\left\{\begin{matrix} I_{12}, \\ I_{21} \end{matrix}\right\}$ | $I_{23}$ | $\left\{\begin{matrix} I_4, \\ I_7, \\ I_{28} \end{matrix}\right\}$ | $\left\{\begin{matrix} I_5, \\ I_6 \end{matrix}\right\}$ | $I_{17}$ | $I_{25}$ | $I_{15}$ | $I_{14}$ |
| $\mathfrak{L}_1$ | $V$ | $= x_0, \sim x_0$ | · | · | · | 1 | 1 | 1 | 1 | · | · | · | · | · | · | · | 1 | 1 | 1 | 1 | · | · |
| $\mathfrak{L}_2$ | $V$ | $= x_0, = x_0, \sim x_0$ | 1 | 1 | 1 | · | · | · | · | 1 | 1 | 1 | 1 | 1 | 1 | 1 | · | · | · | · | · | · |
| $\mathfrak{L}_3$ | $C_{Decl.}$ | $= x_0, C$ | 1 | 1 | · | · | 1 | · | · | · | 1 | 1 | · | · | · | · | · | · | 1 | 1 | 1 | · |
| $\mathfrak{L}_4$ | $C_{ques.}$ | $<= x_0, +z, C$ | · | · | · | · | · | 1 | 1 | 1 | · | · | 1 | 1 | 1 | 1 | 1 | · | · | · | · | · |
| $\mathfrak{L}_5$ | $v$ | $<= x_0, \sim x_0$ | 1 | 1 | 1 | 1 | 1 | 1 | · | · | · | · | · | · | 1 | 1 | · | · | · | · | · | · |
| $\mathfrak{L}_6$ | $C_{ques.}$ | $<= x_0, C$ | · | · | 1 | 1 | · | · | · | · | · | · | · | · | · | · | · | 1 | · | · | · | 1 |
| $\mathfrak{L}_7$ | $v$ | $<= x_0, = x_0, \sim x_0$ | · | · | · | · | · | · | 1 | 1 | 1 | 1 | 1 | 1 | · | · | 1 | 1 | 1 | 1 | 1 | 1 |
| $\mathfrak{L}_8$ | $P$ | $= x_0, \sim x_0$ | 1 | · | · | · | · | · | · | · | · | · | · | 1 | 1 | · | · | · | · | · | · | · |
| $\mathfrak{L}_9$ | $D$ | $= x_0, \sim x_0, -l$ | 1 | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | 1 | 1 | 1 |
| $\mathfrak{L}_{10}$ | $D$ | $= x_0, \sim x_0$ | 1 | · | · | · | · | · | · | 1 | 1 | · | · | · | · | · | · | · | · | 1 | · | · |
| $\mathfrak{L}_{11}$ | $D$ | $\sim x_0, -z$ | · | · | · | · | · | · | · | · | · | · | 1 | 1 | 1 | 1 | 1 | · | · | · | · | · |
| $\mathfrak{L}_{12}$ | $D$ | $\sim x_0, -l, -z$ | · | · | · | · | · | 1 | 1 | 1 | · | · | · | · | · | · | · | · | · | · | · | · |
| $\mathfrak{L}_{13}$ | $T$ | $= x_0, +l, \sim x_0$ | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| $\mathfrak{L}_{14}$ | $V$ | $\sim x_0$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | 1 | 1 |
| $\mathfrak{L}_{15}$ | $N$ | $\sim x_0, -l$ | · | 1 | 1 | 1 | 1 | · | · | · | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | · | · | · |
| $\mathfrak{L}_{16}$ | $N$ | $\sim x_0$ | 2 | 1 | 1 | · | · | · | 1 | 2 | 2 | 2 | 1 | 1 | · | · | · | 1 | 1 | 2 | 1 | 1 |

Table 3.4: Summary of Derivations. This table shows how the optimized inferred lexicon (listed in Table 3.3) is compatible with the primary linguistic data (listed in Table 3.2). Each of the 16 rows code for a distinct lexical feature sequence. Each of the 20 columns codes for a distinct syntactic structure that is constructed by the lexical feature sequences coded for in that column (with the integer values indicating how many times each lexical feature sequence participates). The columns have been seriated (using the hamming distance metric) so as to visually group together similar columns. This table shows that various subsets of the 16 lexical feature sequences can yield at least 20 distinct syntactic structures.

| ID | Model Parameter | Value |
|---|---|---|
| (a) | Maximum Number of Empty Lexical Items | 2 |
| (b) | Maximum Number of Phrasal Movements | 2 |
| (c) | Maximum Number of Head Movements | 2 |
| (d) | Maximum Number of Features per Lexical Item | 3 |
| (e) | Number of Overt Lexical Items per PF | 2 |
| (f) | Number of Covert Lexical Items per PF | 5 |
| (g) | Maximum Number of Overt PF Connections | 1 |
| (h) | Maximum Number of Covert PF Connections | 6 |
| (i) | Selectional Features Labels | $[x_0, x_1, x_2]$ |
| (j) | Licensing Features Labels | $[l, r, z]$ |

Table 3.5: Valuation of model parameters for the computational experiment carried out in §3.2.3. See Table 3.1 for a description of these model parameters.

of ($1x6$) lexical features sequences will be allocated for association with $\epsilon$.

- Model parameter (d) – i.e. "Maximum Number of Features per Lexical Item" – which has a value of 3; thus, 3 syntactic features will be allocated for each of the lexical feature sequence that have been allocated.

The size of the lexicon model is the product of (i) the number of syntactic features allocated per allocated lexical feature sequence, and (ii) the number of allocated lexical feature sequences:

$$|\Omega| = 96 = 3 \times ((26 \times 1) + (1 \times 6))$$

3. The learner then proceeds to process the PLD (arranged as a queue) one entry at a time. After the first pair of interface conditions, $I_0$, is taken off the queue, the learner constructs the first derivation model, constrains it and connects it to the lexicon; the size of the first derivation model – i.e. the size of the *derivation node sort* ($\mathbb{N}$) – can be calculated as a function of the following terms: (i) the number of tokens in the sentence – in this case, $I_0$ has 4 tokens in the expression *"who has eaten icecream?"* – from which is determined the number of overt lexical items that will participate in the derivation; (ii) the maximum number of empty lexical items that may participate in the derivation, in this case, 2 (per model parameter (a)); (iii) the maximum number of features per lexical item, in this case 3 (per model parameter (d)). Then the size of the derivation model, $|\mathbb{N}|$, is:

   $$18 = 3 \times (4 + 2) + 2$$

   Here $3 \times (4+2)$ is the product of the maximum number of features per lexical entry (i.e. 3) and the maximum number of lexical entries that can participate in the derivation $(4 + 2)$ (the addition of 2 arising from the inclusion of the *root node* and the *bottom node*).[26][27]

   The learner next proceeds to process, one by one, the remaining entries in the PLD. As each remaining entry in the PLD is processed, a new derivation model is initialized, constrained by the LF and PF interface conditions listed in that entry, connected to the lexicon model via an uninterpreted function, and added to the solver's stack, thereby growing the acquisition model. Each new derivation model, being tasked with parsing a particular entry in the PLD, will have a different size, as the number of tokens in each entry can vary; furthermore, each derivation model has its own separate set of sorts and uninterpreted functions, and must be connected to the lexicon model with its own bus (i.e. the uninterpreted function $\mu$ that maps nodes in the derivation model to nodes in the lexicon model), so that no two derivation models affect one another in any way except that they are both constrained by the same lexicon model. As each derivation model is connected to the lexicon, thereby further constraining the class of satisfiable interpretations of the lexicon model, the solver is being tasked with having to figure out how to do more with a lexicon that is bounded in size.

4. Having processed the PLD and constructed the acquisition model (i.e. the lexicon model and the derivation models connected to it), the learner next proceeds to add constraints derived from the optimization metrics to the acquisition model (in Step 4), thereby enabling the SMT-solver to infer the optimal grammar. Let us consider the application of each step of this scheme for optimizing the acquisition model; for each metric (including this one), after determining the optimal metric value, the acquisition model is constrained to take on that particular value.

   (a) The learner proceeds to optimize the acquisition model using metric (3.1), so as to minimize the number of lexical entries. The metric is bounded above by the maximum number of lexical entries in the lexicon model, established in the prior

[26] See §**??** for further details.

[27] Note that, up until this point, the procedure has a operated in a manner similar to the procedure for parsing, aside from the lack of a supplied lexicon that is used to constrain the lexicon, and the size of the lexicon model being computed as a function of supplied model parameters rather than, in the case of parsing, the supplied lexicon.

step to be:

$$32 = (26 \times 1) + (1 \times 6)$$

and the metric is bound below by 1 as the lexicon must have at least one lexical feature sequence. Upon evaluating compatibility of optimization metric values between these bounds, the learner determines that the minimal metric value is 16, implying that there is *no* possibility of the learner inferring a lexicon with fewer than 16 lexical feature sequences as that is the minimal number of lexical feature sequences required for a lexicon to be compatible with the PLD.[28] A constraint enforcing that interpretations of the acquisition model have this metric value is then added to the solver stack, thereby restricting the class of solutions to the acquisition model to those in which the lexicon has exactly 16 lexical feature sequences.

(b) The learner next proceeds to optimize the acquisition model using metric (3.5) so as to minimize the number of syntactic features (i.e. selectional and licensing features) in the lexicon. This metric is bounded above by the product of the number of lexical entries (i.e. 16) and the maximum number of features (i.e. 3), which comes out to 48.[29] This metric is bounded below by the product of the minimum number of features per lexical item (i.e. 1) and the minimum number of lexical items (i.e. 16), which comes out to 16. Upon evaluating compatibility of optimization metric values between these bounds, the learner determines the minimal metric value to be 33. A constraint enforcing that interpretations of the acquisition model have this metric value is then added to the solver stack, thereby restricting the class of solutions to the acquisition model to those in which the lexicon has exactly 16 lexical feature sequences and exactly 33 selectional and licensing features.

(c) The learner then proceeds to optimize the acquisition model with metric (3.7) so as to minimize the number of *merge* operations in each derivation.[30] This metric has an upperbound of 585, which is the sum of the maximum size of the derivation node sort in each derivation model, and a lower bound of 29, which is equal to the number of entries in the PLD, as there must be a derivation model for each entry in the PLD, and each derivation must have a non-empty derivation node sort. Upon evaluating compatibility of optimization metric values between these bounds, the learner determines the minimal metric value to be 425. A constraint enforcing that interpretations of the acquisition model have this metric value is then added to the solver stack.

(d) The learner next optimizes the acquisition model using metric (3.8), so as to minimize the number distinct of selectional feature labels in the lexicon; this metric is bounded above by 3, which is the size of the set of selectional feature labels in the supplied valuation of model parameters, and is bounded below by 1 because

[28] Although the optimal metric value was identified, we the optimal lexicon was not inferred; rather, we are just checking for satisfiability so as to determine whether it is possible for the model to comport with the metric taking on a particular value.

[29] If instead we use the maximum number of lexical entries - i.e. 32 – then the metric is instead bounded above by 96; however, whenever possible the system tries to take advantage of knowledge obtained in prior optimization steps.

[30] This is in accordance with Principles of Economy, having the effect of removing spurious movement operations that occur in the derivation; see 2 and 3.2.1 for further discussion.

there must be at least one instance of *external merge* in a derivation. The learner determines that only 1 selectional feature is required; this is expected as there is no requirement preventing the lexicon from yielding a particular derivation, only a requirement that the lexicon generate a finite set of derivations that satisfy the interface conditions specified in the PLD. A constraint enforcing that interpretations of the acquisition model have this metric value is then added to the solver stack.

(e) Finally, the learner optimizes the acquisition model using metric (3.9), so as to minimize the number of licensing feature labels in the lexicon; this metric is bounded above by 3, which is the size of the set of licensing feature labels in the supplied valuation of model parameters, and is bounded below by 0 because it might be the case that no instances of *internal merge* are required. The learner determines that only 2 licensing features are required; this is anticipated as there are entries in the PLD that may be satisfied by derivations involving a crossing of wh-raising and subject-raising, which will require differing licensing features, pursuant to the Shortest Movement Condition. A constraint enforcing that interpretations of the acquisition model have this metric value is then added to the solver stack.

Having completed the process of optimizing the acquisition model, the acquisition model is next checked using the SMT-solver and found to be satisfiable. It must therefore be the case that whatever the particular inferred lexicon turns out to be, it will have exactly 16 lexical items, 33 syntactic features in the lexicon (not including the special feature $C$), a single selectional feature, and two licensing features; likewise, the derivations produced to satisfy the interface conditions in the PLD will require a total of 425 nodes in total over all of the derivations.

5. Finally, the SMT-solver identifies a satisfiable interpretation of the (optimized) acquisition model, and the learner automatically recovers from this model interpretation both the (optimal) inferred lexicon, and the particular derivations that the inferred lexicon yields (to satisfy the interface conditions in the PLD).[31]

The (optimal) inferred lexicon, listed in Table 3.6, shows us which lexical feature sequences can be associated with which phonological forms. The (optimal) inferred lexicon comports with the optimization constraints applied in steps (4a)-(4e) – i.e. the lexicon has: 16 distinct lexical feature sequences, as determined by counting the number of rows in the factored lexicon view, a total of 33 syntactic features (not counting the special feature $C$) among the lexical feature sequences, one selectional feature ($x_0$), and two licensing features ($\{l, z\}$). The factored representation of the lexicon, made explicit in the presentation of the lexicon in Table 3.3, is motivated both by (a) performance considerations of model checking – i.e. the larger the lexicon model is made, the more the solver slows down – and (b) psychological plausibility conditions, in so far as the factored representation allows for a more efficient encoding of the lexicon as compared to the standard representation of the lexicon.[32] (This

[31] An agenda-based MG parser (Harkema, 2001) was used to verify that the recovered (inferred) lexicon can be used to parse each sentence listed in the PLD and recover the same derivation as was recovered. See Fig. 2-3 in Ch. 2.2 for further details about this parser.

[32] In the standard representation of an MG lexicon, each pairing of a phonological form and a lexical feature sequence is listed explicitly, so that the same phonological form may be listed more than once; in the factored representation, a phonological form can only be listed once.

will be covered in more detail later in this section when we evaluate and analyze the inferred lexicon.) Notably, the factored representation of the lexicon allows for a division of labor between the optimization metrics in so far as metrics (3.1) and (3.1) serve to minimize the lexical feature sequences appearing in the lexicon, while the associations between phonological forms and lexical feature sequences are minimized by metric (3.8).

Additionally, the *Summary of Derivations* (see Table 3.4) lists, for each entry in the PLD, the particular lexical feature sequences (from the recovered lexicon) that participate in the (recovered) derivation that satisfied the interface conditions listed in that entry; although this view does not show us *how* the lexical items combine together (via merge), it does show us which lexical items were involved.

### Evaluation and Analysis of the Inferred Grammar.

Let us now evaluate what knowledge of syntax the learner has acquired from the primary linguistic data via the acquisition procedure. The *optimized inferred lexicon* and the *summary of derivations*, listed in Tables 3.3 and 3.4 respectively, will be the subjects of the evaluation and analysis, which will center on evaluating the inferred lexicon with respect to its strong generative capacity – i.e. whether: (i) the lexicon yields, for each entry in the primary linguistic data, the prescribed derivation; (ii) the lexicon yields, for expressions with related interpretations, structures that are systematically related via a transformation; (iii) the lexicon has the capacity to generalize beyond experience, yielding structures distinct from the derivations the lexicon yields in parsing entries in the primary linguistic data; (iv) the lexicon avoids producing ungrammatical expressions. We will consider each of these four points in turn.

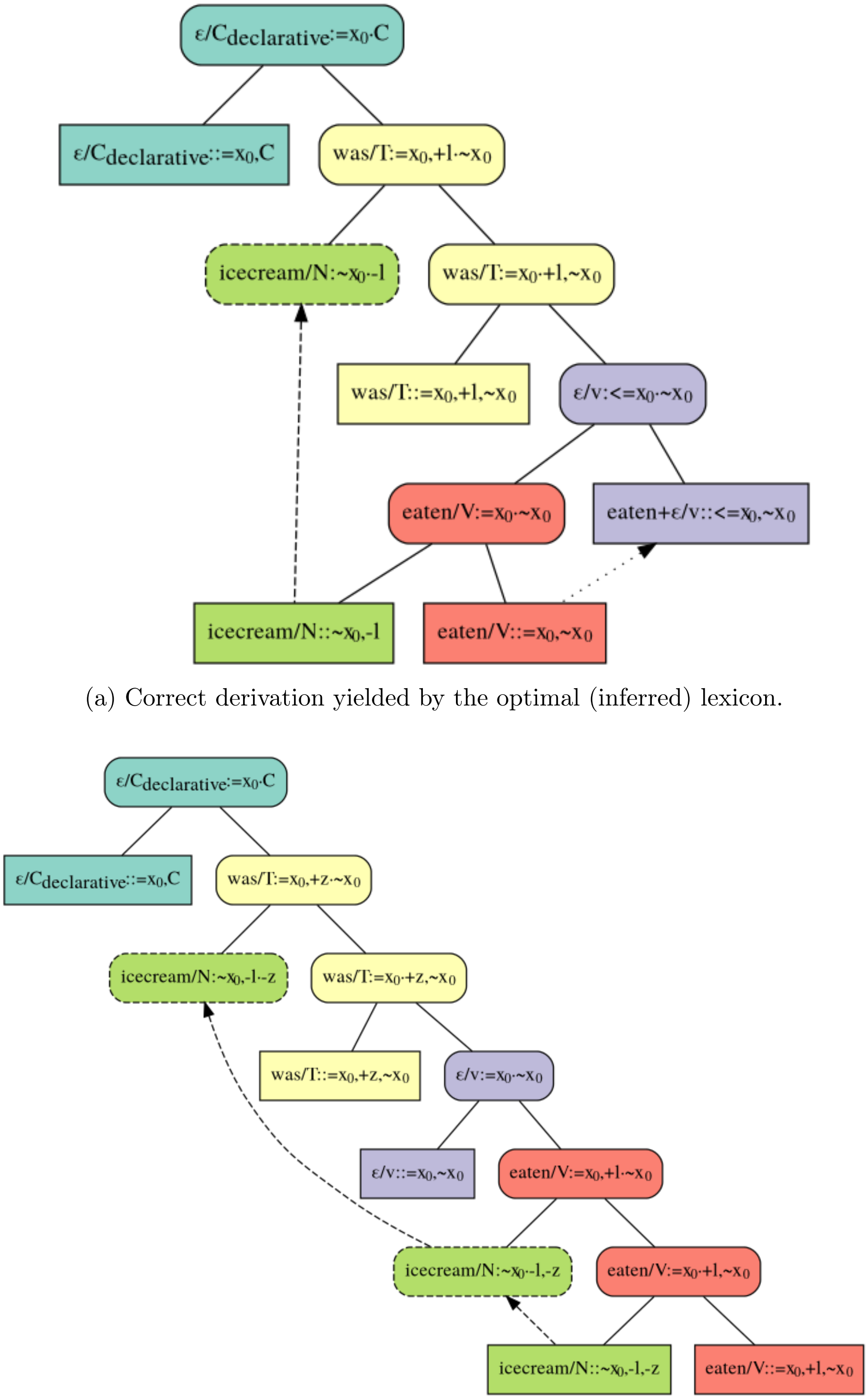
*The inferred lexicon conservatively associates* 26 *overt phonological forms with* 16 *lexical feature sequences, yielding derivations (for the entries in the PLD) that comport with contemporary theories of minimalist syntax*[33]*, employing syntactic movement to enable a phrase to satisfy multiple syntactic relations and utilizing empty (covert) lexical items for representing (little) v and C.*[34]

For example, the derivation (presented in Fig. 3-8) yielded by the (optimized) inferred lexicon to satisfy the interface conditions stipulated in entry $I_{13}$ of the PLD (listed in Table-3.2) demonstrates several of the syntactic phenomenon that are correctly modeled as prescribed by minimalist theories of syntax, including: A' movement (Wh-fronting for question formation); $V$-to-$v$ head-movement (as part of the predicate-argument structure within the derivation; see (Hale and Keyser, 2002)); $T$-to-$C$ head-movement (i.e. subj-auxiliary verb inversion; see (Pesetsky and Torrego, 2001)); and A-movement (subject raising). See Figure 3-9 for additional examples of derivations yielded by the (optimized) inferred lexicon that involve a transitive verb: 3-9a and 3-9b are declaratives; 3-9c and 3-9d are yes/no-questions; 3-9e and 3-9f are wh-questions. Of these examples, Figures 3-9a, 3-9c, and 3-9e are for active-voice expressions, and Figures 3-9b, 3-9d and 3-9f are for passive-voice expressions.

In yielding derivations that satisfied each of the 29 pairs of interface conditions listed in the PLD, the lexicon generated a total of 20 distinct syntactic structures, three of which

[33] E.g. See the presentations in (Hornstein et al., 2005), (Adger, 2003), and (Radford, 1997).

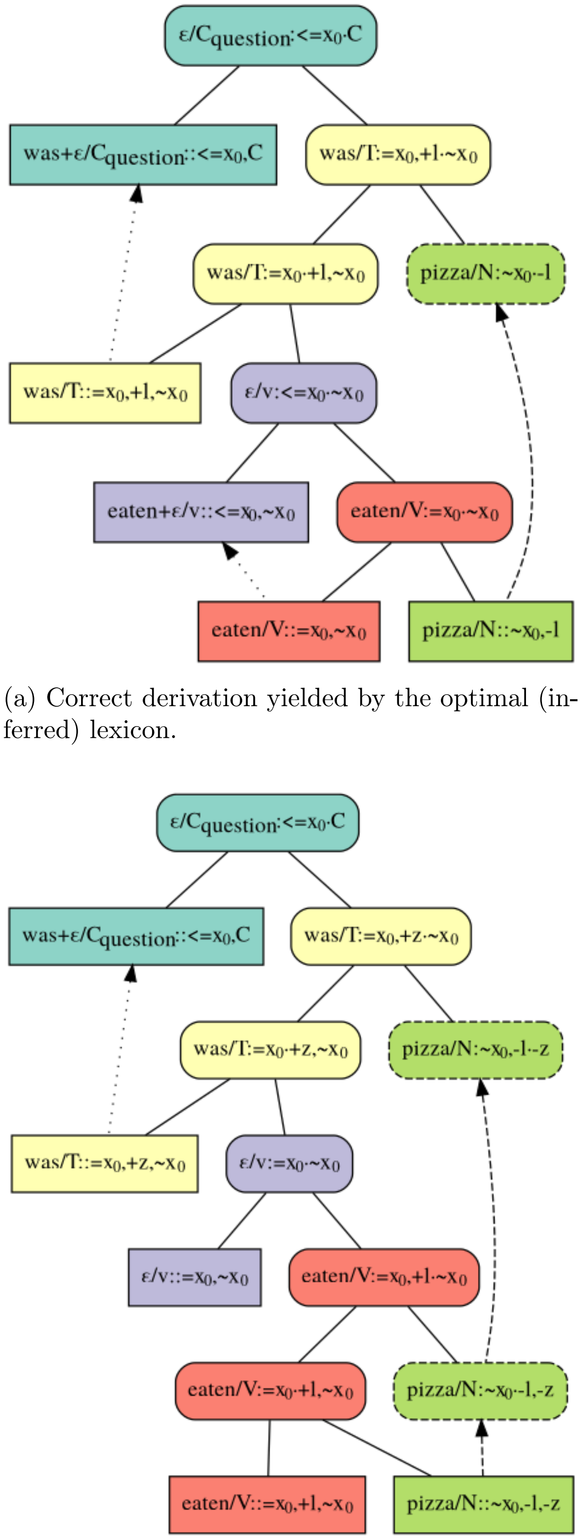
[34] Although the formulae derived from interface conditions to constrain derivation model do not distinguish between overt and covert lexical items, they do sometimes restrict the category of a constituent involved in a syntactic relation.



(a) Correct derivation yielded by the optimal (inferred) lexicon.

(b) Incorrect derivation yielded by the unoptimized (inferred) lexicon.

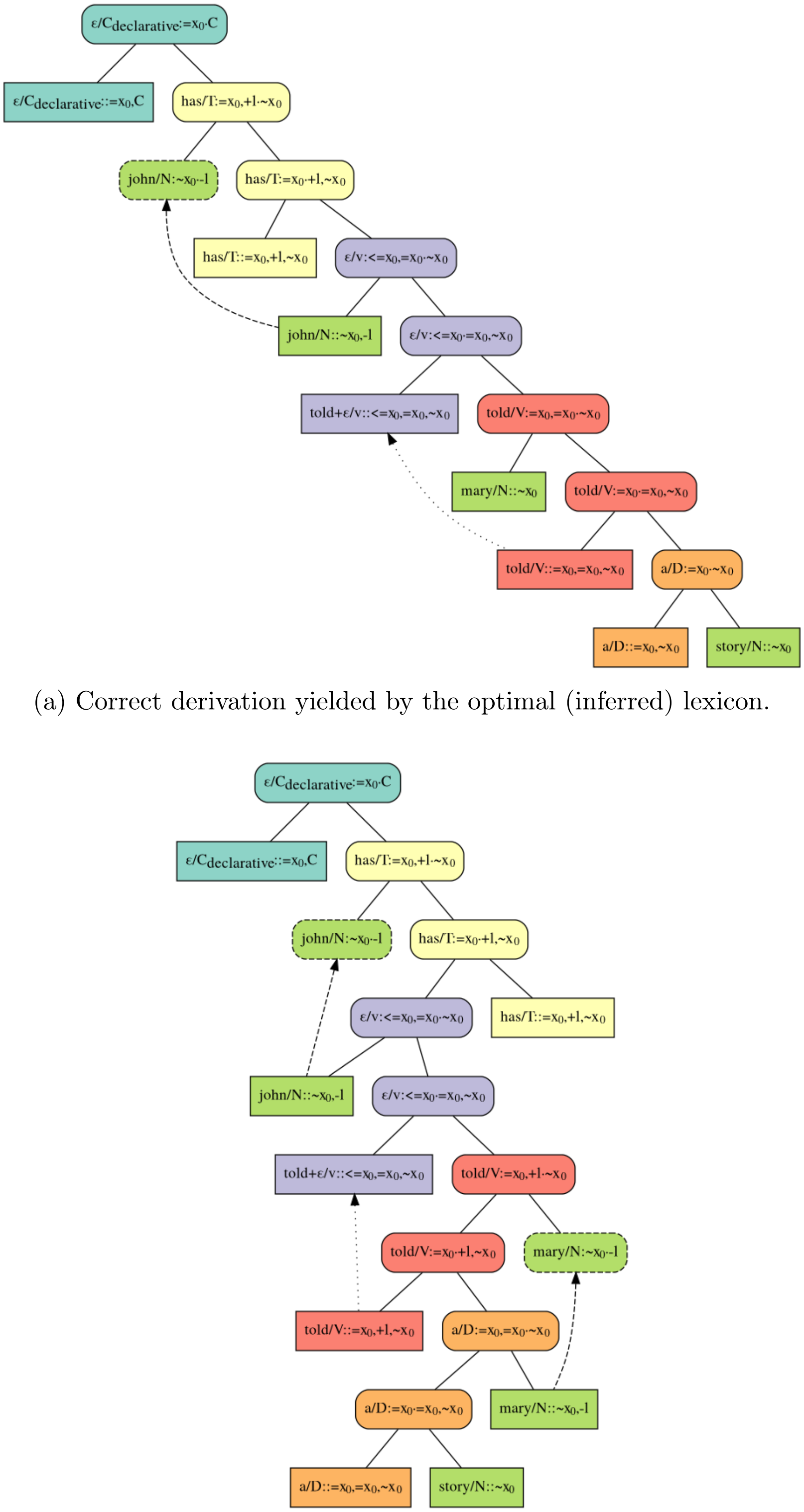
Figure 3-4: Two derivations that satisfy entry $I_1$ in the PLD (i.e. *"Icecream was eaten."*); (3-4a) and (3-4b) are yielded by the optimized and unoptimized inferred lexicons respectively. (3-4a) comports with theory whereas (3-4b) does not.



(a) Correct derivation yielded by the optimal (inferred) lexicon.

(b) Incorrect derivation yielded by the unoptimized (inferred) lexicon.

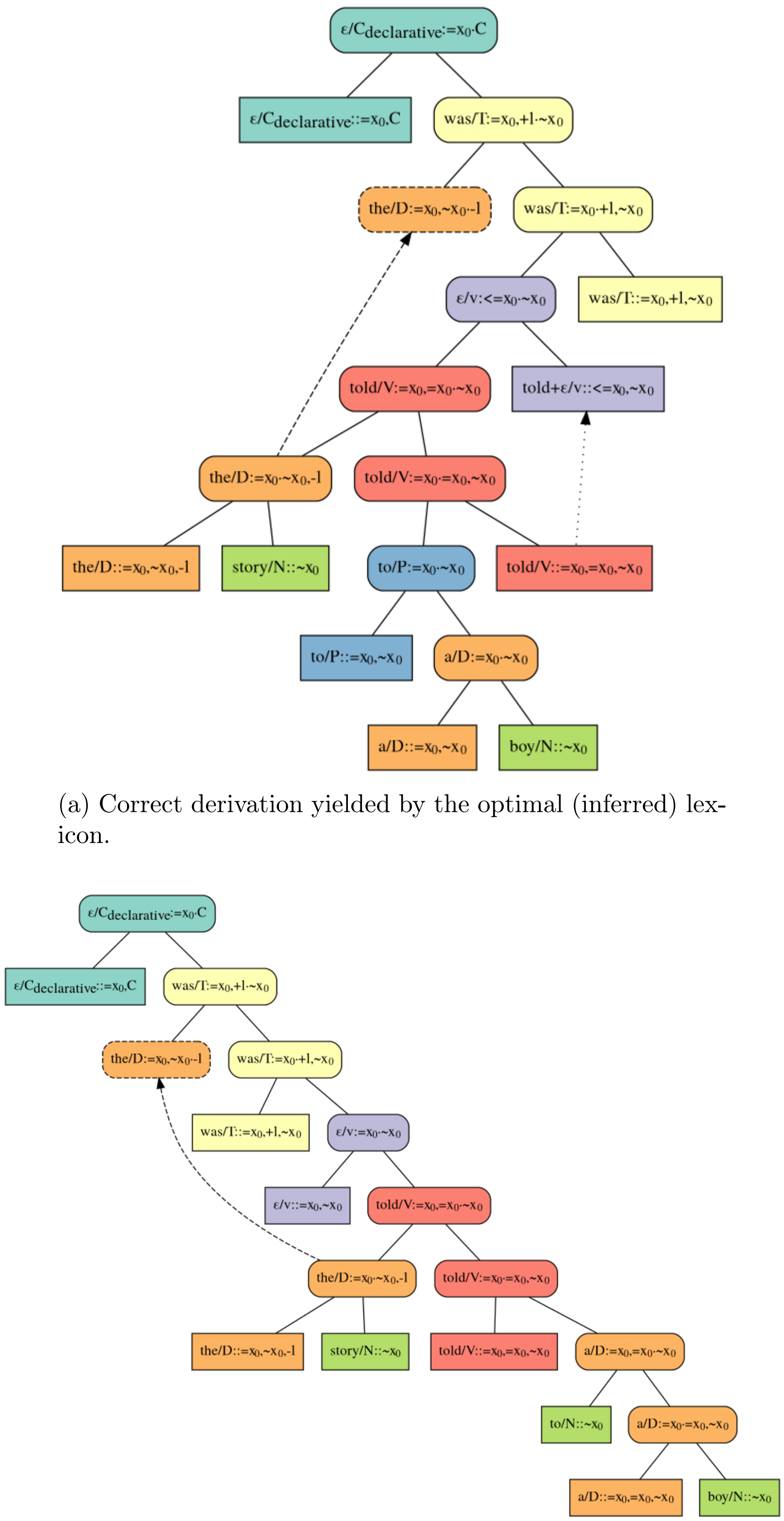
Figure 3-5: Two derivations that satisfy entry $I_3$ in the PLD (i.e. *"Was pizza eaten?"*); (3-5a) and (3-5b) are yielded by the optimized and unoptimized inferred lexicons respectively. (3-5a) comports with theory whereas (3-5b) does not.



(a) Correct derivation yielded by the optimal (inferred) lexicon.

(b) Incorrect derivation yielded by the unoptimized (inferred) lexicon.

Figure 3-6: Two derivations that satisfy entry $I_{26}$ in the PLD (i.e. *"John has told Mary a story."*); (3-6a) and (3-6b) are yielded by the optimized and unoptimized inferred lexicons respectively. (3-6a) comports with theory whereas (3-6b) does not.


$\varepsilon/C_{declarative}:=x_0{\cdot}C$
$\varepsilon/C_{declarative}::=x_0,C$
$was/T:=x_0,+l{\cdot}{\sim}x_0$
$the/D:=x_0,{\sim}x_0{\cdot}{-}l$
$was/T:=x_0{\cdot}+l,{\sim}x_0$
$\varepsilon/v:<=x_0{\cdot}{\sim}x_0$
$was/T::=x_0,+l,{\sim}x_0$
$told/V:=x_0,=x_0{\cdot}{\sim}x_0$
$told{+}\varepsilon/v::<=x_0,{\sim}x_0$
$the/D:=x_0{\cdot}{\sim}x_0,-l$
$told/V:=x_0{\cdot}=x_0,{\sim}x_0$
$the/D::=x_0,{\sim}x_0,-l$
$story/N::{\sim}x_0$
$to/P:=x_0{\cdot}{\sim}x_0$
$told/V::=x_0,=x_0,{\sim}x_0$
$to/P::=x_0,{\sim}x_0$
$a/D:=x_0{\cdot}{\sim}x_0$
$a/D::=x_0,{\sim}x_0$
$boy/N::{\sim}x_0$
$\varepsilon/C_{declarative}:=x_0{\cdot}C$
$\varepsilon/C_{declarative}::=x_0,C$
$was/T:=x_0,+l{\cdot}{\sim}x_0$
$the/D:=x_0,{\sim}x_0{\cdot}{-}l$
$was/T:=x_0{\cdot}+l,{\sim}x_0$
$was/T::=x_0,+l,{\sim}x_0$
$\varepsilon/v:=x_0{\cdot}{\sim}x_0$
$\varepsilon/v::=x_0,{\sim}x_0$
$told/V:=x_0,=x_0{\cdot}{\sim}x_0$
$the/D:=x_0{\cdot}{\sim}x_0,-l$
$told/V:=x_0{\cdot}=x_0,{\sim}x_0$
$the/D::=x_0,{\sim}x_0,-l$
$story/N::{\sim}x_0$
$told/V::=x_0,=x_0,{\sim}x_0$
$a/D:=x_0,=x_0{\cdot}{\sim}x_0$
$to/N::{\sim}x_0$
$a/D:=x_0{\cdot}=x_0,{\sim}x_0$
$a/D::=x_0,=x_0,{\sim}x_0$
$boy/N::{\sim}x_0$

(a) Correct derivation yielded by the optimal (inferred) lexicon.

(b) Incorrect derivation yielded by the unoptimized (inferred) lexicon.

Figure 3-7: Two derivations that satisfy entry $I_{27}$ in the PLD (i.e. *"The story was told to a boy."*); (3-7a) and (3-7b) are yielded by the optimized and unoptimized inferred lexicons respectively. (3-7a) comports with theory whereas (3-7b) does not.

| | |
|---:|:---|
| $\{\text{eating}, \text{eaten}, \text{asking}, \text{asked}, \text{known}\}/V$ | $:: = x_0, \sim x_0$ |
| $\{\text{given}, \text{asked}, \text{told}\}/V$ | $:: = x_0, = x_0, \sim x_0$ |
| $\epsilon/C_{declarative}$ | $:: = x_0, C$ |
| $\epsilon/C_{question}$ | $:: <= x_0, +z, C$ |
| $\epsilon/v$ | $:: <= x_0, \sim x_0$ |
| $\epsilon/C_{question}$ | $:: <= x_0, C$ |
| $\epsilon/v$ | $:: <= x_0, = x_0, \sim x_0$ |
| $\text{to}/P$ | $:: = x_0, \sim x_0$ |
| $\{\text{a}, \text{the}\}/D$ | $:: = x_0, \sim x_0, -l$ |
| $\{\text{a}, \text{the}\}/D$ | $:: = x_0, \sim x_0$ |
| $\{\text{what}, \text{who}\}/D$ | $:: \sim x_0, -z$ |
| $\{\text{what}, \text{who}\}/D$ | $:: \sim x_0, -l, -z$ |
| $\{\text{has}, \text{was}\}/T$ | $:: = x_0, +l, \sim x_0$ |
| $\{\text{sleeping}, \text{slept}\}/V$ | $:: \sim x_0$ |
| $\{\text{pizza}, \text{everything}, \text{john}, \text{mary}, \text{nothing}, \text{icecream}, \text{someone}, \text{money}\}/N$ | $:: \sim x_0, -l$ |
| $\{\text{story}, \text{pizza}, \text{everything}, \text{john}, \text{mary}, \text{boy}, \text{nothing}, \text{icecream}, \text{someone}, \text{money}\}/N$ | $:: \sim x_0$ |

Table 3.6: The *optimal* minimalist lexicon that was inferred from the PLD listed in Table 3.2. For each entry in the PLD, the derivation yielded by the lexicon both agrees with the interface conditions stipulated in that entry, and also aligns with the prescriptions of contemporary theories of minimalist syntax. Each lexical entry consists of a phonetic form paired with a lexical feature sequence (i.e. a sequence of syntactic features); a double-colon indicates that each member of the set of phonetic forms on the left hand side is paired with the lexical feature sequence on the right hand side. The phonetic form $\epsilon$ is covert (unpronounced). A feature has: (i) a value from a finite set of categories; (ii) a type, which is either *selector*, *selectee*, *licensor* or *licensee*, indicated by the prefix $=$, $\sim$, $+$ and $-$ respectively; a $<$ or $>$ prefixed before a selector prefix indicates that the selector triggers left or right head-movement respectively. There is also a special feature, $C$, that serves to indicate the completion of a parse. See Table 3.3 for a *factored view* of this lexicon.

occur thrice, three of which occur twice, and the remaining occurring just once;[35] this was accomplished using only 16 distinct lexical feature sequences. The clusters in the column labels of Table 3.4 (i.e. the *Summary of Derivations*) show that sentences with the same syntactic structures prescribed by the theory use same lexical items; thus, the system is consistent in how it interprets sentences with the same underlying syntactic structure.

To put the analysis of the optimized inferred lexicon into context, let us briefly consider a *suboptimal* lexicon, listed in Table 3.7, that was inferred without any optimization constraints applied – i.e. the lexicon that was inferred at the very end of Step 3 of the procedure (i.e. immediately after the PLD was consumed and prior to optimization).[36] The unoptimized lexicon has 20 distinct lexical feature sequences, 44 syntactic features (not counting the special symbol $C$), one selectional feature label, and two licensing feature labels. For a subset of the entries in the PLD, the optimal inferred lexicon yields derivations that accord with the theory and the unoptimized lexicon yields derivations that do not – e.g. the deriva-

[35] A lower bound of 20 was obtained by counting the number of columns in the factored input sequence view; subsequently it was manually confirmed that there are *exactly* twenty distinct syntactic structures.

[36] We can short circuit the procedure at any time to check the acquisition model, so that if something goes wrong we can isolate the bug down to a particular derivation; this is reminiscent of how in the chapter on parsing, we could check the acquisition model after adding each individual axiom, so as to identify, in the case of the solver declaring the acquisition model to be unsatisfiable, which axiom was responsible.

| | |
|---|---|
| $\epsilon/C_{Decl.}$ | $:: = x_0, C$ |
| $\epsilon/C_{Ques.}$ | $:: <= x_0, C$ |
| $\epsilon/C_{Ques.}$ | $:: <= x_0, +z, C$ |
| $\epsilon/v$ | $:: = x_0, \sim x_0$ |
| $\epsilon/v$ | $:: <= x_0, = x_0, \sim x_0$ |
| $\{\text{a, the}\}/D$ | $:: = x_0, \sim x_0, -l$ |
| $\text{who}/D$ | $:: \sim x_0, -z$ |
| $\text{a}/D$ | $:: = x_0, = x_0, \sim x_0$ |
| $\text{the}/D$ | $:: = x_0, \sim x_0$ |
| $\{\text{has, was}\}/T$ | $:: = x_0, +l, \sim x_0$ |
| $\text{was}/T$ | $:: = x_0, +z, \sim x_0$ |
| $\{\text{eaten, told}\}/V$ | $:: = x_0, +l, \sim x_0$ |
| $\{\text{given, asked, told}\}/V$ | $:: = x_0, = x_0, \sim x_0$ |
| $\{\text{sleeping, slept}\}/V$ | $:: \sim x_0$ |
| $\{\text{asking, asked, eating, eaten, known}\}/V$ | $:: = x_0, \sim x_0$ |
| $\{\text{everything, john, mary, nothing, someone, money}\}/N$ | $:: \sim x_0, -l$ |
| $\{\text{pizza, what, who, icecream}\}/N$ | $:: \sim x_0, -l, -z$ |
| $\{\text{someone, money, story, pizza, everything, john, boy, nothing, icecream, to}\}/N$ | $:: \sim x_0$ |
| $\text{what}/N$ | $:: \sim x_0, -z$ |
| $\text{to}/P$ | $:: = x_0, \sim x_0$ |

Table 3.7: A *suboptimal* minimalist lexicon that was inferred immediately after processing the PLD (listed in Table 3.2) but prior to optimization – i.e. this is the lexicon recovered from a satisfiable interpretation of the model checked in Step 3h of the *instantaneous acquisition procedure*. For each entry in the PLD, the derivation yielded by the lexicon both agrees with the interface conditions stipulated in that entry. Each lexical entry consists of a phonetic form paired with a lexical feature sequence (i.e. a sequence of syntactic features); a double-colon indicates that each member of the set of phonetic forms on the left hand side is paired with the lexical feature sequence on the right hand side. A feature has: (i) a value from a finite set of categories; (ii) a type, which is either *selector*, *selectee*, *licensor* or *licensee*, indicated by the prefix $=$, $\sim$, $+$ and $-$ respectively; a $<$ or $>$ prefixed before a selector prefix indicates that the selector triggers left or right head-movement respectively. There is also a special feature, $C$, that serves to indicate the completion of a parse.

tions that were yielded by the unoptimized lexicon for entries $I_1$ and $I_3$ (presented in Figures 3-4b and 3-5b respectively) both have the problem of an argument phrase merging with a verb first as a complement, and then (illicitly) merging again with the same verb as a specifier. The derivations that were yielded by the unoptimized lexicon for entries $I_{26}$ and $I_{27}$ (presented in Figures 3-6b and 3-7b respectively) both have the problem of a misbehaving determiner *"a"*: in the case of the derivation for $I_{26}$, one internal argument (*"Mary"*) incorrectly originates within the other internal argument (*"a story"*); in the case of the derivation for $I_{27}$, *"to"* is incorrectly associated with category $N$ and merges into the specifier position of the *"a"*. Additionally, the unoptimized lexicon yields derivations that (inappropriately) employ licensors for mixed purposes: $+z$ is used both to trigger wh-movement, but also to trigger subject-raising (necessitated by agreement) as in Figures 3-4b and 3-5b; $+l$ is used both to trigger subject raising, but also is used to drive syntactic movement that enables an internal argument to (illicitly) merge twice with a predicate as seen in Figures 3-4b, 3-5b and 3-7b. Finally, the unoptimized lexicon yields the derivations in Figures 3-4b, 3-5b and 3-7b that do not include $V$-to-$v$ head-movement. To summarize, in each of these derivations yielded by the unoptimized lexicon, the interface conditions are satisfied, yet they are suboptimal with respect to economy considerations as compared to the derivations yielded by the optimized lexicon – e.g. the unoptimized lexicon yields the derivations in Figures 3-4b, 3-5b and 3-6b that involve 6, 6, and 9 merge operations respectively, as opposed to the derivations yielded by the optimized lexicon (for those same PLD entries) that have 5, 5, and 8 merge operations respectively.[37]

Turning back to the evaluation and analysis of the *optimized* inferred lexicon, let us examine the factored view of the inferred lexicon, listed in Table 3.3. This table was seriated along both the rows and columns using the Google OR Tools library so as to (visually) group together similar rows and columns, and thereby display clusters of (syntactically) similar words; upon analyzing these clusters, it was determined that the inferred lexicon correctly associates phonological forms with lexical and functional categories, and that phonological forms can be clustered by the syntactic roles they take on (i.e. phonological forms can be clustered by lexical feature sequences):

$$\{\text{story, boy}\}\,/N\!::\{\mathfrak{L}_{16}\} \tag{3.10}$$

$$\{\text{pizza, everything, john, mary, nothing, icecream, someone, money}\}\,/N\!::\{\mathfrak{L}_{15},\mathfrak{L}_{16}\} \tag{3.11}$$

$$\{\text{a, the}\}\,/D\!::\{\mathfrak{L}_{9},\mathfrak{L}_{10}\} \tag{3.12}$$

$$\{\text{who, what}\}\,/D\!::\{\mathfrak{L}_{11},\mathfrak{L}_{12}\} \tag{3.13}$$

$$\{\text{to}\}\,/P\!::\{\mathfrak{L}_{8}\} \tag{3.14}$$

$$\{\text{has, was}\}\,/T\!::\{\mathfrak{L}_{8}\} \tag{3.15}$$

$$\{\text{sleeping, slept}\}\,/V\!::\{\mathfrak{L}_{14}\} \tag{3.16}$$

$$\{\text{eating, eaten, asking, known}\}\,/V\!::\{\mathfrak{L}_{1}\} \tag{3.17}$$

$$\{\text{asked}\}\,/V\!::\{\mathfrak{L}_{1},\mathfrak{L}_{2}\} \tag{3.18}$$

$$\{\text{told, given}\}\,/V\!::\{\mathfrak{L}_{2}\} \tag{3.19}$$

There are clusters for: nominal arguments that cannot undergo subject raising (3.10); nominal arguments that can optionally under go subject raising (via syntactic movement triggered by agreement) (3.11); determiner phrases that can optionally undergo subject raising (triggered by agreement) but does not undergo wh-raising (3.12); determiners that may first optionally undergo subject raising, and then must undergo wh-raising (hence they may undergo movement twice) (3.13); prepositions used in (non-dative-shifted) ditransitives (3.14); tense markers that require agreement (3.15); intransitive verbs (3.16), transitive verbs (3.17), ditransitive verbs (3.18), and verbs that can be either transitive or ditransitive (3.19).

Finally, the entries in the primary linguistic data may be classified using three parameters: the *valency* of the verb, the *voice* of the verb, and the type of sentence (i.e. declarative, yes/no-question or wh-question).

- The *valency* of the verb is determined by which one of three lexical feature sequences appears in the derivation: $\mathfrak{L}_{14}$ for intransitive verbs, $\mathfrak{L}_1$ for transitive verbs, and $\mathfrak{L}_2$ for ditransitive verbs. To see why, consider that $\mathfrak{L}_{14}$, $\mathfrak{L}_1$, and $\mathfrak{L}_2$ are the only lexical feature sequences that were assigned the categorical variable $V$, and that they have 0, 1 and 2 selector features (i.e. $\sim x_0$ vs. $=x_0, \sim x_0$ vs. $=x_0, =x_0, \sim x_0$) respectively; since a selector feature (i.e. $=x_0$) enables the projection of the lexical head to merge with an (internal) argument, $\mathfrak{L}_{14}$, $\mathfrak{L}_1$, and $\mathfrak{L}_2$ can thus merge with zero, one or two internal arguments respectively.

[37] Both lexicons yield derivations for $I_{27}$ that involve 9 merge operations.

- The *voice* of the verb is determined by which one of two lexical feature sequences appears in the derivation: $\mathfrak{L}_7$ for active voice and $\mathfrak{L}_5$ for passive voice. To see why, consider that $\mathfrak{L}_5$ and $\mathfrak{L}_7$ are the only two lexical feature sequences to be assigned the categorical variable $v$ (i.e. light verb), and that they have 1 and 2 selector features (i.e. $<= x_0, \sim x_0$ vs. $<= x_0, = x_0, \sim x_0$) respectively; the first selector feature, $<= x_0$, merges with the verb to form a double-VP shell (and then triggers $V$-to-$v$ head-movement), and the presence of a second selector feature (i.e. $= x_0$) enables the projection of the lexical head to merge with an (external) argument. Thus $\mathfrak{L}_5$ does not merge with an external argument whereas $\mathfrak{L}_7$ does, so that the former codes for passive-voice and the latter codes for active voice.[38]

- The *sentence type* is determined by which one of three lexical feature sequences appears in the derivation: $\mathfrak{L}_3$ for a declarative, $\mathfrak{L}_6$ for a yes/no question and $\mathfrak{L}_4$ for a wh-question. To see why, consider that: (i) $\mathfrak{L}_3$ is the only lexical feature sequence associated with the category $C_{Decl.}$; (ii) $\mathfrak{L}_6$ and $\mathfrak{L}_4$ are the only lexical feature sequences associated (appropriately) with the category $C_{Ques.}$ and although they both have a selector feature, $<= x_0$, that merges with a TP and then triggers $T$-to-$C$ head-movement (modeling subject aux. verb inversion as traditionally required by both yes/no-questions and wh-questions), only $\mathfrak{L}_4$ has a licensor feature $+z$ that serves to trigger wh-raising.

Every derivation (yielded by the inferred lexicon so as to satisfy the entries listed in the PLD) has exactly one member from each of these three groups of lexical feature sequences, thereby enabling the classification of the entries in the PLD with respect to these three parameters by way of checking for the presence of particular lexical feature sequences as described above. Table 3.8 presents a summary of this classification of the PLD; notably, some of the entries in the table are empty – i.e. there are no expressions in the PLD that are active-voiced wh-questions with either intransitive or ditransitive verbs. We will soon see that the lexicon can yield structures that would fill those empty entries; to do so, we will first examine how the structures in Table 3.8 are related via transformations.

*Expressions with related interpretations have systematically related structures.* Plainly, this means that derivations for sentences with related meanings (e.g. a declarative and the yes/no-question that may be formed from it) should be related to one another via some procedure that transforms the derivation for one into the derivation for the other in a consistent, predictable and productive manner (e.g. aux-raising, or in the parlance of minimalist theories of syntax, T-to-C head-raising). Let us see how the inferred lexicon, and the particular derivations it yielded, embody this notion in a formal setting.

Abstracting away the particular phonological forms associated with particular lexical feature sequences, a structure (i.e. a derivation) is coded for by the multiset of lexical feature sequences that participate in it.[39] Then structures assigned to expressions with related interpretations are systematically related by way of *transformations* that map one structure to another via the addition and subtraction of particular lexical feature sequences

[38] Here it is assumed that by the VP Internal Subject Hypothesis, the external argument (that is present in active-voiced constructions but not in passive-voiced constructions) is to be found in the specifier position of the light-verb.

[39] I.e. if two derivations have the same structure but different phonological forms, we will still consider them the same for the purposes of this present discussion.

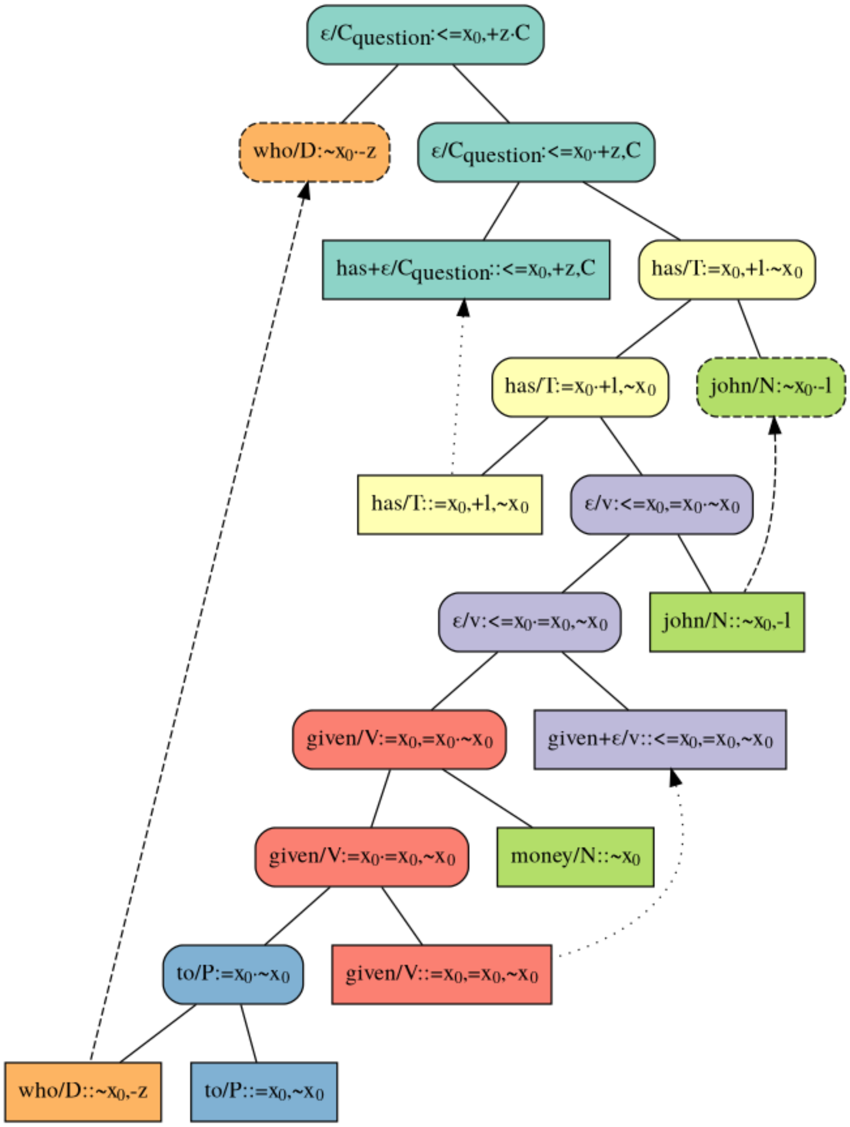


Figure 3-8: A minimalist derivation yielded by the inferred lexicon (listed in Table-3.3) that satisfies the interface conditions stipulated in entry $I_{13}$ of Table-3.2 – i.e. *"Who has John given money to?"* This derivation accords with the derivation prescribed by contemporary theories of syntax. The feature sequences displayed in non-leaf nodes have a dot, $\cdot$ , separating features that have already been consumed (on the left) from those that have not (on the right). The dashed arrows denote phrasal movement. The dotted arrows denote head movement. Nodes with the same *head* have the color. The parse is assembled in a bottom-up manner via merge: "given merges with "to who" (formed by first merging "to" and "who") and then with "money", thus establishing (local) syntactic relations encoding predicate-argument structure; the resulting structure merges with an empty lexical node with category $v$, undergoing $V$-to-$v$ head-movement before merging with the argument "john" in accordance with the VP-Internal Subject Hypothesis (Hale and Keyser, 2002); the resulting structure then merges with the auxiliary verb "has", after which the argument "john" undergoes subject-raising from the VP-shell by (internally) merging with "has", thus establishing morphological agreement between "john" and "has"; next, the head of "has" undergoes $T$-to-$C$ head-movement to merge with the covert complementizer, $\epsilon/C_{question}$, which indicates that the sentence is an interrogative; finally, "who" undergoes wh-fronting by (internally) merging with $\epsilon/C_{question}$. Wh-fronting (of "who") and Subject-raising (of "John"), instances of A'-movement and A-movement respectively, are triggered by different licensor features (pursuant to the Shortest Movement Condition), the former by $+z$ and the latter by $+l$.

| Voice | Valency | Sentence Type | Entries |
|---|---|---|---|
| active | intransitive | declarative | $\{I_{15}\}$ |
| active | intransitive | yes/no-question | $\{I_{14}\}$ |
| active | intransitive | wh-question | $\{\}$ |
| active | transitive | declarative | $\{I_{17}, I_{25}\}$ |
| active | transitive | yes/no-question | $\{I_5, I_6\}$ |
| active | transitive | wh-question | $\{I_0, I_2, I_4, I_7, I_{18}, I_{28}\}$ |
| active | ditransitive | declarative | $\{I_{11}, I_{22}, I_{26}\}$ |
| active | ditransitive | yes/no-question | $\{\}$ |
| active | ditransitive | wh-question | $\{I_{10}, I_{13}, I_{24}\}$ |
| passive | intransitive | declarative | N/A |
| passive | intransitive | yes/no-question | N/A |
| passive | intransitive | wh-question | N/A |
| passive | transitive | declarative | $\{I_{19}, I_{20}, I_1\}$ |
| passive | transitive | yes/no-question | $\{I_3\}$ |
| passive | transitive | wh-question | $\{I_8\}$ |
| passive | ditransitive | declarative | $\{I_{16}, I_{27}\}$ |
| passive | ditransitive | yes/no-question | $\{I_9\}$ |
| passive | ditransitive | wh-question | $\{I_{12}, I_{21}, I_{23}\}$ |

Table 3.8: Classification of the entries in the primary linguistic data.

– e.g. the derivations for a declarative and a yes/no-question are systematically related by a transformation that raises the auxiliary verb. Furthermore, transformations must adhere to the following conditions:

1. A transformation can be applied to a structure only if: (a) the transformation does not require the deletion (subtraction) of a lexical feature sequence from the structure that is not is present in that structure to begin with; (b) it must be possible to derive a *complete* derivation from the multiset of lexical feature sequences that is the output of the transformation, with no lexical feature sequences not participating in said derivation.

2. Two transformations may be composed together (i.e. as successive application) to form a single transformation by summing the additions and subtractions of lexical feature sequences in the two transformations – e.g. the transformation for forming a yes/no-question from a declarative may be composed with the transformation for forming a wh-question from a yes/no-question to yield a transformation that forms a wh-question directly from a declarative.

We will now identify several examples of transformations that hold between structures for related entries in the primary linguistic data – in particular, we will identify transformations for forming yes/no-questions from declarative expressions, for forming wh-questions from yes/no-questions, and for passivization. *Note that in the following analysis of transformations, a (syntactic) structure will be referenced by the PLD entry it satisfies – e.g. $I_{15}$ will be used to reference the derivation yielded by the inferred lexicon to satisfy entry $I_{15}$ in the PLD; this notational convention is a convenience that helps in comparing the differences between (syntactic) structures by comparing the columns in the Summary of Derivations listed in Table 3.4.*

- **Active Voice: Declarative → (Yes/No)-Question → Wh-Question.** Let us first consider structural transformations for forming a yes/no-question from a declarative via aux. raising and for forming a wh-question from yes/no-question via wh-raising. We will consider examples involving intransitives, transitives and ditransitives verbs, all in the active voice.

  1. $(I_{15}, I_{14})$. $I_{15}$ is a active-voice declarative with an intransitive verb. $I_{14}$ is a active-voice yes/no question with an intransitive verb. The transformation $[I_{15} \to I_{14}]$ is accomplished by replacing $\mathfrak{L}_3$ with $\mathfrak{L}_6$; to the see this relation via transformation, note that we can substitute (in $I_{14}$) *"was"* for *"has"* and *"sleeping"* for *"slept"* as they are in the same PF clusters, so that we can then go from "has the boy slept?" to "the boy has slept" by replacing $\mathfrak{L}_3$ with $\mathfrak{L}_6$.
  2. $(I_{17}, I_5, I_2 or I_4)$. $I_{17}$ is an active-voice declarative with a transitive verb. $I_5$ is an active-voice yes/no-question with a transitive verb. $I_5$ may be formed from $I_{17}$ by replacing $\mathfrak{L}_3$ with $\mathfrak{L}_6$. Two wh-questions may be formed from $I_5$ via the application of (wh-raising) transformations that raise the external and internal argument respectively: (i) the transformation $[I_5 \to I_2]$ in which the wh-question $I_2$ is formed from $I_5$ by replacing $\mathfrak{L}_6$ with $\mathfrak{L}_4$, and replacing $\mathfrak{L}_{15}$ with $\mathfrak{L}_{12}$; (ii) the transformation $[I_5 \to I_4]$ in which the wh-question $I_4$ is formed from $I_5$ by replacing $\mathfrak{L}_6$ with $\mathfrak{L}_4$, and replacing $\mathfrak{L}_{16}$ with $\mathfrak{L}_{11}$.
  3. $(I_5, I_0)$. $I_5$ is an active-voice yes/no-question with a transitive verb. $I_0$ is an active-voice wh-question with a transitive verb. The transformation $[I_5 \to I_0]$ is accomplished by replacing $\mathfrak{L}_6$ with $\mathfrak{L}_4$, so as to move from a yes/no question to a wh-question, and replacing $\mathfrak{L}_{15}$ with $\mathfrak{L}_{12}$, so as to replace the nominal argument that was raised to structural subject position with a wh-word that first undergoes subject raising before undergoing wh-raising.
  4. $(I_{17}, I_0)$. $I_{17}$ is an active-voice declarative with a transitive verb. $I_0$ is an active-voice wh-question with a transitive verb. The transformation $[I_{17} \to I_0]$ is accomplished by replacing $\mathfrak{L}_3$ with $\mathfrak{L}_4$, so as to move from a declarative to a wh-question, and replacing $\mathfrak{L}_{15}$ with $\mathfrak{L}_{12}$, so as to replace the nominal argument that underwent subject raising with a wh-word that undergoes subject raising before wh-raising.
  5. $(I_6, I_7)$. $I_6$ is a yes/no-question with a transitive verb. $I_7$ is a wh-question with a transitive verb. The transformation $[I_6 \to I_7]$ is accomplished by replacing $L_6$ with $L_4$, and replacing $\mathfrak{L}_{16}$ with $\mathfrak{L}_{11}$, the latter having the effect of replacing the internal argument that serves as the direct object with a wh-word.
  6. $(I_{11}, I_{10})$. $I_{11}$ is a declarative with a ditransitive verb. $I_{10}$ is a wh-question with a ditransitive verb. The transformation $[I_{11} \to I_{10}]$ is accomplished by replacing $L_3$ with $L_4$ and replacing $\mathfrak{L}_{16}$ with $\mathfrak{L}_{11}$, so as to substitute a wh-word for the internal argument serving as the direct object.
  7. $(I_{26}, I_{24})$. $I_{26}$ is a declarative with an (active) ditransitive verb. $I_{24}$ is a wh-question formed from $I_{26}$ by raising the external argument. The transformation $[I_{26} \to I_{24}]$ is accomplished by replacing $L_3$ with $L_4$ and replacing $\mathfrak{L}_{12}$ with $\mathfrak{L}_{15}$.

  The transformation for aux-raising is to subtract $\mathfrak{L}_6$ and add $\mathfrak{L}_3$ – denote this $(\mathfrak{L}_6 - \mathfrak{L}_3)$. The two transformations observed for wh-raising from a yes/no-question are $(\mathfrak{L}_4 - \mathfrak{L}_6 + \mathfrak{L}_{15} - \mathfrak{L}_{12})$ and $(\mathfrak{L}_4 - \mathfrak{L}_6 + \mathfrak{L}_{11} - \mathfrak{L}_{16})$ respectively; the applicability of these wh-raising transformations depends on which argument is to be the raised wh-word. Finally, the transformations for aux-raising and wh-question formation may be composed to produce the transformation seen in directly going from a declarative to a wh-question in the case of $[I_{17} \to I_0]$ – e.g.:

$$\begin{aligned} [I_{17} \to I_0] &= (\mathfrak{L}_4 - \mathfrak{L}_3 + \mathfrak{L}_{12} - \mathfrak{L}_{15}) && (3.20) \\ &= (\mathfrak{L}_6 - \mathfrak{L}_3) + (\mathfrak{L}_4 - \mathfrak{L}_6 + \mathfrak{L}_{12} - \mathfrak{L}_{15}) && (3.21) \\ &= [I_5 \to I_0] \circ [I_{17} \to I_5] && (3.22) \end{aligned}$$

where the transformations $[I_{17} \to I_5]$ and $[I_5 \to I_4]$ are for aux-raising and wh-raising (of the external argument) respectively.

- **Passive Voice: Declarative → (Yes/No)-Question → Wh-Question.** We will next demonstrate that the transformations applied to active-voice constructions may also be applied to passive-voice constructions. The following examples involving transitives and ditransitives are all in the passive-voice:[40]

    1. $(I_1, I_3, I_8)$. $I_1$ is a passive-voice declarative with a transitive verb, $I_3$ is a passive-voice yes/no question with a transitive verb, and $I_8$ is a passive-voice wh-question with a transitive verb. The transformation $[I_1 \to I_3]$ is accomplished by replacing $L_3$ with $L_6$; $[I_3 \to I_8]$ is accomplished by replacing $L_6$ with $L_4$ and $\mathfrak{L}_{12}$ with $\mathfrak{L}_{15}$.
    2. $(I_{16}, I_9, I_{21} or I_{23})$. $I_{16}$ is a passive-voice declarative with a ditransitive verb, and $I_9$ is a passive-voice yes/no with a ditransitive verb. The transformation $[I_{16} \to I_9]$ is accomplished by replacing $\mathfrak{L}_3$ with $\mathfrak{L}_6$. Two wh-questions may be formed from $I_9$ (by raising each of the two internal arguments): the wh-question $I_{21}$ is formed from $I_9$ by replacing $\mathfrak{L}_6$ with $\mathfrak{L}_4$, adding $\mathfrak{L}_8$ so as to include *"to"*[41], and replacing $\mathfrak{L}_{16}$ with $\mathfrak{L}_{11}$; the wh-question $I_{23}$ is formed from $I_{16}$ by replacing $\mathfrak{L}_3$ with $\mathfrak{L}_4$, and $\mathfrak{L}_{16}$ with $\mathfrak{L}_{11}$.

  The transformations for aux. raising and wh-raising identified in the analysis of active-voice expressions apply in the case of passive-voice expressions as well.

- **Passivization (i.e. Active Voice → Passive Voice).**
  Finally, let us consider how passive-voiced expressions are formed from their active-voiced brethren, both in the case of transitive and ditransitive verbs.

    1. $(I_{17}, I_1)$. $I_{17}$ is an active-voice declarative with a transitive verb, and $I_1$ is a passive-voice declarative with a transitive verb. The transformation $[I_{17} \to I_1]$ is accomplished by: (i) replacing $L_7$ with $L_5$, so as to remove the external argument slot in the VP-shell structure; (ii) removing $\mathfrak{L}_{16}$, so as to remove the nominal that didn't undergo subject raising.
    2. $(I_5, I_3)$. $I_5$ is an active-voice yes/no question with a transitive verb, and $I_3$ is a passive-voice yes/no question with a passive transitive verb. The (passivization)

[40] We will not consider intransitives here as intransitives have no internal argument, only an external argument, whereas passives require an internal argument and cannot have an external argument (per the VP Internal Subject Hypothesis).

[41] To see the role played by $L_8$, consider the transformation $[I_{10} \to I_{13}]$. $I_{10}$ is a question formed by wh-raising of internal argument of ditransitive, and $I_{13}$ is a question formed by wh-raising of internal argument of alternating form of ditransitive; the only addition to $I_{10}$ to form $I_{13}$ is that of $\mathfrak{L}_8$ for introducing the prepositional phrase – i.e. the minimal change that could be.

transformation $[I_5 \to I_3]$ is accomplished by replacing $L_7$ with $L_5$ and deleting $\mathfrak{L}_{16}$.

3. $(I_0, I_8)$. $I_0$ is an active-voice wh-question with a transitive verb, and $I_8$ is a passive-voice wh-question with a transitive verb. The (passivization) transformation $[I_0 \to I_8]$ is accomplished by replacing $L_7$ with $L_5$ and deleting $\mathfrak{L}_{16}$; the wh-word that was serving as the structural subject (prior to undergoing wh-raising) can no longer originate in the external argument position in the VP-shell structure (as $\mathfrak{L}_5$ is a little-v that cannot accept an external subject) and will thus instead originate in the internal argument position.

The transformation for passivization is the same in the case of declaratives and yes/no-questions for transitive verbs, and is the same for declaratives for both transitive and ditransitive verbs; in the case of passivization of a wh-question, the details of the transformation depend on which of the arguments undergoes subject raising, which undergoes wh-raising, and which of the two internal arguments is to become the new structural subject.

Having identified transformations that form yes/no-questions from declaratives, wh-questions from yes/no-questions, and passive-voiced expressions from active-voiced expressions, let us consider three circumstances in which the composition of these transformations is commutative:

(i) passivization of a declarative followed by aux-raising has the same effect as aux-raising followed by passivization;

(ii) passivization of a yes/no question followed by wh-raising of an internal argument has the same effect as wh-raising of an internal argument followed by passivization;

(iii) transforming a declarative into a yes/no-question, which in turn is then transformed into a wh-question (via raising of an internal argument), before undergoing passivization, has the same effect as first undergoing passivization, and then forming first a yes/no-question and then a wh-question.

These three scenarios are demonstrated by the following commutative diagram, in which each arrow is a transformation, with the horizontal arrows, left to right, encoding the transformations for aux-raising (to form a yes/no-question) and wh-raising (to form a wh-question) respectively, and the vertical arrow encoding a transformation for passivization (see Fig. 3-9 for the associated derived trees)).

$$\begin{tikzcd}
I_{17} \arrow[r, "(\mathfrak{L}_6 - \mathfrak{L}_3)"] \arrow[d, "(\mathfrak{L}_5 - \mathfrak{L}_7 - \mathfrak{L}_{16})"] & I_5 \arrow[r, "(\mathfrak{L}_4 - \mathfrak{L}_6 + \mathfrak{L}_{15} - \mathfrak{L}_{12})"] \arrow[d, "(\mathfrak{L}_5 - \mathfrak{L}_7 - \mathfrak{L}_{16})"] & I_0 \arrow[d, "(\mathfrak{L}_5 - \mathfrak{L}_7 - \mathfrak{L}_{16})"] \\
I_1 \arrow[r, "(\mathfrak{L}_6 - \mathfrak{L}_3)"] & I_3 \arrow[r, "(\mathfrak{L}_4 - \mathfrak{L}_6 + \mathfrak{L}_{15} - \mathfrak{L}_{12})"] & I_8
\end{tikzcd}$$

The commutative diagram of transformations, in conjunction with the transformations being invariant with respect to the particular phonological forms involved in a structure, enables the productive application of transformations to structures that the transformation in question has thus far not been applied to; importantly, these transformations (and compositions thereof) may be used to generate novel structures (i.e. structures that were not yielded in the course of processing the primary linguistic data), as we will see next.

*The lexicon can productively generalize to form novel sentences and novel syntactic structures.* Plainly speaking, this means that the inferred lexicon can (i) parse expressions that do not appear the PLD, and (ii) yield derivations that are not associated with entries in the PLD. We will provide examples that demonstrates the inferred lexicon's capacity to produce novel sentences and novel structures; notably, some of these examples can be systematically generated via the application of the earlier identified transformations. *Note that these derivations are presented inline in a bracketed form that only shows the external merge operations; the full derivation can be recovered from a specification of the external merge operations in the derivation because, in the minimalist grammar formalism, internal merge operations happen automatically and deterministically in the course of a derivation – see §2.2 for a discussion of how this is a consequence of the Shortest Movement Condition (SMC).*

Looking at Table 3.8, we see that two of the rows are absent of entries: (i) no active-voice wh-question with an intransitive verb appears in the PLD; (ii) no active-voice yes/no-question with a ditransitive verb appears in the PLD. The first case can be ameliorated by observing that the inferred lexicon can parse the expression *"Who was sleeping?"* and yield the following derivation:

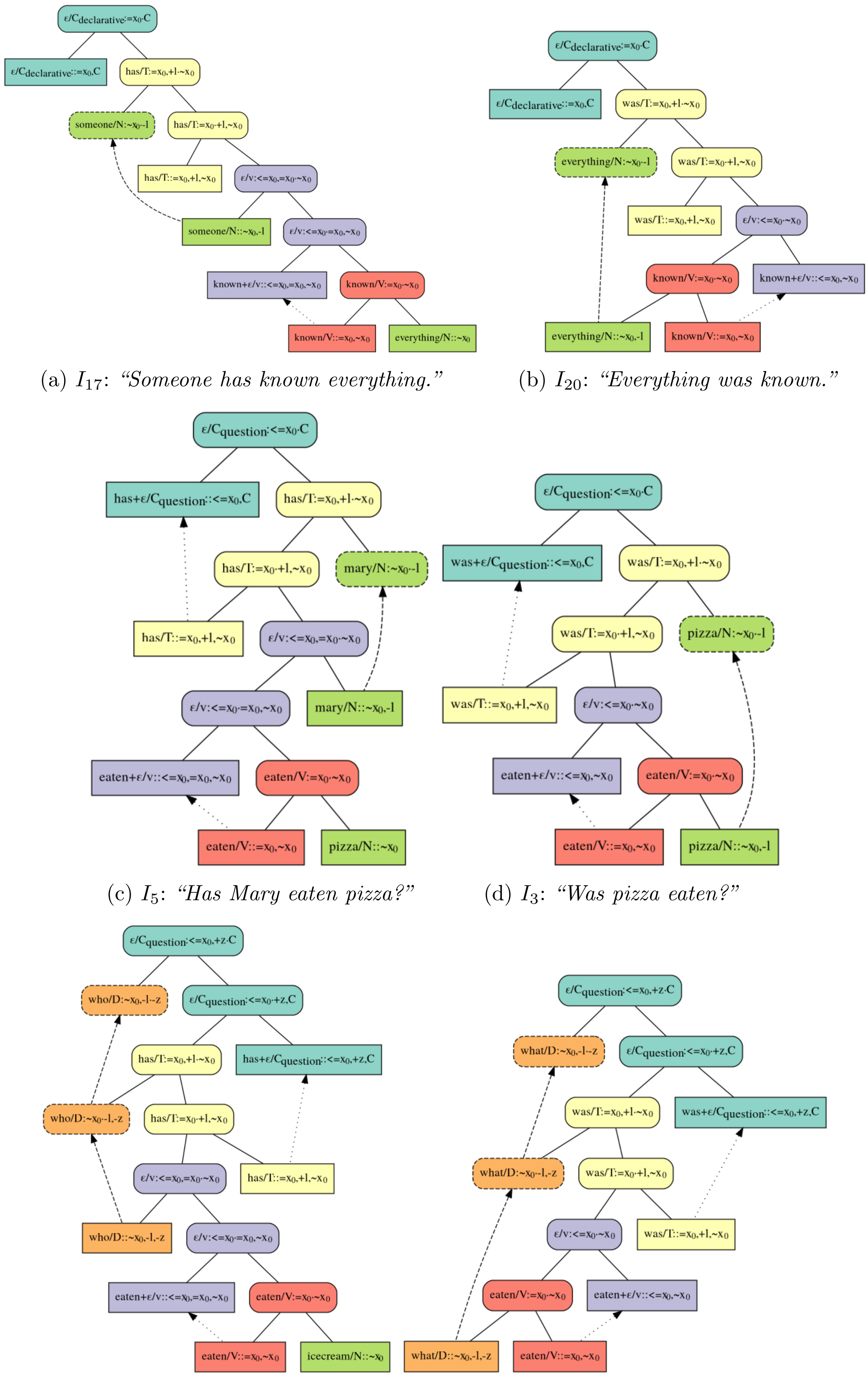
$$\left\{\frac{\epsilon_{C_{Ques.}}}{\mathfrak{L}_4}\left\{\frac{\mathit{was}}{\mathfrak{L}_{13}}\left\{\left\{\frac{\mathit{who}}{\mathfrak{L}_{12}}\right\}\left\{\frac{\epsilon_v}{\mathfrak{L}_7}\left\{\frac{\mathit{sleeping}}{\mathfrak{L}_{14}}\right\}\right\}\right\}\right\}\right\}$$

In this derivation, *"who"* is a wh-word that serves as an external argument to the intransitive verb *"sleeping"*, is then raised to merge with the tense marker *"was"*, thereby serving as the structural subject, and finally undergoes wh-raising so as to form a wh-question. This structure can be produced by consecutively applying two transformations to $I_{14}$ (*"Has the boy slept?"*): firstly, substitute the determiner phrase *"the boy"* with the nominal phrase *"john"* by replacing $\mathfrak{L}_9$ and $\mathfrak{L}_{16}$ with $\mathfrak{L}_{12}$; secondly, apply the transformation for wh-raising by replacing $\mathfrak{L}_6$ with $\mathfrak{L}_4$. The second case can be rectified by observing that the inferred lexicon can parse the (yes/no) question *"Has Mary given John money?"* and yield the following derivation:

$$\left\{\frac{\epsilon_{C_{Ques.}}}{\mathfrak{L}_6}\left\{\frac{\mathit{has}}{\mathfrak{L}_{13}}\left\{\left\{\frac{\mathit{mary}}{\mathfrak{L}_{15}}\right\}\left\{\frac{\epsilon_v}{\mathfrak{L}_7}\left\{\frac{\mathit{john}}{\mathfrak{L}_{16}}\left\{\frac{\mathit{given}}{\mathfrak{L}_2}\ \frac{\mathit{money}}{\mathfrak{L}_{15}}\right\}\right\}\right\}\right\}\right\}\right\}$$

In this derivation, *"given"* is a ditransitive verb and *"Has"* is a tense-marker that undergoes aux. raising via T-to-C head-movement. Observe that this structure can be produced by applying the transformation for forming a yes/no-question from a declarative – i.e. replacing $\mathfrak{L}_3$ with $\mathfrak{L}_6$ – to the structure associated with the entry $I_{26}$ (*"Mary has given John money."*). The inferred lexicon thus has the capacity to yield at least one derivation for each classification in Table 3.8.

The inferred lexicon can also yield a number of novel structures in which determiner phrases undergo (syntactic) movement in ways not seen in processing the derivations that the lexicon yielded to satisfy the interface conditions listed in the PLD. Let us consider examples involving the transitive verb *"eaten"*:



(a) $I_{17}$: *“Someone has known everything.”*

(b) $I_{20}$: *“Everything was known.”*

(c) $I_5$: *“Has Mary eaten pizza?”*

(d) $I_3$: *“Was pizza eaten?”*

(e) $I_0$: *“Who has eaten icecream?”*

(f) $I_8$: *“What was eaten?”*

Figure 3-9: Derivations yielded by the (inferred) lexicon listed in Table 3.3.

1. *"Has the boy eaten the pizza?"*

$$\left\{\frac{\epsilon_{C_{Ques.}}}{\mathfrak{L}_6}\left\{\frac{\textit{has}}{\mathfrak{L}_{13}}\left\{\left\{\frac{\textit{the}}{\mathfrak{L}_9}\ \frac{\textit{boy}}{\mathfrak{L}_{16}}\right\}\left\{\frac{\epsilon_v}{\mathfrak{L}_7}\left\{\frac{\textit{eaten}}{\mathfrak{L}_1}\left\{\frac{\textit{the}}{\mathfrak{L}_{10}}\ \frac{\textit{pizza}}{\mathfrak{L}_{16}}\right\}\right\}\right\}\right\}\right\}\right\}$$

2. *"Was the pizza eaten?"*

$$\left\{\frac{\epsilon_{C_{Ques.}}}{\mathfrak{L}_6}\left\{\frac{\textit{was}}{\mathfrak{L}_{13}}\left\{\frac{\epsilon_v}{\mathfrak{L}_5}\left\{\frac{\textit{eaten}}{\mathfrak{L}_1}\left\{\frac{\textit{the}}{\mathfrak{L}_9}\frac{\textit{pizza}}{\mathfrak{L}_{16}}\right\}\right\}\right\}\right\}\right\}$$

3. *"What has the boy eaten?"*

$$\left\{\frac{\epsilon_{C_{Ques.}}}{\mathfrak{L}_4}\left\{\frac{\textit{has}}{\mathfrak{L}_{13}}\left\{\left\{\frac{\textit{the}}{\mathfrak{L}_9}\ \frac{\textit{boy}}{\mathfrak{L}_{16}}\right\}\left\{\frac{\epsilon_v}{\mathfrak{L}_7}\left\{\frac{\textit{eaten}}{\mathfrak{L}_1}\ \frac{\textit{what}}{\mathfrak{L}_{11}}\right\}\right\}\right\}\right\}\right\}$$

These three expressions and their associated derivations are related as follows: (1) is an active-voice yes/no-question with a determiner phrase (*"the boy"*) that originates as the external argument in the double-VP shell structure (per the VP Internal Subject Hypothesis) before being raised to structural subject position and agreeing with *"has"*; (2) is a passivization of (1), with the determiner phrase (*"the pizza"*) raised from the internal-argument position in the VP-shell to become the (structural) subject of the expression; (3) is a wh-question formed from (1) by replacing the determiner phrase (*"the pizza"*) with a wh-word (*"what"*) that originates in the internal-argument position of the VP shell and undergoes wh-raising.

Having considered novel structures involving determiner phrases that are arguments of a transitive verb, let us consider the case for ditransitives. Observe that there were only two entries in the PLD with two or more determiner phrases – $I_{25}$ and $I_{27}$ – and that both of these were declaratives. The inferred lexicon can generate yes/no-questions and wh-questions that involve ditransitive predicates with at least two determiner phrases serving as arguments:

1. *"Has the boy told the story to the boy?"*

$$\left\{\frac{\epsilon_{C_{Ques.}}}{\mathfrak{L}_6}\left\{\frac{\textit{has}}{\mathfrak{L}_{13}}\left\{\left\{\frac{\textit{the}}{\mathfrak{L}_9}\ \frac{\textit{boy}}{\mathfrak{L}_{16}}\right\}\left\{\frac{\epsilon_v}{\mathfrak{L}_7}\left\{\left\{\frac{\textit{the}}{\mathfrak{L}_{10}}\ \frac{\textit{story}}{\mathfrak{L}_{16}}\right\}\left\{\frac{\textit{told}}{\mathfrak{L}_2}\left\{\frac{\textit{to}}{\mathfrak{L}_8}\left\{\frac{\textit{the}}{\mathfrak{L}_{10}}\ \frac{\textit{boy}}{\mathfrak{L}_{16}}\right\}\right\}\right\}\right\}\right\}\right\}\right\}\right\}$$

2. *"Who has told the story to the boy?"*

$$\left\{\frac{\epsilon_{C_{Ques.}}}{\mathfrak{L}_4}\left\{\frac{\textit{has}}{\mathfrak{L}_{13}}\left\{\frac{\textit{who}}{\mathfrak{L}_{12}}\left\{\frac{\epsilon_v}{\mathfrak{L}_7}\left\{\left\{\frac{\textit{the}}{\mathfrak{L}_{10}}\ \frac{\textit{story}}{\mathfrak{L}_{16}}\right\}\left\{\frac{\textit{told}}{\mathfrak{L}_2}\left\{\frac{\textit{to}}{\mathfrak{L}_8}\left\{\frac{\textit{the}}{\mathfrak{L}_{10}}\ \frac{\textit{boy}}{\mathfrak{L}_{16}}\right\}\right\}\right\}\right\}\right\}\right\}\right\}\right\}$$

3. *"What has the boy told to the boy?"*

$$\left\{\frac{\epsilon_{C_{Ques.}}}{\mathfrak{L}_4}\left\{\frac{\textit{has}}{\mathfrak{L}_{13}}\left\{\left\{\frac{\textit{the}}{\mathfrak{L}_9}\ \frac{\textit{boy}}{\mathfrak{L}_{16}}\right\}\left\{\frac{\epsilon_v}{\mathfrak{L}_7}\left\{\frac{\textit{what}}{\mathfrak{L}_{11}}\left\{\frac{\textit{told}}{\mathfrak{L}_2}\left\{\frac{\textit{to}}{\mathfrak{L}_8}\left\{\frac{\textit{the}}{\mathfrak{L}_{10}}\ \frac{\textit{boy}}{\mathfrak{L}_{16}}\right\}\right\}\right\}\right\}\right\}\right\}\right\}\right\}$$

4. *"Who has the boy told the story to?"*

$$\left\{\frac{\epsilon_{C_{Ques.}}}{\mathfrak{L}_4}\left\{\frac{\textit{has}}{\mathfrak{L}_{13}}\left\{\left\{\frac{\textit{the}}{\mathfrak{L}_9}\ \frac{\textit{boy}}{\mathfrak{L}_{16}}\right\}\left\{\frac{\epsilon_v}{\mathfrak{L}_7}\left\{\left\{\frac{\textit{the}}{\mathfrak{L}_{10}}\ \frac{\textit{story}}{\mathfrak{L}_{16}}\right\}\left\{\frac{\textit{told}}{\mathfrak{L}_2}\left\{\frac{\textit{to}}{\mathfrak{L}_8}\frac{\textit{who}}{\mathfrak{L}_{11}}\right\}\right\}\right\}\right\}\right\}\right\}\right\}$$

5. *“Was the story told to the boy?”*

$$\left\{\frac{\epsilon_{C_{Ques.}}}{\mathfrak{L}_6} \left\{\frac{\textit{was}}{\mathfrak{L}_{13}} \left\{\frac{\epsilon_v}{\mathfrak{L}_5} \left\{\left\{\frac{\textit{the}}{\mathfrak{L}_9}\ \frac{\textit{story}}{\mathfrak{L}_{16}}\right\} \left\{\frac{\textit{told}}{\mathfrak{L}_2} \left\{\frac{\textit{to}}{\mathfrak{L}_8}\ \left\{\frac{\textit{the}}{\mathfrak{L}_{10}}\ \frac{\textit{boy}}{\mathfrak{L}_{16}}\right\}\right\}\right\}\right\}\right\}\right\}\right\}$$

6. *“Who was the story told to?”*

$$\left\{\frac{\epsilon_{C_{Ques.}}}{\mathfrak{L}_4} \left\{\frac{\textit{was}}{\mathfrak{L}_{13}} \left\{\frac{\epsilon_v}{\mathfrak{L}_5} \left\{\left\{\frac{\textit{the}}{\mathfrak{L}_9}\ \frac{\textit{story}}{\mathfrak{L}_{16}}\right\} \left\{\frac{\textit{told}}{\mathfrak{L}_2} \left\{\frac{\textit{to}}{\mathfrak{L}_8}\ \frac{\textit{who}}{\mathfrak{L}_{11}}\right\}\right\}\right\}\right\}\right\}\right\}$$

7. *“What was told to the boy?”*

$$\left\{\frac{\epsilon_{C_{Ques.}}}{\mathfrak{L}_4} \left\{\frac{\textit{was}}{\mathfrak{L}_{13}} \left\{\frac{\epsilon_v}{\mathfrak{L}_5} \left\{\frac{\textit{what}}{\mathfrak{L}_{12}} \left\{\frac{\textit{told}}{\mathfrak{L}_2} \left\{\frac{\textit{to}}{\mathfrak{L}_8}\ \left\{\frac{\textit{the}}{\mathfrak{L}_{10}}\ \frac{\textit{boy}}{\mathfrak{L}_{16}}\right\}\right\}\right\}\right\}\right\}\right\}\right\}$$

The yes/no-question listed in (1) has 9 tokens, which is longer than any expression appearing in the PLD; additionally, (1) demonstrates both triple use of a lexical feature sequence (i.e. $\mathfrak{L}_{10}$), and triple use of a phonological form (i.e. “the”), both of which were not seen in any derivation yielded for entries in the PLD. Any one of the arguments in (1) may be replaced with a wh-word that undergoes wh-raising: raising the (wh-word replacement of the) subject, direct object, and indirect object in (1) produces the wh-questions in (2), (3), and (4) respectively. The inferred lexicon can also parse passive variants of these expressions, yielding novel structures: the yes/no-question listed in (5) is formed by passivizing (1); (6) is a wh-question formed from (5) by raising the wh-word *“who”* from the internal argument position of *“told”* that corresponds to the indirect-object; (7) is a wh-question formed from (5) by raising the wh-word *“what”* from the internal argument position of *“told”* that corresponds to the direct-object.

Having considered several cases of novel syntactic structures that the (optimal) inferred lexicon can yield, let us now briefly consider three kinds of *ungrammatical* expressions that the optimal inferred lexicon will not generate.[42]

(i) The optimal inferred lexicon will not yield a Wh-question in which there is Wh-fronting of the object but no subject-auxiliary verb inversion – e.g. the lexicon will not produce the ungrammatical expression:

(1) * *“What John has eaten?”*

To see why, observe that $\mathfrak{L}_4$ is the only lexical feature sequence in the lexicon that triggers Wh-raising (via the licensor feature $+z$), and $\mathfrak{L}_4$ has the selectional feature $<= x_0$ that triggers $T$-to-$C$ head-movement (i.e. aux-raising).

(ii) The optimal inferred lexicon will not yield a derivation involving a ditransitive verb in which there is no $V$-to-$v$ head-movement – e.g. the lexicon will not produce the following (ungrammatical) expression:

(2) * *“Has John someone asked everything?”*

[42] Note that the three *ungrammatical* expressions presented are variants of the three *grammatical* expressions listed in entries $I_{12}$, $I_{22}$ and $I_1$ of the PLD respectively.

To see why, observe that the axioms (in the derivation model) enforcing extended functional projections (see §2.3.2 for details) require that the projection of the lexical verb ("asked") merge into the complement position of the projection of a light-verb, and both of the lexical feature sequences that code for light-verbs – i.e. $\mathfrak{L}_5$ and $\mathfrak{L}_7$ – have the selectional feature $<= x_0$ that triggers $V$-to-$v$ head-movement.

(iii) The optimal inferred lexicon will not yield a derivation in which an argument is not raised to structural subject position – e.g. the lexicon will not produce the following (ungrammatical) expression:

(3) * *"Was eaten icecream."*

To see why, observe that the axioms enforcing extended functional projections require the presence of a tense-marker; however there is only one lexical feature sequence, $\mathfrak{L}_{13}$, that is associated with the category $T$ (and can therefore serve as a tense-marker), and $\mathfrak{L}_{13}$ has a (licensor) feature $+l$ that triggers subject-raising.

**Further Examination of the Instantaneous Acquisition Procedure.**

Having analyzed the *optimal* lexicon that was inferred by using the *instantaneous acquisition procedure* to process the PLD, let us now briefly examine the acquisition procedure itself by considering an extension of the computational experiment presented earlier in this chapter – specifically, we will examine the (optimal) inferred lexicons that the procedure outputs after processing only entries $I_0$ through $I_k$ in the PLD, for values of $k$ up to and including 28. This examination is motivated by two questions about the procedure. First, does processing only a subset of the PLD produce an (optimal) inferred lexicon that yields derivations different than those yielded by the lexicon inferred when processing the entire PLD? Second, as the procedure processes increasingly larger subsets of the PLD, under what circumstances do increases in the range of information present in the subset of the PLD – i.e increases in the number of distinct phonological forms in the PLD and increases in the number of distinct syntactic features needed to satisfy the interface conditions in the PLD – translate into increases in the size of the inferred lexicon, and what role do the optimization metrics play in this? Let us now turn to detailing the specifics of this examination.

To begin, let $L_k$ denote the optimal lexicon inferred after processing only the first entries $k+1$ of the PLD listed in Table 3.2, so that $L_0$ is the optimal lexicon inferred by the procedure if it only processes the first entry in the PLD (i.e. $I_0$), and $L_{28}$ is the optimal lexicon inferred by the procedure if it processes the entire PLD. Next, for each value of $k$, let $L_k^{'}$ denote the subset of $L_{28}$ that is involved in satisfying the first $k+1$ entries in the PLD – note that the subset $L_k^{'}$ can be obtained from the Summary of Derivations listed in Table 3.4 by identifying the subset of lexical feature sequences involved in the columns for $I_0, ..., I_k$. Finally, to measure the range of information present in the subset of the PLD being processed, we will define, for each value of $k$, the following two values: (i) let $S_k$ be the number of distinct syntactic structures (abstracting away from the particular phonological forms appearing in these structures) required to satisfy the first $k+1$ entries of the PLD; (ii) let $P_k$ be the number of distinct phonological forms that appear in the first $k+1$ entries of the PLD.

We will now consider, for each value of $k$, the statistics of $L_k$ and $L_k^{'}$ – i.e. the number of distinct lexical feature sequences and the total number of syntactic features appearing across these lexical feature sequences; note that these two statistics are the valuations of the first two metrics (3.1 and 3.5) that are optimized in Step 4 of the acquisition procedure. The statistics for each inferred lexicon are presented in Fig. 3-10, along with the statistics for the subsets of $L_{28}$. Let us now touch on several observations about these statistics.

(i) For $k < 4$, $L_k$ is smaller than $L_k^{'}$ (with respect to both the number of distinct lexical feature sequences and the total number of syntactic features found in those lexical feature sequence), suggesting that the acquisition procedure is prone to "over-optimizing" when restricted to processing a small amount of PLD (in this case four or fewer entries of the PLD) – i.e. for $k < 4$, $L_k$ yields derivations that technically satisfy the interface conditions listed in $I_0 - I_k$, but that do not accord with the derivations prescribed by contemporary theories of syntax. For values of $k \geq 4$, $L_k$ and $L_k^{'}$ are the same size, suggesting that when the size of the PLD is large enough – i.e. when the value of $k$ is large enough – the acquisition procedure produces the same lexical entries (in order to yield derivations that satisfy the first $k+1$ entries of the PLD) that it would have after processing the entire PLD. Taken together, these two observations suggest that the acquisition procedure might be modified to (sequentially) process the PLD in batches that are just large enough for the acquisition procedure to avoid "over-optimizing", so that a lexicon might be grown in stages, thereby avoiding the problem of the learner having to retain the entire PLD in memory before inferring the lexicon all at once; looking ahead, modifying the acquisition procedure in this manner is precisely what will be done in the next section.

(ii) The size of $L_k^{'}$ does not increase unless $S_k$ increases – this is because the first two metrics that the procedure optimizes (i.e. metrics 3.1 and 3.5) seek to minimize the size of the lexicon so that the number of distinct lexical feature sequences (and the total number of syntactic features in the lexicon) is only increased if the current number of them is insufficient to yield derivations that satisfy the entries in the PLD – thus if $S_k$ does not increase (i.e. $S_k = S_{k-1}$), then the size of the lexicon will not grow (i.e. $L_k^{'} = L_{k-1}^{'}$).

(iii) Sometimes $S_k > S_{k-1}$ but the size of $L_k^{'}$ is the same as $L_{k-1}^{'}$ (e.g. as when $k = 5$, $k = 10$ or $k = 26$), which implies that the lexical feature sequences in $L_{k-1}^{'}$ that were used to satisfy $I_0 - I_{k-1}$ suffice for deriving a new syntactic structure that satisfies $I_k$; notably, when $k \geq 4$, $L_k$ and $L_k^{'}$ are the same size and thus the lexicon inferred after processing the first $k+1$ entries of the PLD can yield a (novel) syntactic structure (to satisfy $I_k$).

(iv) Sometimes $P_k > P_{k-1}$ but the size of $L_k^{'}$ is the same as $L_{k-1}^{'}$ (e.g. as when $k = 5$, $k = 11$ or $k = 15$), which implies that the set of lexical feature sequences in $L_{k-1}^{'}$ suffice to yield a derivation that satisfies $I_k$; importantly, if $k \geq 4$ and $L_{k-1} = L_k$, then the only updates to $L_{k-1}$ required to produce $L_k$ are: (a) extending the vocabulary to include the new phonological forms, and (b) creating new associations between some of the lexical feature sequences in $L_{k-1}$ (that are required to satisfy $I_0 - I_{k-1}$) with the (new) phonological forms that were added to the vocabulary. This suggests that eventually, once the learner has acquired all of the lexical feature sequences needed to yield the syntactic structures appearing in their language[43], the lexicon will only be

[43] Although in this section the lexicon inferred by the *instantaneous acquisition procedure* can only produce

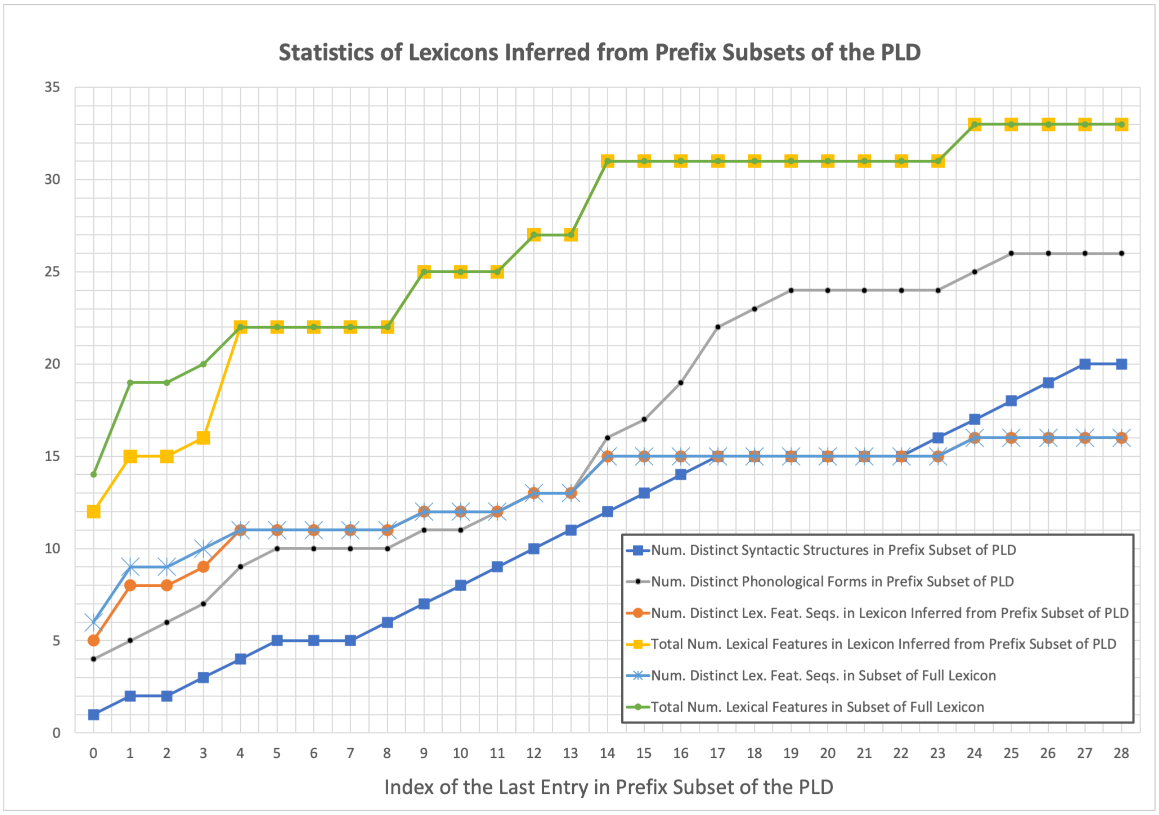


Figure 3-10: Statistics for the Lexicon Inferred from each Prefix Subset of the Primary Linguistic Data (PLD). For each prefix subset $I_0 \ldots I_k$ of the PLD, the x-axis indicates the index of the *last entry* in that prefix subset, and the y-axis indicates the values for the relevant statistics associated with each prefix subset, so that the statistics above position $k$ on the x-axis pertain to the lexicon inferred from entries in the prefix subset $I_0 \ldots I_k$ of the PLD. The chart shows, for each prefix subset $I_0 \ldots I_k$ of the PLD, both the number of distinct lexical feature sequences that appear in the (optimal) lexicon inferred (using the acquisition procedure) from that subset of the PLD, as well as the (total) number of syntactic features that appear in those lexical feature sequences; these two statistics measure the size of the inferred lexicon. The chart also shows, for each prefix subset $I_0 \ldots I_k$ of the PLD, both the number of distinct lexical feature sequences in the *full lexicon* – i.e. the (optimal) lexicon that was inferred from the entire PLD – that were involved in the derivations yielded by the full lexicon to satisfy the entries in the prefix subset, as well as the (total) number of syntactic features that appear in these lexical feature sequences. Finally, for each prefix subset $I_0 \ldots I_k$ of the PLD, the chart also shows two measures of the amount of information in that prefix subset – namely, the number of distinct syntactic structures needed to satisfy the interface conditions listed in the prefix subset, and the number of distinct phonological forms appearing in that subset.

updated with respect to the learner's vocabulary and the associations between lexical feature sequences and phonological forms – this is akin to an adult speaker who does not learn new syntax after childhood, but continues throughout their life to learn new words in the open categories (e.g. nouns, verbs, adjectives and adverbs).

This examination demonstrates that given only a small subset of the PLD, the acquisition procedure is still able to *solve for syntax* and infer a *productive* lexicon.

This examination demonstrates that the acquisition procedure is able to *solve for syntax* given only a small subset of the PLD, and that the acquisition procedure can infer *productive* lexicons from small subsets of the PLD.

### 3.2.4 Summary

The computational experiment presented in this section demonstrates that, given a sequence of interface conditions, it is possible to use a modern, industrial strength SMT-solver to infer, from an empty initial lexicon and a system of first-order logic equations that were derived from the specified sequence of interface conditions, a lexicon that yields derivations that satisfy the stipulated interface conditions. Furthermore, when we introduced optimization constraints that sought to minimize the size of the lexicon and the size of the derivations it yields, we found that the *instantaneous acquisition procedure* infers a lexicon that yields derivations that accord (in form) with the prescriptions of contemporary theories of modern syntax. In particular, the optimized lexicon that was inferred yields derivations for declaratives, yes/no-questions, and wh-questions in both the active and passive voice, and these derivations involve various forms of syntactic movement including wh-raising, subject-raising, T-to-C head-movement and V-to-v head-movement. Additionally, the inferred lexicon includes lexical entries for *covert* complementizers and light-verbs.

Remarkably, the inferred lexicon is composed of a small set of lexical feature sequences, each of which associate with various subsets of the learner's vocabulary (including both *overt* and *covert* phonological forms), and these lexical feature sequences may be combined in many different ways to yield a variety of syntactic structures. Furthermore, these lexical feature sequences can be put into new combinations with one another so as to yield novel structures that were not produced in the course of satisfying the interface conditions listed in the PLD. In particular, novel structures were produced not only by substituting phonological forms with other phonological forms that are in the same cluster (i.e. word class), but also by employing the lexical feature sequences to take on syntactic roles that they had not in the context of any derivation yielded by the inferred lexicon (in order to satisfy an entry in the PLD).

We also saw that the acquisition procedure can infer, from a small sequence of interface conditions, a lexicon that yields the same derivations (to satisfy this small sequence of interface conditions) as a lexicon inferred from a larger sequence of interface conditions of which the smaller sequence is a subset. This suggests a way to address the *instantaneous acquisition procedure*'s requirement that the learner store the entire PLD in their memory before inferring the entire lexicon at once – namely, that a lexicon may be acquired incrementally by processing the PLD in small batches – and the next section will introduce an *incremental acquisition procedure* that takes this approach, enabling it to process a larger PLD while

a finite number of (novel) syntactic structures, in the next section the *incremental acquisition procedure* will be used to infer a finite lexicon that can generate a countably infinite set of distinct syntactic structures.

respecting the condition of psychological plausibility that the learner has a limited memory for remembering the particular interface conditions that appear in the PLD.

## 3.3 An Incremental Model of Acquisition

This section introduces an *incremental procedure for acquisition* that is formed by extending the *instantaneous acquisition procedure* (previously introduced in §3.2.2) to: (i) take as an (optional) input a lexicon that represents the learner's current state of knowledge prior to running the procedure – i.e. the assumption that the initial state of the learner is an empty lexicon is relaxed; (ii) incrementally process the primary linguistic data (PLD) in batches and output an optimized lexicon after each batch is processed that is both compatible with all of the PLD that has been processed up to that point, and includes the input lexicon as a subset. *These two extensions allow for the lexicon that was output by the procedure after processing one batch of the PLD to serve as the (input) initial lexicon to the procedure when processing the subsequent batch of the PLD. The incremental acquisition procedure can thus be run repeatedly so as to sequentially consume the PLD in batches and incrementally acquire a lexicon; in this manner, the incremental theorem prover (i.e. the SMT-solver) may be used to learn a grammar in stages.*

When the *incremental acquisition procedure* processes the first batch of the PLD and no initial lexicon is supplied (as was the case in the experiment carried out in §3.2.3), then the *incremental acquisition procedure* is functionally equivalent to the *instantaneous acquisition procedure.* As the acquisition trajectory proceeds, the learner converges on a lexicon that includes all of the lexical feature sequences needed to yield the derivations that satisfy the pairings of interface conditions enumerated in the PLD (just as a child eventually will, and inevitably does, lock onto the grammar of their first language); the learner can then focus on learning novel associations between phonological forms and lexical feature sequences (as a human does over the course of their entire life). In the limit, the input lexicon will suffice to process the batch of PLD entries that the learner must process, and the *incremental acquisition procedure* will, in effect, become the *procedure for parsing* – i.e. the *incremental acquisition procedure* will output the parse of each entry in the PLD batch with the supplied lexicon, and the output (inferred) lexicon will be the (input) supplied lexicon. *Thus, we see that the parsing procedure and the instantaneous acquisition procedure are both edge cases of this more general procedure for incremental acquisition, thereby satisfying the psychological plausibility condition that the language acquisition system and parsing systems be co-integrated.*

The remainder of this section is organized as follows: §3.3.1 motivates the aforementioned two extensions by pointing out deficits of the *instantaneous acquisition procedure*, and then presents the *incremental acquisition procedure* and detail how it works; §3.3.2 then presents a computational experiment that serves to demonstrate the capacity of the system to infer, from a corpus of sentences with at most one level of embedding, a grammar that can yield structures for sentences with unbounded embedding – i.e. the inferred lexicon can parse sentences with embedded sentences and (restrictive) relative clauses, enabling the learner to create and parse an unbounded set of novel structures and handle sentences with an arbitrary level of embedding.

### 3.3.1 Extending the Instantaneous Model

The *incremental acquisition procedure* (listed on Pg. 157) is an extension of the *instantaneous acquisition procedure* (developed in §3.2.2), that continues to take the form of a computational model of a child language learner that adheres to the criterion for model of language acquisition set out in (Chomsky, 1965, Pg. 30-31).

The *incremental acquisition procedure* solves two problems that arise when using the *instantaneous acquisition procedure*. First, in the case of the *instantaneous acquisition procedure*, whatever grammar $G_i$ is inferred from state $S_i$ (i.e. the SMT-model constructed after processing the first $i$ entries in the PLD), a subsequently inferred grammar $G_{i+1}$ inferred from state $S_{i+1}$ (derived from the first $i + 1$ entries in the PLD) may not be a superset of $G_i$ – indeed, it is possible that $G_i$ and $G_{i+1}$ are disjoint; consequently, the knowledge of language that the learner acquired previously may not in any way help the learner acquire new knowledge of language, and there is no guarantee that previously acquired knowledge of language will even be retained by the learner.[44] Second, as the size of the PLD is increased, the size of the SMT-model constructed by the *instantaneous acquisition procedure* also increases (since a derivation model must be constructed for each entry in the PLD), and the runtime of the solver (as it checks the model) quickly becomes intractable.

The *incremental acquisition procedure* addresses these two problems by consuming the PLD in batches. Upon consuming a batch of the PLD and a specification of a lexicon learned up until that point, the procedure creates a lexicon model and hardcodes it (i.e. values free-variables in the SMT-formula) to ensure that the lexicon that will be inferred is a *superset* of the specified (input) lexicon. The procedure then creates a derivation model for each entry in the batch of the PLD, and "attaches" (only) these derivation model to the lexicon model. Finally, the model is checked and a lexicon is inferred, just as in the *instantaneous acquisition procedure*. As the (output) inferred lexicon is a superset of the lexicon specified in the input to the procedure, the incremental procedure can be applied repeatedly, taking the output lexicon from one run of the procedure as the input lexicon for the next run of the procedure, so as to incrementally acquire a lexicon that is able to parse all of the entries in the PLD consumed up to that point; in this way, the learner is building upon previously acquired knowledge of language, and the first problem with the *instantaneous acquisition procedure* is resolved. Furthermore, since derivation models were only created for the entries in the batch of the PLD being processed, the size of the model is proportional to the size of the batch (of the PLD), and not the size of the entire PLD, thus resolving the second problem with the *instantaneous acquisition procedure*.

The *incremental acquisition procedure* also introduces the capability to *checkpoint* the state of the learner – i.e. it serializes the state of the learner such that later the procedure may be run starting from the checkpoint; importantly, this has a practical consequence – the procedure may be run from the same checkpoint (i.e. input lexicon) with different batches, thereby allowing one to experiment and understand how the particular entries in the PLD affect the inferred grammar.

To summarize, the *incremental acquisition procedure* may initially be used just as the *instantaneous acquisition procedure* was, with an empty set supplied as the initial lexicon when processing the first batch of the PLD; the learner may then consume additional batches

[44] This is because, as the PLD is consumed one entry at a time, the SMT-solver is simply iteratively constructing an SMT-model, and each time the solver is instructed to check the model, there is no guarantee that the solver will use the lemmas it proved earlier in prior check of the model – i.e. the solver is in effect re-solving the model from scratch.

of the PLD by rerunning the *incremental acquisition procedure* and using the lexicon output from the prior run of the procedure as the input lexicon for the next run of the procedure. In this way, the learner is able to incrementally consume an arbitrarily large primary linguistic data in batches by repeatedly running the *incremental acquisition procedure*, each time outputting its state – i.e. the lexicon acquired up until that point – at the end of each run of the procedure. Having outlined the basic form of the *incremental acquisition procedure*, some clarifying comments and observations are now in order.

**Relaxing Model Parameters.**
This procedure introduces two new questions pertaining to parameter-setting that must be addressed on each run of the procedure.

First, the system must determine how many entries from the PLD to include in the next batch of interface conditions to be consumed. To address this point, this thesis assumes, as a condition of psychological plausibility, that the number of entries to process on each run of the acquisition procedure, referred to as that run's *acquisition window*, should be fairly limited – i.e. no more than say 4-5 entries; this is taken to be a middle ground between the computationally-expensive approach of consuming the entire PLD at once as a single batch (employed by the instantaneous model), and consuming the one sentence at a time – i.e. by shrinking acquisition window to a single pair of interface conditions to be presented to the learner – which can lead the inference procedure astray when it "over-optimizes" to handle a single derivation.[45] Although the *incremental acquisition procedure* in effect updates the learner's lexicon after consuming a batch of entries in the PLD, this does not preclude the learner from attempting to parse (using the lexicon they already have) each entry in the PLD one at a time as it is consumed.

Second, the system must determine how to value model parameters that govern the size of the SMT-model that the procedure will construct and then check (using an SMT-solver); this task is made difficult by the fact that the system does not know, a-priori, how large the PLD will be, or how large the target lexicon will be. This thesis takes the approach of re-valuing these model parameters each time the *incremental acquisition procedure* is run, based on the particulars of the batch of the PLD that is about to be processed and the size of the input lexicon, so that the acquisition trajectory need not be constrained by the model parameters used in prior runs of the procedure. In particular, given that the *incremental acquisition procedure* is intended to "incrementally grow" the lexicon as each batch of entries in the PLD is consumed, the size of the model must be gradually increased as more of the PLD is consumed, and the model parameters that constrain the size of the model must be "relaxed." The nuance here is that the system must not relax the model parameters much more than necessary so as to avoid constructing a model that is much larger than needed and thus unnecessarily increasing the time taken by the SMT-solver to check the model. Let us now consider how these valuations are determined for the lexicon model and the derivation models.

1. **Relaxing Parameters that Constrain the Lexicon Model.**

   The lexicon model that the procedure will construct must be, at a minimum, large enough to hold the lexicon supplied in the input to the procedure – i.e. the procedure must allocate enough lexical feature sequences, and enough features per lexical feature sequence, such that the lexicon model can be hard-coded with the supplied lexicon,

[45] See Figure 3-10 and the relevant discussion at the end of §3.2.3.

## INCREMENTAL ACQUISITION PROCEDURE

1. The input consists of:
   - A. a queue of pairs of interface conditions, referred to as the PLD, with $n > 0$ entries;
   - B. a valuation of model parameters;
   - C. an *empty* SMT-solver stack, $S$, with each entry on the stack an SMT-formula, and the conjunction of the entries on the stack referred to as "the acquisition model." (Note that to "check the acquisition model" is to use the SMT-solver to check the conjunction of the terms on the solver's stack.)
   - D. (optional) an initial lexicon;
2. The initial state of the learner, prior to consuming the PLD, is either the lexicon supplied in the input if one was, and otherwise an empty lexicon:
   - A. initialize a lexicon model (i.e. an SMT formula), $m_l$ from the supplied model parameters, the PLD, and the initial lexicon if one was supplied;
   - B. push $m_l$ onto the stack;
   - C. (optional) check the acquisition model.
3. The learner consumes the PLD until it is empty, incrementally constraining the lexicon model:
   - A. pop an entry $I_i$ off of the queue;
   - B. initialize a derivation model (i.e. an SMT formula), $m_d^i$, from model parameters and interface conditions $I_i$;
   - C. push $m_d$ onto the stack;
   - D. translate $I_i$ into an SMT–formula, $m_I^i$, that constrains the derivation model $m_d^i$;
   - E. push $m_I^i$ onto the stack;
   - F. construct an SMT-formula, $m_b^i$, that connects, via an uninterpreted function, the derivation model, $m_d^i$, to the lexicon model, $m_l$;
   - G. push $m_b^i$ onto the stack;
   - H. (optional) check the acquisition model.
4. The learner selects a grammar by optimizing the model:
   - A. optimize the acquisition model using metric (3.1);
   - B. optimize the acquisition model using metric (3.5);
   - C. optimize the acquisition model using metric (3.7);
   - D. optimize the acquisition model using metric (3.8);
   - E. optimize the acquisition model using metric (3.9);
   - F. check the acquisition model using the SMT-solver, and if the acquisition model is found to be satisfiable, recover the identified (satisfiable) model interpretation (i.e. a solution to the acquisition model).
5. The output of the procedure is the final state of the learner:
   - A. for each entry $I_i$ in the PLD, a derivation, $d_i$, that satisfies the conditions imposed by $I_i$;
   - B. the inferred minimalist lexicon that can yield each $d_i$;
   - C. (Optional) the recovered model interpretation;
   - D. the solver stack holds: $m_l$; $m_d^i$ and $m_I^i$ for $1 \leq i \leq n$; constraints associated with each optimization metric.

thereby guaranteeing that the (output) inferred lexicon is a superset of the input lexicon;[46] additionally, the size of the PF node sort must be larger than the number of distinct phonological forms in the supplied lexicon and the PLD. If it is the case that all of the entries in the (input) PLD batch can be parsed with the supplied lexicon, then the lexicon model need not be any larger than what was stipulated above. If this is not the case – i.e. some subset of the PLD batch cannot be parsed with the supplied lexicon – then one of the following three scenarios holds:

(a) new phonological forms appear in the input PLD batch and they are added to the existing vocabulary (present in the supplied lexicon), and no new lexical feature sequences need to be added to the supplied lexicon – in this case, the SMT-solver must work out how to associate these new phonological forms with the lexical feature sequences already found in the supplied (input) lexicon so that the inferred lexicon yields derivations that satisfy the entries in the PLD batch;[47]

(b) no new phonological forms appear in the input PLD batch, but one or more new lexical feature sequences must be added to the supplied lexicon and associated with the existing phonological forms – in this case, the SMT-solver is tasked with both identifying what these new lexical feature sequences should be, and how they should associate with the existing phonological forms, so that the inferred lexicon yields derivations that satisfy the entries in the PLD batch;

(c) the input PLD batch includes new phonological forms that must be added to the existing vocabulary *and* new lexical feature sequences must be added to the lexicon – in this case, the SMT-solver must identify what new lexical feature sequences are needed and also what new associations to establish between the lexical feature sequences and the (both old and newly added) phonological features so that the inferred lexicon can satisfy the entries in the PLD batch.[48]

In the each of these three cases, the system must consider how many additional associations to allow between each phonological form and each lexical feature sequence; note that whether a phonological form can form a (new) association with a lexical feature sequence may depend on whether the lexical feature sequence is associated with an open (i.e. functional) category or a closed (i.e. lexical) category. In the second and third case, the system must additionally consider how many new lexical feature sequences the inferred lexicon shall be allowed to have, how many features each of these lexical feature sequences should be allowed to have, and what member of the Category sort each new lexical feature sequence should be associated with.

2. **Relaxing Parameters that Constrain the Derivation Model.**

The size of each of the derivation models that the procedure will construct is determined as it was in the construction of the *SMT-model of the minimalist parser* – i.e. by inspecting the particulars of the interface conditions that the derivation must supply (see §2.4.1 for the details of this). In the case of the *incremental acquisition procedure*, which is intended to be used to learn a grammar from sentences with one or more levels

[46] The hardcoding of the lexicon model – i.e. the valuation of free-variables in the SMT-formula that is the lexicon model — is done just as it was for the SMT-model of the parser. See §2.4.1 for further discussion.

[47] This case is closely related to how the parsing model may handle out-of-vocabulary phonological forms during parsing.

[48] The *instantaneous acquisition procedure* deals with this case as it starts with an empty lexicon.

of embedding present, the following additional heuristic was used: when more than one predicate is detected in either the LF or PF interface conditions[49], the system will increment, for each additional predicate detected, the values of the following model parameters:

(a) the number of covert (empty) lexical items allocated is increased by 2;

(b) the maximum number of instances of phrasal movement increased by 2;

(c) the maximum number of instances of head movement allowed is increased by 2.

This heuristic is motivated by the assumption that there is an upper-bound on the resources required by a single extended functional projection C-T-v-V; assuming each additional predicate requires its own extended functional projection, the system can thus establish an upper-bound on the resources required by the derivation as the sum of the resources required by each of the extended functional projections that will make up the derivation.

Ultimately, strategies for relaxing the model parameters (prior to consuming each batch of the PLD) must balance the need to (incrementally) grow the lexicon in concert with the requirement that the model not become so large that model-checking becomes intractable. An approach that future efforts to extend this system would do well to consider is to only grow the lexicon when the system is not required to introduce more than a very small handful of new phonological forms and lexical feature sequences; in this approach, the *incremental acquisition procedure* will only be able to learn new syntax that it almost already knows.

**Ordering of the (PLD) Batches Does Matter.**
In the case of the *instantaneous acquisition procedure*, if model parameters are relaxed enough, an optimal lexicon that is compatible with the PLD, if one exists, can be identified given enough time for the solver to check the model. The situation is different with the *incremental acquisition procedure*, which opts to choose a particular strategy for how to construct and solve the model in stages in exchange for losing the guarantee of finding an optimal lexicon. Let us consider a simplified scenario that illustrates why consuming the PLD incrementally in batches (as opposed instantaneously consuming the entire PLD in a single batch) can result in the learner acquiring a suboptimal lexicon with respect to the number of lexical entries in the lexicon.[50]

To begin, suppose the learner will consume a PLD, $I_1, I_2, ..., I_n$, in two consecutive batches, $A = [I_1, ..., I_k]$ and $B = [I_{k+1}, ..., I_n]$. After consuming $A$, the learner has acquired a lexicon $L_A$. Suppose the learner then proceeds to consume $B$, and learns a set of additional lexical items[51], denoted $L_B$, resulting in a new lexicon, $L_{AB}$, that is a proper superset of $L_A$. Let $L^{'}_{AB}$ be the lexicon that would be inferred if the learner consumed the PLD all at once, using the *instantaneous acquisition procedure*, and let $L^{'}_{A}$ and $L^{'}_{B}$ be subsets of $L^{'}_{AB}$ needed to yield the derivations that satisfy entries in $A$ and $B$ respectively. Observe that $|L_A| \leq |L^{'}_{A}|$ because $L_A$ is the minimal set of entries required to yield the derivations required to satisfy $A$ – i.e. entries in $L^{'}_{A}$ must produce derivations required to satisfy $A$, but

[49] For the purposes of this heuristic, a predicate is taken to be either: (i) an overt phonological form associated with the categorical variable "V" (in the case of PF interface conditions), or (ii) a annotation of predicate-argument structure (in the case of LF interface conditions).

[50] Note that here we are referring to the number of distinct lexical feature sequences in the lexicon, factored apart from their associations with phonological forms.

[51] Observe that when the learner consumes $B$, it already has access to all of the lexical entries in $L_A$.

may also need to participate in some of the derivations required to satisfy $B$, which may require additional lexical entries. We can thus write:

$$|L_{AB}| = |L_A \sqcup L_B| = |L_A| + |L_B| \leq |L_A^{'}| + |L_B| \tag{3.23}$$

Then it is the case that:

$$|L_{AB}^{'}| \leq |L_{AB}| \tag{3.24}$$

because the *instantaneous acquisition procedure* will identify the optimal lexicon.

Let us now consider the circumstances under which the inequality 3.24 becomes strict, thereby indicating that the *incremental acquisition procedure* did not acquire the optimal lexicon. We begin by combining 3.23 and the strict form of 3.24 to obtain:

$$\begin{align} |L_{AB}^{'}| < |L_{AB}| \leq |L_A^{'}| + |L_B| \tag{3.25} \\ |L_{AB}^{'}| < |L_A^{'}| + |L_B| \tag{3.26} \\ |L_{AB}^{'}| - |L_A^{'}| < |L_B| \tag{3.27} \\ \tag{3.28} \end{align}$$

Next we observe that as $L_A^{'} \subseteq L_{AB}^{'}$, it is the case that:

$$|L_{AB}^{'} \setminus L_A^{'}| < |L_B| \tag{3.29}$$

The term on the left hand side of equation 3.29 is the number of entries in $L_{AB}^{'}$ that only participate in derivations that satisfy entries in $B$, whereas the term on the right hand side of the equation is the number of entries the learner newly acquired in processing $B$ (using the *incremental acquisition procedure*). The difference $|L_B| - |L_{AB}^{'} \setminus L_A^{'}|$ is referred to as *the cost of $A$ with respect to $B$* and is denoted $C(A, B)$. When $C(A, B) > 0$, the lexicon inferred by the learner using the *incremental acquisition procedure* will be sub-optimal (with respect to the number of distinct lexical feature sequences) as compared to the lexicon inferred by the learner using the *instantaneous acquisition procedure*; this occurs when the learner learns a lexicon from $A$ that may be considered “over-optimized” in so far as it does not include lexical entries that could have been used later when processing $B$.[52]

The ordering in which the learner consumes the batches may also affect whether or not the learner acquires a suboptimal lexicon, with different orderings potentially resulting in different degrees of sub-optimality. To see why this is the case, let us consider the earlier scenario of the PLD being divided into batches $A$ and $B$, but now consider two different incremental learners: learner $X$ will first process $A$ then process $B$ as before, and learner $Y$ will first process $B$ then process $A$. Let us now identify three cases in which the ordering of the batches can affect the optimality of the lexicon:

(i) If learners $X$ and $Y$ each acquire a lexicon that is optimal with respect to the optimization metrics (i.e. both lexicons have the same size as the optimal lexicon inferred by the *instantaneous acquisition procedure*), the ordering of the batches matters only if there is more than one lexicon that is optimal with respect to the optimization metrics.[53]

[52] See Figure 3-10 and the relevant discussion at the end of §3.2.3.

[53] In practice, the case of there being several distinct lexicons that are each optimal with respect to the

(ii) If one of $X$ or $Y$ acquired a suboptimal lexicon and the other did not, then the ordering matters.

(iii) Suppose that both learners acquire suboptimal lexicons (i.e. $C(A, B) > 0$ and $C(B, A) > 0$). If $C(A, B) = C(B, A)$, then again, the relative ordering does not matter as a suboptimal lexicon would be inferred in any case; however, without loss of generality, suppose $0 < C(A, B) < C(B, A)$. In this case the learner, $X$, consuming a presentation of the PLD with less cost, $C(A, B)$, will obtain a less suboptimal lexicon (relative to learner $Y$).

In the latter two of these cases, whether or not the ordering of the two batches will result in the acquisition of a suboptimal lexicon is dependent on the cost of each of the two ordering presentations. Since there is no guarantee that the presentation of the PLD made to the learner will have batches that all have pairwise equivalent cost, we must conclude that the ordering of the batches can affect the optimality of the inferred lexicon, *even if some of the orderings result in the acquisition of an optimal lexicon.*

**Optimization Metrics Remain as Before.**
The optimization metrics used in the *instantaneous acquisition procedure* may also be used in the *incremental acquisition procedure* without any modification *so long as the supplied (input) lexicon is exactly the lexicon output from the prior run of the incremental acquisition procedure.* To see why this is the case, consider that (optimization) metrics (3.1) and (3.5) both have as a lower-bound the valuations of those metrics from the prior run of the *incremental acquisition procedure*, since the lower-bounds for these two metrics are entirely determined by the size of the supplied (input) lexicon. Metric (3.7) remains unchanged because the lower-bound is established by the number of merge operations used by the earlier derivations in the PLD, and since those derivations cannot be changed (as the PLD consumed in earlier batches and the derivations yielded to satisfy the entries in said PLD are not retained in the learner's memory), only the merge operations that occur in the new derivations (that are yielded by the newly inferred lexicon) are a variable factor. Finally, metrics (3.8) and (3.9) remain unchanged because the lower-bound is determined by the set of selectional and licensing feature labels in the supplied (input) lexicon, and will only be increased if it is determined when processing the new batch of PLD that new licensing features are required (as a consequence of the Shortest Movement Condition in the MG formalism); the lower bound on the number of selectional features will not change as, without evidence of what derivations cannot be yielded, one selectional feature will continue to suffice.[54] That the optimization metrics may remain the same without the learner retaining knowledge of the derivations yielded in the past is important for conditions of psychological plausibility.

optimization metrics has not been observed.

[54] Inference can be carried out with only one selectional feature provided, without collapsing the number of lexical items after optimizing the grammar, because the Category variables associated with lexical entries and the axioms encoding extended projections work together to rule out the ruling of (some) overgenerations, a task that is otherwise delegated to the selectional features.

### 3.3.2 Learning Embedded Clauses

We will now illustrate how the *incremental acquisition procedure* works by using it to acquire a grammar that is able to parse sentences with embedded sentences and relative clauses.[55] The presentation is organized as follows. To begin, we will introduce the input to the procedure, which consists of: (i) the lexicon learned in §3.2 (see Table 3.3), which was only able to yield derivations with Degree-0 embedding – i.e. derivations that produce sentences without any subordinate clauses;[56] (ii) the PLD, listed in Table 3.9, which is organized in three batches, and in which every entry has at most one degree of embedding. Following this, we will detail how the procedure is used to drive the state of the learner from the initial state (i.e. the input lexicon) to the final state (i.e. the final outputted lexicon). We will then present the output of the procedure, a lexicon (listed in Table 3.11) that is a superset of the input lexicon, and that can yield, for each entry in the processed PLD, a derivation that satisfies the stipulated interface conditions. Finally, we will prove that this (outputted) lexicon is able to yield a syntactic structure with Degree-$n$ embedding (i.e. $n$-level deep embedding) for any $n \geq 0$, thereby demonstrating that the *incremental acquisition procedure* can infer, from a small, finite set of derivations with Degree-0 or Degree-1 embedding, a lexicon that can generate an unbounded set of syntactic structures, which the initial input lexicon could not do. This demonstrates that the acquisition model can infer a lexicon that makes "infinite use of finite means," a key goal of linguistic theory.

#### Initial Conditions and Input Data.

The initial state of the learner is the (input) lexicon, listed in Table 3.3, that was inferred by using the *instantaneous acquisition procedure* to process the PLD listed in Table 3.2. The learner is now tasked with processing new primary linguistic data, listed in Table 3.9, that consists of 11 entries (i.e. entries $I_{29}$ to $I_{39}$) divided into three batches; these three batches should be understood to follow the "first batch" of primary linguistic data – i.e. entries $I_0$ to $I_{28}$ listed in Table 3.2.

Let us now examine the entries that make up these three new batches of primary linguistic data. The second batch of primary linguistic data (i.e. entries $I_{29}$ to $I_{34}$) consists of sentences with a complementizer phrase (CP) serving as an argument within the matrix clause; the complementizer phrase may be an embedded sentence, as in the case of $I_{31}$ (square brackets are used to designate the embedded clause):

(4) *"Mary has told John [that icecream was eaten]."*

or an embedded question, in the case of $I_{32}$:

(5) *"Mary has asked John [whether she was eating pizza]."*

The third batch of primary linguistic data (i.e. entries $I_{35}$ and $I_{36}$) includes sentences in which a *restrictive* relative clause serves as an argument and the nominal antecedent *is not*

[55] *(Note that, as in §3.2, all of the derivations presented in this section* do *accord with the ordering of phonological forms listed in the associated PLD entry – i.e. if they are redrawn with Specifier–Head-Complement linearization then the correct SVO ordering becomes apparent; the figures were automatically rendered using Graphviz so that the arrows depicting syntactic movement would not overlap with the projections in the derivation.)*

[56] A subordinate clause is a clause that is embedded as a constituent of a matrix sentence, and that functions like a noun, adjective of adverb. A nominal clause is a type of subordinate clause that functions as a noun phrase.

| Batch | $I_i$ | Interface | Interface Conditions |
|---|---|---|---|
| 2 | $I_{29}$ | PF | john/N has asked/V whether pizza/N was eaten/V. |
| | | LF | $\theta_{\text{asked}}$[$s$: john, $o$: whether pizza was eaten], $Agr_{\text{has}}$[$s$: john], $\theta_{\text{eaten}}$[$o$: pizza], $Agr_{\text{was}}$[$s$: pizza] |
| | $I_{30}$ | PF | mary/N was told/V that john/N has eaten/V pizza/N. |
| | | LF | $\theta_{\text{told}}$[$o$: that john has eaten pizza, $i$: mary], $Agr_{\text{was}}$[$s$: mary], $\theta_{\text{eaten}}$[$s$: john, $o$: pizza], $Agr_{\text{has}}$[$s$: john] |
| | $I_{31}$ | PF | mary/N has told/V john/N that icecream/N was eaten/V. |
| | | LF | $\theta_{\text{told}}$[$s$: mary, $o$: that icecream was eaten, $i$: john], $Agr_{\text{has}}$[$s$: mary], $\theta_{\text{eaten}}$[$o$: icecream], $Agr_{\text{was}}$[$s$: icecream] |
| | $I_{32}$ | PF | mary/N has asked/V john/N whether she/N was eating/V pizza/N. |
| | | LF | $\theta_{\text{asked}}$[$s$: mary, $o$: whether she was eating pizza, $i$: john], $Agr_{\text{has}}$[$s$: mary], $\theta_{\text{eating}}$[$s$: she, $o$: pizza], $Agr_{\text{was}}$[$s$: she] |
| | $I_{33}$ | PF | who has mary/N told/V that she/N was eating/V icecream/N? |
| | | LF | $\theta_{\text{told}}$[$s$: mary, $o$: that she was eating icecream, $i$: who], $Agr_{\text{has}}$[$s$: mary], $\theta_{\text{eating}}$[$s$: she, $o$: icecream], $Agr_{\text{was}}$[$s$: she] |
| | $I_{34}$ | PF | who was asked/V whether mary/N has given/V john/N money/N? |
| | | LF | $\theta_{\text{asked}}$[$o$: whether mary has given john money, $i$: who], $Agr_{\text{was}}$[$s$: who], $\theta_{\text{given}}$[$s$: mary, $o$: money, $i$: john], $Agr_{\text{has}}$[$s$: mary] |
| 3 | $I_{35}$ | PF | who has told/V john/N everything/N that mary/N was asked/V? |
| | | LF | $\theta_{\text{told}}$[$s$: who, $o$: everything that mary was asked, $i$: john], $Agr_{\text{has}}$[$s$: who], $\theta_{\text{asked}}$[$o$: everything, $i$: mary], $Agr_{\text{was}}$[$s$: mary] |
| | $I_{36}$ | PF | was someone/N given/V everything/N that she/N has eaten/V? |
| | | LF | $\theta_{\text{given}}$[$o$: everything that she has eaten, $i$: someone], $Agr_{\text{was}}$[$s$: someone], $\theta_{\text{eaten}}$[$s$: she, $o$: everything], $Agr_{\text{has}}$[$s$: she] |
| 4 | $I_{37}$ | PF | mary/N has seen/V everyone/N who john/N was eating/V. |
| | | LF | $\theta_{\text{seen}}$[$s$: mary, $o$: everyone who john was eating], $Agr_{\text{has}}$[$s$: mary], $\theta_{\text{eating}}$[$s$: john, $o$: everyone], $Agr_{\text{was}}$[$s$: john] |
| | $I_{38}$ | PF | john/N has seen/V someone/N who was eating/V icecream/N. |
| | | LF | $\theta_{\text{seen}}$[$s$: john, $o$: someone who was eating icecream], $Agr_{\text{has}}$[$s$: john], $\theta_{\text{eating}}$[$s$: someone, $o$: icecream], $Agr_{\text{was}}$[$s$: someone] |
| | $I_{39}$ | PF | john/N has seen/V someone/N who was eaten/V. |
| | | LF | $\theta_{\text{seen}}$[$s$: john, $o$: someone who was eaten], $Agr_{\text{has}}$[$s$: john], $\theta_{\text{eaten}}$[$o$: someone], $Agr_{\text{was}}$[$s$: someone] |

Table 3.9: A presentation of the three batches of primary linguistic data (PLD) that the incremental acquisition procedure successively consumes (after having consumed the first batch of the PLD, which is listed in Table 3.2). All of the sentences listed here involve one degree (level) of embedding. Batches 2 presents sentences in which the embedded clause is a declarative (e.g. $I_{31}$) or an interrogative (e.g. $I_{34}$). Batch 3 and 4 present sentences in which the embedded clause is a (restrictive) relative clause. In the case of LF interface conditions that show how an embedded clause is an argument, the listed phrase is to be interpreted as a *multi-set* of phonological forms – e.g. in $I_{35}$, the *multi-set* of phonological forms $\{everything, that, mary, was, asked\}$ serves as an internal argument of the lexical verb *"told"*; thus the LF interface conditions do not include any information about the linear ordering of the words that make up the sentence, they only provide information that constrain *hierarchical* relations.

raised to structural subject position within the relative clause.[57] Note that for each of the two entries, the derivation yielded has the antecedent nominal phrase raised directly from an internal argument position (without first being raised to structural subject position). In the case of $I_{35}$, the antecedent is raised from within a passive-voice construction:

(6) *"Who has told John [everything that Mary was asked]?"*

whereas in the case of $I_{36}$ the antecedent is raised from within an active-voice construction:

(7) *"Was someone given [everything that she has eaten]?"*

The fourth batch of primary linguistic data (i.e. entries $I_{37}$ to $I_{39}$) also includes sentences with a relative clause serving as an argument, although now some of the entries have the antecedent nominal phrase *"someone"* raised to structural subject position within the relative clause, as is the case in $I_{38}$:

[57] Restrictive relative clauses typically appear immediately *after* an antecedent (pro)nominal phrase and serve the purpose of identifying or defining (i.e. restricting interpretations of) the antecedent.

(8) *“John has seen [someone who was eating icecream].”*

and $I_{39}$:

(9) *“John has seen [someone who was eaten].“*

**From Inputs to Outputs: Stepping Through the Procedure.**

The learner will successively process these three (new) batches of primary linguistic data over three consecutive runs of the *incremental acquisition procedure*, with the output lexicon from one run serving as the input lexicon for the next run, and the output of the third run of the *incremental acquisition procedure* constituting the final inferred lexicon (listed in Table 3.11).[58] A summary of the derivations yielded by each of the successively inferred lexicons is presented in Table 3.12. Both Table 3.11 and Table 3.12 are structured so as to clearly delineate the new lexical feature sequences that are learned after the learner processes each batch of the primary linguistic data. Valuations of the optimization metrics from each run of the *incremental acquisition procedure* are listed in Table 3.10. Let us now examine each of the three runs of the *incremental acquisition procedure* in greater detail.

To begin, the first run of the *incremental acquisition procedure* involves processing the second batch of the primary linguistic data, and the learner acquires (i.e. infers) a lexicon with 17 distinct lexical feature sequences – i.e. $\mathfrak{L}_1$ to $\mathfrak{L}_{17}$. The input lexicon, consisting of lexical feature sequences $\mathfrak{L}_1$ to $\mathfrak{L}_{16}$, is a strict subset of the (newly inferred) output lexicon, with only a single new lexical feature sequence, $\mathfrak{L}_{17}$, being learned. This new lexical feature sequence serves as a lexical head that projects into a CP; see Figures 3-11, 3-12, 3-13 and 3-14 for examples of derivations, yielded by the (newly) inferred lexicon, that use $\mathfrak{L}_{17}$ to satisfy entries in the second batch of the primary linguistic data (specifically $I_{29}$, $I_{31}$, $I_{32}$ and $I_{34}$ respectively).[59]

The second run of the procedure processes the third batch of the primary linguistic data, with the learner acquiring a lexicon with 19 distinct lexical feature sequences – i.e. $\mathfrak{L}_1$ to $\mathfrak{L}_{19}$. The input lexicon, being the lexicon output by the preceding run of the procedure, has 17 distinct lexical feature sequences – i.e. $\mathfrak{L}_1$ to $\mathfrak{L}_{17}$ – and is a strict subset of the output lexicon, with two new lexical feature sequences introduced: $\mathfrak{L}_{18}$ and $\mathfrak{L}_{19}$. $\mathfrak{L}_{18}$ serves as a lexical head that projects into a CP, and $\mathfrak{L}_{19}$, serving as a lexical head for the antecedent, will be raised to the specifier position of this CP. (see Figure 3-15 for an illustration of this)

The third and final run of the procedure processes the fourth and final batch of the primary linguistic data, with the learner acquiring a lexicon with 20 distinct lexical feature sequences – i.e. $\mathfrak{L}_1$ to $\mathfrak{L}_{20}$. The single new lexical feature sequence that has been learned – i.e. $\mathfrak{L}_{20}$ – is a lexical head that can play the role of an antecedent (of a relative clause) that

[58] N.b. if the incremental acquisition procedure were to be run on the “first batch”, the outcome would be no different than using the instantaneous acquisition procedure since the input lexicon for this run would be an empty lexicon; that is why here we pickup the learner’s acquisition trajectory starting with the second application of the procedure in detail.

[59] As was the case in §3.2.3, the illustrations of the derivations yielded by the lexicon are not depicted in specifier-head-complement ordering so that the (drawn) arrows can depict movement of maximal and minimal projections without obscuring other elements of the derivation; the reader should observe that if these derivations are drawn in the with the convention of specifiers preceding heads and heads preceding complements, then it becomes apparent that these derivations do accord with the linear (SVO) ordering of the phonological forms listed in the PF interface conditions they purport to satisfy. See Figure 3-13 for an illustration of a derivation that *happens to be drawn* so as to accord with the spec-head-comp ordering convention.

| Batch | Optimization Metrics | | | | |
|---|---|---|---|---|---|
| | (3.1) | (3.5) | (3.7) | (3.8) | (3.9) |
| 1 | 16 | 33 | 425 | 1 | 2 |
| 2 | 17 | 35 | 158 | 1 | 2 |
| 3 | 19 | 40 | 58 | 1 | 2 |
| 4 | 19 | 43 | 79 | 1 | 2 |

Table 3.10: Valuations of Optimization Metrics for each run of the Incremental Acquisition Procedure. Valuations of the optimization metrics for the first batch (of primary linguistic data) were determined in §3.2.3 using the *instantaneous acquisition procedure.* Valuations of these same metrics for processing batches 2-4 using the incremental acquisition procedure are introduced here. Metrics (3.1) and (3.5) count the number of total number of distinct lexical feature sequences and total number of syntactic features in the lexicon respectively; the values for these metrics never decrease as each additional batch of primary linguistic data is processed as each run of the incremental acquisition procedure only adds to the input lexicon. Metric (3.7) counts the total number of nodes across all derivations yielded by the inferred lexicon to satisfy the batch of primary linguistic data *being processed during that run*; consequently, the valuation of this metric fluctuates over runs of the incremental acquisition procedure as the number of entries in the batch and the size of the entries in the batch (i.e. number of overt phonological forms) fluctuates. Metrics (3.8) and (3.9) count the number of distinct selectional and licensing features appearing in the inferred lexicon. See §3.2.1 for further discussion of these metrics.

is first raised to structural subject position (i.e. Spec-TP) before within the relative clause before finally being raised to the specifier position of the CP that heads the relative clause; see Figure 3-17 for a derivation yielded by the (final) inferred lexicon that illustrates this arrangement.

Having outlined the process by which the learner incrementally acquires a lexicon by processing the primary linguistic data in batches, we will now turn to more carefully analyzing the lexicon that the learner has acquired, and the derivations it yields.

### Evaluation and Analysis of the Inferred Grammar

The (learned) lexicon is able to produce sentences with embedded complementizer phrases. Acquiring the capacity to yield such sentences requires a small addition to the grammar – i.e. the single (new) lexical feature sequence $\mathfrak{L}_{17}$.[60] This new lexical feature sequence had to be learned when processing the second batch of primary linguistic data because the (existing) three lexical feature sequences that project into complementizer phrases that head the main clause of a sentence – i.e. $\mathfrak{L}_3$, $\mathfrak{L}_4$ and $\mathfrak{L}_6$ – have the special feature $C$ that designates that their maximal projection is the head of the entire derivation, and therefore they cannot be used to head an embedded complementizer clause. Looking ahead, the acquisition of this single new lexical feature, acquired after processing the second batch, will endow the learner with the capacity to generate an unbounded set of syntactic structures. (See Pg. 178 for a proof of this)

[60] This lexical head projects into a CP that doesn't distinguish between embedded sentences and questions, as the system only considers the distinction between the categories $C_{Decl.}$ and $C_{Ques.}$ when interpreting the CP that heads the matrix clause.

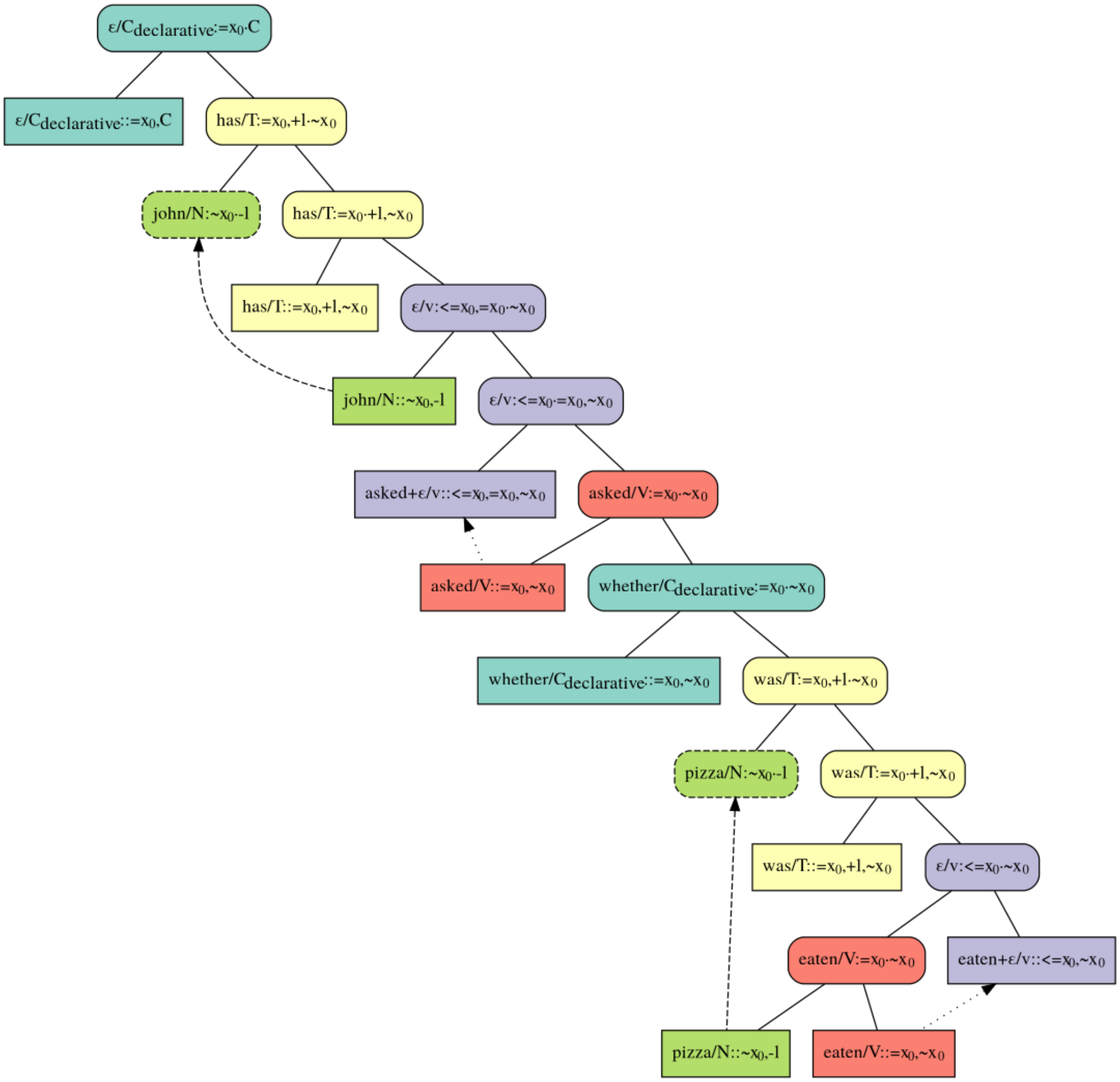


Figure 3-11: The derivation yielded by the final inferred lexicon to satisfy the LF and PF interface conditions in $I_{29}$. This derivation produces the sentence: *"John has asked whether pizza was eaten."*

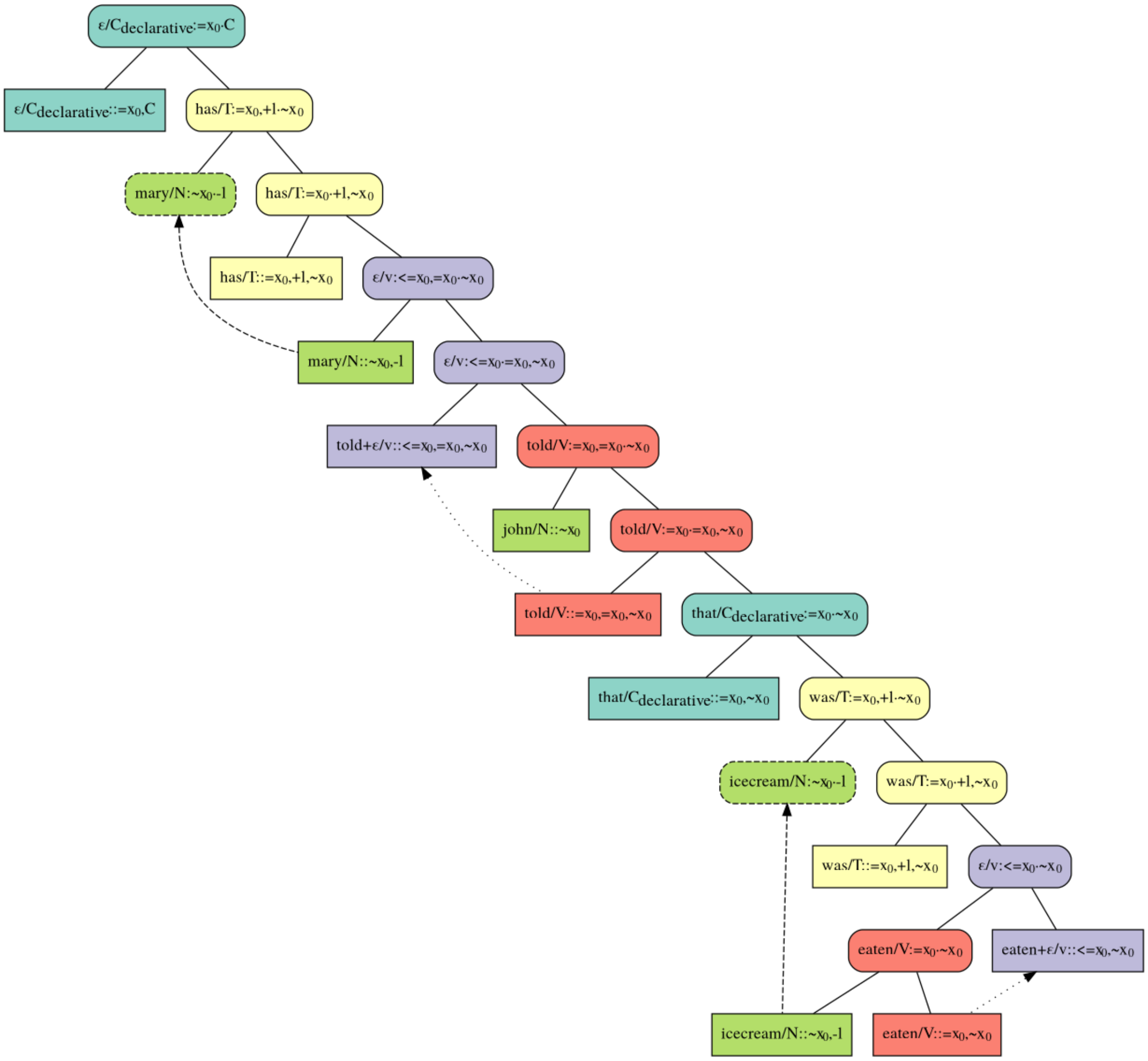


Figure 3-12: The derivation yielded by the final inferred lexicon to satisfy the LF and PF interface conditions in $I_{31}$. This derivation produces the sentence: *"Mary has told John that icecream was eaten."*

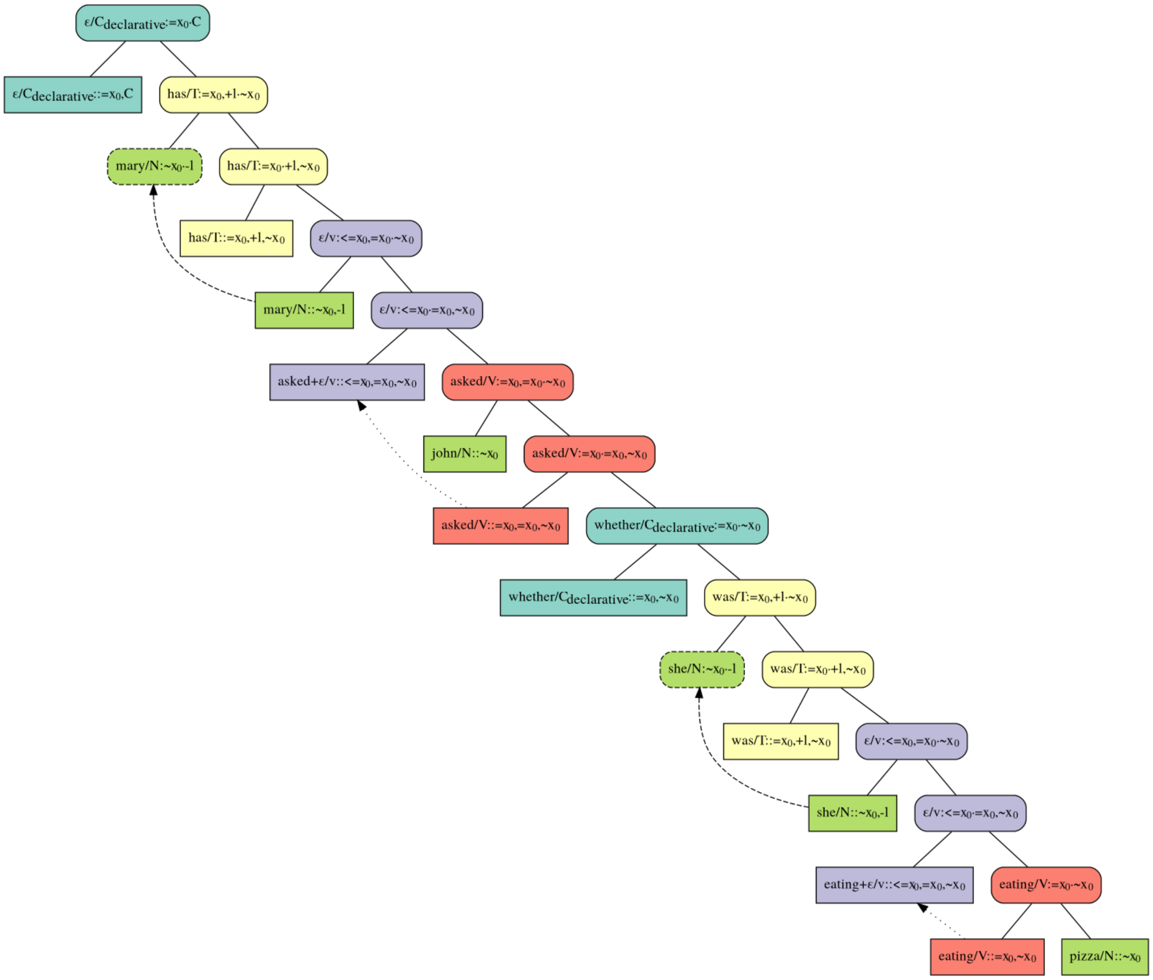

ε/C$_{declarative}$:=x$_0$·C
ε/C$_{declarative}$::=x$_0$,C
has/T:=x$_0$,+l·~x$_0$
mary/N:~x$_0$·-l
has/T:=x$_0$·+l,~x$_0$
has/T::=x$_0$,+l,~x$_0$
ε/v:<=x$_0$,=x$_0$·~x$_0$
mary/N::~x$_0$,-l
ε/v:<=x$_0$·=x$_0$,~x$_0$
asked+ε/v::<=x$_0$,=x$_0$,~x$_0$
asked/V:=x$_0$,=x$_0$·~x$_0$
john/N::~x$_0$
asked/V:=x$_0$·=x$_0$,~x$_0$
asked/V::=x$_0$,=x$_0$,~x$_0$
whether/C$_{declarative}$:=x$_0$·~x$_0$
whether/C$_{declarative}$::=x$_0$,~x$_0$
was/T:=x$_0$,+l·~x$_0$
she/N:~x$_0$·-l
was/T:=x$_0$·+l,~x$_0$
was/T::=x$_0$,+l,~x$_0$
ε/v:<=x$_0$,=x$_0$·~x$_0$
she/N::~x$_0$,-l
ε/v:<=x$_0$·=x$_0$,~x$_0$
eating+ε/v::<=x$_0$,=x$_0$,~x$_0$
eating/V:=x$_0$·~x$_0$
eating/V::=x$_0$,~x$_0$
pizza/N::~x$_0$

Figure 3-13: The derivation yielded by the final inferred lexicon to satisfy the LF and PF interface conditions in $I_{32}$. This derivation produces the sentence: *"Mary has asked John whether she was eating pizza."*

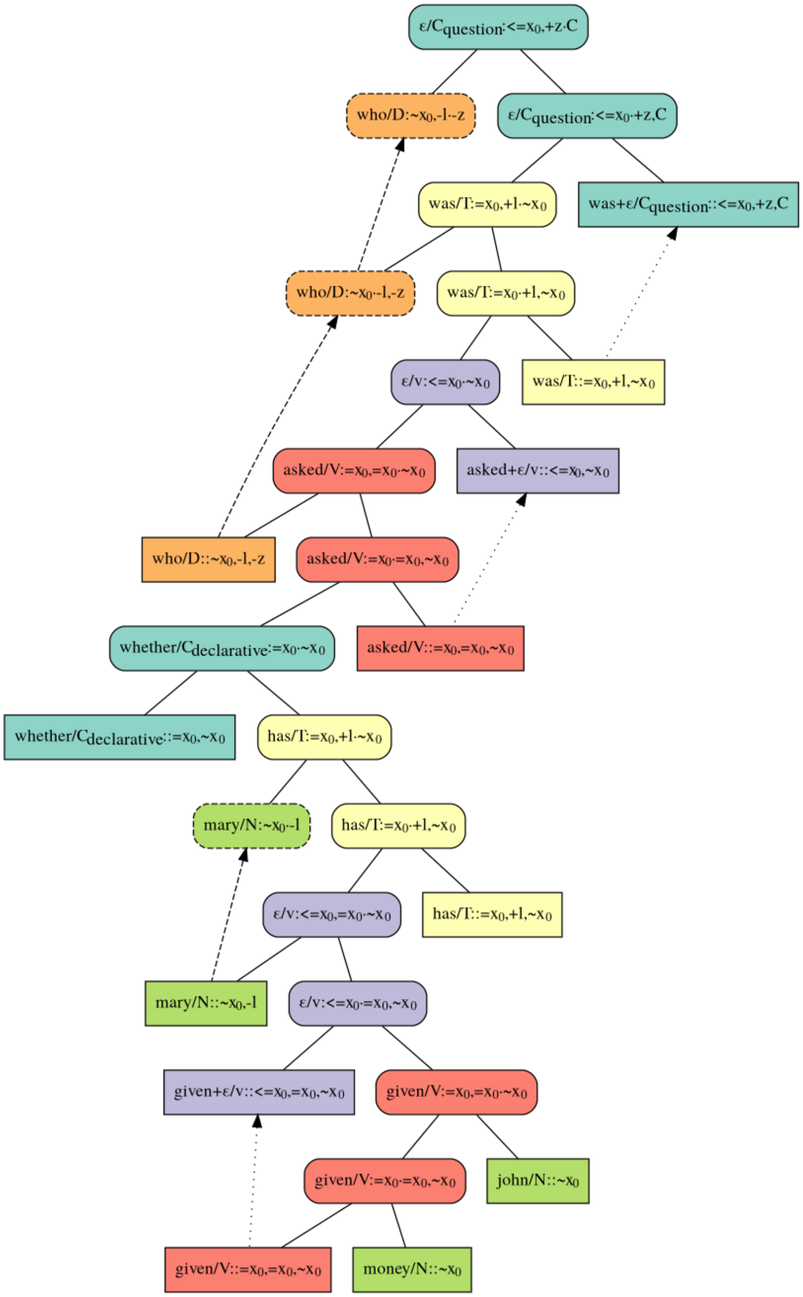


Figure 3-14: The derivation yielded by the final inferred lexicon to satisfy the LF and PF interface conditions in $I_{34}$. This derivation produces the interrogative: *“Who was asked whether Mary has given John money?”*

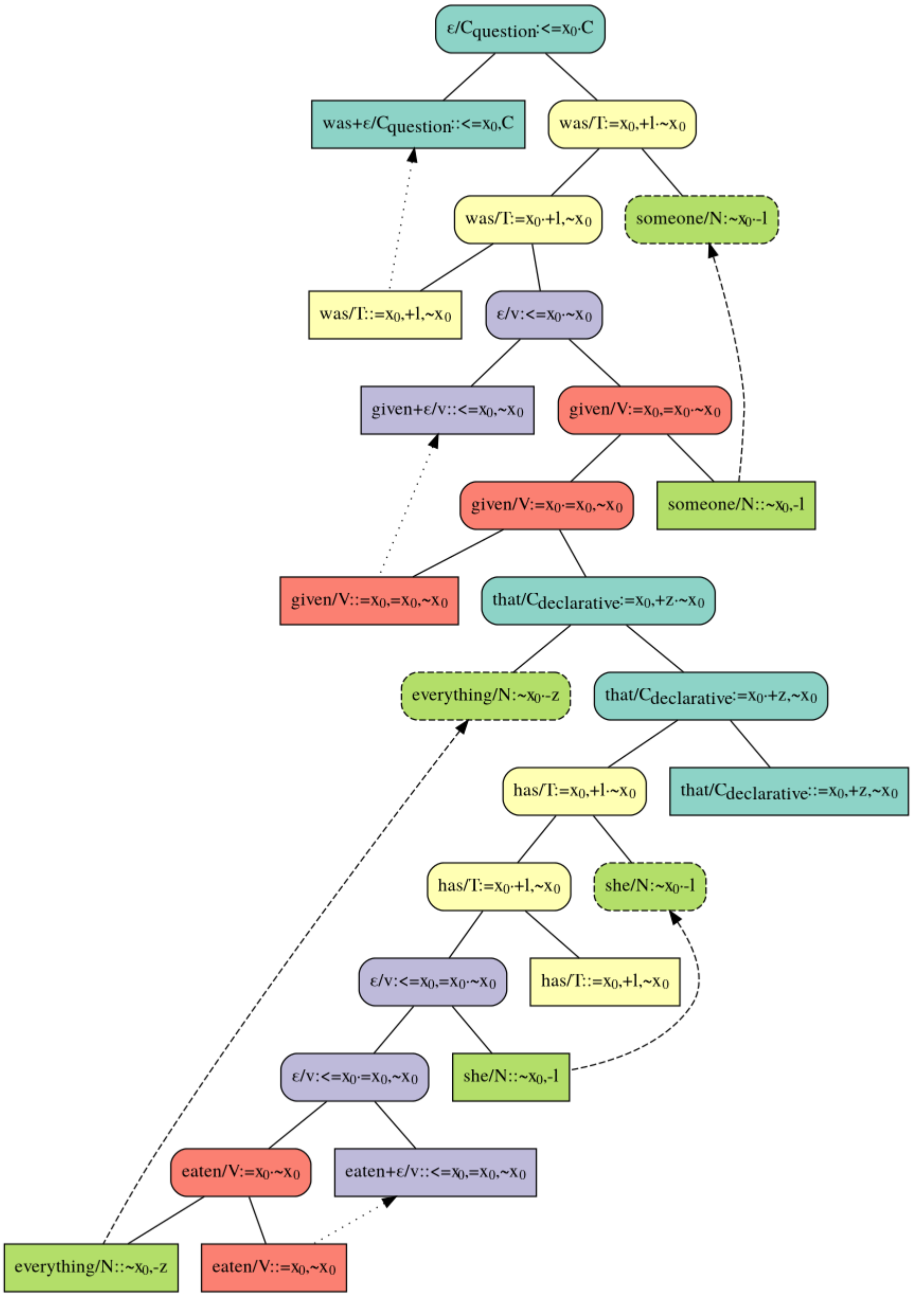

ε/C$_{question}$:<=x$_0$·C
was+ε/C$_{question}$::<=x$_0$,C
was/T:=x$_0$,+l·~x$_0$
was/T:=x$_0$·+l,~x$_0$
someone/N:~x$_0$·-l
was/T::=x$_0$,+l,~x$_0$
ε/v:<=x$_0$·~x$_0$
given+ε/v::<=x$_0$,~x$_0$
given/V:=x$_0$,=x$_0$·~x$_0$
given/V:=x$_0$·=x$_0$,~x$_0$
someone/N::~x$_0$,-l
given/V::=x$_0$,=x$_0$,~x$_0$
that/C$_{declarative}$:=x$_0$,+z·~x$_0$
everything/N:~x$_0$·-z
that/C$_{declarative}$:=x$_0$·+z,~x$_0$
has/T:=x$_0$,+l·~x$_0$
that/C$_{declarative}$::=x$_0$,+z,~x$_0$
has/T:=x$_0$·+l,~x$_0$
she/N:~x$_0$·-l
ε/v:<=x$_0$,=x$_0$·~x$_0$
has/T::=x$_0$,+l,~x$_0$
ε/v:<=x$_0$·=x$_0$,~x$_0$
she/N::~x$_0$,-l
eaten/V:=x$_0$·~x$_0$
eaten+ε/v::<=x$_0$,=x$_0$,~x$_0$
everything/N::~x$_0$,-z
eaten/V::=x$_0$,~x$_0$

Figure 3-15: The derivation yielded by the final inferred lexicon to satisfy the LF and PF interface conditions in $I_{36}$. This derivation produces the interrogative: *"Was someone given everything that she has eaten?"*

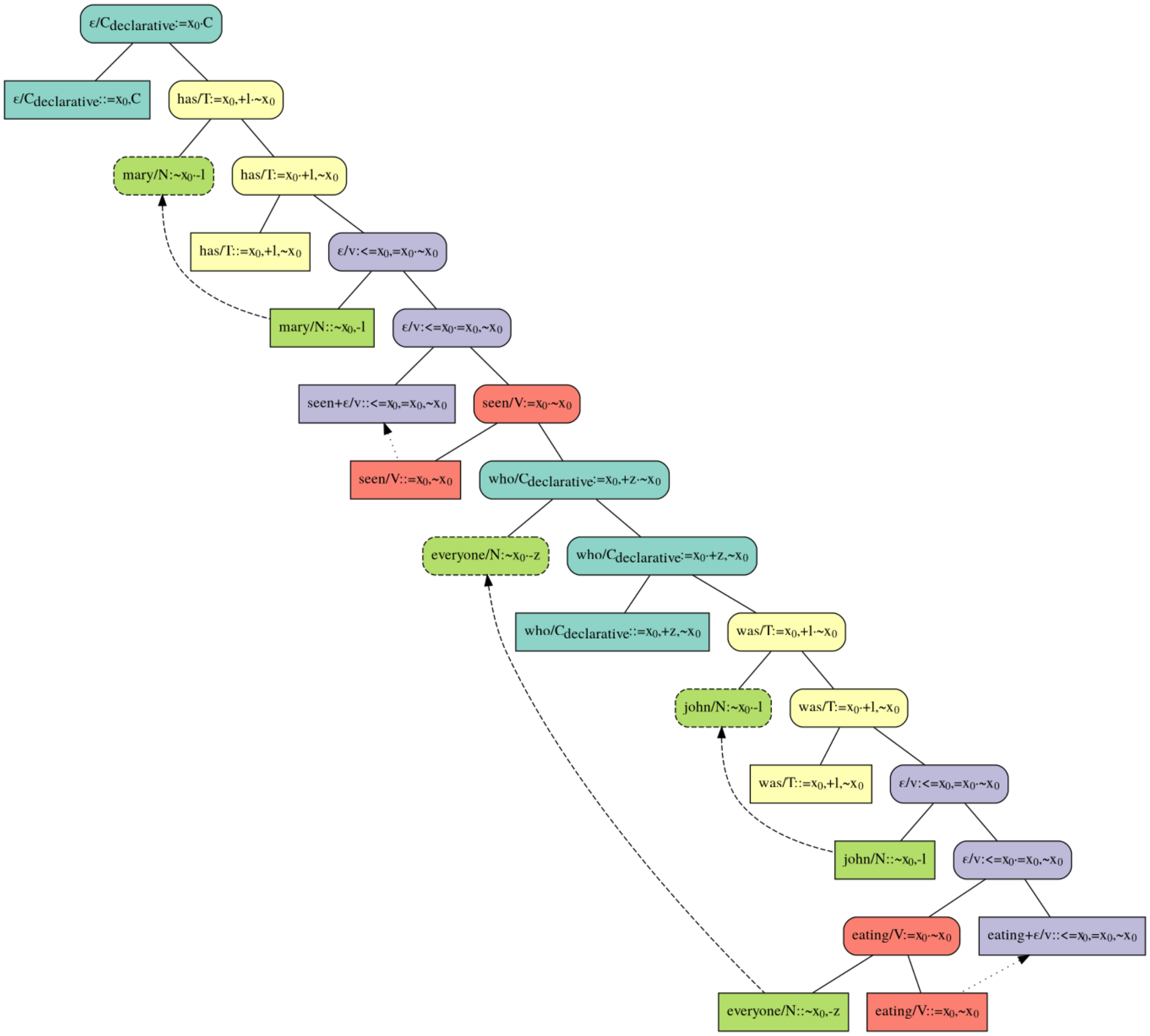

ε/$C_{declarative}$:=$x_0$·C
ε/$C_{declarative}$::=$x_0$,C
has/T:=$x_0$,+l·~$x_0$
mary/N:~$x_0$·-l
has/T:=$x_0$·+l,~$x_0$
has/T::=$x_0$,+l,~$x_0$
ε/v:<=$x_0$,=$x_0$·~$x_0$
mary/N::~$x_0$,-l
ε/v:<=$x_0$·=$x_0$,~$x_0$
seen+ε/v::<=$x_0$,=$x_0$,~$x_0$
seen/V:=$x_0$·~$x_0$
seen/V::=$x_0$,~$x_0$
who/$C_{declarative}$:=$x_0$,+z·~$x_0$
everyone/N:~$x_0$·-z
who/$C_{declarative}$:=$x_0$·+z,~$x_0$
who/$C_{declarative}$::=$x_0$,+z,~$x_0$
was/T:=$x_0$,+l·~$x_0$
john/N:~$x_0$·-l
was/T:=$x_0$·+l,~$x_0$
was/T::=$x_0$,+l,~$x_0$
ε/v:<=$x_0$,=$x_0$·~$x_0$
john/N::~$x_0$,-l
ε/v:<=$x_0$·=$x_0$,~$x_0$
eating/V:=$x_0$·~$x_0$
eating+ε/v::<=$x_0$,=$x_0$,~$x_0$
everyone/N::~$x_0$,-z
eating/V::=$x_0$,~$x_0$

Figure 3-16: The derivation yielded by the final inferred lexicon to satisfy the LF and PF interface conditions in $I_{37}$. This derivation produces the sentence: *"Mary has seen everyone who John was eating."*

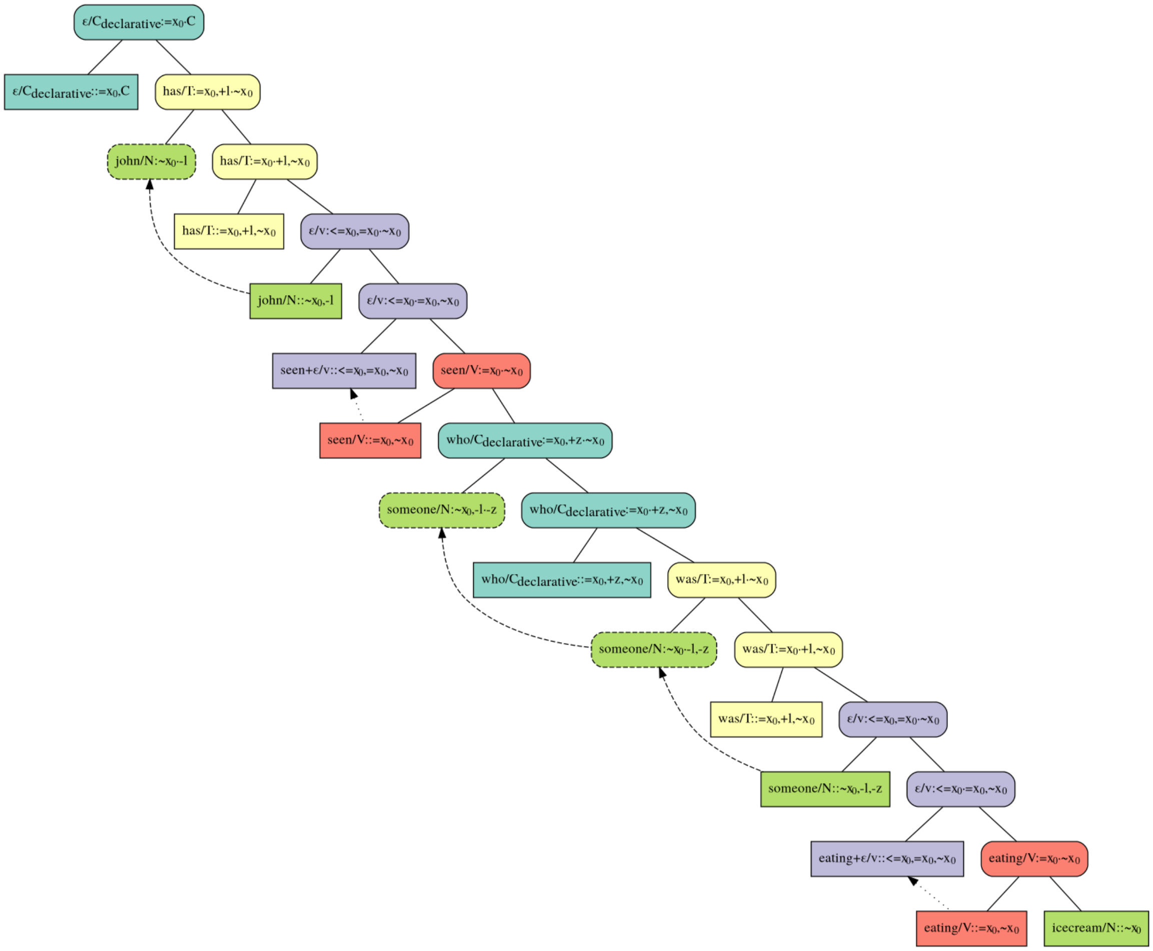


Figure 3-17: The derivation yielded by the final inferred lexicon to satisfy the LF and PF interface conditions in $I_{38}$. This derivation produces the sentence: *"John has seen someone who was eating icecream."*

| ID | Category | Features | Phonological Forms | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | what | who | ***that*** | ***whether*** | has | was | ***she*** | everything | someone | icecream | money | mary | pizza | john | nothing | boy | story | given | told | asked | known | ***seen*** | asking | eating | eaten | to | ***everyone*** | sleeping | slept | the | a | $\epsilon$ |
| $\mathfrak{L}_1$ | $V$ | $= x_0, \sim x_0$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | × | × | × | × | × | × | · | · | · | · | · | · | · |
| $\mathfrak{L}_2$ | $V$ | $= x_0, = x_0, \sim x_0$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | × | × | × | · | · | · | · | · | · | · | · | · | · | · | · |
| $\mathfrak{L}_3$ | $C_{decl.}$ | $= x_0, C$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | × |
| $\mathfrak{L}_4$ | $C_{ques.}$ | $<= x_0, +z, C$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | × |
| $\mathfrak{L}_5$ | $v$ | $<= x_0, \sim x_0$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | × |
| $\mathfrak{L}_6$ | $C_{ques.}$ | $<= x_0, C$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | × |
| $\mathfrak{L}_7$ | $v$ | $<= x_0, = x_0, \sim x_0$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | × |
| $\mathfrak{L}_8$ | $P$ | $= x_0, \sim x_0$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | × | · | · | · | · | · | · |
| $\mathfrak{L}_9$ | $D$ | $= x_0, \sim x_0, -l$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | × | × | · |
| $\mathfrak{L}_{10}$ | $D$ | $= x_0, \sim x_0$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | × | × | · |
| $\mathfrak{L}_{11}$ | $D$ | $\sim x_0, -z$ | × | × | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · |
| $\mathfrak{L}_{12}$ | $D$ | $\sim x_0, -l, -z$ | × | × | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · |
| $\mathfrak{L}_{13}$ | $T$ | $= x_0, +l, \sim x_0$ | · | · | · | · | × | × | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · |
| $\mathfrak{L}_{14}$ | $V$ | $\sim x_0$ | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | × | × | · | · | · |
| $\mathfrak{L}_{15}$ | $N$ | $\sim x_0, -l$ | · | · | · | · | · | · | × | × | × | × | × | × | × | × | × | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · |
| $\mathfrak{L}_{16}$ | $N$ | $\sim x_0$ | · | · | · | · | · | · | · | × | × | × | × | × | × | × | × | × | × | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · |
| $\mathfrak{L}_{17}$ | $C_{decl.}$ | $= x_0, \sim x_0$ | · | · | × | × | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · |
| $\mathfrak{L}_{18}$ | $C_{decl.}$ | $= x_0, +z, \sim x_0$ | · | × | × | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · |
| $\mathfrak{L}_{19}$ | $N$ | $\sim x_0, -z$ | · | · | · | · | · | · | · | × | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | × | · | · | · | · | · |
| $\mathfrak{L}_{20}$ | $N$ | $\sim x_0, -l, -z$ | · | · | · | · | · | · | · | · | × | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · | · |

Table 3.11: A *factored* view of the final inferred lexicon that was produced by first processing the PLD listed in Table 3.2 using the *instantaneous acquisition procedure*, and then processing the PLD listed in Table 3.9 using the *incremental acquisition procedure*. Each row indicates the phonological forms that are paired with the listed category and features in the lexicon. The columns have been seriated (using the hamming distance metric) so as to visually group together similar entries. The rows are divided by horizontal lines to indicate which lexical features sequences were inferred from which batch of the PLD: $\mathfrak{L}_1 - \mathfrak{L}_{16}$ for the first batch (i.e. $I_0$ to $I_{28}$); $\mathfrak{L}_{17}$ for the second batch (i.e. $I_{29}$ to $I_{34}$); $\mathfrak{L}_{18} - \mathfrak{L}_{19}$ for the third batch (i.e. $I_{35}$ to $I_{36}$); $\mathfrak{L}_{20}$ for the fourth (and final) batch (i.e. $I_{37}$ to $I_{39}$). Notably, $\mathfrak{L}_{17}$ and $\mathfrak{L}_{18}$ code for complementizers that can serve as the head of arguments, thereby enabling the inferred lexicon to generate sentences with embedded clauses – i.e. $\mathfrak{L}_{17}$ enables embedded sentences and $\mathfrak{L}_{18}$ enables embedded relative clauses. There are 31 distinct overt phonological forms in the lexicon. As compared with the lexicon inferred by the *instantaneous acquisition procedure* (listed in Table 3.3), this new lexicon introduces five new overt phonological forms: *"that"*, *"whether"* and *"she"* were learned when processing the second batch; *"seen"* and *"everyone"* were learned when processing the fourth batch.

The lexicon is also able to produce sentences with (restrictive) relative clauses. Specifically, the inferred grammar yields derivations that align with the Wh-Movement Analysis for restrictive relative clauses in which the antecedent originates within the relative clause in an argument position, thereby establishing a relation with the predicate within the relative clause, and is then subsequently raised to the specifier position of the complementizer phrase heading the relative clause, thereby achieving the proper word ordering (i.e. the antecedent immediately precedes the remainder of the relative clause). To ground this analysis, let us take a look at the derivation yielded by the lexicon to satisfy the interface condition listed in entry $I_{36}$ of the Table 3.9, as illustrated in Figure 3-15. This derivation is externalized as the sentence: *"Was someone given everything that she has eaten?"* The antecedent "everything" originates as the complement of the lexical verb "eaten" inside the relative clause, before being raised to the specifier position of the projection of the lexical head "that".

| ID | Category | Features | Input Sentence | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | $I_{29}$ | $I_{30}$ | $I_{31}$ | $I_{32}$ | $I_{33}$ | $I_{34}$ | $I_{35}$ | $I_{36}$ | $I_{37}$ | $I_{38}$ | $I_{39}$ |
| $\mathfrak{L}_1$ | $V$ | $= x_0, \sim x_0$ | 2 | 1 | 1 | 1 | 1 | · | · | 1 | 2 | 2 | 2 |
| $\mathfrak{L}_2$ | $V$ | $= x_0, = x_0, \sim x_0$ | · | 1 | 1 | 1 | 1 | 2 | 2 | 1 | · | · | · |
| $\mathfrak{L}_3$ | $C_{Decl.}$ | $= x_0, C$ | 1 | 1 | 1 | 1 | · | · | · | · | 1 | 1 | 1 |
| $\mathfrak{L}_4$ | $C_{ques.}$ | $<= x_0, +z, C$ | · | · | · | · | 1 | 1 | 1 | · | · | · | · |
| $\mathfrak{L}_5$ | $v$ | $<= x_0, \sim x_0$ | 1 | 1 | 1 | · | · | 1 | 1 | 1 | · | · | 1 |
| $\mathfrak{L}_6$ | $C_{ques.}$ | $<= x_0, C$ | · | · | · | · | · | · | · | 1 | · | · | · |
| $\mathfrak{L}_7$ | $v$ | $<= x_0, = x_0, \sim x_0$ | 1 | 1 | 1 | 2 | 2 | 1 | · | 1 | 2 | 2 | 1 |
| $\mathfrak{L}_8$ | $P$ | $= x_0, \sim x_0$ | · | · | · | · | · | · | · | · | · | · | · |
| $\mathfrak{L}_9$ | $D$ | $= x_0, \sim x_0, -l$ | · | · | · | · | · | · | · | · | · | · | · |
| $\mathfrak{L}_{10}$ | $D$ | $= x_0, \sim x_0$ | · | · | · | · | · | · | · | · | · | · | · |
| $\mathfrak{L}_{11}$ | $D$ | $\sim x_0, -z$ | · | · | · | · | 1 | · | · | · | · | · | · |
| $\mathfrak{L}_{12}$ | $D$ | $\sim x_0, -l, -z$ | · | · | · | · | · | 1 | 1 | · | · | · | · |
| $\mathfrak{L}_{13}$ | $T$ | $= x_0, +l, \sim x_0$ | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| $\mathfrak{L}_{14}$ | $V$ | $\sim x_0$ | · | · | · | · | · | · | · | · | · | · | · |
| $\mathfrak{L}_{15}$ | $N$ | $\sim x_0, -l$ | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 2 | 2 | 1 | 1 |
| $\mathfrak{L}_{16}$ | $N$ | $\sim x_0$ | · | 1 | 1 | 2 | 1 | 2 | 1 | · | · | 1 | · |
| $\mathfrak{L}_{17}$ | $C_{decl.}$ | $= x_0, \sim x_0$ | 1 | 1 | 1 | 1 | 1 | 1 | · | · | · | · | · |
| $\mathfrak{L}_{18}$ | $C_{decl.}$ | $= x_0, +z, \sim x_0$ | · | · | · | · | · | · | 1 | 1 | 1 | 1 | 1 |
| $\mathfrak{L}_{19}$ | $N$ | $\sim x_0, -z$ | · | · | · | · | · | · | 1 | 1 | 1 | · | · |
| $\mathfrak{L}_{20}$ | $N$ | $\sim x_0, -l, -z$ | · | · | · | · | · | · | · | · | · | 1 | 1 |

Table 3.12: A summary of the derivations yielded by the final inferred lexicon (listed in Table 3.11) to satisfy the pairings of LF and PF interface conditions presented in batches 2-4 of the PLD (listed in Table 3.9). The derivations yielded to satisfy the interface conditions in each batch (of the PLD) use the new lexical feature sequences introduced by the acquisition procedure after processing that batch. Each of the derivations make use of the lexical feature sequences that were inferred when processing the first batch of the PLD – i.e. $I_0 - I_{28}$, both for constructing the matrix clause for the derivation, but also for constructing the embedded clause.

Although the lexicon is consistent in handling restrictive relative clauses in this way – i.e. the antecedent can always be identified as the specifier of the complementizer that heads the relative clause – this analysis does not align with the *Antecedent Raising Analysis* of restrictive relative clauses (Donati and Cecchetto, 2011) that many contemporary theories of minimalist syntax have adopted.[61] One explanation of this divergence in analysis of restrictive relative clauses is that the *incremental acquisition procedure* developed in this thesis is prone to *over-optimizing* – i.e. inferring the optimal lexicon that is compatible with the PLD leads the learner to acquire a lexicon in which the fewest number of new lexical feature sequences are introduced (beyond those lexical feature sequences already present in the input lexicon). The *Wh-Movement Analysis* only requires that three new lexical feature sequences ($\mathfrak{L}_{18}$, $\mathfrak{L}_{19}$, and $\mathfrak{L}_{20}$) be introduced. In contrast, the *Antecedent Raising Analysis* requires four new lexical feature sequences to be introduced: $\mathfrak{L}_{18}$, which codes for the complementizer that heads the relative clause; a covert nominal that can take $\mathfrak{L}_{18}$ as its complement, and that will then merge into the complement position of the projection of a determiner; variants of $\mathfrak{L}_{19}$ and $\mathfrak{L}_{20}$ (both of which coded for the antecedent) that are extended with

[61] See (Radford, 2016, Pgs. 398-430) for a comparison of the *Wh-Movement Analysis* and *Antecedent Raising Analysis* of (restrictive) relative clauses.

a licensee feature that enables them to undergo antecedent raising to the specifier position of the (aforementioned) nominal lexical feature sequence.[62] Thus, the learner opts for the *Wh-Movement Analysis*, as it requires fewer lexical feature sequences be introduced.

The inferred grammar can also yield novel sentences that employ the newly acquired phonological forms – e.g.:

(10) *"She has seen John."*

This sentence is produced by the following derivation:[63]

$$\left\{\frac{\epsilon_{C_{Decl.}}}{\mathfrak{L}_3}\left\{\frac{\mathit{has}}{\mathfrak{L}_{13}}\left\{\frac{\mathit{she}}{\mathfrak{L}_{15}}\left\{\frac{\epsilon_v}{\mathfrak{L}_7}\left\{\frac{\mathit{seen}}{\mathfrak{L}_1}\frac{\mathit{John}}{\mathfrak{L}_{16}}\right\}\right\}\right\}\right\}\right\}$$

The lexicon inferred by the *instantaneous acquisition procedure* was already able to yield derivations of this form, as can be seen by the use (exclusively) of lexical feature sequences that were already present in that lexicon; what is new is that after using the *incremental acquisition procedure* to process the remaining three batches of the PLD, the learner has acquired knowledge of several new phonological forms, and learned to associate them with the (aforementioned) previously learned lexical feature sequences.

The inferred lexicon is able to yield derivations in which a CP or a (restrictive) relative clauses can serve as an internal argument, so long as the embedded clause is not required to be undergo raising to structural subject position. Let us consider three examples of novel constructions the inferred lexicon can yield that illustrate this point. Firstly, the inferred grammar can yield constructions with a ditransitive verb in which the internal argument corresponding to the direct object is a relative clause – e.g.:

(11) *"John has told [everything that Mary has known] to someone."*

If we let the phrase *"everything that Mary has known"* be designated by $R_\alpha$, then this sentence is produced by the following derivation:

$$\left\{\frac{\epsilon_{C_{Decl.}}}{\mathfrak{L}_3}\left\{\frac{\mathit{has}}{\mathfrak{L}_{13}}\left\{\frac{\mathit{John}}{\mathfrak{L}_{15}}\left\{\frac{\epsilon_v}{\mathfrak{L}_7}\left\{R_\alpha^D\left\{\frac{\mathit{told}}{\mathfrak{L}_2}\left\{\frac{\mathit{to}}{\mathfrak{L}_8}\frac{\mathit{someone}}{\mathfrak{L}_{16}}\right\}\right\}\right\}\right\}\right\}\right\}\right\}$$

where

$$R_\alpha^D=\left\{\frac{\mathit{that}}{\mathfrak{L}_{18}}\left\{\frac{\mathit{has}}{\mathfrak{L}_{13}}\left\{\frac{\mathit{Mary}}{\mathfrak{L}_{15}}\left\{\frac{\epsilon_v}{\mathfrak{L}_7}\left\{\frac{\mathit{known}}{\mathfrak{L}_1}\frac{\mathit{everything}}{\mathfrak{L}_{19}}\right\}\right\}\right\}\right\}\right\}$$

Secondly, the inferred grammar can yield constructions with a ditransitive verb in which the internal argument corresponding to the indirect object is a relative clause – e.g.:

(12) *"John has asked [someone who has eaten nothing] a story."*

Letting the phrase *"someone who has eaten nothing"* be designated by $R_\beta$, this sentence is

[62] See (Radford, 2016, Pg. 420) for an example that illustrates the *Antecedent Raising Analysis* of restrictive relative clauses; note that this illustration corresponds to a slight refinement of the original analysis given in (Donati and Cecchetto, 2011).

[63] Note that, as in §3.2.3, derivations presented inline are in a bracketed form that only shows the external merge operations; the full derivation can be recovered from a specification of the external merge operations in the derivation because, in the minimalist grammar formalism, internal merge operations happen automatically and deterministically in the course of a derivation – see §2.2 for a discussion of how this is a consequence of the Shortest Movement Condition (SMC).

produced by the following derivation:

$$\left\{\frac{\epsilon_{C_{Decl.}}}{\mathfrak{L}_3}\left\{\frac{\textit{has}}{\mathfrak{L}_{13}}\left\{\frac{\textit{John}}{\mathfrak{L}_{15}}\left\{\frac{\epsilon_v}{\mathfrak{L}_7}\left\{R^D_\beta\left\{\frac{\textit{asked}}{\mathfrak{L}_2}\left\{\frac{\textit{a}}{\mathfrak{L}_{10}}\frac{\textit{story}}{\mathfrak{L}_{16}}\right\}\right\}\right\}\right\}\right\}\right\}\right\}$$

where

$$R^D_\beta=\left\{\frac{\textit{who}}{\mathfrak{L}_{18}}\left\{\frac{\textit{has}}{\mathfrak{L}_{13}}\left\{\frac{\textit{someone}}{\mathfrak{L}_{20}}\left\{\frac{\epsilon_v}{\mathfrak{L}_7}\left\{\frac{\textit{eaten}}{\mathfrak{L}_1}\frac{\textit{nothing}}{\mathfrak{L}_{16}}\right\}\right\}\right\}\right\}\right\}$$

Thirdly, the inferred grammar can also yield sentences that have both an embedded complementizer clause and an embedded (restrictive) relative clause as (internal) arguments – e.g.:

(13) *"Mary has given [everything that John was given] to [someone who was sleeping]."*

Designating the phrase *"everything that John was given"* by $R_1$ and the phrase *"someone who was sleeping"* by $R_2$, this sentence is produced by the following derivation:

$$\left\{\frac{\epsilon_{C_{Decl}}}{\mathfrak{L}_3}\left\{\frac{\textit{has}}{\mathfrak{L}_{13}}\left\{\frac{\epsilon_v}{\mathfrak{L}_5}\left\{R^D_1\left\{\frac{\textit{given}}{\mathfrak{L}_2}\left\{\frac{\textit{to}}{\mathfrak{L}_8}R^D_2\right\}\right\}\right\}\right\}\right\}\right\}$$

where

$$R^D_1=\left\{\frac{\textit{that}}{\mathfrak{L}_{18}}\left\{\frac{\textit{was}}{\mathfrak{L}_{13}}\left\{\frac{\epsilon_v}{\mathfrak{L}_5}\left\{\frac{\textit{John}}{\mathfrak{L}_{15}}\left\{\frac{\textit{given}}{\mathfrak{L}_2}\frac{\textit{everything}}{\mathfrak{L}_{19}}\right\}\right\}\right\}\right\}\right\}$$

and

$$R^D_2=\left\{\frac{\textit{who}}{\mathfrak{L}_{18}}\left\{\frac{\textit{was}}{\mathfrak{L}_{13}}\left\{\frac{\textit{someone}}{\mathfrak{L}_{20}}\left\{\frac{\epsilon_v}{\mathfrak{L}_7}\frac{\textit{sleeping}}{\mathfrak{L}_{14}}\right\}\right\}\right\}\right\}$$

Having considered several cases of novel syntactic structures that the (optimal) inferred lexicon can yield, let us now briefly consider two cases of *ungrammatical* forms that the inferred lexicon will not generate.[64]

(i) The inferred lexicon will not yield a derivation in which the complementizer that heads an embedded clause triggers head-movement:[65]

    (14) * *"She has known [that has John eaten pizza]."*

    To see this, observe that neither of the lexical feature sequences that serve to connect an embedded clause to the matrix clause (i.e. the lexical feature sequences $\mathfrak{L}_{17}$ and $\mathfrak{L}_{18}$ that are associated with the phonological form "that") have the (selector) feature $<= x_0$ that triggers T-to-C head-movement (c.f. $\mathfrak{L}_4$ or $\mathfrak{L}_6$ which do have the feature $<= x_0$).

(ii) The inferred lexicon will not yield a derivation in which an argument in the embedded clause is (directly) raised to agree with the tense-marker in the matrix clause, so that the following interrogative cannot be produced by the lexicon:

    (15) * *"Has John told Mary [that money was given to ~~John~~]?"*

[64] Note that these two cases pertain specifically to derivations with embedded clauses and are distinct from the three cases described in §3.2.3.

[65] Cf. the grammatical expression *"She has known [that John has eaten pizza]."*

To see this, first observe that no lexical feature sequence has two instances of the licensee feature $-l$ that enables subject-raising to Spec-TP; this rules out the possibility of an argument originating from within the embedded clause being raised to first check the licensor of the embedded TP and subsequently being raised (again) to check the licensor of the TP in the matrix clause. Next, observe that the single lexical feature sequence for tense markers, $\mathfrak{L}_{13}$ (associated with phonological forms "has" and "was"), has a licensor feature, $+l$, that triggers internal merge and requires an appropriate licensee be raised and check the licensor. Consequently, for the argument "John" (associated with $\mathfrak{L}_{15}$) in the embedded clause to be raised to Spec-TP in the matrix clause, it would have to form either a crossing or nested (i.e. overlapping) movement operation with respect to the phrase "money" that lands at Spec-TP within the embedded clause; such an arrangement is ruled out by the Shortest Movement Condition, which requires that the embedded-TP must be able to determine which internal argument will be raised to its specifier position based on the label of the licensee feature alone, and this is impossible as the lexical feature sequences for "John" and "money" both have the same licensee feature $-l$.[66] (Note that "John" and "money" are both associated with the same lexical feature sequence, $\mathfrak{L}_{15}$.)

### Yielding an Unbounded Set of Syntactic Structures

One of the distinguishing characteristics of the *human language faculty* is the capacity to generate, from a finite set of words, a countably infinite set of (interpretable) hierarchical structures that pair meaning and sound. One can now prove that the inferred lexicon has this capacity, and that in particular, this lexicon can yield, for any $n > 0$, a derivation with Degree-$n$ embedding (i.e. a derivation with $n$ levels of embedding); consequently, there is no upper-bound on how many levels of embedding the structures the lexicon can yield may have. Importantly, the inferred lexicon was acquired from a small, finite set of sentences with at most one level of embedding, thereby demonstrating the capacity of the *incremental acquisition procedure* to generalize from a limited set of examples.

[66] See §2.2 for details about the Shortest Movement Condition.

**Proof-A: Embedded-Sentence Expansion.**
This is a proof by induction that shows that for any $n > 0$, the inferred lexicon can yield a derivation with Degree-$n$ embedding, thereby establishing that the inferred lexicon can generate an unbounded set of distinct syntactic structures. To begin, consider the sentence:

(16) *"A boy has told someone the story."*

This sentence can be generated from the inferred lexicon via the following derivation with Degree-0 embedding:

$$\left\{\frac{\epsilon_C}{\mathfrak{L}_3}\left\{\frac{\mathit{has}}{\mathfrak{L}_{13}}\left\{\left\{\frac{\mathit{a}}{\mathfrak{L}_9}\frac{\mathit{boy}}{\mathfrak{L}_{16}}\right\}\left\{\frac{\epsilon_v}{\mathfrak{L}_7}\left\{\frac{\mathit{someone}}{\mathfrak{L}_{16}}\left\{\frac{\mathit{told}}{\mathfrak{L}_2}\left\{\frac{\mathit{the}}{\mathfrak{L}_{10}}\frac{\mathit{story}}{\mathfrak{L}_{16}}\right\}\right\}\right\}\right\}\right\}\right\}\right\}$$

This derivation with Degree-0 embedding constitutes the base case of this inductive proof.

Next, consider the following rule for expanding this sentence: replace the argument *"the story"* with the (complementizer) phrase *"that a boy has told someone the story"*. Formally, this rule can be written as:

$$\left\{\frac{\mathit{the}}{\mathfrak{L}_{10}}\frac{\mathit{story}}{\mathfrak{L}_{16}}\right\}\rightarrow\left\{\frac{\mathit{that}}{\mathfrak{L}_{17}}\left\{\frac{\mathit{has}}{\mathfrak{L}_{13}}\left\{\left\{\frac{\mathit{a}}{\mathfrak{L}_9}\frac{\mathit{boy}}{\mathfrak{L}_{16}}\right\}\left\{\frac{\epsilon_v}{\mathfrak{L}_7}\left\{\frac{\mathit{someone}}{\mathfrak{L}_{16}}\left\{\frac{\mathit{told}}{\mathfrak{L}_2}\left\{\frac{\mathit{the}}{\mathfrak{L}_{10}}\frac{\mathit{story}}{\mathfrak{L}_{16}}\right\}\right\}\right\}\right\}\right\}\right\}\right\}$$

Importantly, this (substitution) rule can be applied because the features exposed by the head of the phrase on the left hand side – (i.e. $\sim x_0$ in $\mathfrak{L}_{10}$) are the same as the features exposed by the head of the phrase on the right hand side (i.e. $\sim x_0$ in $\mathfrak{L}_{17}$). Applying this rule to the base-case derivation yields the following derivation with Degree-1 embedding:

$$\left\{\frac{\epsilon_C}{\mathfrak{L}_3}\left\{\frac{\mathit{has}}{\mathfrak{L}_{13}}\left\{\left\{\frac{\mathit{a}}{\mathfrak{L}_9}\frac{\mathit{boy}}{\mathfrak{L}_{16}}\right\}\left\{\frac{\epsilon_v}{\mathfrak{L}_7}\left\{\frac{\mathit{someone}}{\mathfrak{L}_{16}}\left\{\frac{\mathit{told}}{\mathfrak{L}_2}\left\{\frac{\mathit{that}}{\mathfrak{L}_{17}}\left\{\frac{\mathit{has}}{\mathfrak{L}_{13}}\left\{\left\{\frac{\mathit{a}}{\mathfrak{L}_9}\frac{\mathit{boy}}{\mathfrak{L}_{16}}\right\}\left\{\frac{\epsilon_v}{\mathfrak{L}_7}\left\{\frac{\mathit{someone}}{\mathfrak{L}_{16}}\left\{\frac{\mathit{told}}{\mathfrak{L}_2}\left\{\frac{\mathit{the}}{\mathfrak{L}_{10}}\frac{\mathit{story}}{\mathfrak{L}_{16}}\right\}\right\}\right\}\right\}\right\}\right\}\right\}\right\}\right\}\right\}\right\}\right\}\right\}$$

which in turn produces the sentence:

(17) *"A boy has told someone [that a boy has told someone the story]."*

Successive applications of this rule yields derivations with increasing degrees of embedding – e.g. when applied twice in succession to the base case derivation, the rule produces the following derivation with Degree-2 embedding:

$$\left\{\frac{\epsilon_C}{\mathfrak{L}_3}\left\{\frac{\mathit{has}}{\mathfrak{L}_{13}}\left\{\left\{\frac{\mathit{a}}{\mathfrak{L}_9}\frac{\mathit{boy}}{\mathfrak{L}_{16}}\right\}\left\{\frac{\epsilon_v}{\mathfrak{L}_7}\left\{\frac{\mathit{someone}}{\mathfrak{L}_{16}}\left\{\frac{\mathit{told}}{\mathfrak{L}_2}\left\{\frac{\mathit{that}}{\mathfrak{L}_{17}}\left\{\frac{\mathit{has}}{\mathfrak{L}_{13}}\left\{\left\{\frac{\mathit{a}}{\mathfrak{L}_9}\frac{\mathit{boy}}{\mathfrak{L}_{16}}\right\}\left\{\frac{\epsilon_v}{\mathfrak{L}_7}\left\{\frac{\mathit{someone}}{\mathfrak{L}_{16}}\left\{\frac{\mathit{told}}{\mathfrak{L}_2}\left\{\frac{\mathit{that}}{\mathfrak{L}_{17}}\left\{\frac{\mathit{has}}{\mathfrak{L}_{13}}\left\{\left\{\frac{\mathit{a}}{\mathfrak{L}_9}\frac{\mathit{boy}}{\mathfrak{L}_{16}}\right\}\left\{\frac{\epsilon_v}{\mathfrak{L}_7}\left\{\frac{\mathit{someone}}{\mathfrak{L}_{16}}\left\{\frac{\mathit{told}}{\mathfrak{L}_2}\left\{\frac{\mathit{the}}{\mathfrak{L}_{10}}\frac{\mathit{story}}{\mathfrak{L}_{16}}\right\}\right\}\right\}\right\}\right\}\right\}\right\}\right\}\right\}\right\}\right\}\right\}\right\}\right\}\right\}\right\}\right\}\right\}\right\}$$

which in turn produces the sentence:

(18) *"A boy has told someone [that a boy has told someone [that a boy has told someone the story]]."*

This leads to the inductive step: given a derivation with Degree-$n$ embedding that was produced by the repeated application of the expansion rule $n$ times to the base-case derivation (with Degree-0 embedding), applying the rule once more to this derivation with Degree-$n$ embedding yields a derivation with Degree-$(n+1)$ embedding; this is the case because the expansion rule has the property that the left hand side of the rule is found within the right hand side of the rule, so that the rule may be applied repeatedly, each time replacing the argument *"the story"* with the phrase *"that a boy has told someone the story"* and thereby increasing the degree of embedding by one.

It follows that for any $n > 0$, the expansion rule can be applied repeatedly $n$ times to the base case derivation to produce a derivation with Degree-$n$ embedding, and thus the inferred lexicon can generate an unbounded set of distinct syntactic structures. See Table 3.13 for a summary of the derivations described in this proof as well as a formula for generating a derivation with Degree-$n$ embedding for any $n > 0$.

**Proof-B: Relative Clause Expansion.**
This is (another) proof by induction that shows that for any $n > 0$, the inferred lexicon can yield a derivation with Degree-$n$ embedding – in this proof, the expansion rule will center around substituting simple arguments with more complex relative clauses. To begin, consider the sentence:

(19) *"Mary has given everything to John."*

This sentence can be generated from the inferred lexicon via the following derivation with Degree-0 embedding:

$$\left\{\frac{\epsilon_C}{\mathfrak{L}_3}\left\{\frac{\mathit{has}}{\mathfrak{L}_{13}}\left\{\frac{\mathit{Mary}}{\mathfrak{L}_{15}}\left\{\frac{\epsilon_v}{\mathfrak{L}_7}\left\{\frac{\mathit{everything}}{\mathfrak{L}_{16}}\left\{\frac{\mathit{given}}{\mathfrak{L}_2}\left\{\frac{\mathit{to}}{\mathfrak{L}_8}\frac{\mathit{John}}{\mathfrak{L}_{16}}\right\}\right\}\right\}\right\}\right\}\right\}\right\}$$

This derivation with Degree-0 embedding constitutes the base case of this inductive proof.

Next, consider the following rule for expanding this sentence: replace the argument *"John"* with the relative clause *"someone who has given everything to John"*. Formally, this rule can be written as:

$$\frac{\mathit{John}}{\mathfrak{L}_{16}} \rightarrow \left\{\frac{\mathit{who}}{\mathfrak{L}_{18}}\left\{\frac{\mathit{has}}{\mathfrak{L}_{13}}\left\{\frac{\mathit{someone}}{\mathfrak{L}_{20}}\left\{\frac{\epsilon_v}{\mathfrak{L}_7}\left\{\frac{\mathit{everything}}{\mathfrak{L}_{16}}\left\{\frac{\mathit{given}}{\mathfrak{L}_2}\left\{\frac{\mathit{to}}{\mathfrak{L}_8}\frac{\mathit{John}}{\mathfrak{L}_{16}}\right\}\right\}\right\}\right\}\right\}\right\}\right\}$$

Importantly, this (substitution) rule can be applied because the feature exposed by the head of the phrase on the left hand side – (i.e. $\sim x_0$ in $\mathfrak{L}_{16}$) are the same as the feature exposed by the head of the phrase on the right hand side (i.e. $\sim x_0$ in $\mathfrak{L}_{18}$). Applying this rule to the base-case derivation (with Degree-0 embedding) yields the following derivation with Degree-1 embedding:

$$\left\{\frac{\epsilon_C}{\mathfrak{L}_3}\left\{\frac{has}{\mathfrak{L}_{13}}\left\{\frac{Mary}{\mathfrak{L}_{15}}\left\{\frac{\epsilon_v}{\mathfrak{L}_7}\left\{\frac{everything}{\mathfrak{L}_{16}}\left\{\frac{given}{\mathfrak{L}_2}\left\{\frac{to}{\mathfrak{L}_8}\left\{\frac{who}{\mathfrak{L}_{18}}\left\{\frac{has}{\mathfrak{L}_{13}}\left\{\frac{someone}{\mathfrak{L}_{20}}\left\{\frac{\epsilon_v}{\mathfrak{L}_7}\left\{\frac{everything}{\mathfrak{L}_{16}}\left\{\frac{given}{\mathfrak{L}_2}\left\{\frac{to}{\mathfrak{L}_8}\frac{John}{\mathfrak{L}_{16}}\right\}\right\}\right\}\right\}\right\}\right\}\right\}\right\}\right\}\right\}\right\}\right\}\right\}\right\}$$

which in turn produces the sentence:

(20) *“Mary has given everything to [someone who has given everything to John].”*

Successive applications of the expansion rule yield derivations with increasing degrees of embedding – e.g. applying the rule twice in succession to the base case derivation yields a derivation with Degree-2 embedding that produces the sentence:

(21) *“Mary has given everything to [someone who has given everything to [someone who has given everything to John]].”*

and applying the rule three times in succession to the base case derivation yields a derivation with Degree-3 embedding that produces the sentence:

(22) *“Mary has given everything to [someone who has given everything to [someone who has given everything to [someone who has given everything to John]]].”*

This leads us to the inductive step: given a derivation with Degree-$n$ embedding that was produced by the repeated application of the expansion rule $n$ times to the base-case derivation (with Degree-0 embedding), applying the expansion rule once more to this derivation with Degree-$n$ embedding will yield a derivation with Degree-$(n+1)$ embedding; to see why this is the case, note that the expansion rule has the property that the left hand side of the rule is found within the right hand side of the rule, so that the rule may be applied repeatedly, each time replacing the argument *“John”* with the relative clause *“someone who has given everything to John”* and thereby increasing the degree of embedding by one.

It follows that, for any $n > 0$, the expansion rule can be applied repeatedly $n$ times to the base-case derivation to produce a derivation with Degree-$n$ embedding, and thus the set of structures the inferred lexicon can generate is unbounded. See Table 3.13 for a summary of the derivations described in this proof as well as a formula for generating a derivation with Degree-$n$ embedding for any $n > 0$.

| ID | Category | Features | Expansions | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | $A_0$ | $A_1$ | $A_2$ | $A_n$ | $B_0$ | $B_1$ | $B_2$ | $B_n$ |
| $\mathfrak{L}_1$ | $V$ | $=x_0, \sim x_0$ | · | · | · | · | · | · | · | · |
| $\mathfrak{L}_2$ | $V$ | $=x_0, =x_0, \sim x_0$ | 1 | 2 | 3 | $n+1$ | 1 | 2 | 3 | $n+1$ |
| $\mathfrak{L}_3$ | $C_{Decl.}$ | $=x_0, C$ | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| $\mathfrak{L}_4$ | $C_{ques.}$ | $<=x_0, +z, C$ | · | · | · | · | · | · | · | · |
| $\mathfrak{L}_5$ | $v$ | $<=x_0, \sim x_0$ | · | · | · | · | · | · | · | · |
| $\mathfrak{L}_6$ | $C_{ques.}$ | $<=x_0, C$ | · | · | · | · | · | · | · | · |
| $\mathfrak{L}_7$ | $v$ | $<=x_0, =x_0, \sim x_0$ | 1 | 2 | 3 | $n+1$ | 1 | 2 | 3 | $n+1$ |
| $\mathfrak{L}_8$ | $P$ | $=x_0, \sim x_0$ | · | · | · | · | 1 | 2 | 3 | $n+1$ |
| $\mathfrak{L}_9$ | $D$ | $=x_0, \sim x_0, -l$ | 1 | 2 | 3 | $n+1$ | · | · | · | · |
| $\mathfrak{L}_{10}$ | $D$ | $=x_0, \sim x_0$ | 1 | 1 | 1 | 1 | · | · | · | · |
| $\mathfrak{L}_{11}$ | $D$ | $\sim x_0, -z$ | · | · | · | · | · | · | · | · |
| $\mathfrak{L}_{12}$ | $D$ | $\sim x_0, -l, -z$ | · | · | · | · | · | · | · | · |
| $\mathfrak{L}_{13}$ | $T$ | $=x_0, +l, \sim x_0$ | 1 | 2 | 3 | $n+1$ | 1 | 2 | 3 | $n+1$ |
| $\mathfrak{L}_{14}$ | $V$ | $\sim x_0$ | · | · | · | · | · | · | · | · |
| $\mathfrak{L}_{15}$ | $N$ | $\sim x_0, -l$ | · | · | · | · | 1 | 1 | 1 | 1 |
| $\mathfrak{L}_{16}$ | $N$ | $\sim x_0$ | 3 | 5 | 7 | $2n+3$ | 2 | 3 | 4 | $n+2$ |
| $\mathfrak{L}_{17}$ | $C_{decl.}$ | $=x_0, \sim x_0$ | · | 1 | 2 | $n$ | · | · | · | · |
| $\mathfrak{L}_{18}$ | $C_{decl.}$ | $=x_0, +z, \sim x_0$ | · | · | · | · | · | 1 | 2 | $n$ |
| $\mathfrak{L}_{19}$ | $N$ | $\sim x_0, -z$ | · | · | · | · | · | · | · | · |
| $\mathfrak{L}_{20}$ | $N$ | $\sim x_0, -l, -z$ | · | · | · | · | · | 1 | 2 | $n$ |

Table 3.13: A summary of the unbounded set of derivations that the final inferred lexicon is able to yield. Derivations $A_i$ correspond to Degree-$i$ embedding constructions as described in Example-A (see Pg. 178). Derivations $B_i$ correspond to Degree-$i$ embedding constructions as described in Example-B (see Pg. 179). Notably, the columns for $A_n$ and $B_n$ provide formulas for generating derivations with Degree-$n$ embedding.

### 3.3.3 Summary

By extending the *instantaneous acquisition procedure* to take an input lexicon and requiring that the inferred lexicon be a superset of the input lexicon, the *incremental acquisition procedure* can incrementally learn a lexicon by processing a PLD in small batches, thereby obviating the psychologically implausible requirement that the learner to store the entire PLD in memory, as was the case for the *instantaneous acquisition procedure*.

Notably, the *incremental acquisition procedure* enables the learner to incrementally acquire a grammar from an arbitrarily large PLD as only a single batch of the PLD needs to be modeled at a time; indeed, the PLD may even be infinite in size if it is presented to the learner as a stream. This enabled us to carry out computational experiments involving the processing of a PLD that was larger than the PLD processed in §3.2.3 and that included sentences with a single level of (structural) embedding in the form of both embedded sentences and embedded (restrictive) relative clauses.

The *incremental acquisition procedure* also enables the computationally tractable inference of larger grammars. To see this, let us assume that on each run of the *incremental acquisition procedure* after the first, the learner already has knowledge of most of the lexical entries needed to process the next batch of PLD (this knowledge being supplied in the input lexicon that was the output of the prior run of the procedure); then on each run of the procedure, the majority of the lexicon model is hard-coded with the supplied (input) lexicon, with only a small allocation of lexical feature sequences remaining unconstrained so that they are available to be used for new syntactic roles as required to produce derivations that the supplied lexicon could not yield.

Importantly, the computational experiment in §3.3.2 demonstrated that the *incremental acquisition procedure* can generalize from a small set of expressions with at most one level of embedding and infer an (optimal) lexicon that can (provably) generate a countably infinite set of interpretable hierarchical structures that pair meaning with sound, including structures with unbounded depth of (structural) embedding; remarkably, this inferred lexicon only has four additional lexical feature sequences beyond the sixteen in the lexicon inferred in §3.2.3.

Finally, the *incremental acquisition procedure* integrates language acquisition and language processing. When processing the first batch of the PLD, if the procedure is not supplied with an initial lexicon, then the procedure is functionally equivalent to the instantaneous acquisition procedure; as more batches of the PLD are processed over successive runs of the incremental acquisition procedure, in the limit, when the learner can already parse the next PLD batch with the (up to then) acquired lexicon, the procedure in effect becomes the procedure for parsing each entry in the PLD batch. This transition from acquisition to parsing is possible because both center on the same underlying axiomatization of minimalist syntax (detailed in §2.3), with the difference between the two being the degree to which the lexicon model is hard-coded with the lexicon that the learner has acquired up until that point.

# Chapter 4

# Towards Explanatory Adequacy

This thesis has shown how a logic engine may be used to drive procedures for parsing and acquisition. The SMT-models constructed by the procedures for parsing and acquisition are flexible by design and may be extended by adding or removing axioms and the uninterpreted functions and sorts that they constrain. Furthermore, the procedures for parsing and acquisition only employ a small subset of the capabilities of modern SMT-solvers, which support a variety of background theories – e.g. theories for real and integer arithmetic, arrays, algebraic datatypes and strings – that we can leverage when extending the models developed in this thesis.[1][2][3] It is therefore worthwhile to briefly consider, as this final chapter does, several ways in which the SMT-models underlying parsing and acquisition may be extended, by leveraging the diverse capabilities of modern SMT-solvers, so as to better align them with the goal of modeling a linguistic theory that meets the condition of *explanatory adequacy*, which is the gold standard for a scientific theory of language acquisition. Let us now briefly review what meeting the condition of explanatory adequacy requires, and what motivates the meeting of this condition.

A linguistic theory meets the condition of explanatory adequacy if it can explain how to select a *descriptively adequate* grammar based on the *primary linguistic data* (PLD) from which a child acquires knowledge of language.[4] A grammar for a particular language is said to be descriptively adequate if it can provide, for each sentence in that language, a structural representation from which the (linguistic) facts about that sentence may be systematically recovered,[5] and a theory of linguistics is descriptively adequate if it provides a descriptively adequate grammar for each natural language.[6] Note that descriptive adequacy is a necessary

[1] See (Bjørner et al., 2018) for a survey of the background theories supported by the Z3 SMT-solver and other capabilities thereof.

[2] See (Dutertre and de Moura, 2006) for details of the linear-arithmetic solver underlying Z3.

[3] See (Fichte et al., 2020) for an analysis of the impact hardware and algorithm improvements have had on performance increases for SAT-solvers (which make up the core of modern SMT-solvers).

[4] See (Chomsky, 1965, Pg. 25): *"A linguistic theory must contain a definition of 'grammar,' that is, a specification of the class of potential grammars. We may, correspondingly, say that a linguistic theory is* descriptively adequate *if it makes a descriptively adequate grammar available for each natural language. ... To the extent that a linguistic theory succeeds in selecting a descriptively adequate grammar on the basis of primary linguistic data, we can say that it meets the condition of explanatory adequacy."*

[5] In particular, a descriptively adequate grammar has strong generative capacity. See (Berwick, 1984) for a review of strong generative capacity.

[6] See (Chomsky, 1965, Pg. 24): *"A grammar can be regarded as a theory of a language; it is descriptively adequate to the extent that it correctly describes the intrinsic competence of the idealized native speaker. The structural descriptions assigned to sentences by the grammar, the distinctions that it makes between well-formed and deviant, and so on, must, for descriptive adequacy, correspond to the linguistic intuition of the*

condition for a theory of linguistics to meet the condition of explanatory adequacy, but it is not sufficient, as there may be multiple (distinct) grammars that meet the condition of descriptive adequacy for a particular natural language. A theory of linguistics that meets the condition of explanatory adequacy is able to select, by means of an evaluation function, which of these descriptively adequate grammars is acquired by the child language learner.

A theory of linguistics that meets the condition of explanatory adequacy is desirable because it provides a solution to the *logical problem of language acquisition*, sometimes referred to as *Plato's Problem*, which asks how a child is able to acquire language pursuant to a *Poverty of the Stimulus*[7] In (Chomsky, 1986), Chomsky addresses *Plato's Problem* by proposing that a child language learner is innately endowed with Universal Grammar (UG), a set of constraints on the space of possible generative grammars that compensates for the lack of discriminating evidence in the PLD. Hence, if the procedure for acquisition can be extended to meet the condition of explanatory adequacy, then we can use it to run (computational) experiments by carefully setting up particular input lexicon and PLD, and understanding how the axioms that underlie $C_{HLF}$ and that encode and the principles of learning impact the acquired target grammar. Additionally, we can use the procedure for acquisition in computational experiments that help us understanding how language might be acquired in a staged manner – i.e. are there specific axioms (encoding particular rules or learning principles) that appear or disappear in particular stages of a child's language acquisition trajectory?[8]

To this end, the extensions outlined in this chapter form a roadmap for extending the models developed in this thesis for language processing and language acquisition so that they may more faithfully implement a theory of linguistics that meets the condition of explanatory adequacy. (Note that the extensions may be considered independently of one another, and each extension does not assume that any of the other extensions have been implemented.)

## 4.1 Incorporating a Theory of Phases

This section outlines how the SMT-model of the derivation and the SMT-model of the lexicon (detailed in §2.3) may be modified so as to incorporate a theory of phases (Chomsky, 2001; Chomsky, 2008) that supports a Probe/Goal-system employing feature valuation.[9]

*native speaker. … a linguistic theory is descriptively adequate if it makes a descriptively adequate grammar available for each natural language."*

[7] Poverty of the Stimulus is the name given to the observation that children reliably acquires a language from input data that does not in of itself contain sufficient information for the child to derive knowledge of that language. (Chomsky, 1986; Chomsky, 2013a) See (Legate and Yang, 2002; Crain and Pietroski, 2002; Berwick et al., 2011; Berwick et al., 2013; Lasnik and Lidz, 2017) for detailed reviews of arguments from the Poverty of the Stimulus and challenges thereto.

[8] With respect to engineering-oriented applications, meeting the condition of descriptive adequacy translates to building a better parser that avoids overgenerating while accounting for the facts of language, so that we can confidently use the PF and LF representations recovered from a derivation in downstream applications. Meeting the condition of explanatory adequacy enables the procedure for acquisition to infer a generative grammar for a language from a relatively small corpus without needing a treebank of derivations; notably, the grammar learned by the acquisition procedure could be used to generate a treebank of minimalist derivations.

[9] This approach is informed by the work presented in (Chesi, 2004; Chesi, 2007) on formally incorporating phases into the Minimalist Grammar framework.

We will begin by briefly reviewing the Probe/Goal-system and then consider what modifications to the models of the lexicon and the derivation are necessary for implementing the Probe/Goal-system.

The Probe/Goal-system involves a probe with an unvalued feature that searches for a goal with a valued feature, with the requirement that the types of the unvalued feature match the type of the valued feature; the goal then moves to merge with the probe (i.e. the probe is the target of movement), with the (unvalued) feature on the probe being assigned the value of the (valued) feature on the goal; finally, the feature on the probe that has been valued is "deleted" in so far as it is not available for interpretation at the LF interface or in later syntactic computations, although it may be interpreted at the PF interface. The valued and unvalued features (in the Probe/Goal-system) correspond to the interpretable and the uninterpretable features (in checking theory) respectively. Note that uninterpretable features do not enter into a derivation already valued as in checking theory; rather, they enter into a derivation unvalued, and must eventually be valued and then "deleted" by an interpretable feature. In this way, the Probe/Goal based system allows for feature valuation, which reduces the size of the lexicon by reducing the number of distinct feature matrices present within the lexicon.

Implementing the Probe/Goal-system requires that the lexicon model be modified by: (i) associating with each lexical item a "set" of "value-able" features that are either valued (i.e. being assigned a feature label) or unvalued, with each feature being associated with a "feature type" (e.g. person, number, case, gender, tense), and (ii) associating each lexical item with a "set" of c(onstituent)-selectional features, each of which is associated with a "feature label" and is either a "selector" or a "selectee." An unvalued feature may be assigned the value of a valued feature if they both have the same "feature type." Notably, these modifications entail that lexical feature sequences are no longer sequentially ordered.

The derivation model is then modified as follows. First, we take external merge to be driven by the argument (of the merge operation) that projects having a c-selectional "selector" feature that selects a "selectee" feature in the other argument subject to the constraint that the selector and selectee features both have the same feature label. Whenever external merge takes place, the system tries, for each unvalued feature in the argument that projects (i.e. the probe), to assign it the value of one of the "valued" features in the selected argument (i.e. the goal) if the unvalued and valued feature both have the same feature type;[10] importantly, the system will require that an unvalued feature may *only* be valued once in the derivation and that it *must* be valued as soon as such an opportunity arises. Next, we take all instances of internal merge to be driven by the projection of a lexical head – i.e. a probe – with an unvalued feature identifying a subordinate node in the derivation – i.e. a goal – the head of which is associated with a valued feature that may value the unvalued feature (thus requiring that the valued and unvalued feature both have the same feature type); as in the case of external merge, when a goal is raised to merge with the probe that identified it, the system tries, for each unvalued feature in the argument that projects (i.e. the probe), to assign it the value of one of the "valued" features in the selected argument (i.e. the goal) if the unvalued and valued feature both have the same "feature type". In this way, there are no more "licensing features" as are found in the conventional Minimalist Grammar framework. Finally, the constraints derived from interface conditions (that are added to the derivation model) are modified such that: (i) the LF interface can only read a value-able feature if it was already valued when it entered into the derivation (as from the

[10] Note that this implies that a goal may value more than one unvalued feature on a probe.

perspective of the LF interface, "unvalued" features are deleted as soon as they are valued within a derivation); (ii) the PF interface can read a value-able feature so long as it has been valued at some point in the derivation.

Let us next review the theory of phases and consider what modifications need to be made to the model of the derivation so as to incorporate this theory.

The theory of phases centers on the assertion that derivations are (cyclically) built in *phases*, with a phase made up of a domain headed by either a complementizer ($C$) or a (transitive) light-verb ($v$). After a phase is constructed, the complement of the phase head is sent out to the interfaces, and is unavailable for later syntactic computations in the derivation. The Phase Impenetrability Condition (PIC) stipulates that the domain of a phase cannot be accessed by (syntactic) operations outside of the phase domain; such operations can only access the phase head or specifiers of the projection of the phase head. Notably, this entails that a probe cannot search past a phase-head, thereby restricting movement operations to be local within a phase domain. The PIC entails that $C_{HLF}$ (i.e. the computational system underlying the Human Language Faculty) operates with a limited memory capacity for representing structures in so far as it places strict limits on how much of a derivation it constructs at a time.

The PIC may be implemented by adding to the derivation model a unary uninterpreted function, $\phi_{\mathbb{N}}$, with signature:

$$\phi_{\mathbb{N}} : \mathbb{N} \to \mathbb{N} \tag{4.1}$$

that is constrained (by model axioms) so as to associate each node, $x \in \mathbb{N}$, in a derivation with the closest phase head that dominates it.[11] Specifically, if the parent of $x$ is a phase head – i.e. $p(x)$ associates with the category $C$ or the category (transitive) $v$ – then $p(x)$ is the closest phase head that dominates $x$, so that $\phi_{\mathbb{N}}(x) = h(p(x))$; alternatively, if the parent of $x$ is not a phase head, then the closest phase head that dominates the parent of $x$ is also the closest phase head that dominates $x$, so that $\phi_{\mathbb{N}}(x) = \phi_{\mathbb{N}}(p(x))$. (Note that the root node of the derivation must be handled as a special case as $p(R_{\mathbb{N}}) = \bot$.) Then the PIC may be expressed as an axiom that requires that in an instance of movement from a source (i.e. goal) node $s \in (\mathbb{L}_{\mathbb{N}} \cup D)$ to a target node $t \in (\mathbb{L}_{\mathbb{N}} \cup D)$ (that merges with the probe that matched the goal), if $\phi_{\mathbb{N}}(s) \neq \phi_{\mathbb{N}}(t)$ then $t$ must merge with the projection of the closest phase head that dominates $s$ (i.e. the phase boundary for the domain in which $s$ is located), so that

$$(h(p(t)) = \phi_{\mathbb{N}}(s)) \wedge (\beta_{\mathbb{N}}(h(p(t))) \in \{\mathfrak{c}_C, \mathfrak{c}_{v_{transitive}}\}) \tag{4.2}$$

(Note that if $\phi_{\mathbb{N}}(s) = \phi_{\mathbb{N}}(t)$, then the source and target of movement are both in the same phase domain.)

## 4.2 Extending the LF and PF Interfaces

This section considers how the diversity of representations that the LF and PF interfaces can process (as detailed in §2.3.3) may be expanded.

To begin, we observe that the LF interface conditions were all translated into constraints (i.e. axioms expressed as SMT-formulae) that required a (local) structural relation between

[11] If the node $x$ does not participate in the derivation – i.e. $h(x) = \bot$ then we let $\phi_{\mathbb{N}}(x) = \bot$.

two constituents be established by an instance of merge (e.g. between a predicate and an argument). However there are a number of syntactic phenomenon that require a different structural relation, c(onstituent)-command, be established between two constituents within a syntactic structure.[12] The c-command relation is defined as follows: given two distinct nodes $x$ and $y$ in a syntactic (tree) structure, $x$ c-commands $y$ if and only if $x$ does not dominate $y$, $y$ does not dominate $x$, and the parent of $x$ dominates $y$.[13] C-command may be readily expressed using the components of the derivation model as follows:[14]

$$\text{c-command}(x, y) = (\neg d(x, y) \wedge \neg d(y, x) \wedge d(p(x), y)) \vee (\neg d^{\star}(x, y) \wedge \neg d^{\star}(y, x) \wedge d^{\star}(p(x), y)) \quad (4.3)$$

If we include LF interface conditions that translate into constraints expressed using the c-command relation, then the following additional syntactic phenomena may be modeled:[15]

1. Co-reference. An anaphor (e.g. a reflexive pronoun) that is bound to (i.e. co-referenced with) an antecedent must be c-commanded by that antecedent (corresponding to Principle A of Binding Theory); note that if the LF interface is extended to handle co-reference relations, axioms encoding Principle B and C of Binding Theory[16] should also be added, in the form of axioms expressed as SMT-formulae, to the derivation model.

2. Quantifier Scope. A pronoun bound to a quantifier (e.g. "all," "some," "every") must fall within the scope of the quantifier that it is bound to; as the constituents in a syntactic structure that are in the scope of a quantifier phrase are those that are c-commanded by the quantifier phrase, it follows that a pronoun bound to a quantifier must be c-commanded by that quantifier.

3. Negative Polarity Items and scope interaction. A negation particle that licenses a negative polarity item (e.g. "all", "any", "a single") must c-command that negative polarity item.

Extending the LF interface in this way will both enable us to specify, as input to the acquisition procedure developed in this thesis, *primary linguistic data* that can include a more diverse range of sentences, and also enable a broader range of use cases for the procedure for parsing (developed in this thesis) by enabling quantifier scope, licensing of negative polarity items, and resolutions of co-reference to be (automatically) recovered from an inspection of the LF interface conditions satisfied by a derivation yielded by the parser (e.g. when only PF interface conditions, and not LF interface conditions, are stipulated as input to the parser).

Turning next to extensions of the PF interface, we observe that the axioms presented in §2.3.3 are intended to derive a specifier-head-complement (SHC) linear-order (in which

[12] The c-command relation was explicitly introduced in (Reinhart, 1976). See (Sportiche et al., 2013) for a review of the c-command relation.

[13] N.b. the c-command relation may be derived from sisterhood and immediate-dominance relations that are established by *merge*.

[14] Note that this c-command definition involves "domination" with respect to the derivation tree *or* the derived tree.

[15] Note that constraints derived from LF interface conditions involving the c-command relation can be expressed using the sorts and uninterpreted functions already present in the derivation model; in particular, these constraints may be expressed without requiring the introduction of any new sorts or uninterpreted functions in the derivation model, and without requiring any modification or addition to the axioms in the derivation model.

[16] See (Chomsky, 1981) for a presentation of Binding Theory.

the specifier precedes the head, and the head precedes its complement) over the phonological forms appearing in the derivation, which enables the parser to be used with an SVO (i.e. subject-verb-object) language such as English. However, many human languages (e.g. French, Hindi, Japanese) have an SOV (subject-object-verb) ordering that corresponds to a linearization procedure in which specifiers come before a head and its complement, and a complement comes before its head.[17] Notably, the difference between the two orderings is whether the head comes before or after its complement; within the (linguistic) theory of Principles and Parameters, this observation is captured by the "Head-Directionality" parameter.[18] The PF interface may be extended to handle both SVO and SOV languages as follows:

1. Add to the SMT-model of the lexicon[19] a boolean (free) variable, $\text{param}_{HD}$, that codes for the *Head-Directionality parameter*; the system models a *head-initial* or *head-final* language depending on whether $\text{param}_{HD}$ is valued True or False respectively. In the case of the procedure for parsing, the value of $\text{param}_{HD}$ may be set (via a model parameter supplied as input to the procedure) if we know whether the language of the expression being parsed is head-initial or head-final. In the case of the procedure for acquisition, the value of $\text{param}_{HD}$ may be left unspecified (as in the case of a child language learner who does not know beforehand whether its first language will be head-initial or head-final), and its value may be recovered from the (satisfiable) interpretation of the model identified by the SMT-solver.

2. Modify axioms (2.118), (2.119), (2.120) and (2.122) (listed in §2.3.3) by parameterizing them on $\text{param}_{HD}$ (via the introduction of conditional terms), such that, if $\text{param}_{HD}$ is valued True (because the language is head-initial), the axioms remain as they currently are (i.e. heads appear before their complement), and otherwise the axioms require that a head appears after its complement (because the language is head-final).

This extension not only enables us to use the procedure for parsing on a larger set of languages (by manually specifying the value of $\text{param}_{HD}$), but also presents an opportunity to run acquisition experiments in which try to learn how much primary linguistic data a child must consume before it "decides" the valuation of $\text{param}_{HD}$.[20] In this way, extending the PF interface to handle both SVO and SOV languages serves to increase the descriptive adequacy (and by extension, the explanatory adequacy) of the theory of linguistics implicitly modeled by the acquisition procedure.

[17] In fact, a majority of (documented) human languages have an SOV ordering (a total of 564 languages), with SVO being the second most popular ordering among human languages (with a total of 488 languages). (This information was obtained by searching for feature "81A" in the World Atlas of Language Structures (Dryer and Haspelmath, 2013).)

[18] The theory of Principles and Parameters asserts that the language acquisition device selects one of the possible human languages by setting the values of a finite set of parameters. See (Ayoun, 2005, Pg. 79) for a comprehensive review of proposed linguistic parameters – e.g. the *Null-Subject Parameter* as detailed in (Rizzi, 1982; Rizzi, 1986) and later reviewed in (Rizzi, 2011).

[19] We assume the *Borer-Chomsky hypothesis* (Borer, 1984), and consequently, take all parameters to be located within the lexicon.

[20] The approach taken by this extension may be adapted to incorporate other linguistic parameters into the models developed in this thesis, in which case the acquisition procedure might also be employed to carry out experiments that seek to understand the order in which the parameters are set by the language learner.

## 4.3 Curtailing Overgenerations

As we observed in §3.2 and §3.3, the lexical entries that make up the inferred lexicon may be productively combined in new ways, so as to generate derivations not produced in the course of processing the *primary linguistic data*. However, this capacity may also enable the learner to generate a derivation that yields (via SPELLOUT) an ungrammatical surface form or incorrectly establishes a pairing between logical form (LF) and phonological form (PF) representations. Such a derivation is referred to as an *overgeneration*; the production of an overgeneration is considered erroneous (with respect to meeting the condition of descriptive adequacy), and is to be avoided. To that end, this section considers how the *instantaneous* acquisition (presented in §3.2.2) can be modified so as to identify a grammar that both respects the optimization constraints employed by the acquisition procedure while seeking to reduce the set of overgenerations that the inferred lexicon may generate.

Let us now outline a particular kind of overgeneration that we seek to rule out. Some overgenerations may be ruled out when selecting lexical items from the lexicon – i.e. by prohibiting selection of lexical items that are not involved in a complete derivation that fully satisfies the specified interface conditions. However, even after selecting lexical entries from the lexicon, an overgeneration may still be produced: given a multi-set of lexical entries, $L$, that are all combined together, using *merge*, to generate a derivation that satisfies specified interface conditions, a derivation generated by $L$ that does not satisfy the specified interface conditions constitutes an overgeneration. E.g. suppose the following lexical entries have been selected from the lexicon:

$$\text{what}/N :: \sim x_0, -p \tag{4.4}$$

$$\text{has}/T :: = x_0, +q, \sim x_0 \tag{4.5}$$

$$\text{the}/D :: = x_0, \sim x_0, -q \tag{4.6}$$

$$\text{man}/N :: \sim x_0 \tag{4.7}$$

$$\text{eaten}/V :: = x_0, \sim x_0 \tag{4.8}$$

$$\epsilon/v :: <= x_0, = x_0, \sim x_0 \tag{4.9}$$

$$\epsilon/C_{ques.} :: <= x_0, +p, C \tag{4.10}$$

These lexical entries can be combined (via *merge*) to produce the following derivation that satisfies the interface conditions listed in entry $I_1$ of Table 2.4 and associates the (grammatical) surface form *"What has the man eaten?"* with its standard interpretation (i.e. that "the man" is the subject of "eaten" and agrees with "has", and that "what" is the object of "eaten"):

[$_{\text{CP}}$ what [$_{C'}$ has+$\epsilon_{\text{C}_{\text{ques.}}}$ [$_{TP}$ [$_{DP}$ the man] [$_{T'}$ ~~has~~ [$_{vP}$ ~~[$_{\text{DP}}$ the man ]~~ [$_{v'}$ eaten+$\epsilon_v$ [$_{VP}$ ~~eaten~~ ~~what~~]]]]]]] (4.11)

However, these lexical entries can also be combined to produce the following derivation that is similar to (2-2), but has the (initial) positions of the arguments "the man" and "what" swapped within the double VP-shell structure:

[$_{\text{CP}}$ what [$_{C'}$ has+$\epsilon_{\text{C}_{\text{ques.}}}$ [$_{TP}$ [$_{DP}$ the man] [$_{T'}$ ~~has~~ [$_{vP}$ ~~what~~ [$_{v'}$ eaten+$\epsilon_v$ [$_{VP}$ ~~eaten~~ ~~[$_{\text{DP}}$ the man ]~~ ]]]]]]] (4.12)

This structure is an overgeneration because it does not satisfy the specified interface conditions, pairing the (grammatical) surface form "What has the man eaten?" with an incorrect interpretation in which "what" is mislabeled as the subject of "eaten" and "the man" is mislabeled as the object of "eaten". The production of such an overgeneration may be overlooked so long as the learner is *generating* an expression for some particular interpretation that they have in mind, in which case the learner stipulates a-priori known LF interface conditions that suffice to rule out the production of the overgeneration. However, in the context of *recognition*, where the PF interface conditions (encoding word-ordering) to be satisfied are presented to the learner and the LF interface conditions are not known[21], the learner is not prohibited from producing the overgeneration and then recovering from it an incorrect interpretation. This motivates the development of an extension of the instantaneous acquisition procedure that rules out this scenario from arising even in the absence of the LF interface conditions that would otherwise restricted the illegitimate production.

We can prohibit the production of such overgenerations by adding the following constraint to the *instantaneous acquisition procedure* that seeks to minimize the number of ways in which lexical items may be combined by merge:

> Two features appearing in a derivation, one a selector and the other a selectee, may not have the same label unless at least one of the following two conditions are met:
>
> (i) the two features together drive an instance of external merge at some point in the derivation;
>
> (ii) at least one of the labels is not in the matrix clause of the derivation.

Condition (i) follows from external merge being driven by a constituent with a selector feature selecting an constituent with a selectee feature, subject to the constraint that the selector and selectee features both have the same feature label. Condition (ii) serves to ensure that if the same lexical entry is to participate multiple-times in a derivation – e.g. once in the matrix clause and once in an embedded clause – that a contradiction does not arise from the system requiring that the two copies of the lexical entry have different feature labels.[22] Note that although this constraint increases the number of selectional features present in the inferred lexicon, the constraint derived from optimization metric (3.8) (defined in §3.2.1) will ensure that the inferred lexicon uses as few distinct selectional features as possible while respecting the constraint introduced in this extension.[23]

This approach is motivated by the notion that a lexical feature sequence in a minimalist grammar should correspond to a trajectory through a derivation that is specific to the kind of lexical items will be merged with – e.g. a lexical feature sequence for a determiner that be selected as an external argument (by a light-verb) should be different from the lexical feature sequence for a determiner that can be selected as an internal argument (by a lexical verb). This will prevent, in the case that only PF (and not LF) interface conditions are input to the parser (as in the case of *recognition*), the possibility of the set of lexical entries that is able to construct a "correct" derivation (i.e. from which the correct logical form may

[21] Indeed, discovering what LF interface conditions are satisfied may well be what motivates the parser to be used for the task of *recognition* to begin with.

[22] Condition (ii) may be checked, with respect to whether a node $x$ in the derivation (that corresponds to a feature) by verifying that the only complementizer head that c-commands $x$ is the root node of the derivation.

[23] This extension requires that a substantively larger set of selectional features be made available to the SMT-model, and that the size of the lexicon, with respect to the number of lexical feature sequences, also be increased; these changes require increasing the values of several of the model parameters that are supplied as input to the acquisition procedure.

be recovered) also being able to produce an overgeneration (more specifically, a derivation from which an incorrect logical form may be recovered).

With this approach, we expect the resulting, modified set of lexical entries (a subset of the inferred lexicon) to look something like the following:

$$\text{what}/N :: \sim x_0, -p \tag{4.13}$$

$$\text{has}/T :: = x_4, +q, \sim x_5 \tag{4.14}$$

$$\text{the}/D :: = x_2, \sim x_3, -q \tag{4.15}$$

$$\text{man}/N :: \sim x_2 \tag{4.16}$$

$$\text{eaten}/V :: = x_0, \sim x_1 \tag{4.17}$$

$$\epsilon/v :: <= x_1, = x_3, \sim x_4 \tag{4.18}$$

$$\epsilon/C_{ques.} :: <= x_5, +p, C \tag{4.19}$$

Observe that every lexical entry has been modified such that every instance of *external merge* occurring in the prescribed derivation is driven by a unique selectional feature label (requiring the introduction of five new selectional feature labels), thereby preemptively ruling out the production of *any* derivation that does not satisfy both the LF and PF interface conditions; in particular, this rules out the production of *both* the overgeneration associated with the incorrect interpretation of the surface form *"what has the man eaten"*, *and* the overgeneration associated with the meaningless (ungrammatical) expression *"what has the eaten man?"* (which satisfies neither the PF nor or the LF interface conditions listed in $I_1$ of Table 2.4).

To summarize, by eliminating the production of a class of overgenerations, the implementation of this extension would improve the strong generative capacity of the lexicon inferred by the *instantaneous acquisition procedure*, thereby improving the descriptive adequacy of the inferred lexicon (which is a precondition for the acquisition procedure to meet the condition of explanatory adequacy).

## 4.4 Incorporating Principles of Language Learning

Finally, let us consider how a principle of language learning, the *Tolerance Principle* (Yang, 2016), may be incorporated into the acquisition procedures introduced in this thesis.[24]

The acquisition procedures introduced in this thesis are conservative in so far as the (inferred) lexicon they output only associates a phonological form with a lexical feature sequence if the lexical entry formed by that pair appears in a derivation that was used to satisfy one of the entries (i.e. a pairing of LF and PF interface conditions) in the primary linguistic data that was the input to the procedure. However, there are times when productivity is justified by the (positive) evidence that a child obtains from *primary linguistic data*: if the child obtains sufficient (positive) evidence from the PLD, they may be justified in assuming that a phonological form associating with one lexical feature sequence (productively) implies that the phonological form also associates with a second lexical feature

[24] The procedures for acquisition developed in this thesis most closely align with the Triggering Model of language acquisition (Gibson and Wexler, 1994; Berwick and Niyogi, 1996; Niyogi and Berwick, 1996), in which the learner is identified with a single grammar in the hypothesis space – if the current grammar fails to parse an input sentence, a different grammar is selected. This model of language acquisition has the benefit that, under certain conditions established in formal learning theory (Gold, 1967), the learner will converge to a single grammar; however, this model suffers from being vulnerable to noise in the input data.

sequence. This is illustrated by the classic example, introduced by (Baker, 1979), of children learning for which (ditransitive verbs) dative alternations are possible. Specifically, there are many ditransitive verbs that take two internal arguments – one a direct argument and the other an indirect argument headed by a (dative) prepositional phrase (e.g. "I will give the money to the museum.") – and some of these ditransitive verbs (e.g. assign, extend) have an alternate form in which the dative prepositional phrase is shifted (e.g. "I will give the museum the money."), while others do not (e.g. donate, introduce, return).[25] Notably, children are able to learn that sentences with certain ditransitive verbs may under go dative-shift, while also rarely making the mistake of producing the dative shift form of a sentence with a (ditransitive) verb for which this is not possible; furthermore, children appear to acquire this knowledge without relying on (direct or indirect) negative evidence.

Let us now put this problem into the context of the procedure for parsing using a lexicon output by the procedure for acquisition. Suppose there exist two distinct lexical feature sequences in the lexicon, $\mathfrak{L}_\alpha$ and $\mathfrak{L}_\beta$, with $\mathfrak{L}_\alpha$ used for the standard construction of a ditransitive verb and $\mathfrak{L}_\beta$ used for the alternative (dative shift) construction of a ditransitive verb. Suppose now that the acquisition procedure, in processing the PLD, has seen $N$ (distinct) phonological forms that have been associated with $\mathfrak{L}_\alpha$ and $M$ (distinct) phonological forms that are associated with *both* $\mathfrak{L}_\alpha$ *and* $\mathfrak{L}_\beta$ (note that $M \leq N$). Then the question is: should a (ditransitive) verb that has only been observed to associate with $\mathfrak{L}_\alpha$ also associate with $\mathfrak{L}_\beta$, and if so, how does the quantity of evidence – i.e. the values of $M$ and $N$ – impact this conclusion? Fortunately, this problem has been studied in detail and an answer to this problem is provided by the *Sufficiency Principle*, which follows directly from the *Tolerance Principle*:[26]

> *"A (strong) prediction of the derivational account would be that verbs that appear in one construction should be automatically extendable to other construction. This can be formalized quantitatively. Suppose that there are $N$ lexical items that participate in construction A, out of which $M$ also appear in construction B. Following the Sufficiency Principle, if $N - M < \theta_N$, then learners are justified to conclude that construction A and B are productively implicational"* (Yang, 2016, Pg. 205)

Here the threshold, $\theta_N$, above which one construction (productively) implies another is defined as:

$$\theta_N = \frac{N}{\ln N}$$

The *Sufficiency Principle* may be incorporated into the procedure for parsing as follows:

1. Associating each lexical feature sequence, $\mathfrak{L}_a$, with an integer variable, $N_a$ that is constrained to count the number of distinct phonological forms that associates with

[25] See (Levin, 1993) for a comprehensive classification of ditransitive verbs that can and cannot serve as the predicate in a double-object construction (via dative-shift).

[26] Notably, Tolerance Principle-based models of language acquisition go beyond Variational models of language acquisition (Yang, 2002). In a Variational model of language acquisition, the learner uses general learning mechanisms to search the hypothesis space given by UG – that is, this model views language learning as a probabilistic process; the learner has a probability distribution over the parameters of the space of grammars; the distribution changes in response to input stimulus. However, while a Variational model of language acquisition is a probabilistic process that can tolerate noise in the primary linguistic data (to a degree) but cannot distinguish between exceptions and noise, a Tolerance Principle based model of language acquisition is able to distinguish between noise and exceptions without having to maintain probability distributions.

that lexical feature sequence. (This association may be implemented via a unary uninterpreted function that maps members of the lexicon node sort to integers.)

2. Associating each distinct pair of lexical feature sequences $(\mathfrak{L}_a, \mathfrak{L}_b)$ with an integer variable $M_{a,b}$ that is constrained to count the number of distinct phonological forms that associates with those two lexical feature sequences. (This association may be implemented via a binary uninterpreted function that maps pairings of members of the lexicon node sort to integers.)

Given this, if in addition the Sufficiency Principle is satisfied – i.e. the following inequality holds:

$$N_a - M_{a,b} < \frac{N_a}{\ln N_a}$$

then the parser will assume that a phonological form that associates with $\mathfrak{L}_a$ will also associate with $\mathfrak{L}_b$.[27] As the incremental acquisition processes the PLD in batches and outputs a sequence of (successively) inferred lexicons, the values of $N_a$ and $M_{a,b}$ may increase, and as they do, whether the Sufficiency Condition is satisfied may fluctuate, and consequently whether the child assumes that a phonological form associating with $\mathfrak{L}_a$ also implicates association with $\mathfrak{L}_b$ will also fluctuate. In this way, adopting the sufficiency principle provides a mechanism for a child to decide both: (i) whether there is sufficient evidence to start (productively) applying a rule (i.e. that association with one lexicon feature sequence implicates association with another lexicon feature sequence), and (ii) whether to back-off from an earlier the decision to start applying the rule.[28] Finally, note that the approach outlined in this section may be expanded, so as to refine the conditions under which a (productive) generalization is considered, by requiring that multiple specific lexical feature sequences be associated with a phonological form before inferring that the phonological form also associates (productively) with another lexical feature sequence; here we have only considered the simplest case of association with one lexical feature sequence implies another.

## 4.5 Summary

Implementing the extensions outlined in this chapter will serve to bring the procedures for parsing and acquisition developed in this thesis into closer alignment with the goal of (faithfully) modeling linguistic theories that meet the conditions of *descriptive adequacy* and (more importantly) *explanatory adequacy*, and this has ramifications for the applications of these models to problems of engineering and science. More broadly, the approach this thesis

[27] Note that these integer variables and the inequality encoding the Sufficiency Principle utilize the theory of linear-arithmetic for integers and real numbers.

[28] For example, as detailed in (Yang, 2016, Pg. 211-212), when at first the number of ditransitive communication-verbs (e.g. ask, admit, declare, report, teach, tell) that a child has learned is small enough (and consequently the threshold of the Sufficiency Principle is small), the child may believe there to be sufficient evidence for them to assert that all communication-verbs can appear in the double-object construction (e.g. the child will assert that the verb "say" can appear in the sentence "I said her no." even though they have (likely) never heard an adult produce that sentence); however, later, as the child learns more (ditransitive) communication-verbs (and consequently the threshold of the Sufficiency Principle increases), the child will come to understand that there is no longer sufficient evidence for them to assert that all communication-verbs can appear in double-object constructions, and will instead lexicalize the list of communication-verbs that they know can appear in double-object constructions (because they have heard adult speakers produce double-object constructions with those particular communication-verbs).

has taken in developing models of language processing and language acquisition is inspired by the *Strong Minimalist Thesis*, which asserts that:

> *"language is an optimal solution to interface conditions that FL must satisfy; that is, language is an optimal way to link sound and meaning, where these notions are given a technical sense in terms of the interface systems that enter into the use and interpretation of expressions generated by an I-language."*
> (Chomsky, 2008)

By way of enabling and disabling the axioms underlying the SMT-models constructed by the procedures for parsing and acquisition, it is possible to carry out experiments (using an interactive SMT-solver) that aim to determine which of these axioms are superfluous, as the purpose they serve is instead accounted for by the requirement of the system to accord with the facts of language are redundant, and *thereby gain insight into whether the (universal) linguistic principles, from which the model-axioms are derived, are justified or may be discarded.* Additionally, as LF and PF interface conditions are translated into axioms that constrain the model, in the case that the procedure *cannot* infer an MG lexicon (because no valid interpretation of the model can be identified by the SMT-solver), the solver may be used to trace the provenance of the impeding constraints back to specific axioms derived from the interface conditions, so that we may better understand how the identified axioms conflict with out axiomatization of minimalist syntax, and consequently, better understand how the constraints imposed by the requirement to satisfy interface conditions may conflict with the stipulations of the principles of UG. In this way, the procedures for parsing and acquisition developed in this thesis, and the SMT-models underlying them, may serve as a vehicle for running computational experiments that evaluate theories of syntax within the framework of the Minimalist Program.

Ultimately, it is our hope that after reading this thesis, you the reader will have your own questions to ask of the models of language processing and language acquisition developed herewithin, and that you will be able to use an interactive theorem prover (such as the Z3 SMT-solver) to query these models and gain insight into the workings of the computational system underlying the human language faculty.

# Bibliography

Adger, D. (2003). *Core syntax: A minimalist approach*, volume 33. Oxford University Press Oxford.

Adger, D. and Svenonius, P. (2011). Features in minimalist syntax. *The Oxford handbook of linguistic minimalism*, pages 27–51.

Ayoun, D. (2005). *Parameter Setting in Language Acquisition.* Bloomsbury Publishing.

Baker, C. L. (1979). Syntactic theory and the projection problem. *Linguistic Inquiry*, 10(4):533–581.

Baker, M. C. (1988). *Incorporation: A theory of grammatical function changing.* University of Chicago Press.

Baker, M. C. (2008). The macroparameter in a microparametric world. *The Limits of Syntactic Variation*, 132.

Barrett, C. and Tinelli, C. (2018). Satisfiability modulo theories. In *Handbook of Model Checking*, pages 305–343. Springer.

Berwick, R. C. (1984). Strong generative capacity, weak generative capacity, and modern linguistic theories. *Computational Linguistics*, 10(3-4):189–202.

Berwick, R. C. (1985). *The acquisition of syntactic knowledge.* MIT Press.

Berwick, R. C. and Chomsky, N. (2016). *Why only us: Language and Evolution.* MIT Press.

Berwick, R. C., Chomsky, N., and Piattelli-Palmarini, M. (2013). Poverty of the stimulus stands: Why recent challenges fail. In *Rich languages from poor inputs*, pages 19–42. Oxford University Press.

Berwick, R. C. and Niyogi, P. (1996). Learning from triggers. *Linguistic Inquiry*, pages 605–622.

Berwick, R. C., Pietroski, P., Yankama, B., and Chomsky, N. (2011). Poverty of the stimulus revisited. *Cognitive Science*, 35(7):1207–1242.

Bjørner, N., de Moura, L., Nachmanson, L., and Wintersteiger, C. M. (2018). Programming z3. In *International Summer School on Engineering Trustworthy Software Systems*, pages 148–201. Springer.

Bloom, P. (1994). *Language Acquisition.* MIT Press.

Borer, H. (1984). *Parametric syntax: Case studies in Semitic and Romance languages*, volume 13 of *Studies in Generative Grammar*. Foris Publications, Holland.

Chesi, C. (2004). *Phases and cartography in linguistic computation: Toward a cognitively motivated computational model of linguistic competence*. PhD thesis, Università degli Studi di Siena.

Chesi, C. (2007). An introduction to phase-based minimalist grammars: why move is top-down from left-to-right. *Studies in Linguistics*.

Chomsky, N. (1965). *Aspects of the theory of syntax*. MIT Press.

Chomsky, N. (1981). *Lectures on Government and Binding*. Studies in generative grammar. Foris.

Chomsky, N. (1986). *Knowledge of language: Its nature, origin, and use*. Greenwood Publishing Group.

Chomsky, N. (1990). *Some Concepts and Consequences of the Theory of Government and Binding*. MIT Press.

Chomsky, N. (1995). *The Minimalist Program*. Volume 28 of Current studies in linguistics series. MIT Press.

Chomsky, N. (2001). Derivation by phase. In Kenstowicz, M., editor, *Ken Hale: A life in language*, pages 1–52. MIT Press.

Chomsky, N. (2005). Three factors in language design. *Linguistic Inquiry*, 36(1):1–22.

Chomsky, N. (2008). On phases. *Current Studies in Linguistics Series*, 45:133.

Chomsky, N. (2013a). Poverty of the stimulus: Willingness to be puzzled. In *Rich languages from poor inputs*, pages 61–67. Oxford University Press.

Chomsky, N. (2013b). Problems of projection. *Lingua*, 130:33–49.

Chomsky, N., Gallego, Á. J., and Ott, D. (2019). Generative grammar and the faculty of language: insights, questions, and challenges. *Catalan Journal of Linguistics*, pages 229–261.

Collins, C. (2001). Economy conditions in syntax. *The Handbook of Contemporary Syntactic Theory*, pages 45–61.

Collins, C. and Stabler, E. (2016). A formalization of minimalist syntax. *Syntax*, 19(1):43–78.

Crain, S. and Pietroski, P. (2002). Why language acquisition is a snap. *The Linguistic Review*, 19(1-2):163–183.

Davis, M., Logemann, G., and Loveland, D. (1962). A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397.

De Moura, L. and Bjørner, N. (2008). Z3: An efficient smt solver. In *Proceedings of the Theory and Practice of Software, 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, TACAS'08/ETAPS'08, pages 337–340, Berlin, Heidelberg. Springer-Verlag.

Donati, C. and Cecchetto, C. (2011). Relabeling heads: A unified account for relativization structures. *Linguistic Inquiry*, 42(4):519–560.

Dryer, M. S. and Haspelmath, M., editors (2013). *WALS Online.* Max Planck Institute for Evolutionary Anthropology, Leipzig.

Dutertre, B. and de Moura, L. (2006). A fast linear-arithmetic solver for DPLL(T). In Ball, T. and Jones, R. B., editors, *Computer Aided Verification*, pages 81–94, Berlin, Heidelberg. Springer Berlin Heidelberg.

Everaert, M. B., Huybregts, M. A., Chomsky, N., Berwick, R. C., and Bolhuis, J. J. (2015). Structures, not strings: linguistics as part of the cognitive sciences. *Trends in Cognitive Sciences*, 19(12):729–743.

Fichte, J. K., Hecher, M., and Szeider, S. (2020). A time leap challenge for sat-solving. In Simonis, H., editor, *Principles and Practice of Constraint Programming*, pages 267–285, Cham. Springer International Publishing.

Fong, S. and Ginsburg, J. (2019). Towards a minimalist machine. In Berwick, R. C. and Stabler, E. P., editors, *Minimalist Parsing*, pages 16–38. Oxford University Press.

Gibson, E. and Wexler, K. (1994). Triggers. *Linguistic Inquiry*, 25(3):407–454.

Gold, E. M. (1967). Language identification in the limit. *Information and control*, 10(5):447–474.

Graf, T. (2013). *Local and transderivational constraints in syntax and semantics.* PhD thesis, University of California at Los Angeles.

Grimshaw, J. (1981). Form, function, and the language acquisition device. *The logical problem of language acquisition*, 165:178.

Grimshaw, J. B. (2005). *Words and structure.* CSLI Publications.

Hale, K. and Keyser, S. J. (1993). On argument structure and the lexical expression of syntactic relations. In Hale, K. and Keyser, S. J., editors, *The view from Building 20: Essays in linguistics in honor of Sylvain Bromberger*, pages 53–109. MIT Press.

Hale, K. and Keyser, S. J. (2002). *Prolegomenon to a theory of argument structure*, volume 39 of *Linguistic Inquiry Monographs.* MIT Press.

Harkema, H. (2001). *Parsing Minimalist Languages.* PhD thesis, University of California Los Angeles.

Hirsh-Pasek, K., Treiman, R., and Schneiderman, M. (1984). Brown & Hanlon revisited: Mothers' sensitivity to ungrammatical forms. *Journal of Child Language*, 11(1):81–88.

Hornstein, N., Nunes, J., and Grohmann, K. K. (2005). *Understanding minimalism.* Cambridge University Press.

Hornstein, N. and Pietroski, P. (2009). Basic operations: Minimal syntax-semantics. *Catalan Journal of Linguistics*, 8(1):113–139.

Hunter, T. and Dyer, C. (2013). Distributions on minimalist grammar derivations. In *Proceedings of the 13th Meeting on the Mathematics of Language (MoL 13)*, pages 1–11.

Indurkhya, S. (2020). Inferring minimalist grammars with an SMT-solver. *Proceedings of the Society for Computation in Linguistics*, 3(1):476–479.

Jackendoff, R. (1983). *Semantics and cognition*, volume 8. MIT press.

Joshi, A. K. (1985). Tree adjoining grammars: How much context sensitivity is required to provide a reasonable structural description? *Natural Language Parsing*, pages 206–250.

Joshi, A. K., Shanker, K. V., and Weir, D. (1990). The convergence of mildly context-sensitive grammar formalisms. *Technical Reports (CIS)*, page 539.

Kobele, G. M. (2011). Minimalist tree languages are closed under intersection with recognizable tree languages. In *International Conference on Logical Aspects of Computational Linguistics*, pages 129–144. Springer.

Kobele, G. M., Retoré, C., and Salvati, S. (2007). An automata-theoretic approach to minimalism. *Model Theoretic Syntax at 10*, pages 71–80.

Lasnik, H. and Lidz, J. L. (2017). The argument from the poverty of the stimulus. *The Oxford Handbook of Universal Grammar*, pages 221–248.

Legate, J. A. and Yang, C. D. (2002). Empirical re-assessment of stimulus poverty arguments. *The Linguistic Review*, 19(1-2):151–162.

Levin, B. (1993). *English verb classes and alternations: A preliminary investigation*. University of Chicago press.

Lust, B. C. (1999). Universal grammar: The strong continuity hypothesis in first language acquisition. In Ritchie, W. C. and Bhatia, T. K., editors, *Handbook of child language acquisition*, pages 111–155. Academic Press.

Lust, B. C. (2006). *Child language: Acquisition and growth*. Cambridge University Press.

Marcus, G. F. (1993). Negative evidence in language acquisition. *Cognition*, 46(1):53–85.

Marques Silva, J. P. and Sakallah, K. A. (1996). Grasp - a new search algorithm for satisfiability. In *Proceedings of International Conference on Computer Aided Design*, pages 220–227.

Marques-Silva, J. P. and Sakallah, K. A. (1999). Grasp: a search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, 48(5):506–521.

Michaelis, J. (1998). Derivational minimalism is mildly context-sensitive. In *International Conference on Logical Aspects of Computational Linguistics*, volume 98, pages 179–198. Springer.

Michaelis, J. (2001). Transforming linear context-free rewriting systems into minimalist grammars. *Logical Aspects of Computational Linguistics*, pages 228–244.

Michaelis, J., Mönnich, U., and Morawietz, F. (2000). Algebraic description of derivational minimalism. *Algebraic Methods in Language Processing*, 16.

Morawietz, F. (2008). *Two-Step Approaches to Natural Language Formalism*, volume 64. Walter de Gruyter.

Newport, E., Gleitman, H., and Gleitman, L. (1977). Mother, id rather do it myself: Some effects and non-effects of maternal speech style. In Snow, C. E. and Ferguson, C. A., editors, *Talking to Children*, pages 109–149. Cambridge University Press.

Nieuwenhuis, R., Oliveras, A., and Tinelli, C. (2006). Solving sat and sat modulo theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T). *Journal of the ACM (JACM)*, 53(6):937–977.

Niyogi, P. and Berwick, R. C. (1996). A language learning model for finite parameter spaces. *Cognition*, 61(1-2):161–193.

Niyogi, S. and Berwick, R. C. (2005). A minimalist implementation of Hale-Keyser incorporation theory. In *UG and External Systems: Language, Brain and Computation*, pages 269–288. John Benjamins Publishing.

Pereira, F. C. N. and Warren, D. H. D. (1983). Parsing as deduction. In *Proceedings of the 21st Annual Meeting on Association for Computational Linguistics*, ACL '83, pages 137–144. Association for Computational Linguistics.

Pesetsky, D. and Torrego, E. (2001). T-to-c movement: Causes and consequences. *Current Studies in Linguistics Series*, 36:355–426.

Pinker, S. (1984). *Language Learnability and Language Development.* Harvard University Press.

Radford, A. (1997). *Syntactic theory and the structure of English: A minimalist approach.* Cambridge University Press.

Radford, A. (2009). *An introduction to English sentence structure.* Cambridge University Press.

Radford, A. (2016). *Analysing English sentences: A minimalist approach (Second Edition).* Cambridge University Press.

Rayner, M., Hugosson, A., and Hagert, G. (1988). Using a logic grammar to learn a lexicon. In *Coling Budapest 1988 Volume 2: International Conference on Computational Linguistics.*

Reinhart, T. M. (1976). *The syntactic domain of anaphora.* PhD thesis, Massachusetts Institute of Technology.

Rizzi, L. (1982). Negation, wh-movement and the null subject parameter. *An Annotated Syntax Reader*, page 169.

Rizzi, L. (1986). Null objects in italian and the theory of pro. *Linguistic Inquiry*, 17(3):501–557.

Rizzi, L. (2011). On the elements of syntactic variation. *Studies in Linguistics*, 4:142–162.

Rizzi, L. (2017). The concept of explanatory adequacy. In *The Oxford Handbook of Universal Grammar*. Oxford University Press.

Rogers, J. and Nordlinger, R. (1998). *A descriptive approach to language-theoretic complexity*. MIT press.

Shieber, S. M., Schabes, Y., and Pereira, F. C. (1995). Principles and implementation of deductive parsing. *The Journal of Logic Programming*, 24(1-2):3–36.

Sportiche, D., Koopman, H., and Stabler, E. (2013). *An introduction to syntactic analysis and theory*. John Wiley & Sons.

Stabler, E. (1996). Derivational minimalism. In *International Conference on Logical Aspects of Computational Linguistics*, pages 68–95. Springer.

Stabler, E. P. (1998). Acquiring languages with movement. *Syntax*, 1(1):72–97.

Stabler, E. P. (2001). Recognizing head movement. In *International Conference on Logical Aspects of Computational Linguistics*, pages 245–260. Springer.

Stabler, E. P. (2004). Varieties of crossing dependencies: structure dependence and mild context sensitivity. *Cognitive Science*, 28(5):699–720.

Stabler, E. P. (2013). Two models of minimalist, incremental syntactic analysis. *Topics in Cognitive Science*, 5(3):611–633.

Stabler, E. P. and Keenan, E. L. (2003). Structural similarity within and among languages. *Theoretical Computer Science*, 293(2):345–363.

Stanojević, M. (2016). Minimalist grammar transition-based parsing. In *International Conference on Logical Aspects of Computational Linguistics*, pages 273–290. Springer.

Torr, J., Stanojevic, M., Steedman, M., and Cohen, S. B. (2019). Wide-coverage neural a* parsing for minimalist grammars. In *Proceedings of the 57th annual meeting of the Association for Computational Linguistics*, pages 2486–2505.

Vijay-Shanker, K., Weir, D. J., and Joshi, A. K. (1987). Characterizing structural descriptions produced by various grammatical formalisms. In *Proceedings of the 25th Annual Meeting on Association for Computational Linguistics*, ACL '87, pages 104–111, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yang, C. (2002). *Knowledge and learning in natural language*. Oxford University Press.

Yang, C. (2015). Negative knowledge from positive evidence. *Language*, 91(4):938–953.

Yang, C. (2016). *The Price of Productivity: How Children Learn and Break the Rules of Language*. MIT Press.