

Number 675



UNIVERSITY OF  
CAMBRIDGE

Computer Laboratory

## Computational models for first language acquisition

Paula J. Buttery

November 2006

15 JJ Thomson Avenue  
Cambridge CB3 0FD  
United Kingdom  
phone +44 1223 763500  
<http://www.cl.cam.ac.uk/>

© 2006 Paula J. Buitery

This technical report is based on a dissertation submitted March 2006 by the author for the degree of Doctor of Philosophy to the University of Cambridge, Churchill College.

Technical reports published by the University of Cambridge Computer Laboratory are freely available via the Internet:

*<http://www.cl.cam.ac.uk/techreports/>*

ISSN 1476-2986

# Abstract

This work investigates a computational model of first language acquisition; the Categorical Grammar Learner or CGL. The model builds on the work of Villavicencio, who created a parametric Categorical Grammar learner that organises its parameters into an inheritance hierarchy, and also on the work of Buszkowski and Kanazawa, who demonstrated the learnability of a  $k$ -valued Classic Categorical Grammar (which uses only the rules of function application) from strings. The CGL is able to learn a  $k$ -valued General Categorical Grammar (which uses the rules of function application, function composition and Generalised Weak Permutation). The novel concept of Sentence Objects (simple strings, augmented strings, unlabelled structures and functor-argument structures) are presented as potential points from which learning may commence. Augmented strings (which are strings augmented with some basic syntactic information) are suggested as a sensible input to the CGL as they are cognitively plausible objects and have greater information content than strings alone. Building on the work of Siskind, a method for constructing augmented strings from unordered logic forms is detailed and it is suggested that augmented strings are simply a representation of the constraints placed on the space of possible parses due to a string's associated semantic content. The CGL makes crucial use of a statistical Memory Module (constructed from a Type Memory and Word Order Memory) that is used to both constrain hypotheses and handle data which is noisy or parametrically ambiguous. A consequence of the Memory Module is that the CGL learns in an incremental fashion. This echoes real child learning as documented in Brown's Stages of Language Development and also as alluded to by an included corpus study of child speech. Furthermore, the CGL learns faster when initially presented with simpler linguistic data; a further corpus study of child-directed speech suggests that this echoes the input provided to children. The CGL is demonstrated to learn from real data. It is evaluated against previous parametric learners (the Triggering Learning Algorithm of Gibson and Wexler and the Structural Triggers Learner of Fodor and Sakas) and is found to be more efficient.



# Contents

<b>Abstract</b>	<b>3</b>
<b>List of Figures</b>	<b>9</b>
<b>Acknowledgments</b>	<b>11</b>
<b>1 Introduction</b>	<b>13</b>
<b>2 Analysis of Child I/O</b>	<b>19</b>
2.1 Linguistic Input . . . . .	19
2.1.1 Child-Directed Speech . . . . .	20
2.1.2 Subcategorization Frames . . . . .	20
2.1.3 Subcategorization Frames in Acquisition . . . . .	22
2.1.4 Automatic Extraction of Subcategorization Frames . . . . .	22
2.1.5 Methods of SCF Analysis . . . . .	24
2.1.6 Differences in Verb Types . . . . .	25
2.1.7 SCF Comparison . . . . .	25
2.2 Linguistic Output . . . . .	29
2.2.1 Stages of Language Acquisition . . . . .	30
2.2.2 Internal vs. External Language . . . . .	31
2.2.3 Child Production and Computational Modelling . . . . .	32
2.2.4 Subcategorization Frames in Child Speech . . . . .	32
2.2.5 Differences in Verb Types . . . . .	33
2.2.6 SCF Comparison . . . . .	33
<b>3 Learnability and Learning Models</b>	<b>37</b>
3.1 Principles and Parameters . . . . .	38
3.1.1 Triggers . . . . .	40
3.1.2 The Triggering Learning Algorithm . . . . .	41
3.1.3 The Structural Trigger Learner . . . . .	43
3.2 Noise and Learning Models . . . . .	43
3.2.1 How do Errors Affect Learning? . . . . .	45
3.2.2 Introducing a Hypothesis Bias . . . . .	45
3.3 Statistical Models of Language Acquisition . . . . .	46
3.3.1 The Variational Learner . . . . .	47
3.4 Connectionist Modelling . . . . .	48
3.4.1 Connectionism as a Model of Language Acquisition . . . . .	50

3.5	Learning from Semantics . . . . .	51
3.6	Summary . . . . .	52
<b>4</b>	<b>Categorial Grammar Learners and the Input they Receive</b>	<b>55</b>
4.1	Categorial Grammars . . . . .	55
4.1.1	Classic Categorial Grammars . . . . .	55
4.1.2	Combinatory Categorial Grammars . . . . .	57
4.1.3	Generalised Weak Permutation vs. Type Raising . . . . .	61
4.1.4	Semantics and Categorial Grammars . . . . .	62
4.2	Previous Categorial Grammar Learners . . . . .	64
4.2.1	The Waldron/Villavicencio Learning System . . . . .	64
4.2.2	Formal Categorial Grammar Learners . . . . .	67
4.3	Sentence Objects . . . . .	73
4.3.1	Simple Strings . . . . .	74
4.3.2	Augmented Strings . . . . .	74
4.3.3	Unlabelled Structures . . . . .	75
4.3.4	Functor-Argument Structures . . . . .	75
4.3.5	Information Content of Sentence Objects . . . . .	75
4.4	Mapping Words to their Semantic Representation . . . . .	78
4.4.1	Mechanics of the Semantic Learner . . . . .	80
4.4.2	Forming Augmented Strings . . . . .	82
<b>5</b>	<b>The Categorial Grammar Learner</b>	<b>85</b>
5.1	Learning from Augmented Strings . . . . .	85
5.2	Improving the Categorial Grammar Learner . . . . .	86
5.2.1	Introducing a Memory Module . . . . .	86
5.2.2	Setting the Memory and Memory Integrity . . . . .	89
5.2.3	Reducing Duplicated Effort . . . . .	89
5.2.4	Using Additional Rules . . . . .	90
5.3	Categorial Grammar Learner—Worked Example . . . . .	92
5.4	Categorial Grammar Learner Summary . . . . .	98
<b>6</b>	<b>Evaluation of Model</b>	<b>99</b>
6.1	Comparison to Previous Parameter Based Learners . . . . .	99
6.1.1	Influence of and Differences to Waldron/Villavicencio Learning System . . . . .	101
6.1.2	Possible Problems with the Principle of Categorial Type Transparency . . . . .	102
6.2	Learning in an Ideal Environment . . . . .	103
6.3	Learning from Real Data . . . . .	112
6.3.1	Evaluation of the Learning System . . . . .	113
6.3.2	Noise Introduced by Indeterminacy of Meaning . . . . .	113
6.3.3	Noise Introduce by Indeterminacy in Parameter Setting . . . . .	117
6.4	Developmental Compatibility . . . . .	118
6.5	Summary . . . . .	120
<b>7</b>	<b>Conclusions</b>	<b>123</b>
<b>A</b>	<b>SCF Classification</b>	<b>127</b>

<b>B Verb Distributions in Adult Speech vs. Child Directed Speech</b>	<b>147</b>
<b>C Full SCF Data in Adult Speech vs. Child Directed Speech</b>	<b>151</b>
<b>D SCF Distributions in Adult Speech vs. Child Directed Speech</b>	<b>161</b>
<b>E SCF Distribution in Child Speech</b>	<b>165</b>





# Figures

2.1	Linguistic input to a child . . . . .	20
2.2	40 most frequent verbs in adult speech (BNC) corpus vs. child-directed speech (CHILDES1) corpus . . . . .	26
2.3	Average similarity values: BNC vs. CHILDES1 . . . . .	27
2.4	30 most frequent verbs in adult speech (BNC) corpus vs. child-directed speech (CHILDES1) corpus vs. child speech (CHILDES2) corpus. . . . .	33
2.5	Frame comparison: BNC, CHILDES1, CHILDES2. . . . .	34
2.6	Average similarity values: BNC vs. CHILDES1, CHILDES1 vs. CHILDES2 . . . . .	34
2.7	SCFs acquired for the verbs <i>hit</i> and <i>pull</i> . . . . .	36
3.1	A property array; identifying the location of grammar $G$ in the intersection of sets $A$ and $B$ (where set $A$ , for example, contains all those grammars exhibiting property $A$ ). . . . .	40
3.2	The hypothesis-space; the shaded area being the location of the same grammar, $G$ . . . . .	40
3.3	The 3-parameters of the Triggering Learning Algorithm. . . . .	42
3.4	English and German type languages in Gibson and Wexler’s parameter-space. . . . .	42
3.5	Bush and Mosteller’s Linear Reward-Penalty scheme: given an input sentence $s$ and total number of grammars $N$ , the learner selects a grammar $G_i$ with probability $p_i$ . . . . .	47
3.6	A schematic diagram of a typical neuron. . . . .	48
3.7	The operation of a unit in a neural network. . . . .	49
3.8	The architecture of a neural network. . . . .	50
4.1	A selection of the rules used in Combinatory Categorical Grammars. . . . .	58
4.2	An example of Type Raising in CCG . . . . .	60
4.3	An example of Generalised Weak Permutation in CCG. . . . .	61
5.1	A fragment of a categorial type hierarchy . . . . .	87
5.2	An example state of the Type Memory . . . . .	88
6.1	Illustration of the interrogative “was the cat who grinned disappearing?” . . . . .	100
6.2	Partial hierarchy of syntactic categories . . . . .	100
6.3	Derivation of <i>Ian seems to be happy</i> . . . . .	103
6.4	Parameter settings and surface-strings of Gibson and Wexler’s English-like language. . . . .	104
6.5	Gibson and Wexler’s TLA as a Markov structure. . . . .	105

6.6	Probability and expected number of steps to convergence from each starting grammar to an English-like grammar (SVO -V2) when using the TLA. . . . .	105
6.7	Alternative derivations for the surface-string <i>subj verb obj</i> . . . . .	108
6.8	The STL as a Markov structure. . . . .	109
6.9	Category hierarchy required to parse Gibson and Wexler's English-like language.	110
6.10	The CGL as a Markov structure. . . . .	111
6.11	The Categorial Grammar Learner . . . . .	114
6.12	Illustration of the use of unary rules versus multiple lexical entries. . . . .	115
6.13	Experiment 1a: graph of size of semantic hypothesis set vs. $F_1$ . . . . .	116
6.14	Graph of relative frequency of / against % of misinterpreted thematic roles; × indicates UNDERGOER, + indicates ACTOR. . . . .	119
6.15	Category hierarchy of all SCFs that appears at least 100 times in CHILDES2. .	121
6.16	The syntactic categories involved in Brown Stage 1 . . . . .	122
6.17	The syntactic constructions that have been mastered by Brown Stage 2. . . . .	122

# Acknowledgments

I would like to thank my supervisor Ted Briscoe for all his help over the last three years. I am also very grateful to my second supervisor Ann Copestake and to Mark Steedman for their detailed suggestions for revisions to the original draft. Thanks also to Simone Teufel, Karen Spark-Jones, Anna Korhonen and Aline Villavicencio for coffee room discussions and really helpful advice. Naila, Fabre and Advaith you were fantastic office mates, thank you. In the LCE I want to thank Brian Jones for letting me use the machines and for providing well timed sponge cake; also Jon Davies for his most excellent taste in music. Last but not least are my family (especially mum, dad, Helen and Bob) for their love and encouragement; Ian Felce for being a brilliant friend and for always having confidence in me; Andy whose external optimism has been a blessing through good times and bad; and Christopher who sat beside me on the darkest days and convinced me that it could be finished.

for my grandad...

# Chapter 1

## Introduction

For a normal child, acquiring language is a natural phenomena that requires no conscious effort. After only 12 months, and with no formal teaching, a normal child is making verbal reference to objects, people and actions. These first words occur in isolation and might not sound like any recognisable adult words but within the subsequent 12 months a child starts forming understandable declarative sentences such as “i want cookie” (Naomi at 1;11—the Sachs corpus [87], CHILDES [63]). Over the next few years a child’s linguistic ability rapidly develops. A child is able to form questions, use inflections, produce clauses and use a whole range of other linguistic construction.

Eventually a child’s linguistic growth settles down. At the age of seven she is speaking with the full fluency of an adult; by which time she has the ability to process the infinite number of sentences belonging to her language and she is capable of producing an infinite number of sentences herself. This would be remarkable even if the procedure for producing (and understanding) sentences could be described by a well defined set of rules: however, human languages are so complicated that we have not yet managed to settle upon a theory that describes all linguistic phenomena satisfactorily.

Despite huge advances in cognitive science over the last 50 years, little is understood of how the human brain accomplishes the task of storing, processing and acquiring language. As native speakers, we are not conscious of how we put together sentences, how we access our mental lexicon or how this lexicon is stored; any rules that we think we know about our language are prescriptive rules for “standard” usage that have been repeated to us at school.

A key issue that remains to be resolved is that of *why* child language differs from adult language. We know that ability to speak a specific language is not genetically bestowed: a child growing up in a French speaking environment will become a Francophone; the same child growing up in a English speaking environment would speak English. Clearly, a child *learns* their native language on exposure to it. However, the exact description of what is being *learnt* is an issue of much controversy.

The *Continuity Hypothesis* ([73], [62]) says that, given there is no evidence to the contrary, a child’s cognitive system is to be assumed identical to that of an adult’s (i.e. the mechanics of brain operation are the same in both an adult and child). If this is true, then we can explain the differences between child and adult language from two possible standpoints:

1. At one extreme we can assume a child to be born with a complete language faculty: the grammar for every possible language is innately available to the child (often referred to as *nativism*). For such a bestowed child, language acquisition is the task of selecting one

grammar from all those available to it. This can be achieved by subconsciously “noticing” the linguistic properties of the environmental language.

The mechanism for selecting the correct grammar is sometimes described as a process of setting language parameters (e.g. [39], [90]). For example, a possible language parameter may relate to subject drop (to be clear, Italian is an example of a language which allows subject drop—in Italian it is possible to say *credo nel destino* (“[I] believe in destiny”) without explicitly specifying the subject of the sentence). If a child is exposed to a language where subjects can be dropped she will hopefully set her *subject drop parameter* to true; thus excluding all grammars that don’t allow subject drop and reducing the number of possible grammars left to choose from. At any given time the child selects one of the remaining possible grammars in order to produce and understand language; the idea being that once all parameters are set, one grammar is uniquely identified from all possibilities.

An alternative method is to associate each possible grammar with a score indicating its likelihood of describing the target (parental) language. The probability of a grammar is incremented (or decremented) in response to evidence of linguistic features in the language environment. As such, grammars are “competing” for selection by the child (e.g. [109] or [12]); the child’s current grammar being the most highly ranked (or perhaps selected according to the probability distribution over the grammars). By using a scoring mechanism the child never has to rule out any grammar completely (as may occur with the parameter setting model). Instead, the probability of ill-fitting grammars becomes incrementally smaller so that the probability of them being selected by the child becomes very small.

Under both these models, differences between child and adult language are explained by the child’s current grammar being different to that of the adults in their environment.

Nativist theories come under criticism for relying too heavily on the requirement of innate knowledge. In particular, the question of how an innate language faculty might develop is much debated (see [77] for a review).

2. At the other extreme we assume a child is a “blank-slate” in which linguistic skills develop in response to language stimulus. This viewpoint is referred to as *empiricism* since language is being acquired purely from observation.

Empiricist theories come under fire from Arguments from the Poverty of Stimulus [23]. One such argument states that *given the limited evidence in the language examples that children are exposed to, it is impossible to extrapolate the complicated generalizations required to acquire a language*. The problem here is that, given unconstrained scope, there are an enormous number of ways to describe the examples presented and not enough evidence is provided to refute all false grammatical hypotheses.

Since empiricist theories rely on the child’s ability to extract linguistic patterns from examples of the parental language, they are often investigated with connectionist models (computational models based on the architecture of the brain which are useful for pattern recognition from raw data).

Unfortunately, connectionist models tend to require exposure to hundreds (or even thousands) of iterations of a training set before they can reliably recognise patterns. Here again the empiricist models come under scrutiny regarding data sparsity; the argument

being that a connectionist model can not be a viable model of acquisition because of the amount of data it requires to learn and the comparatively limited number of language examples that children are exposed to.

The real explanation of acquisition may exist on some middle ground where there is some innate linguistic information (albeit as simple as a desire to communicate and build mental compositional structures) but there is also much inferred from the empirical evidence in the language environment.

Either way, any model of acquisition is going to have to be able to learn despite the varying quality of data presented to a language learner. Frequently the language examples that a child hears contain mistakes: “umms” and “ahs”; slips of the tongue; or even incomplete sentences due to distraction or interruption. A child must be able to learn the correct grammar despite these errors or be able to identify sentences that contain errors and “choose” not to learn from them. A possible solution here is to consider the statistical properties of the language; hopefully grammatical sentences will greatly outweigh the error prone sentences and consequently linguistic rules or patterns can be extracted with an associated confidence measure from evidence that is greatly skewed towards the grammatical.

There are further difficulties regarding the design and evaluation of *realistic* acquisition models; it is one matter to design a model that can acquire a target grammar, but it is another to design a model that does so in the same way as a child. If the goal of acquisition is to acquire a target grammar, then the success of a model may easily be measured by comparing the finally acquired grammar with the grammar of the target language. However, for a realistic model, it is required that the model’s internal grammar is similar to that of a child’s at all stages of the acquisition. Unfortunately, a child’s internal grammar is difficult to ascertain because articulatory skills are not fully developed at birth. Linguistic performance is hindered throughout acquisition due to linguistic competence. This brings the challenge of separating these issues to the study of acquisition and creates problems for design since it is unclear exactly what to model.

It should be noted here that it is NOT possible to investigate first language acquisition post development of articulatory skills by studying the acquisition of second languages later in life. First language and second language acquisition are two very different processes. The acquisition of a second language is a labour of much hard work and memory effort whereas acquiring your native language requires no conscious effort at all. The study of second language acquisition can perhaps tell us as much about our memory as our language faculty.

The study of first language acquisition can be approached from many directions. The oldest method is that of corpus analysis. Several types of corpora have been compiled: *diary studies* are the traditional method for tracking child language development and usually involve a single child who’s caretaker keeps a diary of when new constructions are first produced; *large sample studies* involve a large number of children and generally record an experiment looking for a specific language phenomena; *longitudinal language sampling* involves regularly recording/transcribing a single child’s language (for instance, a child may be recorded for a few hours once a week over a period of several months or years). Linguists analyse such studies to collate data and speculate on how acquisition may proceed given the evidence. An example that is relevant to this thesis is Snow’s small scale corpus analysis of speech directed towards children [94]. In this study Snow recorded conversational speech between mothers and children. She analysed the mothers’ speech and concluded that adults talk amongst each other in a way different from that in which they talk to their children.

Psycho-linguists study language acquisition by carefully analysing the way we respond to differing linguistic stimuli; this will often involve measuring tiny differences in response times that we can not ourselves perceive. An example is Jean Gleason's famous *wug test* [40] that was designed to investigate the acquisition of the inflection. The experiment proceeded by presenting a child with a picture of a fictional creature and stating "*This is a wug*". The child was then shown a picture of two of these creatures and invited to complete the sentence "*now there are two ...?*". A child that learnt to generalise the inflection rule for plurals was able to answer *two wugs*. Younger children were just confused and at best answered *two wug*. Furthermore, it was demonstrated that the three plural allophones (/z/, as in dogs; /s/, as in cats; and /ɛz/, as in horses) are acquired separately. The experiment shows that children can generalise rules (or patterns) from the language that they have been exposed to and apply these generalisations to novel terms. The wug test has since been used to investigate the acquisition of other inflection rules (such as the past tense and possessives).

Computational linguistics can contribute to the study of language acquisition formally by investigating the mathematical constraints of linguistic theory. The most well-known example of such work must be Gold's theory [42] which showed that an infinite language (i.e. one that has a hierarchical structure capable of recursion) is unlearnable from examples of the language alone (see Chapter 3 for a more precise description). Computational linguists can also contribute to the study of acquisition experimentally by producing learning simulations: two examples relevant to this work include Gibson and Wexler's [39] simulation of acquisition in a parametrically defined grammar-space; and also Rumelhart and McClelland's connectionist model for acquiring past tenses [86]. Simulations can provide useful insight into the problems of time and space complexity for acquisition algorithms.

This work investigates a computational model of first language acquisition. The model can neither be classed as a pure nativist or a pure empiricist model, rather it lies on the middle ground between the two extremes. As with a nativist model, it assumes some innate linguistic functionality; in particular the ability to associate a meaning with a sentence, to recognise objects, segment words and combine constituents. The possibility that such functionality can be provided by some previous empirical processing will not be discussed in this thesis. As with an empirical model, this model infers linguistic patterns from language examples. Furthermore, it makes crucial use of a linguistic memory and is able to deal with mistakes in the input by employing statistical techniques to filter noise. The model acquires language from real data. The functionality of the model is described algorithmically however its statistical nature leads one to believe that it could possibly lend itself to a connectionist model if one had the time and resources.

In order to aid the design and evaluation of the model, corpus studies have been conducted on the model's input (child-directed speech) and output (child speech). Its mathematical rigidity has also been explored and a comparison is made between this and previous computational models.

The outline of this thesis is as follows: Chapter 2 investigates verbal constructions in a child's input stimulus and output productions. Two corpora are extracted from parts of the CHILDES database [63]; one containing child speech and another containing child-directed speech. For comparison a corpus of adult speech has also been constructed from the spoken section of the British National Corpus [56]. Corpora are compared for verb frequency and verb subcategorization frames. We discuss the possible role of child-directed speech as an aid to learning and notice that, in concurrence with Brown's stages, children might acquire verbal frames



incrementally—using more complex constructions only after simpler constructions have been learnt. We use this idea of incremental learning as a basis for the model presented in this thesis. Chapter 3 investigates the problems of modelling language acquisition in general and discusses the pros and cons of methods that have previously been adopted. Chapter 4 looks at previous categorial grammar learners—categorial grammar being the chosen formalism for this work. The Learning System of Waldron/Villavicencio ([105], [104]) and the algorithms of Buszkowski/Kanazawa ([17], [49]) are summarised in detail since they form the basis of the learner presented here. This investigation of previous learners leads us to explore the different types of input that a categorial grammar learner could be presented with in order to learn from. The concept of a *sentence object* is introduced to refer to any input structure that carries at least as much information as a string. The complexity of learning from different types of sentence objects is discussed as well as the cognitive plausibility of such objects. We select one particular type of sentence object (the augmented string) as suitable input for models learning from real data. We finish the chapter by outlining how augmented strings could be created and suggest that they are simply representations of the constraints placed on the search space of possible parses by the semantics associated with a string.

Chapter 5 presents the Categorial Grammar Learner. This chapter builds on the work of Waldron/Villavicencio and Buszkowski/Kanazawa. It details a learner that uses the categorial grammar rules of function application, function composition and Generalised Weak Permutation. The learner makes use of a memory module which is used to constrain hypothesised grammars and also makes the learner robust to noise. An example of the operation of the learner is given. In Chapter 6 the Categorial Grammar Learner is evaluated. The efficiency of the model is investigated in comparison with two previous models (the Triggering Learning Algorithm [39] and the Structural Triggers Learner [38]). Further experiments demonstrate the learner to be robust to noise caused by indeterminacy of meaning and indeterminacy in parameter setting. This chapter concludes with a discussion of the developmental compatibility of the model in relation to Brown's stages of acquisition.



# Chapter 2

## Analysis of Child I/O

In order to design and evaluate a model of language acquisition it is necessary to have a good understanding of the properties of the input and output. With this mind, the following chapter details a corpus study on some of the lexical properties of child-directed speech and child speech.

### 2.1 Linguistic Input

A child's environment is rich with stimulus but the degree to which this stimulus contributes to learning is much debated. With respect to language acquisition, stimulus is provided in the form of language examples rather than direct teaching and is primarily provided by the child's caretaker. The importance of the linguistic interaction between caretaker and child is still unclear; acquisition theories vary considerably in their input requirements. However, there is one requirement beyond dispute; that of some sort of linguistic interaction. A child growing up in a French speaking environment acquires French, the same child growing up in an English speaking environment would acquire English; thus, a certain quantity of linguistic interaction is needed to determine which language is to be acquired. Furthermore, linguistically isolated children are known not to develop language spontaneously (the most well documented example being Genie [30]).

The exact quantity and quality of linguistic input required for successful acquisition is unknown. Linguistic interaction with children is very much culturally defined ([92], [91]), consequently broad statements regarding the nature of the input are very difficult to make. However, there is evidence to suggest that there is a minimum threshold *quantity* for successful acquisition; a study by Sachs and Johnson [88] reported that a hearing child of deaf parents did not learn spoken English despite having been exposed to television. Other research has suggested that the *quality* of language can be fairly poor and acquisition still be successful; hearing children of deaf parents do not just imitate the limited spoken language of their parents [6] but add to it. The linguistic input to a child arrives from a number of sources; not all of which are specifically directed at the child. Furthermore, it is apparent that a child will not attend to all the linguistic input she is exposed to. Input can thus be divided into subcategories as shown in Figure 2.1: the outer circle shows all the linguistic input available to a child (including television, overheard conversations etc.); the circle labelled Child-Directed Speech (CDS) indicates the input which is spoken directly to the child; the remaining circle indicates the input which the child actually attends to.

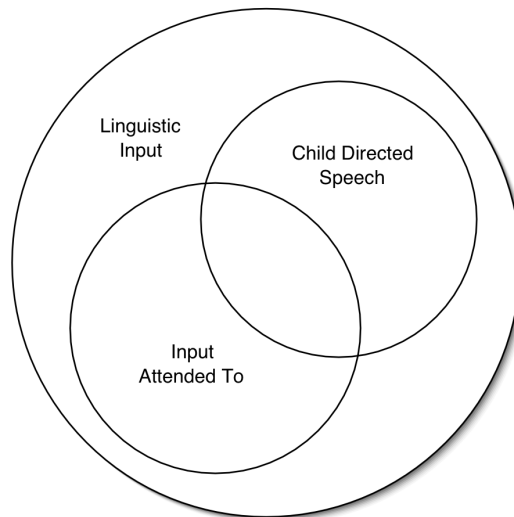


Figure 2.1: Linguistic input to a child

### 2.1.1 Child-Directed Speech

Understanding the role of child-directed speech (CDS) is of fundamental importance to language acquisition. Several manual small scale studies (see Snow [94] for an overview) have suggested that CDS is very different from speech between adults: intonation is often exaggerated, a specific vocabulary can be used, and sometimes even specific syntactic structures that are not found in adult speech appear. Perhaps contrastingly, there is considerable evidence that accurate and complex syntactic structures are informative during language acquisition (e.g. [58], [67] and [36]). Consequently, the role of CDS is by no means clear. Pine [72], amongst others, speculates that the purpose of CDS is to merely engage the child in conversation. Snow [94], on the other hand, suggests that CDS is actually teaching the child language. Clearly, larger-scale studies into the nature of CDS are required before we can begin to establish its role in acquisition. In this work we will look in particular at the role of subcategorization frames in CDS.

### 2.1.2 Subcategorization Frames

Verbs may be categorised according to the types of *complement* they take; the partially ordered list of complements being referred to as the verb's subcategorization frame (SCF). The term *complement* includes obligatory and optional arguments but NOT *adjuncts*: complements are understood to be selected by the verb and complete its meaning; adjuncts extend the meaning of the central predication.

Examples 1a, 1b and 1c below demonstrate how verbs vary in their number of obligatory arguments; i.e arguments which are *required* in order to complete the meaning of the sentence. In 1a, *surf* does not require any arguments; we can assign *surf* to the SCF category NULL. In 1b, *bought* selects the noun phrase **a juicer** to complete its meaning; we can assign *buy* to the SCF category NP. In 1c *put* requires both the noun phrase **Harvey** and the prepositional phrase **on the floor** to be selected in order to complete its meaning; we shall assign it to SCF category NP-PP.

1. (a) *Stephen surfs*  
 (b) *Andrew bought a juicer*  
 (c) *Lindsay put **Harvey on the floor***

Obligatory arguments may be identified by the “*elimination*” test (e.g. [95]), which involves eliminating an element from a sentence and observing whether the remaining sentence is still grammatical; sentences that do not contain all their obligatory arguments are not grammatical (see 2a, 2b and 2c).<sup>1</sup>

2. (a) \**Andrew bought*  
 (b) \**Lindsay put **on the floor***  
 (c) \**Lindsay put **Harvey***

Now consider the sentences 3a, 3b and 3c. We have already stated that SCFs are classified by complements—which includes both obligatory and optional arguments but not adjuncts. Obligatory arguments may be identified by elimination but how do we identify optional arguments from adjuncts? For instance, the sentence in example 3b does not require the prepositional phrase **in the garden** to be grammatical. So is **in the garden** an optional argument to *work* or an adjunct?

3. (a) *William drank **lager***  
 (b) *Vic worked **in the garden***  
 (c) *Harvey sat **licking his paws***

Unfortunately, there is disagreement in the literature over the classification of optional arguments versus adjuncts. Some linguists address this problem by proposing an argument-adjunct scale ( [64], [95]). Somers distinguishes a six-point scale:

- i *integral complements* (as in *Jon doesn't have **a hope***);
- ii *obligatory complements* (as in 1c);
- iii *optional complements* (as in 3a);
- iv *middles* as (in 3b);
- v *adjuncts* (as in 3c);
- vi *extra-peripherals* (as in *Bobby can eat, **as you know***).

The COMLEX lexicographers [66] distinguish adjuncts from arguments using a set of criteria and heuristics. For instance, they state that PPs headed by *to* tend to be arguments, whereas PPs expressing time, manner, or place are mostly adjuncts. They also state that adjuncts occur with a large variety of verbs at a similar frequency whereas arguments occur with a high frequency with specific verbs.

In general SCFs can be made more specific (i.e. we can increase the number of possible frames) by parameterising the frames for lexically-governed particles and prepositions. 4a illustrates a SCF containing the particle *up* (NP-*up*-NP) and 4b a SCF with the preposition *to* (NP-*to*-NP).

---

<sup>1</sup>It should be noted that we are referring to syntactic grammaticality: Somers [95] points out that the elimination test is not foolproof in that it may be complicated by the distinction between syntactic and semantic obligatoriness.

4. (a) *Cheryl took Ewan up a pint.*  
(b) *Tom took the light sabre to Dave.*

Semantic constraints might also be usefully captured within subcategorization frames. Such constraints are usually referred to as selectional restrictions. For instance, the verb *moo* prefers a cow as its subject and *eat* prefers an edible things as its object. Sentences that violate selectional restrictions sound jarring.

5. (a) *The cat was mooing because she was hungry.*  
(b) *Stephen ate geography for breakfast.*

Henceforth, the set of SCFs referred to in this chapter will be the union of the SCFs found in the ANLT [7] and COMLEX [43] dictionaries. These SCFs abstract over specific lexically governed particles, prepositions and specific predicate selectional preferences but include some derived semi-predictable bounded dependency constructions, such as particle and dative movement (see Appendix A for a complete listing).

### 2.1.3 Subcategorization Frames in Acquisition

Landau and Gleitman [54] suggest that children use verb subcategorization frames (SCFs) to identify novel word meanings; arguing that in many cases surface-structure/situation pairs are insufficient or even misleading about a verb's interpretation. Consider the sentences *Did you eat your cookie?* and *Do you want me to take that away?* According to Landau and Gleitman the SCFs of *eat* and *want* cue their interpretations, i.e. *want* occurs with sentential complements, suggesting a mental component to its interpretation. Furthermore, they suggest that SCFs provide convergent evidence on the meaning of a verb. For instance, if *John zirks bill the book* the learner assumes *zirk* to be an active verb of transfer (such as *bring, throw, explain*), whereas if *John is zirking that the book is dull* the learner interprets *zirk* to be a mental verb.

Such a syntactically intensive theory of acquisition can only be supported if the input to children is sufficiently complex and diverse in its SCFs. In general, CDS is thought to be syntactically simpler than adult speech [94]. If the role of CDS is to teach language, as Snow suggests, then we may have a conflict with acquisition theories that require syntactic complexity and diversity.

### 2.1.4 Automatic Extraction of Subcategorization Frames

Manual analysis of SCFs is very costly and therefore not ideal for large scale studies in specific domains, such as CDS. Automatic acquisition of SCFs from corpora now produces fairly accurate lexical data useful for (psycho)linguistic research (e.g. Roland et al. [85]). However, these methods are yet to be applied to CDS.

In the following, the most comprehensive English subcategorization system available is used to automatically acquire large scale empirical data related to verb SCFs from CDS. Both qualitative and quantitative methods are used to compare the resulting data against that obtained from a corpus of adult speech. Section 2.1.4 describes the method for subcategorization frame acquisition and section 2.1.4 introduces the corpora used. The metrics involved in the analysis are explained in section 2.1.5. Sections 2.1.6 and 2.1.7 look at the difference in verb frequencies and SCF distribution between the two corpora respectively. We conclude with a discussion and summary of the observations. This work was carried out in conjunction with Anna Korhonen and is originally published in [21].

## Methodology

For subcategorization acquisition, Korhonen’s version [51] of Briscoe and Carroll’s system [10] was used. This system incorporates 163 SCF distinctions; the union of those found in the ANLT [7] and COMLEX [43] dictionaries (see Appendix A).

The system first extracts sentences containing specific predicates from a corpus. The resulting data is tagged, lemmatised and parsed using the RASP system (Robust Accurate Statistical Parser; [11]). Local syntactic frames including the syntactic categories and head lemmas of constituents are then extracted from parses. The resulting patterns are classified to SCFs on the basis of the feature values of syntactic categories and the head lemmas in each pattern. Finally a lexical entry is constructed for each verb and SCF combination.

## Corpora

In order to make valid comparisons between SCF frequencies in CDS against those in adult speech it is necessary to first ensure that the corpora are controlled for all other variables. Roland and Jurafsky [84] have shown that there are subcategorization differences between written and spoken corpora and, furthermore, that subcategorization is affected by genre and discourse type. Hence, we use only spoken data for both corpora and restrict data to face-to-face conversation between family members and friends.

To ensure sufficient data for subcategorization acquisition, we have had to use an American English source for the CDS corpus although we had a British English source for the adult speech corpus. However, we do not expect this to be a problem: Roland *et al* [85] have shown that subcategorization probabilities are fairly stable across American vs. British English corpora; finding any exceptions to be the result of subtle shifts in verb sense due to genre rather than the dialect. The two corpora that were investigated are described below:

**Child-Directed Speech—CHILDES1 Corpus** The CHILDES database [63] contains transcripts (and also media data) collected from conversations with young children. Most of the transcripts record spontaneous conversational interactions. The speakers involved are mostly young, monolingual, normally developing children talking with their parents or siblings. There are also transcripts from “*bilingual children, older school-aged children, adult second-language learners, children with various types of language disabilities, and aphasics who are trying to recover from language loss*” [63]. The transcripts cover 26 different languages.

The CDS (or CHILDES1) corpus has been created from several sections of the CHILDES database: Demetras1 [32]; Demetras2 [31]; Higginson [45]; Post [81]; Sachs [87]; Suppes [99]; Warren-Leubecker [106]. These sections of the database contain natural interactions between a child and caretaker (average child age 2;7). Speakers are both male and female, from a variety of backgrounds and from several locations around the USA. Child speech has been removed from the corpus and there is no reading. The corpus contains 534,782 words and has an average utterance length of 4.8 words.

**Adult Speech—BNC Corpus** The British National Corpus (BNC) [56] is a 100 million word collection of samples of written and spoken language from a wide range of sources, designed to represent a wide cross-section of current British English, both spoken and written.

Our adult speech corpus has been manually selected from the demographic part of the spoken BNC such that it contains friend/family interactions where no children were present. The speakers were recruited by the British Market Research Bureau and come from a variety of social backgrounds. Speakers are both male and female, from several locations around the UK and all have an age of at least 15. Conversations were recorded unobtrusively over two or three days, and details of each conversation were logged. The corpus contains 835,461 words and has an average utterance length of 7.3 words.

## 2.1.5 Methods of SCF Analysis

For each corpus, verbs with more than 50 occurrences were identified. Subsequently, a set of up to 5000 verb-occurrence-utterances was extracted for each of these verbs. In practice the set size was often much smaller than the maximum 5000. This was due to the highly Zipfian nature of verb distributions in corpora; most verb types occur extremely infrequently in language (see e.g. Korhonen [51] for a discussion). Verb sets containing less than 50 utterances were not used since experience has shown that SCF acquisition from such small sets is unreliable. To make the results comparable, an equal number of utterances were used per verb per corpus. Hence, set size was often constrained by CHILDES1, which was the smaller of the two corpora.

Both qualitative and quantitative methods were used to compare the data in two SCF lexicons. Korhonen and Krymolowski [52] found that, when comparing subcategorization frame distributions, similarity measures vary in their robustness depending on the noise in the original data. Consequently, similarity between SCF distributions in the lexicons has been examined using several measures of distributional similarity (described below). In the following  $p = (p_i)$  and  $q = (q_i)$  where  $p_i$  and  $q_i$  are the probabilities associated with  $SCF_i$  in the two distributions.

**Intersection:** the intersection of non-zero probability SCFs in  $p$  and  $q$  [61].

$$IS(p, q) = \frac{2 * com(p, q)}{supp(p) + supp(q)}$$

where  $supp(p)$  = the number of SCFs with non-zero  $p_i$  and  $supp(q)$  similarly.  $com(p, q)$  is the number of SCFs with both non-zero  $p_i$  and non-zero  $q_i$ .

**Rank correlation:** calculated by first ranking the SCFs in each distribution by probability and then finding the Pearson correlation,  $corr()$ , between ranks [96]. More specifically, if  $p_i$  is the  $k$ th smallest of the  $p$ 's then define  $r_i^p$  to be equal to  $k$ , (similarly for  $q$ ).

$$RC(p, q) = corr(r^p, r^q)$$

Rank correlation lies in the range  $[-1; 1]$ , with values near 0 denoting a low degree of association and values near -1 and 1 denoting strong association.

**Cross entropy:** a measure of the information needed to describe a true distribution  $p$  using a model distribution  $q$ . Cross entropy is minimal when  $p$  and  $q$  are identical.

$$CE(p, q) = \sum_i -p_i \log(q_i)$$



**Kullback-Leibler distance:** a measure of the additional information needed to describe  $p$  using  $q$ . KLD is always  $\geq 0$  and  $= 0$  only when  $p \equiv q$ .

$$KLD(p||q) = CE(p, q) - H(p) = \sum_i p_i \log\left(\frac{p_i}{q_i}\right)$$

where  $H(p)$  is the Shannon entropy of  $p$ .

**Jenson-Shannon divergence:** a measure which relies on the assumption that if  $p$  and  $q$  are similar, they are close to their average [61].

$$JS(p, q) = \frac{1}{2} [KLD(p||\frac{p+q}{2}) + KLD(q||\frac{p+q}{2})]$$

**Skew divergence:** smoothes  $q$  by mixing with  $p$  [55].

$$SD(p, q) = KLD(p||\alpha * q + (1 - \alpha) * p)$$

$SD(p, q)$  approximates KLD as  $\alpha \rightarrow 1$ . In this work  $\alpha = 0.99$ .

## 2.1.6 Differences in Verb Types

Before conducting the SCF comparisons, the 100 most frequent verbs in the BNC corpus versus the CHILDES1 corpus are examined in order to obtain a more complete picture of the differences between the two data. It was discovered that some verbs tend to be frequent in both corpora, e.g. *go, get, think, like, make, come, take*. However, closer analysis of the data revealed large differences. In general, simple action verbs (e.g. *put, look, let, sit, eat, play*) are more frequent in CHILDES1, while mental state verbs (e.g. *know, mean, suppose, feel, seem*)—which tend to have richer argument structure—are more frequent in BNC. The 40 most frequent verbs in the two corpora are listed in Figure 2.2 in the order of their frequency, starting from the highest ranked. Mental state verbs have been roughly grouped and highlighted in bold. For a list of the top 100 verbs see Appendix B.

Notice that in general the mental state verb counts are much higher in the BNC corpus; the exceptions are the verbs **want, try** and **need**. These verbs appear to be intrinsically tied with the demands of the child (in the case of **want, need**) or with gaining the corporation of the child (in the case of **try**). This could be seen to back the claims of Pine [72] who claims that the main purpose of CDS is to simply engage the child in conversation.

## 2.1.7 SCF Comparison

A subset of the constructed lexicons were compared for subcategorization similarities between the BNC corpus and CHILDES1 corpus. To obtain reliable results, we restricted our scope to 104 verbs; those for which the total number of sentences analysed for SCFs was greater than 50 in both corpora, and which were thus less likely to be affected by data sparsity problems during SCF acquisition.

The average number of SCFs taken by studied verbs in the two corpora proved quite similar, although verbs in BNC took on average a larger number of SCFs (17) than those in CHILDES1 (13). However, we found that most verbs (regardless of their frequency in the corpora) showed

Rank	BNC	n	CHILDES1	n
1	<i>get</i>	5000+	<i>go</i>	5000+
2	<i>go</i>	5000+	<i>be</i>	5000+
3	<i>say</i>	5000+	<i>do</i>	5000+
4	<i>be</i>	5000+	<i>see</i>	4200
5	<b>know</b>	5000+	<i>put</i>	4037
6	<i>do</i>	5000+	<i>get</i>	4018
7	<b>think</b>	4074	<b>want</b>	3411
8	<i>see</i>	2852	<i>can</i>	3409
9	<b>like</b>	2827	<i>let</i>	2771
10	<i>can</i>	2710	<i>look</i>	2585
11	<i>come</i>	2602	<b>think</b>	2280
12	<b>want</b>	2148	<b>like</b>	2038
13	<b>mean</b>	2078	<b>know</b>	1768
14	<i>look</i>	1930	<i>say</i>	1755
15	<i>put</i>	1776	<i>come</i>	1693
16	<i>take</i>	1443	<i>make</i>	1692
17	<i>tell</i>	1122	<i>okay</i>	1593
18	<i>make</i>	1092	<i>take</i>	1356
19	<i>use</i>	1016	<i>eat</i>	1172
20	<i>will</i>	1007	<i>give</i>	990
21	<i>give</i>	920	<i>play</i>	944
22	<i>buy</i>	590	<i>tell</i>	860
23	<i>leave</i>	548	<i>find</i>	661
24	<i>keep</i>	545	<i>happen</i>	581
25	<i>pay</i>	543	<i>sit</i>	580
26	<i>let</i>	536	<i>read</i>	571
27	<b>remember</b>	517	<b>remember</b>	563
28	<i>work</i>	495	<b>try</b>	556
29	<b>suppose</b>	489	<i>fall</i>	546
30	<i>play</i>	477	<i>will</i>	537
31	<i>talk</i>	475	<b>need</b>	531
32	<i>ask</i>	469	<i>hold</i>	527
33	<i>find</i>	464	<i>turn</i>	492
34	<i>start</i>	445	<i>call</i>	439
35	<b>need</b>	443	<i>talk</i>	426
36	<i>call</i>	431	<i>thank</i>	408
37	<b>try</b>	430	<i>show</i>	404
38	<i>eat</i>	394	<i>wait</i>	395
39	<i>hear</i>	370	<i>bring</i>	389
40	<i>stop</i>	345	<b>mean</b>	379

Figure 2.2: 40 most frequent verbs in adult speech (BNC) corpus vs. child-directed speech (CHILDES1) corpus

	CHILDES1 vs. BNC
intersection	0.608
rank correlation	0.463
KL distance	1.022
cross entropy	2.698
JS divergence	0.083
skew divergence	0.533

Figure 2.3: Average similarity values: BNC vs. CHILDES1

substantially richer subcategorization behaviour in the BNC than in CHILDES1. A total of 78 frame types were hypothesised for the 104 studied verbs in the BNC, while 69 were hypothesised in CHILDES1. The intersection between the frame distributions in the corpora was not large (0.61). The maximum possible intersection in this case is 0.92 (when the SCFs in CHILDES1 are always a proper subset of those in the BNC); the figure of 0.61 indicates that CHILDES1 is actually substantially different to the BNC.

The distributions of SCFs in the two corpora are fairly different. In order to compare distributions we have included only a verb's SCFs whose relative frequency is higher than a defined threshold (in this case 0.015); this should remove some noise from the data. We looked at several measures when comparing subcategorization frame distributions since similarity measures vary in their robustness depending on the noise in the original data [52]. However, in this case, all the measures show the same lack of similarity (see Figure 2.3). For instance, a rank correlation measure lies in the range of -1 to 1 with values near 0 denoting the lowest degree of association; here there is only a weak rank correlation between the frames in the distributions (0.46). The Kullback-Leibler distance is 0 when two distributions are identical; the value 1.02 denotes a low degree of correlation. The cross entropy would have a value of 1.68 if the distributions were identical; the value of 2.70 again shows low correlation. Neither JS distance nor skew divergence have any significance as stand alone values but are included here for comparison with a second SCF study of child speech later in this chapter.

Thorough qualitative analysis of SCF differences in the two corpora reveals reasons for these differences. The most basic SCFs (e.g. intransitive and simple NP and PP frames; which describe *he slept*, *he ate an apple* and *he put the book on the table*) appear equally frequently in both corpora. The same is true for prepositional and nominal complements (*she asked him his name*, *he put the toy in the box*). However, a large number of more complex frames are either very low in frequency or altogether absent in CHILDES1. For example, the verb *hear* appears only in the following kind of constructions in CHILDES1:

1. *I heard you* ( 24 NP )
2. *I heard* ( 22 INTRANS )
3. *I heard that you came* ( 106 S-SUBJUNCT )

while in BNC it also appears in the following kind of constructions:

1. *I heard it from him* ( 49 NP-PP )

2. *Can you hear this out?* ( 76 NP-PART )
3. *I heard about it* ( 87 PP )
4. *I heard him singing* ( 35 NP-ING-OC )

Several types of SCFs are poorly covered or largely absent in CHILDES1. Many of these were frames involving sentential, adjectival and predicative complementation (e.g. *they admit that they did it* ( 97 PP-THAT-S ), *he painted the car black* ( 25 NP-ADJP ), *I considered him foolish* ( 26 NP-ADJP-PRED )). However, particle constructions (e.g. *I picked up the ball* ( 76 PART-NP )) are well covered. The total number of subjunctive constructions (e.g. *I want that you stop now* ( 106 S-SUBJUNCT )) was much higher in CHILDES1 than in BNC; as was the number of infinitival constructions (e.g. *you wanted to go* ( 112 TO-INF-SC )). For a full listing of all acquired frames per verb see Appendix C and for a comparison of the total number of acquired SCFs see Appendix D.

While the SCF differences seem fairly big, they are perhaps not altogether arbitrary. Rather, they seem to be correlated with different verb senses and SCFs typically permitted by the senses. To gain a better understanding of this, we looked into Levin's taxonomy [59] which divides English verbs into different classes on the basis of their shared meaning components and similar syntactic (mostly subcategorization) behaviour. For example, in Levin's resource, verbs such as *fly*, *move*, *walk*, *run* and *travel* belong to the same class since they not only share a similar meaning but also take similar SCFs.

By grouping verbs together into their Levin's classes, it was noticed that the SCFs within a Levin class in CHILDES1 were a subset of those in the BNC for the same Levin class. For example, Levin classes that take multiple sentential and predicative complements took a small range of those SCFs in CHILDES1 and a greater number in the BNC. In the light of this small scale investigation with Levin classes, it seems that to gain a better understanding of SCF differences in adult and CDS speech and the role of SCFs in language acquisition, it would be useful, in the future, to investigate to what extent SCF learning is mediated by the sense of the predicate and its membership in classes such as Levin's.

## Observations

A great number of subjunctive and infinitival constructions were present in CHILDES1. This can be explained by the fact that the semantic content of CDS is mostly concerned with the child's desires and wishes or with giving commands to the child; for instance, the phrase *want to* counts for almost half of the infinitival constructions. The prevalence of these structures is therefore a consequence of the subject matter.

In general, the empirical results shown here, obtained from SCF analysis of large-scale data, suggested that CDS is not only significantly simpler but also syntactically very different than speech between adults.

Some prevailing theories of language acquisition (e.g. that of Landau & Gleitman [54]) suggest that verb SCFs provide convergent evidence on the meaning of a verb. These theories rely on the assumption that the frames provided in a child's input are adequately diverse to support learning. Meanwhile, Snow [94] suggests that CDS plays an important role in the facilitation of acquisition. If Snow and Landau & Gleitman are both correct then we would perhaps expect to find that CDS is diverse in terms of its SCFs. However, previous small-scale empirical studies

(e.g. [94]) suggest that, while CDS is quite complex (displaying, for example, the full range of conventional indirectness), it is syntactically much simpler than speech between adults. Perhaps then, the role of CDS is to encourage the acquisition of simple frames, providing a basis from which more complex frames may be developed.

The fact that there is not significant correlation between the SCFs in two corpora is a little surprising; one might expect CDS to contain a subset of adult speech's SCFs. However, as the small scale experiment with Levin classes suggests, the SCFs seem nevertheless correlated via related verb meanings. While this issue requires further investigation, it is important to also note that some CHILDES1 SCFs absent in BNC may not be altogether absent in adult speech. Due to the Zipf-like nature of the SCF data, they may just occur in adult speech with a very low frequency and may have been cut off by our relative frequency threshold on frames. If this turns out to be the case after further larger scale experiments, it would indicate that most CDS SCFs are indeed a subset of those in adult speech but the frequencies of the SCF in the two corpora differ substantially.<sup>2</sup>

The results may also support Valian's [102] findings that 4% of parental replies to children are ungrammatical, and 16% grammatical but not fully acceptable (examples from our CDS corpus include "*play this together?*", "*another one missing.*"). Such utterances explain at least partly why there are SCFs present in the CHILDES1 lexicon that are missing from the BNC.<sup>3</sup> Valian also found that adults tend to reply to children using an utterance which is lexically and structurally similar to the child's sentence (5% verbatim, 30% structurally similar). Since child speech at 2;7yrs (the average age of child subject in our CDS corpus) is usually simpler than adult speech ([69] and [13]) such repetition could help to boost the relative frequency of simpler frames in the CHILDES1 lexicon.

## 2.2 Linguistic Output

Language development and production in children has been more widely studied than the input they learn from. Data has been collected in a variety of forms:

**Diary Studies:** the traditional method for tracking child language development. The studies usually involve a single child whose caretaker keeps a diary of when new constructions etc. are first produced. The diary is generally kept over a long period of time (several years).

**Large Sample Studies:** involve a large number of children and generally record an experiment looking for a specific language phenomena.

**Longitudinal Language Sampling:** generally involves regularly recording/transcribing of a single child's language; for instance, a child may be recorded for a few hours once a week. Longitudinal studies are generally continued over a period of several months or years.

---

<sup>2</sup>It should also be noted that, despite painstaking attempts to ensure transcriptions were standard across corpora, any inconsistencies may cause systematic erroneous SCFs to be acquired.

<sup>3</sup>Future work will check the precision of the CDS corpus against a suitable gold standard.

## 2.2.1 Stages of Language Acquisition

The most generally cited stages of language acquisition are “Brown’s Stages” [13]. These stages provide a framework within which to discuss and predict the path that normal language development usually takes. Brown’s stage boundaries are defined by the learner’s average utterance length. Utterance length, in this case, is calculated as the average number of morphemes per sentence. It is this particular measurement of average utterance length that differentiates Brown’s Stages from other stage definitions. The alternative measure would be the average number of words per utterance but this measure is less sensitive to changes in the acquired grammar; inflectional morphology, for instance, is bound to words. The utterance “Ducks eating bread” has 5 morphemes whereas “Duck eat bread” has only 3 but both phrases contain exactly the same number of words. Brown refers to his measure as the Mean Length of Utterance or MLU and to make it reliable he defined a set of criteria to specify precisely what constitutes a morpheme. There are five Brown Stages specified by ranges of MLU’s:

**Stage 1 (1.0–2.0 MLU):** At around 12 months old children start to make reference to objects, people and actions that are important to them. The child’s first words do not necessarily sound much like adult words and are usually produced in isolation. At this part of Stage 1 the child will generally use a raised intonation to indicate that they are asking a yes/no type of question.

At around 18 months vocabulary starts to increase rapidly and the child starts to produce two-word utterances. Words which express negativity such as “no”, “gone” and “allgone” are generally the first to be used in two-word combinations. These are followed by two-word combinations of the type *agent+action* and *action+object* i.e. “I sit” or “See baby”. Children’s two-word combinations are similar across cultures [77]:

Children tend to announce when objects appear, disappear, and move about, point out their properties and owners, comment on people doing things and seeing things, reject and request objects and activities, and ask about who, what, and where.

During late Stage 1 (at the time when about half of the child’s utterances are two words long) three and four word utterances begin to be introduced. Now children begin to form declarative statements of the form *subject+verb+object* and start to introduce the prepositions “in” and “on” as well as the conjunction “and”.

**Stage 2 (2.0–2.5 MLU):** During Stage 2 grammatical morphemes appear such as “ed”, “ing” and “s”. The child overextends their use during most of this stage; possibly using words like “go-ed”. Possessive pronouns start to be used as well as preliminary auxiliary verb forms (such as “wanna” and “gonna”). Question forms also become more complex during this stage. The child begins to use “what”, “where” and “why” and also uses a rising intonation at the end of a phrase to indicate a yes/no question.

The child is now more aware of the interactive nature of language. She will attempt to repair utterances that were not understood and is able to sustain a topic for one or two turns.

**Stage 3 (2.5–3.0 MLU):** Possessive pronouns and the modal verbs “can”, “will” and “do” begin to be used consistently and the copular and auxiliary forms of “to be” are introduced.

The child also starts using a few quantifiers such as “two” and “some”. The question set now expands to incorporate “who” and “how”.

The child is still only capable of holding a topic for one or two turns. Its primary method for doing this is to repeat part or all of the utterance of its conversation partner. Conversational repairs now involve trying to use another word for that which has been misunderstood, even if sometimes it is an inappropriate word.

**Stage 4 (3.0–3.75 MLU):** The child starts to use past tenses of the common modal verbs such as “could”, “would” and “should”. Contractions like “didn’t” become common in negative sentences. “When” questions begin to be asked.

The child has now learnt that short pauses indicate that conversation exchange will continue whereas long pauses indicate that responses will not be forthcoming. Conversation can be sustained for more than two turns by the end of this stage and has become more interactive. The child has become aware of what information the listener will need and tries to provide it.

**Stage 5 (3.75–4.5 MLU):** More than half of the grammatical morphemes have been mastered by Stage 5. The remaining morphemes (such as the irregular past tense, regular and irregular third person) are mastered just after the child has finished Stage 5 and has MLU of greater than 4.5. The child now understands superlatives but not comparatives. She is also using negative past tense forms like “weren’t”.

Question forms tend to now have properly inverted words e.g. “Is he playing?”. The child is also starting to use indirect requests although the majority of utterances still refer to direct requests.

### **Sources of Variation Amongst Children**

One child’s language development may vary from another’s for several reasons. For instance, variation may be due to biologically determined individual capacities or abilities of the child that lead to preferences for (or better skill at) particular linguistic subsystems [46]. Further variation may be caused by environmental effects. These range from obvious differences such as the need to hear a language to speak it, to more subtle differences such as the effect of the frequency of specific language forms [46] in the input.

### **2.2.2 Internal vs. External Language**

Data-collection for the study of language development is a laborious but straightforward task; the interpretation of the data, however, is not so straightforward. First, children are not born with a fully developed articulatory system and consequently their productions, particularly in the early years, are likely to be hindered by their linguistic performance. Furthermore, children (as well as adults) understand a great many lexemes and constructs that they do not ever use themselves. These points would suggest that we are likely to under-estimate a child’s competence. However, it is also easily possible to over-estimate a child’s linguistic competence by generalising from a few isolated productions; a child that has produced the word *dogs* has not necessarily learnt how to form plural inflections. Psycho-linguistic experiments (such as

Gleason's previously mentioned wug test [40]) can help to provide evidence for linguistic competence that could not easily be recognised from child production transcripts alone, but since these experiments are difficult to design and time consuming to perform, we cannot expect to discover a child's entire internal grammar from experimentation. By using the following guidelines we can reduce the risk of over- or under-estimate a child's linguistic competence:

**The Competence Assumption** assumes that a child's linguistic performance is relatively close to their linguistic competence. We can never presume a linguistic construct is known by the child until there is evidence for it in the child's performance [46].

**The Productive Performance Criteria** states that a child's linguistic production can only be said to have been produced by a rule when there is evidence that the rule is productive, i.e. when the child creates new instances of the structure under discussion [46].

### 2.2.3 Child Production and Computational Modelling

From a computational linguistic viewpoint an interesting question is can a computer model stages of language acquisition echoing those of a child? It is one matter to design a model that acquires a language but another to design one that does so in the same way as a child. A realistic model of acquisition (i.e. one that learns from real data and is attempting to learn a real grammar) could hopefully exhibit similar learning stages to that of a child. However, we can not expect too much of a model in this respect since there are other factors to take into account. A child's cognitive functionality is developing in parallel with language ability. It is therefore possible that some aspects of language are completely barred to a learner until a particular milestone in cognitive development has been reached. Having passed such a milestone a learner would be capable of processing language units in such a way that new grammatical information can be acquired. A computer model does not develop in this manner. Its ability to manipulate symbols remains constant throughout the learning process.

At the very least a computational model of language acquisition should be generally supportive of the observed stages of child language acquisition. I refer to this property as the *developmental compatibility* of the model. In Chapter 6 I shall evaluate the developmental compatibility of the learning model presented in this thesis with reference to Brown's stages and the evidence of produced subcategorization frames from a child speech corpora (presented below).

### 2.2.4 Subcategorization Frames in Child Speech

In the following section the subcategorization frames found in child speech are compared with the frames found in linguistic input. In particular the CDS corpus (CHILDES1 corpus) and the adult speech corpus (BNC corpus) are contrasted to a child speech corpus. The child speech corpus is constructed from all of the child utterances that were removed from the CHILDES database to construct CHILDES1. The child speech corpus is referred to as CHILDES2. It contains 273831 words and 81086 utterances; the MLU over the whole corpus is 3.4 and the children's average age is 2;7.<sup>4</sup>

---

<sup>4</sup>Note that future work will investigate the interaction and alterations in SCF distribution within CDS and child speech at different child ages; the sizes of the currently available corpora are not sufficient for such a task.



Rank	BNC	n	CHILDES1	n	CHILDES2	n
1	<i>get</i>	5000+	<i>go</i>	5000+	<i>go</i>	3018
2	<i>go</i>	5000+	<i>be</i>	5000+	<i>get</i>	2361
3	<i>say</i>	5000+	<i>do</i>	5000+	<i>want</i>	2069
4	<i>be</i>	5000+	<i>see</i>	4200	<i>put</i>	1682
5	<i>know</i>	5000+	<i>put</i>	4037	<i>see</i>	1188
6	<i>do</i>	5000+	<i>get</i>	4018	<i>let</i>	971
7	<i>think</i>	4074	<i>want</i>	3411	<i>make</i>	849
8	<i>see</i>	2852	<i>can</i>	3409	<i>eat</i>	781
9	<i>like</i>	2827	<i>let</i>	2771	<i>look</i>	765
10	<i>can</i>	2710	<i>look</i>	2585	<i>take</i>	699
11	<i>come</i>	2602	<i>think</i>	2280	<i>okay</i>	623
12	<i>want</i>	2148	<i>like</i>	2038	<i>know</i>	563
13	<i>mean</i>	2078	<i>know</i>	1768	<i>come</i>	496
14	<i>look</i>	1930	<i>say</i>	1755	<i>need</i>	467
15	<i>put</i>	1776	<i>come</i>	1693	<i>give</i>	442
16	<i>take</i>	1443	<i>make</i>	1692	<i>play</i>	427
17	<i>tell</i>	1122	<i>okay</i>	1593	<i>like</i>	391
18	<i>make</i>	1092	<i>take</i>	1356	<i>do</i>	335
19	<i>use</i>	1016	<i>eat</i>	1172	<i>fall</i>	329
20	<i>will</i>	1007	<i>give</i>	990	<i>read</i>	315
21	<i>give</i>	920	<i>play</i>	944	<i>say</i>	287
22	<i>buy</i>	590	<i>tell</i>	860	<i>sit</i>	272
23	<i>leave</i>	548	<i>find</i>	661	<i>thank</i>	254
24	<i>keep</i>	545	<i>happen</i>	581	<i>hold</i>	253
25	<i>pay</i>	543	<i>sit</i>	580	<i>sleep</i>	224
26	<i>let</i>	536	<i>read</i>	571	<i>cause</i>	207
27	<i>remember</i>	517	<i>remember</i>	563	<i>open</i>	199
28	<i>work</i>	495	<i>try</i>	556	<i>watch</i>	194
29	<i>suppose</i>	489	<i>fall</i>	546	<i>be</i>	182
30	<i>play</i>	477	<i>will</i>	537	<i>find</i>	169

Figure 2.4: 30 most frequent verbs in adult speech (BNC) corpus vs. child-directed speech (CHILDES1) corpus vs. child speech (CHILDES2) corpus.

### 2.2.5 Differences in Verb Types

Figure 2.4 shows the 30 most frequent verbs in all three corpora; the child speech corpus is shown in the third column. Notice that, as predicted by Brown, the most frequent verbs tend to describe the properties of objects and their owners (e.g. *take, give, fall, hold, open*); or the actions of people (e.g. *get, put, make, say, watch*); or relate to the child's desires (*want, eat, need, sleep*).

### 2.2.6 SCF Comparison

The distribution of subcategorization frames found in the child speech corpus (CHILDES2) was compared to both the child-directed speech corpus (CHILDES1) and adult speech corpus (BNC). As in the previous experiment the verbs selected for SCF comparison were chosen

	BNC	CHILDES1	CHILDES2
Average No. of Frames	15	11	10
Total No. of Frames	73	67	58

Figure 2.5: Frame comparison: BNC, CHILDES1, CHILDES2.

	CHILDES1 vs. BNC	CHILDES1 vs. CHILDES2
intersection	0.608	0.621
rank correlation	0.463	0.492
KL distance	1.022	0.682
cross entropy	2.698	1.986
JS divergence	0.083	0.074
skew divergence	0.533	0.404

Figure 2.6: Average similarity values: BNC vs. CHILDES1, CHILDES1 vs. CHILDES2

because of their quantity of occurrence in the corpora; each corpus had to contain at least 50 utterances for a verb if it was to be analysed.

The average number of SCFs taken by the verbs studied in the child speech (CHILDES2) corpus was 10; this compared to an average of 15 SCFs for the same verbs in the adult speech (BNC) corpus and an average of 11 SCFs for the child-directed speech (CHILDES1) corpus. In the child speech corpus a total of 58 frames were acquired for the 78 verbs studied, while the CHILDES1 and BNC hypothesised 67 and 73 respectively. The adult speech corpus is clearly the most syntactically rich of the three and, as one might expect, the child speech corpus is the least so. It is interesting, however, to notice that the child-directed speech corpus sits right between the two (see Figure 2.5).

The average similarity values, shown in Figure 2.6, clearly indicate that the distribution of SCFs in the child speech corpus is much closer to that of the child-directed speech corpus than the adult speech corpus. To remind the reader, rank correlation values closer to 1 denote a stronger similarity; KL distances closer to 0 indicate the same. If the distributions were identical the cross entropy would have a value of 1.68 for CHILDES1 vs. BNC and 1.30 for CHILDES1 vs. CHILDES2. Both JS divergence and skew divergence are comparative measures; the fact that JS divergence and skew divergence are lower for CHILDES1 vs. CHILDES2 indicates that they are more closely correlated than CHILDES1 vs. BNC.

The average intersection between SCFs was higher for the child speech and child-directed speech corpora (0.621) than for the child speech and adult speech corpora (0.590). This difference might suggest that children more readily pick up the frames occurring in speech directed to them than from the speech between adults around them. However, we can not be sure of this since we are unable to tell how greatly the similarity between CHILDES1 and CHILDES2 is due to the utterances being two halves of the same conversations. This shall have to be investigated further.

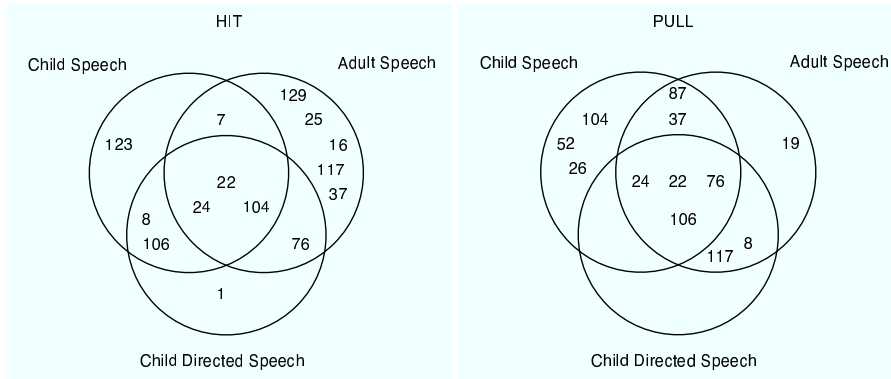
For some verbs, *how-to* constructions (such as *he explained how to do it* (17 HOW-TO-INF)) and verb particle plus infinitive constructions (such as *he set out to win* (139 SC-INF, PRT, SUBTYPE EQUI)) are found both in CHILDES1 and CHILDES2 but are entirely missing from the adult speech corpus. It is possible that these frames, missing from the adult

speech, might have been acquired by the SCF system in error due to the issue, as reported by Valian [102], that adults often (30% of the time) reply to a child in a manner that structurally echoes the child's last utterance, even if that utterance is ungrammatical.

The total set of acquired SCFs for the child speech corpus form a proper subset of those acquired for the child-directed speech corpus; largely this also applies on a per verb basis. However, we can not infer from this that children never generalise incorrectly from one verb to another since the phenomena might be both too rare and subtle to be picked up in this data.

Figure 2.7 shows the SCFs acquired from the three corpora for the verbs *hit* and *pull*. Notice that, for the verb *hit*, the two SCFs that are common to child speech and child-directed speech but not to adult speech (*to hit pleases you* and *hit it that it breaks*) are most likely an illustration of parental imitation. Also notice that, for the verb *pull*, the SCFs common to child speech and adult speech but not child-directed speech (*he hit her the ball* and *he pulled it to him*) are answers to questions (e.g. *what is he doing?*); this suggests that the child-directed speech is also being used to engage children in conversation as suggested by Pine [72].

The corpora studies presented here show that child-directed speech is syntactically less diverse than speech between adults and that it contains a similar distribution of SCFs to child speech. Why parents alter their language in this way is still a matter of debate; are they attempting to match the child's linguistic competence or trying to aid acquisition by providing simpler constructions to learn from? Either way, it is clear that a good deal of the language that children are exposed to contains a reduced set of less complex SCFs. Furthermore, children produce language using a small and less complex set of SCFs than adults. Adhering to the Competence Assumption these observations lead towards the conclusion that human learners acquire and use complex syntactic constructions only after simpler constructions have been learnt (concurring with Brown's stages); the acquisition model presented will operate in a similar manner.



No.	SCF	Example
1	ADJP	his ball hit high
7	S-SUBJ-NP-OBJ	that she hit amazed them
8	TO-INF-SUBJ-NP-OBJ	to hit pleases him
16	HOW-S	he hit where she told him to
19	ING-NP-OMIT	her hair needs pulling
22	INTRANS	you hit
24	NP	he hit her
25	NP-ADJP	he hit the ball hard
26	NP-ADJP-PRED	it pulled her hard
37	NP-NP	he hit her the ball
52	NP-S	he pulled it so it would go higher
76	PART-NP	he pulled his socks up
87	PP	he pulled it to him
104	S	it hits that it knocks it off
106	S-SUBJUNCT	pulls it that it breaks
117	NP-NP-up	he pulled him up a chair
123	MP	he hit 5
129	SFIN, AGR S[FIN +], SUBTYPE EXTRAP	that it hits counts

Figure 2.7: SCFs acquired for the verbs *hit* and *pull*

# Chapter 3

## Learnability and Learning Models

A normal child will learn the language of their environment and is theoretically capable of learning any language; a child living in an English speaking environment learns English but the same child brought up in a Chinese speaking environment would learn Chinese. Assuming that low level brain functionality is standard in normal humans, it follows that it must be possible to learn any member of the class of natural languages using a fixed set of mental mechanisms. Formally, a class of languages is *learnable* if there exists a learning function that can successfully learn the grammar of any language in the class. The definition of *successful learning* will depend on the learning model.

**Gold's Model:** Gold [42] modelled language learning as an infinite process in which a learner is presented with an infinite stream of sentences of the target language. Every time the learner encounters a new sentence a guess is made as to the grammar of the target language on the basis of all the sentences encountered so far.

Gold made two assumptions about the input stream: first, only grammatical sentences of the target language appear in the stream; secondly, every sentence of the target language eventually appears in the infinite stream.

Formally, we have:

1.  $\Omega$ —an hypothesis-space of grammars (or grammar-space);
2.  $\Phi$ —a sample set of grammatical sentences;
3.  $F$ —a learning function that maps finite subsets of  $\Phi$  (languages) to elements of  $\Omega$ .

$$G_i = F(\{s_0, s_1, s_2, \dots, s_i\})$$

where  $G_i \in \Omega$  and  $s_0, s_1, \dots, s_i \in \Phi$

Gold's criterion for success was if the learner reached a point after which its guess no longer changed; i.e. if the learner converged on a grammar. He called this *identification in the limit*. Formally, let  $S_i$  be the set of sentences  $\{s_0, s_1, s_2 \dots s_i\}$  then  $F$  converges to  $G \in \Omega$  if there exists an  $n \in \mathbb{N}$  such that for all  $i \geq n$ ,  $F$  is defined on  $S_i$  and is equal to  $G$ .

Unfortunately, using this criterion, it is impossible to ever distinguish if learning has been successful, since the learner may always guess a new grammar when presented with the next sentence.

**Statistical Models:** Other work has shown that learning can be modelled as a statistical competition between all the grammars within the hypothesis-space (see [12] and [109] for natural language examples). Using a statistical model,  $F$  returns a probability distribution over the grammar-space. The distribution represents each grammar’s fitness to describe the sentences encountered so far. The current grammar,  $G_i$ , (after encountering  $S_i$ ) is selected according to the distribution. Under this model of learning, we can give a similar criterion for success:  $F$  converges to  $G \in \Omega$  if there exists and  $n \in \mathbb{N}$  such that for all  $i \geq n$ ,  $F$  is defined on  $S_i$  and returns a distribution over  $\Omega$  such that  $G$  is most likely.<sup>1</sup>

For learning in general, Gold provides us with a model from which specific details must be fleshed out; in particular the definition of the hypothesis-space and the learning function. Linguistically, the spectrum of learning models is marked at one end by nativism and by empiricism at the other. The pure nativist viewpoint asserts that the input stimulus presented to children is too impoverished for successful acquisition. Consequently nativists assume the existence of some innate linguistic knowledge or language faculty (referred to by Chomsky as the Universal Grammar). Models derived from this viewpoint will generally have a comparatively small grammar hypothesis-space since it is constrained by the innate knowledge. Learning functions for nativist models tend to be algorithmic in nature—analysing an input string and then moving systematically from one grammar to the next within the small hypothesis-space. The pure empiricist, on the other hand, believes that language may be acquired without the aid of any innate language faculty. For empiricist models, the hypothesis-space is unconstrained and consequently very large. Learning functions for empiricist models tend to be highly statistical and consequently data demanding—identifying the target grammar only after a great deal of data has been observed.

### 3.1 Principles and Parameters

Chomsky is a particular advocate of nativism. He claims [22] that, given the “relatively slight exposure” to examples and “remarkable complexity” of language, it would be “an extraordinary intellectual achievement” for a child to acquire a language if not specifically designed to do so. His *Argument from the Poverty of the Stimulus* suggests that if we know  $X$ , and  $X$  is undetermined by learning experience, then  $X$  must be innate. For an example consider structure dependency in language syntax:

A question in English can be formed by inverting the auxiliary verb and subject noun-phrase: (1a) “*Dinah was drinking a saucer of milk*”; (1b) “*was Dinah drinking a saucer of milk?*”

Upon exposure to this example, a child could hypothesise infinitely many question-formation rules, such as: (i) *swap the first and second words in the sentence*; (ii) *front the first auxiliary verb*; (iii) *front words beginning with w*.

The first two of these rules are refuted if the child encounters the following: (2a) “*the cat who was grinning at Alice was disappearing*”; (2b) “*was the cat who was grinning at Alice disappearing?*”

If a child is to converge upon the correct hypothesis unaided, she must be exposed to sufficient examples so that all false hypotheses are refuted. Unfortunately such examples are not

---

<sup>1</sup>Note that the statistical nature of competitive models makes them robust to noise and amenable to language change. These properties will be discussed later in this chapter.

readily available in child-directed speech; even the constructions in examples (2a) and (2b) are rare [57]. To compensate for this lack of data, Chomsky suggests that some principles of language are already available in the child's mind. For example, if the child had innately "known" that all grammar rules are structurally-dependent upon syntax, she would never have hypothesised rules (i) and (iii). Thus, Chomsky theorises that a human mind contains a Universal Grammar which defines a hypothesis-space of "legal" grammars.<sup>2</sup> This hypothesis-space must be both large enough to contain grammars for all of the world's possible languages and small enough to ensure successful acquisition given the sparsity of data. Language acquisition is the process of searching the hypothesis-space for the grammar that most closely describes the language of the environment. With estimates of the number of living languages are around 6800 [35] (and this being only a sample of all possible languages), it is not sensible to model the hypothesis-space of grammars explicitly, rather it should be modelled parametrically. Language acquisition is then the process of setting these parameters.

A grammar can be located in the hypothesis-space by its properties: if set  $A$  contains all the grammars that produce a *subject-verb-object* (SVO) ordering and set  $B$  contains all grammars that produce *subject-drop*, then the grammar of Italian, for example, lies in the intersection of  $A$  and  $B$ . Once enough properties are specified then a grammar can be uniquely identified in the intersection of all the sets it belongs to. The hypothesis-space itself is defined by all the possible combinations of properties. The properties are defining the grammar-space parametrically; and the innate knowledge of the existence of these properties is an example of Chomsky's Universal Grammar (UG). To be clear, Chomsky speculated that the UG is composed of *principles* (the aspects of language that are common to all languages) and *parameters* (language variables to be observed during the process of acquisition). Here, the learner is provided with the innate knowledge that all grammars are definable by the presence or absence of particular properties, and also the knowledge of what those properties are; these are Chomsky's language principles. The grammar search is refined by observing properties in the language environment; observing a property is analogous to setting one of Chomsky's parameters.

The properties of a grammar may be represented as an array that contains as many elements as there are property sets; the value in each element indicating whether or not the grammar in question belongs to the property set associated with that element. Each grammar will have a unique configuration of the array. Language acquisition is the process of finding the right configuration from all the possibilities; a search on a search-space of size  $2^N$  where  $N$  is the number of property sets (henceforth referred to a parameters). Acquisition would be most efficient when the number of parameters needed to distinguish between all grammars is small; i.e. when each set is maximally discriminatory. With 6800 world languages, at least 13 parameters are required for a realistic model of language acquisition ( $2^{13} = 8192$ ). However, it is unlikely that children are predisposed to maximally discriminate between all the world's languages. Chomsky [24] suggested that parameters should represent points of variation between languages; following this idea it has suggested that perhaps 30+ parameters are required. If this is the case then the UG is describing a hypothesis-space of over a billion grammars.

---

<sup>2</sup>Discussion of structural dependence as evidence of the Argument from the Poverty of Stimulus is illustrative, the significance being that innate knowledge in any form will place constraints on the hypothesis-space.

<i>A</i>	<i>B</i>	<i>C</i>
1	1	0

Figure 3.1: A property array; identifying the location of grammar  $G$  in the intersection of sets  $A$  and  $B$  (where set  $A$ , for example, contains all those grammars exhibiting property  $A$ ).

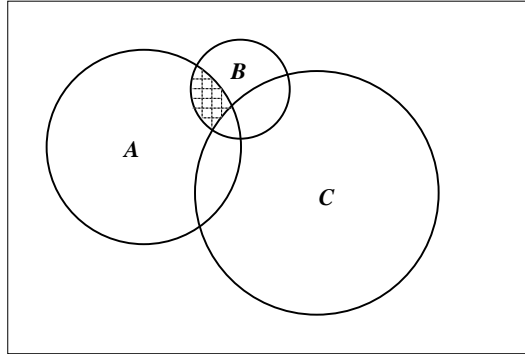


Figure 3.2: The hypothesis-space; the shaded area being the location of the same grammar,  $G$ .

### 3.1.1 Triggers

To converge upon the grammar of the environment language the learner needs to set the values of all  $N$  parameters. The learner can set an element when evidence has been provided from language examples to show that a property is (or is not) exhibited; these language examples are referred to as triggers. In the best case a learner need only be exposed to  $N$  triggers before acquisition is complete.

But what exactly constitutes a trigger? And how does a learner know whether an utterance contains a trigger. Clark [28] suggests that every parameter is associated with a trigger that causes the learner to set the parameter's value immediately upon exposure to it. Fodor's Structural Trigger Learner (STL) [38] adopts this definition of a trigger.

Triggers are easily found if the properties the learner is looking for are independent of each other. However, there is a conflict between using independent properties and using sufficient properties to uniquely identify all grammars; Clark [27] and Kayne [50] estimate that between 30 and 40 properties are needed (which is significantly more than the minimal 13). This has the consequence that in practice, triggers are difficult to come by; very often language examples contain ambiguous evidence for the properties that the learner is looking for [27]. For instance, sentences of English (subject-verb-object ordering) might be misclassified as a subject-object-verb ordering with an active  $V2$  (verb movement), as in German. When faced with an ambiguous trigger the learner has two choices:

- (a) choose one of the possible interpretations and set the parameter values according to that interpretation;
- (b) ignore the trigger entirely and wait for an unambiguous one.

The first approach is adopted in Gibson and Wexler's Triggering Learning Algorithm (TLA) [39]. This learner analyses incoming triggers using the current property settings and modifies their



values if they conflict with the properties of the incoming trigger. Unfortunately, this method of dealing with an ambiguous trigger has been found to put the learner at risk of never converging on the correct grammar.

Fodor's original Structural Triggers Learner (STL) [38] recognises ambiguous triggers by carrying out some structural analysis and then ignores them entirely; instead waiting for unambiguous triggers. This method avoids converging on the wrong grammar but relies on the learner encountering unambiguous triggers for every property. As mentioned above, the existence of such triggers is questionable [27]. Also, the STL learner is wasteful of the language examples it is provided with; exerting a lot of effort in the structural analysis of triggers only to throw most of them away — so, neither approach **(a)** or **(b)** are without problem.

Dresher and Kaye [33] suggest that the problem of ambiguous triggers can be avoided if there is a constraint placed on the order in which the learner looks for properties; so the learner ignores triggers demonstrating property *B* until a trigger demonstrating property *A* has been observed. Dresher and Kaye's model was designed to learn metrical phonology but we can extend the general idea; the hope being that by placing a careful ordering on the properties, the decision of which interpretation to use for an ambiguous trigger can be made for free. For example, consider a trigger that may be interpreted as either exhibiting properties *B* and *C* or properties *A* and *D*. If the ordering on properties is *A* before *B* before *C* before *D*, then a learner will choose the second interpretation (properties *A* and *D*) and set the element relating to property *A*. Note, that the learner can not set property *D* element because properties *B* and *C* have not yet been observed. Having seen property *A*, the learner can now wait for a trigger exhibiting property *B*. Using this type of model increases the size of the learner's innate knowledge since now it not only knows what properties to look for but also which order to look for them in.

### 3.1.2 The Triggering Learning Algorithm

Gibson and Wexler [39] designed the Triggering Learning Algorithm (TLA) to investigate learning over a grammar-space defined by binary valued parameters. The original work investigated learning in a grammar-space defined by three binary parameters: specifier, complement and *V2* (see Figure 3.3). The specifier and complement parameters defined word order. The specifier parameter is concerned with the location of the specifier (determiner etc.) with regard to its head (in this case the main verb). Setting the specifier parameter to 1 for example, indicates that the language is specifier-final (i.e. specifiers occur after the head-word); setting the same parameter to 0 would then indicate a specifier-initial language. This follows similarly for the complement (or object) parameter. The *V2* parameter (when set) indicates that verb movement is allowed from its base position as defined by the word-order parameters to second position. In other words, if the *V2* parameter is set then the surface order of the words may be altered even though the base order is unchanged. This phenomena is seen in the Germanic languages. Using these parameters an English type subject-verb-object language and Germanic type language would be represented as in Figure 3.4.

This setting of the parameters allows for the base word order to be complement-initial but for the verb to take 2nd position in root declarative clauses.

- ... dass Christopher der Ball kauft.
- Christopher kauft der Ball.

PARAMETER NAME	VALUE 0	VALUE 1
Specifier	initial	final
Complement	initial	final
V2	on	off

Figure 3.3: The 3-parameters of the Triggering Learning Algorithm.

LANGUAGE	SPECIFIER	COMPLEMENT	V2
English	0 ( <i>initial</i> )	1 ( <i>final</i> )	0 ( <i>off</i> )
German	0 ( <i>initial</i> )	0 ( <i>initial</i> )	1 ( <i>on</i> )

Figure 3.4: English and German type languages in Gibson and Wexler’s parameter-space.

The TLA is error driven; put simply, its function is to randomly modify a parameter value every time the learner cannot parse the current input. The algorithm is bound by two constraints: the first is the Single Value Constraint [26], which ensures that the TLA only ever considers grammars that differ from the current hypothesis by one parameter; and the second is the Greediness Constraint [26], which ensures that the current hypothesis is only changed if there is something to gain in doing so. For an  $n$ -parameter-space a single iteration of the algorithm proceeds as follows:

1. Attempt to parse the current input utterance,  $S_1$ :
  - (a) **if**  $S_1$  can be parsed **then** leave the parameters unchanged;
  - (b) **else** randomly select and toggle one parameter (with probability  $1/n$  of selecting each parameter).
    - i. **if**  $S_1$  can be parsed with the new settings **then** adopt the new settings;
    - ii. **else** revert to the original parameter settings.

The TLA has the following problems (which will be discussed further in Chapter 6):

**local maxima:** the phenomena where a non-target grammar is reached from which the learner can never reach the target grammar;

**ambiguous triggers:** some input examples can be parsed by more than one grammar in the hypothesis-space (i.e. there is more than one way to configure the parameter settings to achieve a successful parse). By choosing grammars blindly (randomly choosing a parameter to toggle) an “unwise” grammar could be adopted that, in the worst case, might lead to a local maxima;

**noise:** the algorithm is deterministic and will modify parameters even when the input example is erroneous.

### 3.1.3 The Structural Trigger Learner

The Structural Triggers Learner (STL) [90] addresses some of the problems with ambiguity that are faced by the TLA. The model provides a set of schematic treelets as part of its Universal Grammar. A treelet can be thought of as a subtree that is used in the derivation of a full parse. The STL associates a treelet with each feature of the language and consequently they may be thought of as parameters. A language is identified by the subset of treelets that are required to parse the language.

During learning, all Universal Grammar treelets are available to the algorithm. If an unambiguous parse is found that requires a treelet that has not already been collected into the subset required for the language, then that treelet is adopted. There are several versions of the STL which differ in how to handle ambiguous parses. The algorithm with the least processing is called the *Weak STL*; it proceeds as below:

1. Attempt to parse the current input utterance,  $S_1$ , with current subset of treelets:
  - (a) **if**  $S_1$  can be parsed **then** the subset of treelets remains unchanged;
  - (b) **else** attempt to parse  $S_1$  with all treelets in the Universal Grammar.
    - i. **if** at some point during the parse there is a choice of treelets then disregard  $S_1$  for all learning.
    - ii. **else** adopt all novel treelets that have been used in the parse into the subset of treelets.

The STL allows several treelets (parameters) to be learnt during a single parse. This is useful for speedy learning. However, the wait for an unambiguous parse to learn from might be very long; especially at the early stages of acquisition when all the treelets are “in play”.

Although able to deal with ambiguity, the STL (like the TLA) can not handle noisy input. The following section discusses the problem of noisy input data and explains an important requirement—that models should be robust to noise.

## 3.2 Noise and Learning Models

A child is exposed to evidence of her target language that must exclusively belong to one of three possible classes: positive evidence is information that describes which utterances are allowed in the target language; negative evidence is information that describes which utterances are *not* allowed in the target language; errors are pieces of information that have been mistakenly classified as either positive or negative evidence.<sup>3</sup>

**Positive Evidence:** Positive evidence can be presented to a child in the form of example utterances spoken by proficient members of her language community. A large proportion of the language a child is exposed to will be positive evidence. In fact Pinker [77] goes as far as saying that “effectively the input to the learner *only* includes grammatical sentences”. Following Gold’s paradigm [42], a child hypothesises her language based on accumulated positive evidence; all previously heard utterances form a subset of the current hypothesised language. Learning is completed once the hypothesised language no longer needs to be updated.

---

<sup>3</sup>This discussion is previously published as [20].

**Negative Evidence:** Negative evidence might be provided by correcting a child when they produce an ungrammatical sentence. Evidence of this sort could be used to constrain the child from hypothesising a grammar that describes a superset of the target language. A child that is only ever exposed to positive evidence can not be corrected if she hypothesises a grammar that is too unspecific. However, in general children do not learn from correction [14]. This indicates that there must be some other mechanism for constraining the hypothesised language: a possible solution is Minimum Description Length learning [83] (where the child only ever hypothesises the simplest language that describes the evidence seen so far).

**Errors and Noise:** Lacking any discerning information, a child is likely to assume that all the utterances she hears are grammatical and therefore constitute positive evidence. However, spoken language can contain ungrammatical utterances, perhaps in the form of interruptions, lapses of concentration or slips-of-the-tongue. When a child misclassifies such utterances as positive evidence, an error has occurred.

Situations also arise where entirely grammatical sentences can produce an error because of misclassification due to indeterminacy. For instance, indeterminacy of the input may lead to noise within the parameter settings of the Universal Grammar [27]: sentences of English (subject-verb-object ordering) can be misclassified as a subject-object-verb ordering with an active  $V2$  (verb movement) parameter, as in German.

In general, any environment that contains ambiguity can introduce errors. Often a child is exposed to input from more than one target language and yet manages to learn one (or more) consistent grammar(s), rather than a grammar that allows all possible combinations of the sampled input. Specific examples of this include diglossia [53] and language change [60]. In such situations, misclassification of one of the input languages is an example of an error. Furthermore, there are documented situations where a conflicting and inconsistent input is “regularised” and fashioned into a single consistent generative grammar; as in the cases of rapid creolization [5] and the acquisition of sign language from a source of non-expert signers (the case of Simon [68]).

Errors of these sorts are always accidental and lead to the false assignment of an utterance to the class of positive evidence. A child is somehow able to cope with such erroneous assignments.

A malicious error would occur if a deliberate attempt was made to confound the child’s acquisition of language. An example might be if the child is corrected on her grammatically correct utterances or if she is deliberately exposed to utterances that are ungrammatical. Malicious errors are unlikely in spoken language but deliberate errors do occur in some very early child-directed speech in the form of nonsense words, and also in later child-directed speech due to parental imitation of children’s ungrammatical utterances.

To summarise: in an ideal learning situation a learner would have access to an oracle [103] that can correctly identify every utterance heard as either a positive evidence, negative evidence or noise. However, a child learning its first language can not rely on receiving *any* negative evidence; at best she can hope to receive positive evidence or, more realistically, positive evidence that is also noisy. Without an oracle a child is, of course, unaware of when an erroneous utterance has been encountered. She is also unaware of when an ambiguous utterance has caused an error to occur. Any simulation or explanation of language acquisition should therefore attempt

to learn from every utterance it encounters and should be robust to errors whether caused by erroneous utterances or general ambiguity.

### 3.2.1 How do Errors Affect Learning?

Consider a simplified learning problem, a game for two players: the first player, the exemplar, thinks of a set of numbers that can be defined by a rule, such as multiples of two  $\{x|x/2 \in Z\}$ ; the second player, the guesser, attempts to reproduce the set by discovering the rule which defines it. The only information available to the guesser is a continuous stream of examples provided by the exemplar.

A possible scenario might be that the first two examples provided are 4 and 8. At this point the guesser may well hypothesise that the set contained multiples of four  $\{x|x/4 \in Z\}$ . The guesser doesn't need to revise this hypothesis until she encounters an example that breaks the rule. If the guesser ever arrives at the hypothesis that the set contains multiples of two she'll never have to revise her hypothesis again.<sup>4</sup>

Now if the same game was played in a noisy room or with a distracted exemplar the guesser might receive erroneous examples. For instance, in attempting to guess the set  $\{x|x/2 \in Z\}$ , the guesser may have heard the examples 2, 4, 7, 8,... If the guesser classifies all the examples as positive evidence then there are two possible outcomes: either the guesser fails to find a rule or she hypothesises the wrong rule.

The guesser could only arrive at the correct hypothesis if she is aware that some of the examples may be erroneous. The guesser's best chance of winning is to figure out which hypothesis is *most likely* to be correct. Before the game begins the guesser will consider all hypotheses equally likely. As the game proceeds the working hypothesis is selected if it is the most likely given the accumulated evidence. In other words the guesser must adopt a statistical methodology to cope with the erroneous examples.

### 3.2.2 Introducing a Hypothesis Bias

Now, the interesting problem is: how many erroneous examples could the guesser encounter before she is completely unable to guess the rule. The answer lies in the type and the frequency of the errors encountered as well as any bias the guesser may have towards certain hypotheses. For example, consider the set  $\{x|x/5 \in Z\}$ . With no examples the guesser considers all hypotheses equally likely. After being exposed to the examples 15, 30, 45 the guesser has to consider the hypotheses  $\{x|x/5 \in Z\}$  and  $\{x|x/3 \in Z\}$  to be equally likely. Too many erroneous examples that happen to be multiples of three but not multiples of five may lead the guesser to eventually choose the later and incorrect hypothesis,  $\{x|x/3 \in Z\}$ . However, if the guesser had been initially biased towards the  $\{x|x/5 \in Z\}$  hypothesis, perhaps because her favourite number is five, then she may have continued to select this hypothesis despite the accumulated evidence. In terms of language acquisition, hypothesis biases would need to be part of the innate principles of the Universal Grammar.

---

<sup>4</sup>The situation where the guesser hypothesises a rule that produces a superset of the original set is not discussed here. This situation would be avoided by allowing the exemplar to provide negative evidence (i.e. examples of numbers not in the set) or by constraining the hypothesis-space.

### 3.3 Statistical Models of Language Acquisition

The Triggering Learning Algorithm and the Structural Triggers Learner are deterministic models. The next parameter to be set is ascertained from the current trigger and the current parameter settings only; the models have no memory of the utterances that have been previously seen. Since parameters can be set on the basis of evidence from a single trigger, the models rely on the input to the learning system being free from error. Considering the evidence above, these models can not be considered realistic.

A statistical model can “consider” the distribution of evidence before committing to a course of action. In terms of the principles and parameters paradigm, a parameter is only set when enough evidence is accumulated; when there is not enough evidence the parameter remains unchanged. This might be achieved by setting a lower bound on the number of triggers that need to be encountered in support of a parameter before that parameter is set. As long as the target parameter values are the most statistically likely in the data then the learner will eventually obtain enough evidence to set the parameters correctly despite any erroneous utterances. Furthermore, a statistical parametric learner has some ability to deal with the problem of ambiguous triggers. For a simple illustration consider a language which exhibits property  $A$  and not property  $B$  and a trigger from that language that may be interpreted either as exhibiting property  $A$  or as exhibiting property  $B$ . If the learner has been keeping track of the distribution of properties over all triggers then it will have hopefully seen evidence for the property  $A$  many more times than evidence for property  $B$ . The learner can choose the correct interpretation on the basis of the accumulated evidence and update the parameters’ values and distribution of properties accordingly.

For a more complicated but realistic example, consider again Gibson and Wexler’s parameter-space and the confusable SVO  $V2$ - (English-like) and SOV  $V2+$  (German-like) languages. The 3 strings *subj-verb*, *subj-verb-obj* and *subj-aux-verb* occur in both languages and may be parsed with parameter settings [010] or [001] (see Figure 3.3 for a reminder of the parameter meanings). However, for the English-like language the 5 strings *subj-aux-verb-obj*, *adv-subj-verb*, *adv-subj-verb-obj*, *adv-subj-aux-verb-obj* and *subj-aux-verb-obj* also occur. These strings can not parse if the  $V2$  parameter is set to 1. If all strings are equally likely then we will get more parses with  $V2$  set to 0 than set to 1. Eventually there will be enough accumulated evidence to confidently set the  $V2$  parameter to 0.

Unfortunately, this method will not work as efficiently in a realistic model because there is not a uniform distribution over sentence constructions in real data. The constructions *subj-verb* and *subj-verb-obj* are the most common in both English and German. Consequently, we might be a long time waiting for enough evidence to set the  $V2$  parameter. If the learner has a method for detecting ambiguity (such as the STL), then the parameter setting rate could be increased by adding greater weighting to evidence acquired from unambiguous triggers.

In general, if we assume that the majority of language provided to a learner is grammatical, then linguistic evidence from erroneous utterances will become statistically insignificant and will be ignored. Furthermore, a statistical model has scope for dealing with errors caused by ambiguity; the only downfall being the need for a larger amount of input data.

By using a statistical model we are required to endow the learner with an innate ability for statistical retention of data. How this is achieved and exactly what is retained is a matter of debate. Should the learner analyse the triggers and record the distribution of all possible interpretations (a possible statistical extension to Fodor’s STL) or perhaps, attempt to parse the

$$\begin{array}{ll}
\text{if } G_i \text{ parses } s \text{ then} & p'_i = p_i + \gamma(1 - p_i) \\
& p'_j = (1 - \gamma)p_j \quad \text{if } j \neq i \\
\\
\text{if } G_i \text{ does not parse } s \text{ then} & p'_i = (1 - \gamma)p_i \\
& p'_j = \frac{\gamma}{N-1} + (1 - \gamma)p_j \quad \text{if } j \neq i
\end{array}$$

Figure 3.5: Bush and Mosteller’s Linear Reward-Penalty scheme: given an input sentence  $s$  and total number of grammars  $N$ , the learner selects a grammar  $G_i$  with probability  $p_i$ .

trigger with the current settings, rewarding parameters for a successful parse and penalising them when unsuccessful? In the second case, how does the learner discover which parameters contributed to a failed parse and subsequently decide which parameters to punish and which to reward? Briscoe [12] has implemented a Bayesian Incremental Parameter Setting (BIPS) algorithm which addresses this issue. In this model a partially-ordered hierarchy of parameters are each associated with a probability. During learning, the probability associated with a “successful” parameter is increased while the probabilities of “unsuccessful” parameters are reduced. A parameter is considered successful if it is involved in a valid parse of a trigger. Probabilities are evaluated using Bayes theorem,

$$p(a|b) = \frac{p(b|a)p(a)}{p(b)}$$

which has the added benefit of allowing priors to be assigned to parameters before learning commences. By assigning priors the hypothesis-space can be biased as described in section 3.2.2. Another reward-penalty learner, implemented by Yang [109], is discussed below.

### 3.3.1 The Variational Learner

Yang’s Variational Model involves associating each parameter with a weight representing the prominence of that parameter in the learner’s hypothesis-space; consequently this associates a probability to every grammar in the hypothesis-space. Grammar selection for parsing (and also production) is a function on the probability distribution over the grammars. The parameters in the selected grammar are either rewarded or penalised depending on whether or not it is able to parse the incoming utterance. Rewards and penalties are calculated in accordance with the Linear Reward-Penalty Scheme [15] (see Figure 3.5 for a general Linear Reward-Penalty Scheme for competing grammars): a successful grammar has all of its parameters probabilities increased; an unsuccessful grammar has all its parameters probabilities decreased. Acquisition is complete when the grammar weights become approximately constant. The method is passive and requires the learner to do nothing other than select a grammar from the distribution and then reward or punish the parameters as appropriate.

Even in a noiseless environment, this method of setting parameter weights will never allow them to become constant; they will tend towards a limit. In reality, due to the noise inherent in speech, the parameter weights are likely to fluctuate within a tolerance threshold.

A problem with the Variational Model is that it does not reward or penalise parameters individually. When a grammar parses, all parameter’s probabilities are increased; so every time the successful grammar is not actually the target grammar then some parameters have been rewarded unjustifiably. Contrastingly for unsuccessful grammars, unless the grammar’s param-

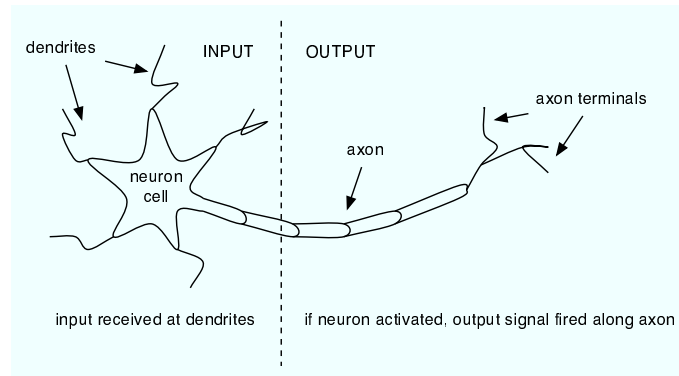


Figure 3.6: A schematic diagram of a typical neuron.

eters are the inverse of the target grammar then some parameters are being penalised unjustly. Yang admits that this approach is naive but hopes that “in the long run, the correct parameter values will prevail”. Experimentation backs this claim.

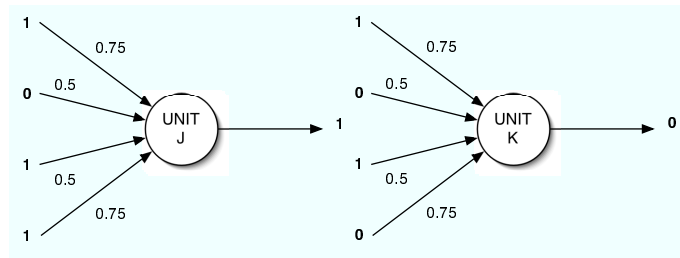
Another problem with this learner is that the problem of local maxima is not definitely solved. Yang’s model selects a grammar for parsing according to the current probability distribution: if that particular grammar is successful its parameters are rewarded while all others are penalised. The learner has no method of knowing if the input utterance was ambiguous (being parsable by another grammar), unlike in the STL for example. So what happens if a grammar that describes a superset of the actual target takes an early lead in the probability distribution? This grammar will be selected more often than any other grammar as a possibility for parsing and it will always have its parameters rewarded since it is capable of parsing all sentences that the actual target could. The more it is rewarded, the more often it is selected, until finally the learner converges on the wrong target. During the entire learning process the learner was unaware that the utterances it received were ambiguous; and since the superset grammar can never be penalised, it could never realise its mistake.

These problems aside, Yang’s model has been shown to be a formally specified model capable of making quantitative predictions over developmental language patterns, as well as giving an explanation for continuity between adult and child language. Furthermore, he is able to “formalise historical linguists’ intuition that grammar competition is a mechanism for change”. It is also possible to envisage Yang’s model extending to other problems in acquisition. In fact, the model could be used in any situation where learning can be expressed as a competition between forms. For an example, consider the acquisition of a phoneme set: adults identify and use only the phonemes of their own language despite being born able to recognise the phonemes of all languages; such a learning process is easily modelled with a distribution on a phoneme-space.

### 3.4 Connectionist Modelling

The models discussed so far have all fallen within the principles and parameters paradigm. However, a study of language models is not complete without a discussion of connectionism. Connectionist models are often closely associated with empiricism since they begin as a “blank state”. There is no concept of an innate Universal Grammar. Everything is learnt from linguistic





$$\begin{aligned}
 \text{Net input to unit } J &= (1 * 0.75) + (0 * 0.5) + (1 * 0.5) + (1 * 0.75) \\
 &= 0.75 + 0 + 0.5 + 0.75 \\
 &= 2.0
 \end{aligned}$$

$$\begin{aligned}
 \text{Net input to unit } K &= (1 * 0.75) + (0 * 0.5) + (1 * 0.5) + (0 * 0.75) \\
 &= 0.75 + 0 + 0.5 + 0 \\
 &= 1.25
 \end{aligned}$$

Figure 3.7: The operation of a unit in a neural network.

input.

The models are based upon the architecture of neural networks in the brain. Each network consists of units (which are analogous to neurons) that are connected together by weighted links (modelling synapses). Each unit is assigned an activation level which determines whether the unit will produce data at its output: much like the activation energy required by a neuron in order to fire. Thus neural networks consist of four parts (units, activations, connections, and connection weights) each of which corresponds to a particular structure or process in biological neural networks (Figure 3.6).

Figure 3.7 shows the operation of a typical unit. To determine the net input to the unit, all the input values are scaled by the weight of their connection and then summed. In Figure 3.7 both units  $J$  and  $K$  have a simple activation threshold of 1.5: if the net input is  $\geq 1.5$  the output will be 1; and if  $> 1.5$ , the output will be 0. The net input to unit  $J$  exceeds the activation threshold so the unit responds with output 1. The input to unit  $K$  does not exceed the activation threshold; the unit responds with output 0.

More generally, connection weights are found to be expressed both positively and negatively; a negative weight being taken to represent inhibition of the receiving unit due to the activity of a sending unit. The function for calculating activation may be any function that is dependent on all the weighted input. Furthermore, the unit output need not simply toggle at a threshold value but may fall within the range  $[0, 1]$  as determined by a function on the weighted input (e.g. a simple scaling on the activation function). Whatever the activation method, it is assumed that all units in a particular network operate in the same way.

Units in a neural net are usually arranged in several layers (see Figure 3.8): the input layer, the output layer and one or more hidden layers. The units in the input layer receive input stimulus in the form of an encoded pattern. The activated output of these units is transferred to units in the hidden layer. Finally the signal propagates to the output level where the response signal is recorded.

Training a neural network involves finding the right set of connection weights to prompt the correct response for a given input. The simplest method of training is called *back-propagation*.

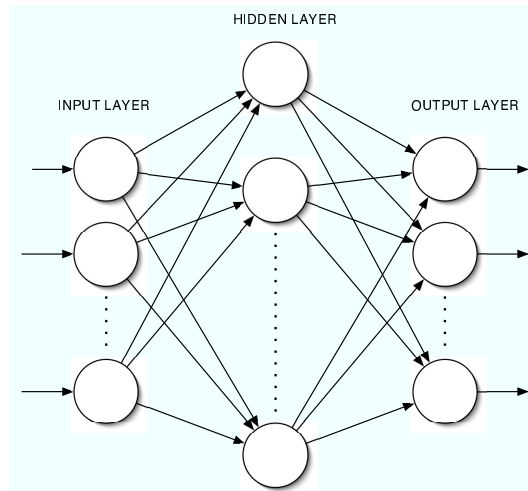


Figure 3.8: The architecture of a neural network.

Initially the weights of the net are set randomly. Next, a pattern from the training set is placed at the input units. This pattern is allowed to propagate through the hidden layer to the output units where the resulting pattern is compared to the desired output. Then all the weights in the net are adjusted slightly in the direction that brings the output pattern closer to the desired output. This process is continued with the rest of the training set. In order to achieve the required connection weights the network will often have to undergo hundreds or thousands of iterations of training.

### 3.4.1 Connectionism as a Model of Language Acquisition

The first connectionist models used to model language were concerned with verb morphology. Rumelhart and McClelland [86] trained a neural network to predict the past tense of English verbs. The network performed well, trained on a set of mostly irregular verbs. It was even shown to have generalised for patterns within the irregular verbs that were not in the training set. The network was then trained on an additional set of verbs containing mostly regular forms (420 verbs in total). During training, the network was shown to have over-generalised regular forms or even combine regular and irregular forms: producing *breaked* or *broked* instead of *broke*. A similar phenomena is seen in children learning English; between the ages of 2 and 5 children appear to over-regularize often producing such errors:

“It *breaked*”—Naomi, age 2, The Sachs Corpus, CHILDES.

Over-generalisation in the network was corrected with sufficient training; after 200 iterations all verbs had been correctly learnt. The similarity between over-generalisation in children and Rumelhart and McClelland’s neural network has prompted the suggestion that this network is a good model of acquisition. However, this claim is fiercely opposed by classical computational linguists and by those who tend towards algorithmic models. Pinker and Prince [79] argue that the network has a problem generalising rules and that this is a failing of neural networks in principle. The *words and rules* algorithm provided by Pinker [78] assigns regular verbs to the default -ed rule. Meanwhile, irregular verbs pre-empt the default rule; their past tenses are memorised by rote. Hence, Pinker’s model will never give an incorrect past tense for a regular

verb. This is not the case for the Rumelhart and McClelland’s model which might very well predict the past tense of *think* to be *thunk* by “generalising” from the pattern of the irregular verb *sink/sunk*; whether or not this is a useful trait is still up for discussion.

Indeed, the apparent inability of neural networks to master general rules is a major criticism of connectionist modelling. However, Elman’s work using a recurrent network [34] has demonstrated that it is possible for a neural network to exhibit behaviour similar to that of a very simple ruled-based context-free grammar (albeit on a very small vocabulary of only 23 words). It should be noted that the success of this model relies heavily on the manner in which it is trained (much like the model presented in Chapter 5).

Another problem that need to be tackled within connectionist theories is that of “one-time” learning. Humans display an ability to learn from single events (such as touching a hot object). Since connectionist training techniques require many iterations they are as yet unable to explain this phenomena. However, it should be remembered that a unit in a neural network is only a crude model of the neuron. It is easy to forget that there are many types of neuron in the brain whereas a neural network will generally only use one type of unit. Furthermore, the effects of neurotransmitters are not modelled.

Neural networks are robust to noisy data in a similar manner to the statistical models; they can exhibit robustness because they are both recording information about the previously seen data. For a neural network, the values of its connection weights are a consequence of every input that has been observed. Training requires an enormous amount of data, the majority of which will be grammatically accurate. The networks “work” by generalising from patterns and are consequently reasonably unaffected by the occasional erroneous input during training. Neural networks do not have to be robust to ambiguous triggers (as the parametric learners do) since there are no parameters to be set. Moreover, connectionist models are robust to “physical” destruction of the model. If units are destroyed the gradual degradation of functionality is observed; responses are still appropriate, though somewhat less accurate.

It is perhaps unfortunate that research in language acquisition has often vehemently adhered to either connectionism or the principles and parameters paradigm. There is often an analogy that can be found between the two. For instance, the introduction of a hypothesis bias in a nativist model might also be modelled by initialising connection weights in a neural net (although the latter is certainly more difficult to get right). All said, the main reason for rejecting connectionism to model language acquisition is that the training is just too expensive (with regard to both data and computation).

### **3.5 Learning from Semantics**

Thus far our discussion of learning and learnability has focused on learning from strings (or surface-strings in the case of the TLA—see 3.1.2). This next section will investigate learning from the semantic representations associated with strings. Learning from semantic representations is of particular interest when modelling language acquisition; we might assume that if a child has attended to an utterance then she will have associated some semantic content to it. However, most of the work in this area has not been carried out with child language acquisition in mind; rather it has focused on the automated construction of natural language interfaces for data-base queries—and as such, has been generally concerned with fairly limited domains.

Some of the earliest work in this area attempted to learn case-role assignments (e.g. agent, patient) and was carried out using connectionist models (e.g. [65]). Subsequently, Mooney

and colleagues improved on these connectionist models using their semantic acquisition system called CHILL ([110], [111]) and its successor COCKTAIL [100] both of which employ Inductive Logic Programming techniques. These systems can train on a corpus of sentences paired with their corresponding semantic representations and induce a grammar (actually a shift reduce parser) that can subsequently be used to transform sentences directly into semantic representations. Very recently further work has been published that automatically induces a categorial grammar to map natural language sentences to lambda calculus encodings of their meanings [112]. The major difference between this work and that of Mooney is that CHILL and COCKTAIL require lexical semantics to be known before the commencement of learning (although Thompson and Mooney demonstrate how a lexicon for CHILL may be acquired in [101]).

All of the above work has been carried out on database query tasks: i.e inducing grammars to map sentences to database queries. This is a neat semantic learning problem since a) system evaluation for the task is straightforward—simply check that the system returns the correct answer to the natural language query; and b) the training data is reasonably easy to come by—it can be generated from users of the database in question with little extra effort. However, by the nature of database queries, this work has focused on (sometimes fairly artificial) language within a limited domain. As such, these methods are not easily extended to the task of real language learning. For instance, Zettlemoyer and Collins postulate trigger rules which generate only 8 categorial grammar categories for their induced categorial grammar. Real non-domain-specific language requires many more categories available to the lexicon: a categorial grammar manually constructed by Villavicencio [104] required 89 categories to describe just 2000 utterances of child directed speech (which, as shown in Chapter 2 is syntactically less complicated than speech between adults). Furthermore, for real language acquisition, all trigger rules must be considered to be part of the principles of a Universal Grammar—thus imposing innate burdens on the learner.

The basic principle of these semantic learners is very similar to the work presented in this thesis; they are all attempting to induce a grammar from a string paired with a semantic representation. The difference is in the application and implementation: the systems described above are trying to solve a functional problem (that of mapping natural language interfaces to database queries) whereas the work in this thesis attempts to be cognitively plausible and developmentally compatible with human learning.

## 3.6 Summary

In this chapter we have seen that for a language acquisition model to handle real data, it must have a method of dealing with noise. Both connectionist and statistical models are robust to noise because they record information about the language distribution as a whole; any model that determines its next state based only on its current state and the current input can not be robust to erroneous data.

A connectionist model, although most robust to noisy data, requires an enormous amount of data and computation for learning. Models that follow a more nativist theme (the principles and parameters learners) potentially require much less data since the hypothesis-space is constrained. However, to eliminate the possibility of reaching a local maxima these parametric models must also have some mechanism for recognising ambiguity.

The acquisition model presented in this thesis, will learn from input utterances paired with semantic representations and will induce a grammar for parsing. It will make some use of a Universal Grammar. However, the model will not employ parameters in the traditional sense, but instead make use of a memory module that keeps a statistical record of the syntactic constructs that have been learnt so far. As alluded to by the analysis of child speech and child-directed speech (see Chapter 2) the model learns incrementally. A benefit of this incremental learning is that it avoids many of the issues of ambiguous input (similar to the ordered parameter learner of Dresher and Kay).



# Chapter 4

## Categorial Grammar Learners and the Input they Receive

This chapter discusses the mechanisms of categorial grammar learners as well as the input they receive. First, Section 4.1 presents an overview of categorial grammars; including a description of common rules and the association between semantics and categorial grammar categories. Section 4.2 then discusses previous categorial grammar learners. We notice that the input provided to categorial grammar learners is not standard. For instance, the categorial grammar learner of Buszkowski ([16], [17]) learns from structures, whereas the learning system of Waldron/Villavicencio ([105], [104]) learns from string/semantic-form pairs. In order to determine the complexity and cognitive plausibility of various types of inputs, Section 4.3 introduces the concept of *sentence objects* (the term I use for conceptual elements that carry at least as much information as a string). Sentence objects are simply possible starting points from which learning can commence. A discussion is presented on the complexity of learning from different types of sentence objects and on their cognitive plausibility. Having selected the type of sentence objects we will use as an input for our categorial grammar learner, Section 4.4 presents a learning system that provides a cognitive model for creating such objects from semantic representations.

### 4.1 Categorial Grammars

#### 4.1.1 Classic Categorial Grammars

In a *classic* categorial grammar all constituents and lexical items are associated with a finite number of types. Types are formed from *primitive types* using two operators,  $\backslash$  and  $/$ . If  $Pr$  is the set of primitive types then the set of all types,  $Tp$ , satisfies:

$$\begin{aligned} Pr &\subset Tp \\ \text{if } A \in Tp \text{ and } B \in Tp, \text{ then } A \backslash B &\in Tp \\ \text{if } A \in Tp \text{ and } B \in Tp, \text{ then } A / B &\in Tp \end{aligned}$$

One member of  $Pr$  is the *sentence type*,  $s$ . All other members of  $Pr$  are referred to as the *variables*,  $Var$ , such that:

$$Pr = \{s\} \cup Var.$$

Members of  $Tp$  may be combined by considering them as arguments and functors. A type that is acting as a functor encodes the following information:

1. the type of the argument(s) the function takes;
2. the directionality of the argument(s) (i.e. the whether the argument can be found to the left or right of the function);
3. the type of the result.

Arguments are shown to the right of the operators and the result to the left. The forward slash operator (/) indicates that the argument must appear to the right of the function and a backward slash (\) indicates that it must appear on the left.

### Function Application

For a classic categorial grammar, types may be combined using the rules of function application:

$$\text{Forward Application}(>) : A/B \ B \rightarrow A \quad (4.1)$$

$$\text{Backward Application}(<) : B \ A \backslash B \rightarrow A \quad (4.2)$$

where  $A$  and  $B$  range over types.

A sentence has a valid parse if the lexical types (the types assigned to the words in the sentence) may be combined to produce a derivation tree with root  $s$  (the primitive sentence type). Consider a grammar using variable  $Var = \{np\}$ , (thus primitive types  $Pr = \{s, np\}$ , allowed types  $Tp = \{s, np, s \backslash np, s / np, s \backslash s \dots\}$ ) and lexicon,  $L_1$ , which associates words with types—a grammar that associates at most one type to each member of the lexicon is called a *rigid grammar* whereas a grammar that assigns at most  $k$  types to a member of the lexicon is a *k-valued grammar*.

$$L_1 = \left\{ \begin{array}{ll} \text{smudge} & \rightarrow np, \\ \text{chases} & \rightarrow (s \backslash np) / np, \\ \text{mice} & \rightarrow np \end{array} \right\}$$

Given  $L_1$ , the sentence *Smudge chases mice* is parsed as follows:

$$\frac{\frac{\text{Smudge} \quad \frac{\text{chases} \quad \text{mice}}{(s \backslash np) / np}}{np} \quad >}{\frac{np \quad s \backslash np}{s} <}$$

The types of the constituents in this parse are combined by acting as either functors or arguments. The type of *chases* for instance,  $(s \backslash np) / np$ , is acting as a functor; taking an  $np$  argument from the right to give the return type  $(s \backslash np)$ . For the purpose of this work, and for categorial



grammars in general, the primitive categories must be regarded to include minor syntactic information such as gender and number agreement. If inflectional information is explicitly required it may be provide via agreement features (usually represented as subscripts to the syntactic category) [98].

Note that classic categorial grammars have a simple correspondence with context-free grammars. Given a classic categorial grammar,  $G$ , with lexicon  $\Sigma$ , then the *range* of  $G$  is the set of types that  $G$  assigns to words in  $\Sigma$ , so  $Tp(G) = \{A \mid A \in range(G) \vee A \text{ is a subtype of } range(G)\}$ . The set of context-free grammar rules to describe  $G$  may now be defined as:

$$\begin{aligned} CF(G) &= \{B \rightarrow A \ B \setminus A \mid B \setminus A \in Tp(G)\} \\ &\cup \{B \rightarrow B/A \ A \mid B/A \in Tp(G)\} \\ &\cup \{A \rightarrow lex \mid lex \in \Sigma\} \end{aligned}$$

## 4.1.2 Combinatory Categorial Grammars

Real natural language can not be modelled using the rules of function application alone. Several other rules have been posited to provide more extensive syntactic descriptions. A selection are illustrated in Figure 4.1 where  $A, B, C, T \in Tp$  and  $|$  is a variable over  $\setminus$  and  $/$  (see [98] for an overview and also [12] for the rule of Generalised Weak Permutation). A grammar which employs any of these rules in addition to those of the classic categorial grammar, is called a *combinatory* categorial grammar.

### Function Composition

The rules of composition (4.3 and 4.4) allow non-constituents to be created. A constituent is considered to be a sequence of text that can be assigned a type using only the rules of function application. Consider the sentence "Somebody might eat you":

$$\begin{array}{cccc} \text{somebody} & \text{might} & \text{eat} & \text{you} \\ | & | & | & | \\ np & s \setminus np / s \setminus np & s \setminus np / np & np \end{array}$$

The sequence *eat you* for example is a constituent whereas *might eat* is not. The composition rules can be used to assign a type to *might eat* by allowing the argument of a function to be a function itself:

$$\frac{\frac{\frac{\text{Somebody}}{np} \quad \frac{\frac{\frac{\text{might}}{(s \setminus np) / (s \setminus np)} \quad \frac{\text{eat}}{(s \setminus np) / np}}{s \setminus np / np}}{s \setminus np}}{s}}{s}}{np} \quad \frac{\frac{\text{you}}{np}}{np}}{s \setminus np} > B >$$

Forward Composition	$(> B) : A/B \ B/C \rightarrow A/C$	(4.3)
Backward Composition	$(< B) : B \setminus C \ A \setminus B \rightarrow A \setminus C$	(4.4)
Forward Type Raising	$(> T) : A \rightarrow T / (T \setminus A)$	(4.5)
Backward Type Raising	$(< T) : A \rightarrow T \setminus (T/A)$	(4.6)
Forward Substitution	$(> S) : (A/B) / C \ B / C \rightarrow A / C$	(4.7)
Backward Substitution	$(< S) : B \setminus C \ (A \setminus B) \setminus C \rightarrow A \setminus C$	(4.8)
Forward Crossed Substitution	$(> S_x) : (A/B) \setminus C \ B \setminus C \rightarrow A \setminus C$	(4.9)
Backward Crossed Substitution	$(< S_x) : B / C \ (A \setminus B) / C \rightarrow A / C$	(4.10)
Generalised Weak Permutation	$(P) : ((A \mid B_1) \dots \mid B_n) \rightarrow ((A \mid B_n) \dots \mid B_1)$	(4.11)

Figure 4.1: A selection of the rules used in Combinatory Categorical Grammars.

## Type Raising

The type raising rules (4.5 and 4.6) allow arguments to become functions (or more precisely, allow arguments to become functions over functions over arguments). The utility of this is in capturing co-ordinate structures. For instance, consider the sentence *Andrew throws and Smudge chases the ball* (see Figure 4.2); the forward type raising rule can be used to transform the *np*-subjects (*Andrew* and *Smudge*) into  $s/(s\backslash np)$ 's, which may then be composed with the verbs (*throws* and *chases*) to allow co-ordination with the direct object, (*the ball*):

The unary type raising rules may be applied to any type at any point during the parse—providing the undesirable possibility of an infinite parse tree. To avoid this pitfall the use of the type raising rules is generally constrained. For instance, type raising can be restricted to only produce types that already exist in the lexicon; or restricted to act only on the lexicon; or restricted to use only as a back-off parsing option when all other rules fail (although this can be dangerous if there are ungrammatical strings).

## Function Substitution

The substitution rules (4.7, 4.8, 4.9 and 4.10) are used for parasitic gap constructions; allowing a single argument to be used by two functors. Consider the sentence “*Christopher watched without enjoying the match between Manchester United and Villarreal*”. Here *Christopher* is the argument of both *watched* and *without enjoying*.

$$\begin{array}{c}
 \frac{\frac{\frac{\frac{\textit{watched}}{(s\backslash np)/np}}{(s\backslash np)/np}}{(s\backslash np)/np}}{(s\backslash np)/np}}{np} \quad \frac{\frac{\frac{\frac{\textit{without enjoying}}{((s\backslash np)\backslash (s\backslash np))/np}}{(s\backslash np)\backslash (s\backslash np))/np}}{(s\backslash np)\backslash (s\backslash np))/np}}{(s\backslash np)\backslash (s\backslash np))/np}}{s\backslash np} < S_x \quad \frac{\textit{the match}}{np} > \\
 \frac{\frac{\frac{\frac{\frac{\textit{Christopher}}{np}}{(s\backslash np)/np}}{(s\backslash np)/np}}{(s\backslash np)/np}}{(s\backslash np)/np}}{s} <
 \end{array}$$

## Generalised Weak Permutation

The Generalised Weak Permutation (GWP) rule is a unary rule that allows arguments in a functional category to be reordered. After reordering, the permuted category may be able to combine with adjacent categories that were previously not compatible with the rules. The allowed permutations of arguments is limited to rotations of the original ordering; for a functional category of  $n$  arguments there will be only  $n$  permutations.

The GWP rule allows a flexible ordering on the constituents of oblique transitive verbs like *donate* whose original category is  $(s\backslash np)/np/pp$ . The allowed argument rotations would parse both sentences *Kim donated the money to charity* and *Kim donated to charity the money*.

The GWP rule can also be used to parse sentences containing unbounded dependencies (relationships between non-adjacent constituents), which includes many wh-questions and relative clauses. Figure 4.3 shows the sentence *Chickens make the eggs that we eat* and uses the GWP rule to parse the relative clause *that we eat*:

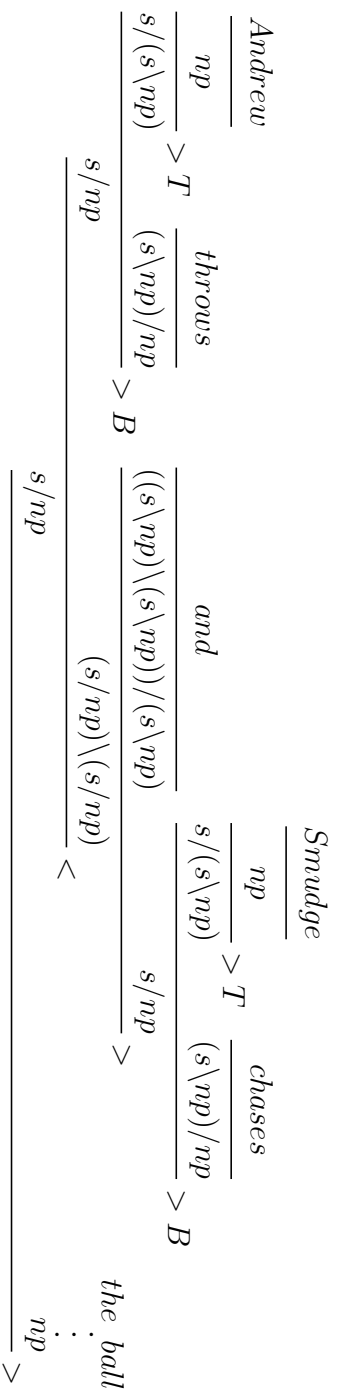


Figure 4.2: An example of Type Raising in CCG

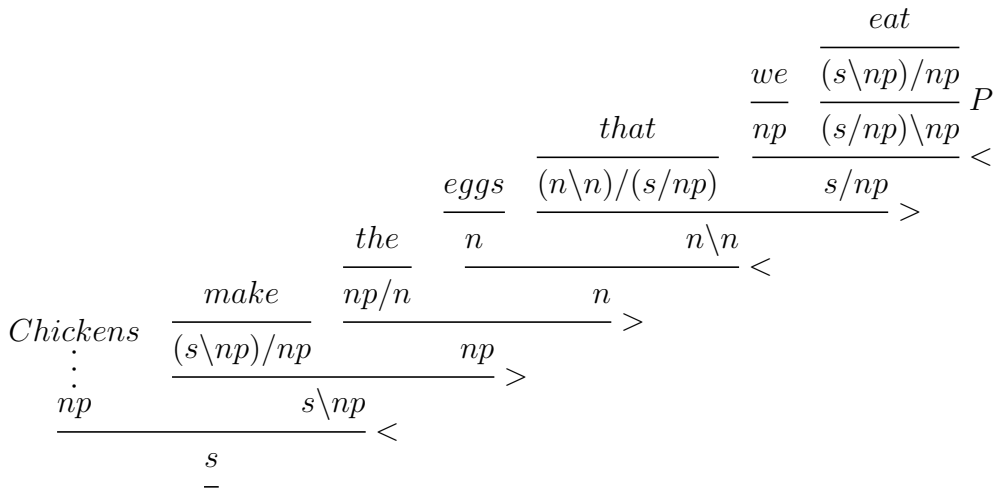
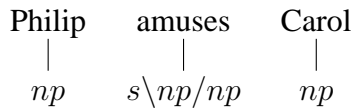


Figure 4.3: An example of Generalised Weak Permutation in CCG.

### 4.1.3 Generalised Weak Permutation vs. Type Raising

Note that GWP and the type raising rules can be used to capture similar linguistic constructs. A Combinatory Categorical Grammar will tend to use either type raising or GWP but not both. Since they are unary rules, both type raising and GWP can be applied almost anywhere in the parse tree; consequently, their use is normally constrained. It is usually asserted that these rules should only be used on items in the lexicon. The reasons for adopting one of these rules in a CCG rather than the other will depend on the application. Type raising rules are more linguistically pleasing since they functionally capture the nature of the phenomena being expressed. However, for work on grammar inference (such as the work here) type raising increases the search space over GWP.

If we restrict use of the rules to the lexicon, GWP may be applied as many times as there are arguments in the lexical items of the sentence. Type raising, however, is bounded by the size of  $Tp$  (i.e. the number of lexical types in the grammar—which is arguably infinite but at least undefined in a learning situation). For an example consider the sentence ‘*Philip amuses Carol*’.



Now, imagine we are trying to derive a parse tree for this sentence from the syntactic categories by exhaustive application of the rules of a CCG. For this sentence the GWP rule can only be used on the verb *amuses* and it can only be applied once (applying it twice yields the original category). The type raising rule may be applied to every word in the sentence in  $|Tp|$  possible ways; since any word can be raised by any category in  $Tp$ . This in itself is a problem because while we are still learning  $Tp$  is undefined: unless we make some very strong assumptions about what is known by the learner before learning commences (cf. the predefined categories of Villavicencio’s categorial grammar learner described in Section 4.2.1).

In general, the number of arguments in a sentence is proportional to the number of words,  $n$ . So, for the GWP rule we add complexity  $O(n)$  but for type raising we add complexity  $O(nTp)$ . For

the example above, “*Chickens make the eggs that we eat*”, we can make 3 possible applications of GWP whereas for type raising we have at least  $9 * 7 = 63$  (since  $Tp$  must contain at least as many types as occur in this derivation and we can raise every word in the sentence by each of these types). Although most of these applications of the type raising rule would lead to parses that fail, it would have been preferable to not have considered them at all. It has been noted that in English, type raising need only be used on primitive types (like  $np$  and  $pp$ ) [97]. This would reduce the number of possible applications to 27 which is still 9 times more than for GWP. Furthermore, if use of type raising is restricted to primitive types then there is a requirement for the learner to be aware of this. In a real language learning situation this equates to either a) having innate knowledge of which categories type raising can apply to or b) having some method of learning which categories type raising can apply to.

#### 4.1.4 Semantics and Categorical Grammars

A close relationship can be found between a categorical grammar type and semantic type. To explain how we can represent and exploit this relationship it is first necessary to know a little about lambda calculus. All lambda calculus expressions  $E$  can be expressed using the following context free grammar, where  $x$  is a variable:

$$\begin{aligned} E &\rightarrow x \\ E &\rightarrow \lambda x.E \\ E &\rightarrow (EE) \end{aligned}$$

Consider the function  $g(x) = 2x$ . In lambda calculus we may represent this function as  $\lambda x.2x$ . The  $x$  in this expression is bound by the  $\lambda$ . Lambda expressions may be applied to arguments using function application. For example,  $g(3)$  would be written  $(\lambda x.2x)3$ . Function application is left associative; we can express and evaluate the addition of numbers  $a$  and  $b$  as follows:

$$\begin{aligned} &(\lambda x \lambda y. x + y)ab \\ \equiv &(\lambda x (\lambda y. x + y))ab \\ \rightarrow &(\lambda y. a + y)b \\ \rightarrow &(a + b) \end{aligned}$$

Instead of mathematical functions let us now consider semantic predicates. Let the expression (**LOVE'** **bob'** **helen'**) be the semantic representation of “*Helen loves Bob*”. Here the predicate **LOVE'** has two arguments; **helen'** is the actor and **bob'** the undergoer. Using function application we could represent the same semantic expression in lambda calculus as:

$$(\lambda x \lambda y. \mathbf{LOVE}'xy)\mathbf{bob}'\mathbf{helen}'$$

The expression  $\lambda x \lambda y. \mathbf{LOVE}'xy$  tells us how the word “*love*” behaves semantically and can be recorded in the lexicon along with its syntactic category.

$$\text{love} : (s \setminus np) / np : \lambda x \lambda y. \mathbf{LOVE}'xy$$

Notice the relationship between the syntactic category and lambda expression; they both express a functor that takes an argument and becomes another functor that takes another argument in order to be evaluated. The only difference is that the syntactic category expresses the directionality of the arguments.

Indeed, the *Principle of Categorial Type Transparency* [98] states that:

For a given language, the semantic type of the interpretation together with a number of language-specific directional parameter settings uniquely determines the syntactic category of a word.

A consequence of this principle is that if the semantic type of a lexical item is known then a certain amount about its syntactic category is also known. For example, from the semantic expression (**LOVE' bob' helen'**) we know that **LOVE'** is a two argument predicate. Hence, the syntactic category of “love” will also take two arguments (but we don't know their directionality). This knowledge can be represented in a skeleton syntactic category as  $A|B|C$  where  $A, B, C \in Tp$  and  $|$  is a variable over  $\setminus$  and  $/$ . Furthermore, we know that if the argument is a semantic entity (such as **helen'** or **bob'**) then the associated syntactic argument will be a primitive syntactic category. This could be represented as  $A|a|b$  where  $a, b \in Pr$ . The ability to extract syntactic information from semantic representations is fundamental to the learner presented here.

All of the rules of the categorial grammar can be represented in lambda calculus; this allows us to combine semantic constituents to evaluate the semantic content of a sentence. Some CCG rules and their semantic lambda expression are shown below;  $\eta$ -conversion<sup>1</sup> has been used to simplify the reading:

$$\begin{array}{ll} \text{Forward Application}(>) : & A/B : f \quad B : a \rightarrow A : fa \\ \text{Backward Application}(<) : & B : a \quad A \setminus B : f \rightarrow A : fa \\ \text{Forward Composition}(> B) : & A/B : f \quad B/C : g \rightarrow A/C : \lambda x. f(gx) \\ \text{Backward Composition}(< B) : & B \setminus C : g \quad A \setminus B : f \rightarrow A \setminus C : \lambda x. f(gx) \\ \text{Forward Type Raising}(> T) : & A : x \rightarrow T / (T \setminus A) : \lambda f. fx \\ \text{Backward Type Raising}(< T) : & A : x \rightarrow T \setminus (T / A) : \lambda f. fx \\ \text{Generalised Weak Permutation}(P) : & ((A | B_1) \dots | B_n) : \lambda y_n \dots \lambda y_1. f y_1 \dots y_n \rightarrow \\ & ((A | B_n) \dots | B_1) : \lambda y_1 \lambda y_n \dots. f y_1 \dots y_n \end{array}$$

The following shows an example of how these rules can be applied to the sentence “*Smudge chases mice*” to derive the semantic predicate **CHASE' mice' smudge'**.

---

<sup>1</sup> $\lambda x. fx \equiv f$  when  $x$  is not free in  $f$

$$\frac{\frac{Smudge}{np : smudge'} \quad \frac{\frac{chases}{(s \setminus np) / np : \lambda x \lambda y. CHASE' xy} \quad \frac{mice}{np : mice'}}{s \setminus np : \lambda y. CHASE' mice' y} >}{s : CHASE' mice' smudge'} <$$

## 4.2 Previous Categorical Grammar Learners

The purpose of this work is to create a categorial grammar learner that can learn from real data and echos real learning. Several previous attempts have been made to learn categorial grammars computationally. However, many of these attempts have produced learners that are unable to acquire language from real data. These systems have generally involved learning categorial grammars that use only the rules of functional application (e.g. Osborne and Briscoe [70], Buszkowski [16] and Kanazawa [49]); as explained above, real natural language can not be modelled using the rules of function application alone. Often learning has been from an artificially generated corpus (e.g. Watkinson and Manandhar [107]) and even negative feedback has been allowed (e.g. Adriaans [1]), which does not reflect real learning (see Chapter 3 for a discussion).

Below two previous categorial grammar learners are discussed in detail. First, the learning system of Waldron/Villavicencio ([105], [104]) is presented. This system learns from real data using a parametric model that has greatly influenced the direction of my own work. Secondly, the formal categorial grammar learners of Buszkowski ([16], [17]) and its extensions [49] are discussed. Both of these learning systems have influenced the categorial grammar learner presented in Chapter 5.

### 4.2.1 The Waldron/Villavicencio Learning System

#### Waldron's Semantic Learner

Waldron [105] has implemented a system that learns categories using the rule of Generalised Weak Permutation as well as those of function application. As with this learner, Waldron's system does not use negative evidence and will learn categories from real data.

The learner assumes that the mapping from words to primitive syntactic types (such as  $np$  or  $n$ ) is known. The input to the learner is an utterance paired with a set of associated semantic predicates or primitive syntactic types. First, the algorithm enumerates all possible ways of combining the members of this set using function application. These are recorded as a list of equations. For example if we receive the utterance *naomi likes georgi* with the set  $\{np, \mathbf{LIKE}', np\}$  then the algorithm would build the following equations (where  $>$  and  $<$  represent forward and backward application respectively):

$$\begin{aligned}
(np > \mathbf{LIKE}') &> np \\
(np > \mathbf{LIKE}') &< np
\end{aligned}$$



$$\begin{aligned}
(np < \mathbf{LIKE}') &> np \\
(np < \mathbf{LIKE}') &< np \\
np > (\mathbf{LIKE}' > np) \\
np > (\mathbf{LIKE}' < np) \\
np < (\mathbf{LIKE}' > np) \\
np < (\mathbf{LIKE}' < np)
\end{aligned}$$

Next, equations which will definitely result in failure are removed from the list. These equations are identified by means of a set of *equation validity rules* such as: a primitive syntactic type must act as the argument in function application. This would reduce the example list of equations to:

$$\begin{aligned}
(np < \mathbf{LIKE}') &> np \\
np < (\mathbf{LIKE}' > np)
\end{aligned}$$

The remaining equations are associated with a weighting. This weighting is  $1/n$  where  $n$  is the number of equations left. The predicate names are then looked up in a lexicon of predicate-category mappings and substitutions are made for as many predicates as possible. If after substitutions, the equations still have more than one unknown in them then nothing can be learnt so the equations are stored along with their weightings. If, however, there is only one unknown then its category is inferred using the *inference rules*. These rules are shown below where  $a, b, c, d$  are syntactic categories,  $?$  is the unknown predicate and  $>, <$  are forward and backward application:

$$\begin{aligned}
(a < ?) = b &\rightarrow ? = b \backslash a \\
(? > a) = b &\rightarrow ? = b / a \\
(c/d > ?) = c &\rightarrow ? = d \\
(? < c \backslash d) = d &\rightarrow ? = d
\end{aligned}$$

If the syntactic category associated with a predicate is successfully inferred it is recorded in the lexicon together with a weighting derived from the weight of the equation it was inferred from. Armed with this new predicate-category mapping all the previously stored equations are re-examined. If they contain the predicate it is substituted for its (now known) category and further inferences are attempted.

Waldron's algorithm makes large demands on memory since it is necessary to remember all past equations as well as the lexicon. This is possibly not a very realistic model of learning. It is unlikely that children store all utterances they've heard for batch processing. Using an unrestricted search-space the system was found to over-generate considerably; producing the following category for *red* in the phrase *red ball* rather than  $n/n$ :

$$red \mapsto ((((((s \backslash np) \backslash (np \backslash n)) \backslash ((s/np)/np))/n)/np)$$

Performance was improved by using Villavicencio's Universal Grammar and Parameter Learner to restrict the search-space to only categories known to exist in the target language.

## Villavicencio’s Universal Grammar and Parameter Learner

Villavicencio’s Universal Grammar is represented by an under-specified unification-based categorial grammar. The Principles of her Universal Grammar specify a subset (cardinality 89) of all the possible categorial grammar categories that take up to five arguments. The grammar uses the primitive types *s*, *n*, *np*, *pp* and *pvt*. Syntactic categories are defined using *attribute-value specifications*. Both syntactic and semantic properties are specified as attribute-value pairs; for instance, the specification for an intransitive verb (which has categorial grammar type  $s \setminus np$ ) might include the attribute pairs *subject*:  $\mapsto np$  and *subject-dir*:  $\mapsto backward$ . Syntactic categories are arranged in a hierarchy so that child types inherit all attribute-value pairs from their parents; this encodes the Universal Grammar more efficiently than specifying each category separately.

The parameters of this Universal Grammar are embedded within the attribute-value specifications. Each category has an attribute (e.g. *intransitive-parameter*) that will take a Boolean value. If the category is part of the current grammar then this attribute is set to *true*; otherwise it is *false*. The learner is further endowed with the ability to group these parameters according to the type of the category they are associated with. These groups are arranged into hierarchies according to the similarity of attribute-value pairs within the categories they are associated with; the hierarchies follow Pollard and Sag [80]. Villavicencio also defines a hierarchy on the direction-attributes (such as *subject-dir*); this hierarchy is very flat being only 3 levels deep. Additionally, each direction-attribute is associated with a score that is used to determine whether the value for that attribute should be *forward* or *backward*.

The input to Villavicencio’s grammar is a set of possible syntactic assignments (as hypothesised by Waldron’s syntactic learner) and a semantic representation of the utterance. On receiving this input the set of assignments is filtered by the *Valid Category Assignment Detection Model*. Filtering is achieved by removing all assignments that have invalid syntactic categories; a syntactic category is deemed valid if it both adheres to the *Principle of Categorial Type Transparency* (see Section 4.1.4) and is type compatible with its semantic predication. For an example of the latter, the word *do* hypothesised with the syntactic category  $np/np$  is invalid because the verb *do* is not compatible with a nominal predication. Note here that this degree of semantic validation places large innate requirements on the learner; for instance, in the example above, the learner is required to ‘know’ which predications *do* is compatible with. This is not information that can be directly extracted from the semantics unless the verb acts as a predicate on semantic entities (which have a direct mapping to primitive types in the categorial grammar). Also, the need to check at this stage for adherence to the *Principle of Categorial Type Transparency* highlights an inefficiency of this learning system: the syntactic categories were inferred from the semantic representations in the first instance and should inherently adhere to the principle.

After the filtering process, surviving assignments are analysed for triggers. Following Drescher and Kaye [33], Villavicencio models a learner that knows how to detect triggers in the input, and also knows which parameters are being expressed. This is implemented by means of a *Trigger Detection Module* using the properties of the syntactic categories in the Universal Grammar. Once detected, triggers are used to set parameters.

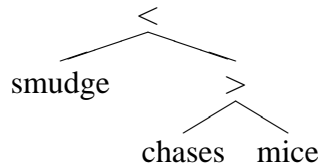
First, category parameters (such as the *intransitive-parameter* described above) are updated. A categorial parameter can be set to *true* if its associated trigger has been detected and if its direct parent in its group hierarchy is also true. Next, if the trigger relating to a direction-attribute has been seen, then the score associated with that attribute is recalculated; the Bayesian Incremental Parameter Setting algorithm is used for this purpose (see Chapter 3 for a description). Since

the Universal Grammar is defined hierarchically the direction of the direction-attributes are inherited in all allowed categories.

Together, Waldron and Villavicencio present a system for learning to set the parameters of a Universal Grammar. The acquired grammar can then be used to parse and produce language. However, in order to acquire the parameter settings, the learner has to first use Waldron’s syntactic learner to hypothesis categories for the utterance heard. The mechanisms for acquisition are distinct to those for utilising the grammar being acquired.

## 4.2.2 Formal Categorical Grammar Learners

Buszkowski ([16], [17], [18]) developed an algorithm for learning rigid grammars from functor-argument structures. A functor-argument structure is a binary branching derivation tree whose leaf nodes are labelled with the words of the input sentence and whose internal nodes are labelled with either  $>$  or  $<$ ; indicating whether forward or backward application is used at that node. The functor-argument structure for the sentence *Smudge chases mice* would be as follows:



Buszkowski’s algorithm proceeds by inferring types from the available data and then unifying variables across all encountered structures.

**Inferring types:** if the functor-argument structure shows an instance of forward application  $>$  then there must be an argument on the right,  $A$ , and a functor on the left  $B/A$  (where  $A, B \in Tp$ ).

1. Forward Application:

$$\begin{array}{c} > \\ \swarrow \quad \searrow \\ ? \quad ? \end{array} \rightarrow \begin{array}{c} B \\ \swarrow \quad \searrow \\ B/A \quad A \end{array} \quad (4.12)$$

2. Backward Application:

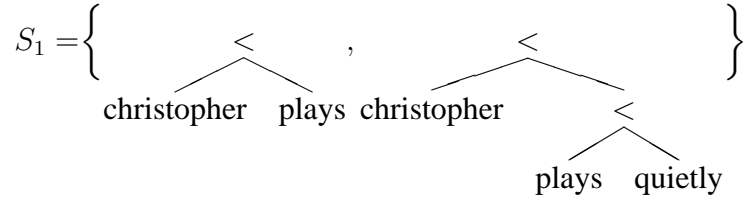
$$\begin{array}{c} < \\ \swarrow \quad \searrow \\ ? \quad ? \end{array} \rightarrow \begin{array}{c} B \\ \swarrow \quad \searrow \\ A \quad B \setminus A \end{array} \quad (4.13)$$

**Unifying variables:** a substitution  $\sigma$  (or unifier) unifies a set of types  $\mathbf{A}$  if for all types  $A_1, A_2 \in \mathbf{A}$ ,  $\sigma(A_1) = \sigma(A_2)$ . Furthermore,  $\sigma$  unifies a set of sets of types  $\mathcal{A}$  if  $\sigma$  unifies all sets of types  $\mathbf{A}, \mathbf{B} \in \mathcal{A}$ .  $\sigma_1$  is a *more general unifier* than  $\sigma_2$  if there is a substitution  $\sigma_3$  such that  $\sigma_2(\mathbf{A}) = \sigma_3(\sigma_1(\mathbf{A}))$ .

Baader and Siekman [2] summarise algorithms for deciding whether a unifier exists for  $\mathcal{A}$  and, if so, discovering the most general unifier. The most efficient algorithms are linear in time complexity.

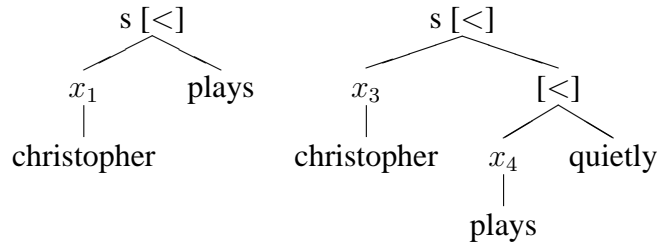
The algorithm, which will be referred to as the Bus-CGL (Buszkowski’s categorical grammar learner), is illustrated below:

Recall that  $S_i = \{s_0, s_1, s_2 \dots s_i\}$  is the set of the first  $i$  sentence objects in the input stream. Consider:

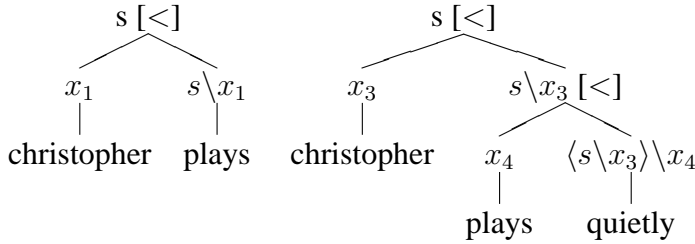


**Step 1:** Assign types to the structures.

- (a) Assign the sentence type primitive  $s$  to each root node.
- (b) Assign distinct variables to the argument nodes:



- (c) Compute types for functor nodes using equations 4.12 and 4.13:



**Step 2:** Collect the types assigned to the leaf nodes:

$$\begin{aligned} \text{christopher} &\rightarrow x_1, x_3 \\ \text{plays} &\rightarrow s \backslash x_1, x_4 \\ \text{quietly} &\rightarrow (s \backslash x_3) \backslash x_4 \end{aligned}$$

$$\text{so } \mathcal{A} = \{ \{x_1, x_3\}, \{s \backslash x_1, x_4\}, \{(s \backslash x_3) \backslash x_4\} \}$$

**Step 3:** Unify  $\mathcal{A}$  (the sets of types). If unification fails then fail:

$$\sigma = \{x_3 \mapsto x_1, x_4 \mapsto s \backslash x_1\} \tag{4.14}$$

where  $\{a_1 \mapsto b_1, \dots, a_n \mapsto b_n\}$  denotes the unifier  $\sigma$  such that  $\sigma(a_1) = b_1, \dots, \sigma(a_n) = b_n$  and  $\sigma(y) = y$  for all other variables  $y$ .

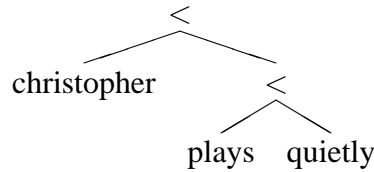
**Step 4:** Output the grammar (a lexicon with associated types):

$$\begin{aligned} G_1 : \text{christopher} &\rightarrow x_1 \\ \text{plays} &\rightarrow s \backslash x_1 \\ \text{quietly} &\rightarrow (s \backslash x_1) \backslash (s \backslash x_1) \end{aligned}$$

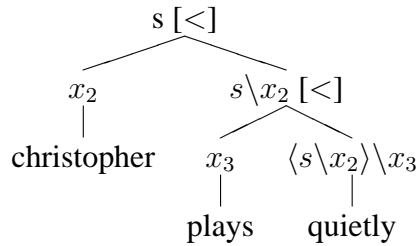
Buszkowski’s algorithm was designed to learn a rigid grammar from a finite *set* of functor-argument structures. Kanazawa [49] constructed a proof to show that the algorithm could learn the class of rigid grammars from an *infinite stream* of functor-argument structures—as required to satisfy Gold’s learning model. The following is a suggestion for modification to Bus-CGL so that it may learn from a *stream* of structures rather than a *set*. The algorithm will be referred to as Bus-stream-CGL. The main difference between Bus-CGL and Bus-stream-CGL is that in Bus-CGL unification occurs over all structures in the language in one go whereas in Bus-stream-CGL unification occurs repeatedly (every time the next structure in the stream is presented). Let  $G_i$  be the current grammar (or guess) of Gold’s learning model after having seen  $i$  structures from the stream:

$$G_i : \text{christopher} \rightarrow x_1 \\ \text{plays} \rightarrow s \backslash x_1$$

and let the next encountered structure in the stream be:



**Step 1:** Assign types to the new functor-argument structure as in Bus-CGL.



**Step 2:** Look up words at the leaf nodes of the new structure in  $G_i$ . If the word exists in  $G_i$ , add types assigned to current leaf nodes to the existing set of types for that word; else create new word entry.

$$\text{christopher} \rightarrow x_1, x_2 \\ \text{plays} \rightarrow s \backslash x_1, x_3 \\ \text{quietly} \rightarrow (s \backslash x_2) \backslash x_3$$

$$\mathcal{A} = \{\{x_1, x_2\}, \{s \backslash x_1, x_3\}, \{(s \backslash x_2) \backslash x_3\}\}$$

**Step 3:** Unify the set of types. If unification fails then fail.

$$\sigma = \{x_2 \mapsto x_1, x_3 \mapsto s \backslash x_1\} \tag{4.15}$$

**Step 4:** Output the lexicon

$$\begin{aligned} G_{i+1} : \text{christopher} &\rightarrow x_1 \\ &\text{plays} \rightarrow s \backslash x_1 \\ &\text{quietly} \rightarrow (s \backslash x_1) \backslash (s \backslash x_1) \end{aligned}$$

Presented with structures from a  $k$ -valued grammar, Bus-CGL (and Bus-stream-CGL) will either over-generalise (arriving at a grammar that generates a proper superset of the language) or fail on unification. Natural language is better described by  $k$ -valued grammars so Kanazawa [49] has modified Buszkowski's algorithm to learn from  $k$ -valued structures. The modification relies on the notion of  $k$ -partial unification, i.e. unification that will construct at most  $k$  types for each member of the lexicon. Note that a 1-partial unifier is simply a unifier as in 4.14. A case of 2-partial unification is illustrated below.

Consider that after **Step 2** of the Bus-CGL algorithm we have collected the following word-type associations from the leaf nodes:

$$\text{word}_1 \rightarrow x_1, x_2 \backslash s, s / x_2 \quad (4.16)$$

$$\text{word}_2 \rightarrow x_2, x_3, s / x_3 \quad (4.17)$$

so  $\mathcal{A} = \{\{x_1, x_2 \backslash s, s / x_2\}, \{x_2, x_3, s / x_3\}\}$

Then  $\sigma_1$  is an example of a 2-partial unifier:

$$\sigma_1 = \{x_1 \mapsto (x_1/s) \backslash s, x_2 \mapsto x_1/s, x_3 \mapsto x_1/s\} \quad (4.18)$$

giving us:

$$\text{word}_1 \rightarrow (x_1/s) \backslash s, s / (x_1/s)$$

$$\text{word}_2 \rightarrow (x_1/s), s / (x_1/s)$$

Kanazawa showed that the complete set of  $k$ -partial unifiers for  $\mathcal{A}$  may be found in exponential time—by running the linear unification algorithm over all *partitions* of  $\mathcal{A}$  (where a partition of  $\mathcal{A}$ ,  $\mathcal{B}$ , is defined to be:

$$\mathcal{B} = \bigcup \{\mathcal{B}_i \mid 1 \leq i \leq n\} \quad (4.19)$$

where  $\mathcal{B}_i = \{\mathbf{B}_{(i,1)}, \dots, \mathbf{B}_{(i,n)}\}$  is a partition of  $\mathbf{A}_i$ , for  $1 \leq i \leq n$ .)

Continuing with examples 4.16 and 4.17, Steps 3 and 4 of Buszkowski's algorithm are now adjusted as follows to give the Kan-CGL (Kanazawa's categorial grammar learner):

**Step 3:** Unify  $\mathcal{A}$ . If unification fails then allow  $k$ -partial unification. If  $k$ -partial unification fails then fail:

$$\sigma_1 = \{x_1 \mapsto (x_1/s) \backslash s, x_2 \mapsto x_1/s, x_3 \mapsto x_1/s\}$$

$$\sigma_2 = \{x_1 \mapsto x_2 \backslash s, x_3 \mapsto x_2\}$$

$$\sigma_3 = \{x_1 \mapsto (s/x_3) \backslash s, x_2 \mapsto s/x_3\}$$

$$\sigma_4 = \{x_1 \mapsto s/x_2, x_3 \mapsto x_2\}$$

**Step 4:** Output all possible grammars.

Applying  $\sigma_1$  to  $\mathcal{A}$ :

$$\begin{aligned} G_1 : \text{word}_1 &\rightarrow (x_1/s) \setminus s, s/(x_1/s) \\ \text{word}_2 &\rightarrow x_1/s, s/(x_1/s) \end{aligned}$$

Applying  $\sigma_2$  or  $\sigma_4$  to  $\mathcal{A}$ :

$$\begin{aligned} G_2 : \text{word}_1 &\rightarrow x_2 \setminus s, s/x_2 \\ \text{word}_2 &\rightarrow x_2, s/x_2 \end{aligned}$$

Applying  $\sigma_3$  to  $\mathcal{A}$ :

$$\begin{aligned} G_3 : \text{word}_1 &\rightarrow (s/x_3) \setminus x_3, s/(s/x_3) \\ \text{word}_2 &\rightarrow s/x_3, x_3 \end{aligned}$$

**Step 5:** Select one grammar from the output set by placing an ordering on the languages they produce and then choosing the minimal element.

Kanazawa showed that **Step 5** of Kan-CGL is exponential in time in the size of the grammars. As with Bus-CGL, Kan-CGL learns from a *set* of structures rather than a *stream*. Kan-CGL may be modified to learn from a *stream* in exactly the same manner that Bus-CGL was modified to give Bus-stream-CGL. The stream accepting version of Kan-CGL will be referred to a Kan-stream-CGL.

## Learning from Strings

Kanazawa shows that  $k$ -valued classic categorial grammars are also learnable from strings. The algorithm (String-CGL) is very expensive; applying the Kan-CGL to all possible functor-argument structures of the input strings. String-CGL is described below:

**Step a:** Form a set  $\mathcal{S}$  containing all the possible functor-argument structure combinations for the input strings in  $S_i$ .

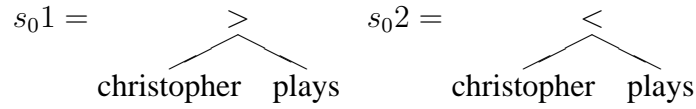
**Step b:** For each element of  $\mathcal{S}$ , carry out **step 1** to **step 4** of Kan-CGL.

**Step c:** Select one grammar using **step 5** of Kan-CGL.

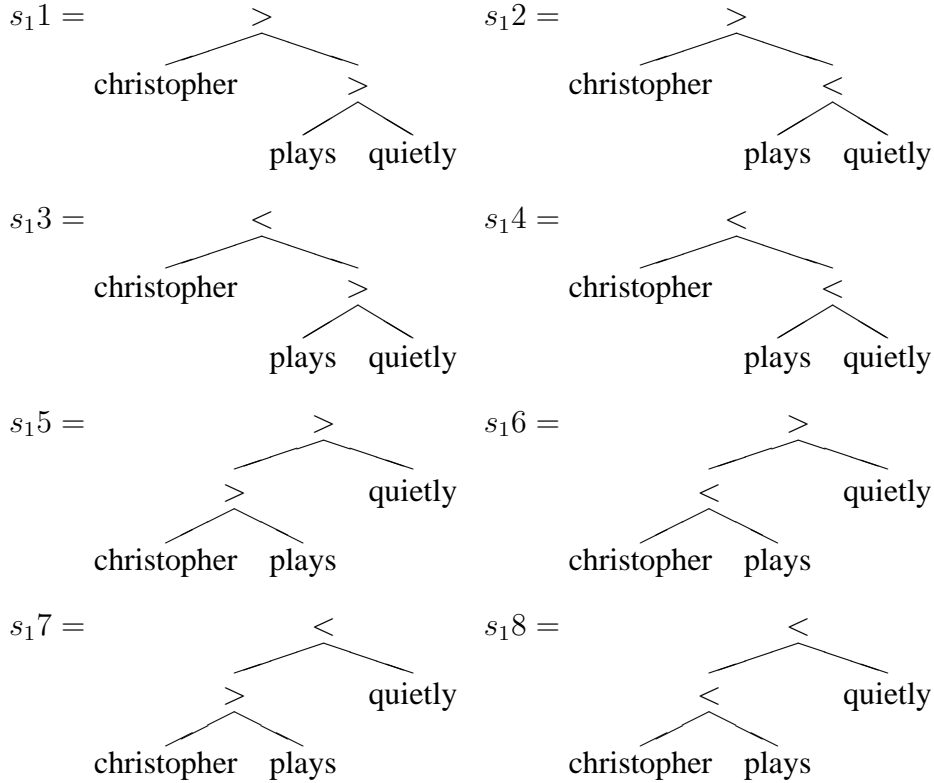
Consider example input set  $S_1 = \{s_0, s_1\}$  with:

$$\begin{aligned} s_0 &= \text{christopher plays} \\ s_1 &= \text{christopher plays quietly} \end{aligned}$$

There are two functor-argument structures that could yield  $s_0$ :



And there are eight functor-argument structures that could yield  $s_1$ :



Giving the sixteen element set  $\mathcal{S} = \{\{s_01, s_11\}, \{s_01, s_12\}, \dots, \{s_02, s_18\}\}$ . Kan-CGL will succeed on six elements from this set.

For example  $\{s_02, s_14\}$  yields:

$$\begin{aligned} G_1 : \text{christopher} &\rightarrow x \\ \text{plays} &\rightarrow s \backslash x \\ \text{quietly} &\rightarrow (s \backslash x) \backslash (s \backslash x) \end{aligned}$$

and  $\{s_02, s_18\}$  yields:

$$\begin{aligned} G_2 : \text{christopher} &\rightarrow x \\ \text{plays} &\rightarrow s \backslash x \\ \text{quietly} &\rightarrow s \backslash s \end{aligned}$$

For learning from a *stream* of strings rather than a *set* the algorithm would be:

**Step a:** Form a set  $\mathcal{S}$  containing all the possible functor-argument structures for the new input



string.

**Step b:** For each element of  $\mathcal{S}$ , carry out **step 1** to **step 4** of Kan-stream-CGL.

**Step c:** Select one grammar using **step 5** of Kan-stream-CGL.

### Basic Categorical Grammar Learner Summary

Buszkowski created an algorithm to learn rigid classic-categorical grammars from a set of functor-argument structures. Kanazawa modified this algorithm to learn  $k$ -valued categorical grammars by introducing partial unification. Further, he showed that  $k$ -valued categorical grammars may also be learnt from a set of strings. Simple modification to these algorithms allow the input to be a stream rather than a set (in line with Gold’s learning model).

There are three main sections to a general algorithm for learning a  $k$ -valued classic-categorical grammar from a stream of strings:

**Step a—Form Search-Space:** Form a set  $\mathcal{S}$  of all possible functor-argument structures to describe the strings.

**Step b—Hypothesise Grammars:** Iterate through  $\mathcal{S}$  assigning types to each functor-argument structure. Unify (using partial unification) the assigned types with the current grammar. If unification fails, then fail, else add possible grammar to grammar set,  $\mathbf{G}$ .

**Step c—Select Grammar:** Choose one grammar from set  $\mathbf{G}$ .

## 4.3 Sentence Objects

In our discussion of previous categorical grammar learners we noted that the form of input provided to learners is not standard: for instance, the Waldron/Villavicencio system ([105], [104]) takes strings with an accompanying semantic representation; and more recently Clark and Curran [29] developed a method for learning parameters of a CCG given fully annotated parse trees.

Traditionally, studies in CCG learnability (and learnability in general) have concentrated on models that take streams of strings (literally the ordered words of a sentence) as input. However, Buszkowski ([16] and [17]), developed an algorithm for learning categorical grammars from a stream of *structures*, which carry more information than strings.

We would like to know what is the most sensible form of input (in terms of efficiency and cognitive plausibility) to a categorical grammar learner that is attempting to learn from real language. This section discusses methods of learning categorical grammars from streams of *sentence objects*, which is the collective term for elements that carry at least as much information as a string.

Gold’s general learning model (see Chapter 3) is updated as follows:

1.  $\Omega$ —a hypothesis-space of grammars;
2.  $\Phi$ —a sample set of grammatical sentence objects;
3.  $F$ —a learning function that maps finite subsets of  $\Phi$  (languages) to elements of  $\Omega$ .

$$G_i = F(\{s_0, s_1, s_2 \dots s_i\})$$

where  $G_i \in \Omega$  and  $s_0, s_1, s_i \in \Phi$

The cognitive plausibility of 4 different types of sentence object (simple strings, augmented strings, unlabelled structures and functor-argument structures) is considered below followed by methods for learning from them:

### 4.3.1 Simple Strings

A simple string is simply the words of the input sentence, presented in the correct order. The string for the sentence *Smudge chases mice* is simply:

smudge    chases    mice

Cognitively this is equivalent to a real learner being able to recognise word boundaries and creating word token for each word. This is clearly plausible (otherwise we would not be able to use language at all) but it seems likely real learner will have access to more information than just word tokens.<sup>2</sup>

### 4.3.2 Augmented Strings

In addition to simple word order an augmented string has some word type information. The type of words which are associated with primitive types (belonging to  $Pr$ ) are declared. Words with complex types are not fully declared but some general information about their structure is displayed. For example, the information declared for the word *chases* might be that it is of the form  $(A|B)|C$  where  $A, B, C \in Tp$  and  $|$  is a variable over  $\backslash$  and  $/$ . The extra syntactic information here may be extracted from the semantics following the Principle of Categorical Type Transparency as explained in Section 4.1.4. This of course requires that the semantic type of each word is known. For a real learner to form augmented strings from an utterance they would have to:

- a) segment the utterance on word boundaries;
- b) hypothesise a semantic expression for the utterance;
- c) map parts of the semantic expression to word tokens.

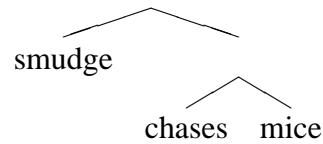
smudge	chases	mice
<i>np</i>	$\langle A   B \rangle   C$	<i>np</i>

Augmented strings appear to be fairly cognitively plausible. It is extremely likely that when a child attends to an utterance they hypothesise some semantic meaning to associate with it. Pinker [76] and Siskind [93] among others have offered methods for mapping parts of semantic expressions to individual words. This will be discussed in more detail in Section 4.4 where we will explain how an augmented string is simply a means of representing the constraints placed on a space of possible parses due to the string's associated semantic form.

<sup>2</sup>There is no particular reason why we talk about word tokens rather than morpheme tokens.

### 4.3.3 Unlabelled Structures

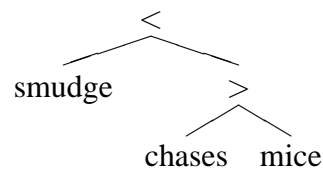
An unlabelled structure is a binary branching derivation tree whose leaf nodes are labelled with the words of the input sentence. The unlabelled structure for *Smudge chases mice* would be:



Cognitively it is difficult to imagine how a learner would come to associate such a structure with an input utterance prior to learning a grammar. Some of the information is derivable from the Principle of Categorical Type Transparency but not without prior analysis.

### 4.3.4 Functor-Argument Structures

A functor-argument structure is much like a unlabelled structure except that the internal nodes are labelled with either  $>$  or  $<$ ; indicating whether forward or backward application is used at that node. The functor-argument structure for the sentence *Smudge chases mice* would be as follows:

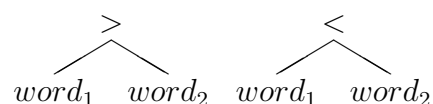


Again some of the information is derivable from the semantics (i.e. which word is the functor) but, as for unlabelled structures, it is difficult to imagine how the learner could associate this structure with their input utterance.

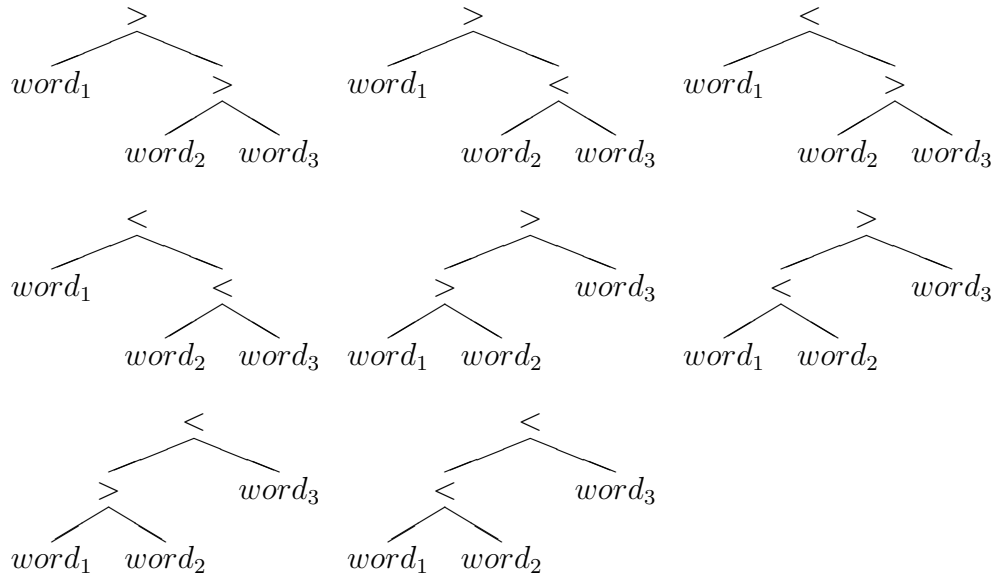
### 4.3.5 Information Content of Sentence Objects

This section discusses the information content of sentence objects. Each sentence object will be compared to the functor-argument structure (since this is the object that carries the most content); every other type of sentence object may be thought of as an under-specified functor-argument structure. In each case the number of functor-argument structures that could yield the sentence object will be counted.

**Simple Strings:** To find the number of functor-argument structures that derive a string of length  $n$  it is necessary to find all of binary trees with  $n$  leaves and then enumerate all the ways of labelling the nodes of those trees with forward or backward application ( $>$  or  $<$ ). For example, a two word string ( $word_1 word_2$ ) may be derived from two functor-argument structures; there is one binary tree with two leaves and two ways to label it with  $>$  or  $<$ .



A three word string may be derived from eight functor-argument structures; there are two binary trees with three leaves and four ways to label each tree.



In general the iterative rule for finding the number of binary trees with  $n$  leaves is:

$$X_n = \sum_{r=1}^{n-1} X_r X_{(n-r)} \quad (4.20)$$

The rule uses the fact that a tree with  $n$  leaves is equivalent to a tree with 1 leaf joined to a tree with  $n - 1$  leaves, and also equivalent to a tree with 2 leaves joined to a tree with  $n - 2$  leaves etc; so, the rule sums the number of ways each combination of trees may be drawn. From this iterative rule, the general rule for the number of binary trees with  $n$  leaves may be derived:

$$\text{number of binary trees with } n \text{ leaves} = \frac{(2n - 2)!}{n!(n - 1)!} \quad (4.21)$$

To find the number of functor-argument structures for a string of length  $n$ , it is necessary to multiply the number of binary trees with  $n$  leaves by the number of ways of labelling the tree nodes with  $<$  or  $>$ . A tree with  $n$  leaves is always made up from  $n - 1$  binary branch nodes of the form,  $\wedge$ . Each node may be labelled one of two ways ( $<$  or  $>$ ) so for a tree with  $n$  leaves there are  $2^{n-1}$  labellings. Hence the general formula for the number of functor-argument structures that derive a string of length  $n$  is:

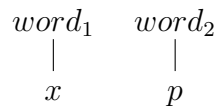
$$\text{number of functor-argument structures} = 2^{n-1} \frac{(2n - 2)!}{n!(n - 1)!} \quad (4.22)$$

**Augmented Strings:** The information content of augmented strings is not simply dependent on the length of the string but also on the types associated with each word. At worst, the information content of an augmented string is the same as for a simple string of the same length. The occurrence of primitive types within an augmented string reduces the number

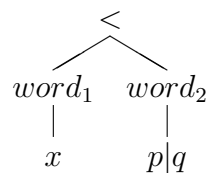
of possible functor-argument structures. Primitive types will always be the argument of a function (they can not be a function themselves). Thus, the presence of primitive types can provide us with information regarding allowed function directionality and also viable tree structures.

1. Function direction:

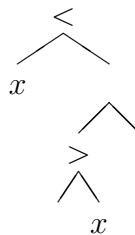
Consider the following augmented string where  $x$  is a primitive type and  $p \in Tp$ :



Since  $x$  is a primitive type it must be acting as a argument and therefore  $p$  must be a function. Thus, there is only one possible functor-argument structure:



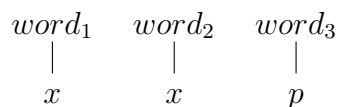
Whenever a primitive type occurs, the function directionality of the associated node will be known, as illustrated in the partial tree below.



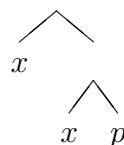
For every primitive type in the augmented string we can reduce the number of possible functor-argument structures by a factor of 2.

2. Tree Structure:

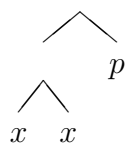
The location of primitive types within an augmented string places constraints on the deriving tree structures. Consider the following augmented string, where  $x$  is a primitive type and  $p \in Tp$ :



The tree structure for this string has to be of the form:



The other possible tree structure would leave two primitive types together on one branch node with no means of being combined.



Whenever primitive types occur adjacently in an augmented string, all trees that would place them on the same branch node can be ruled out. Recall that the number of binary trees with  $n$  leaves is  $X_n$ . Two adjacent primitive types in an  $n$ -length augmented string will reduce the number of possible binary trees by  $X_{n-1}$ .

**Unlabelled Structures:** An unlabelled structure with  $n$  leaves is made up from  $n - 1$  binary branch nodes of the form,  $\wedge$ . Each node may be labelled one of two ways ( $<$  or  $>$ ) so for a tree with  $n$  leaves there are  $2^{n-1}$  labellings.

The following table shows a comparison of information content for sentence objects.  $n$  refers to the number of words in a sentence. The column figures indicate the number of functor-argument structures that can derive the given sentence object. In the case of augmented strings (whose information content is word-type dependent as well as sentence-length dependent), two illustrative columns are included; one to indicate a string with two adjacent primitive types (Augmented String (a)), and another to indicate a string with two non-adjacent primitive types (Augmented String (b)).

$n$	Functor-Argument Structure	Unlabelled Structure	Augmented String (a)	Augmented String (b)	Simple String
2	1	2	-	-	2
3	1	4	1	2	8
4	1	8	6	10	40
5	1	16	36	56	224
6	1	32	168	280	1120
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

For unlabelled structures there are always  $2^{n-1}$  functor-argument structures that can be derived. It can be shown mathematically that  $X_n$  (the number of binary trees with  $n$  leaves) can be bounded by  $4^{n-1}$ . Hence, for the number of derivable functor-argument structures for the simple strings is bounded by  $8^{n-1}$ . An augmented string of type (b) will have exactly a quarter as many structures. Augmented strings of type (a) will have roughly  $\frac{3}{4}$  of those for type (b).

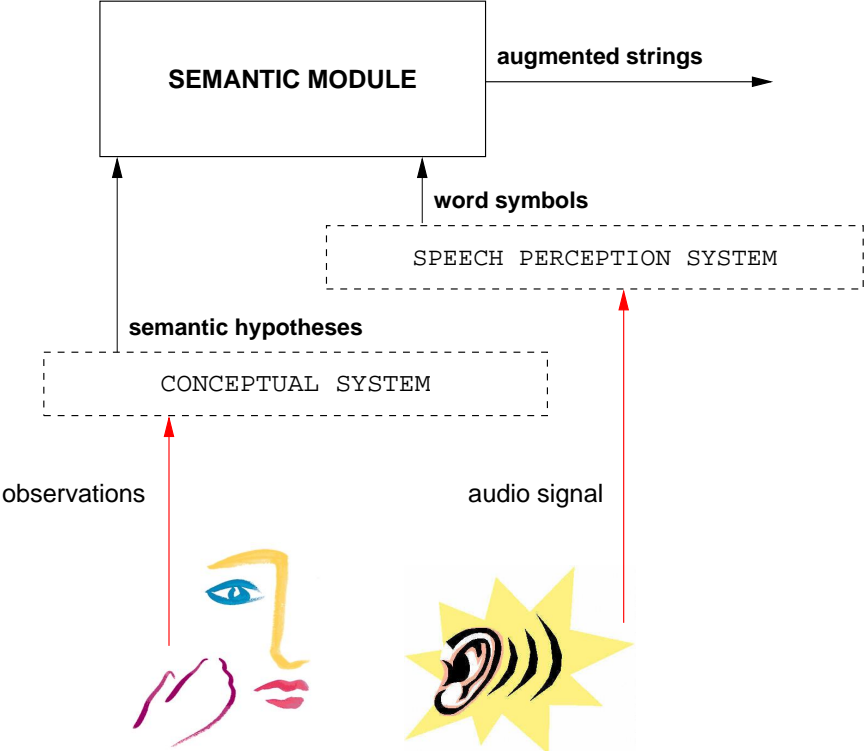
The type of sentence object that carries the most cognitively realistic data is the augmented string since the extra information it carries can be learnt directly from semantics. Learning from augmented strings also reduces the search space by a constant factor, whose size is shown above to be dependent on the content of the string. In order to create augmented strings a mapping is required from the words in the string to their semantic representation. Section 4.4 below describes a mechanism by which free order semantic representations may be mapped to their associated words.

## 4.4 Mapping Words to their Semantic Representation

In order to create augmented strings from real world data it is necessary to provide a mapping between words and their semantic representations. Pinker [76] and Siskind [93] have offered

methods for how children map parts of semantic expressions to individual words. Following these methods, Buttery [19] implemented a semantic learner that can be used to provide the word-meaning mapping required to form augmented strings. The inputs to this learner are utterances annotated with semantic representations and the output is a lexicon of word-to-meaning mappings.

For this work Siskind’s cognitive model [93] is assumed. This is a simple model representing the interaction between a speech perception system and a conceptual system. On hearing an utterance, the purpose of the speech perception system is to break up the acoustic signal and pass a series of word symbols to its output. At the same time, the conceptual system is responsible for producing semantic hypotheses for the utterance.



The word symbols (produced by the speech perception system) have thus far been written in italics; the semantic symbols (produced by the conceptual system and forming the constituent parts of the semantic hypotheses of an utterance) have been written in bold. So, a child hearing the utterance “Kitty eats biscuits” would theoretically produce the word symbols *Kitty*, *eats* and *biscuits* from their speech perception system. If the child was simultaneously observing the cat eating something they would hopefully also produce the semantic expression **EAT’biscuit’kitty’** from their conceptual system.<sup>3</sup>

A possible problem with the conceptual system is that it could produce an infinite number of semantic hypotheses for a given utterance. Siskind avoids this problem by stating that the learner will only entertain likely semantic hypotheses. However, he does not specify the distinction between likely and unlikely. Pinker [76], on the other hand, suggests that semantic hypotheses are constrained for two reasons: first, they are constrained by the semantic structures that constitute mental representations of a word’s meaning (this is referred to as the Universal Lexical

<sup>3</sup>Note that the conceptual system need not only rely on visual observation to produce semantic hypotheses.

Semantics [48] and is somewhat analogous to Chomsky’s Universal Grammar for syntax); secondly, hypotheses may be constrained by the way the child’s lexicon is constructed—it seems that children are fairly unwilling to admit true synonyms to their lexicon and consequently a child would rather not hypothesise an existing word’s meaning for a new word [25]. Even when these constraints are taken into account there may still be several plausible semantic hypotheses for a given utterance. The conceptual system is therefore expected to produce a set of semantic hypotheses.

The output of the the speech perception system and the conceptual system are simulated as follows:

**Simulation of word symbols:** Segmenting an audio signal to produce word symbols is not a straight-forward task [9]; speech doesn’t contain any reliable markers analogous to the blank spaces between words in English text. For adults segmentation is an easier (although not fool-proof) task. For this task we will assume a fully developed speech perception system; i.e. we will not deal with a child’s segmentation problem. The output of the system will be very simply simulated by using a written corpus; the symbols required can be simply created from the textual representation of each utterance.

**Simulation of the semantic hypotheses:** Simulating the output of the conceptual system is more difficult. The approach used here was to parse utterances using an existing grammar and then extract the semantic representations produced. Let the set of all semantic representations be called *Sem*. The set of semantic expressions hypothesised for an input utterance is then simulated by selecting a subset of *Sem*. This method gives control over how much noise the learner is exposed to; at one extreme the set of semantic hypotheses could contain the one single correct hypothesis and at the other extreme many incorrect hypotheses.

#### 4.4.1 Mechanics of the Semantic Learner

The mapping mechanisms of the semantic learner are based on Siskind’s investigation of Cross Situational Techniques [93] and include the following: cross situational learning; covering constraints; constraining hypotheses with partial knowledge; and using the principle of exclusivity.

**Cross Situational Learning:** cross situational learning has been suggested as a method of learning for hundreds of years but more recently by Pinker [75] amongst others. The theory speculates that lexical acquisition may be achieved by finding the common factors across all observed uses of a word. Hearing a word in enough contexts should therefore allow the learner to rule out all incorrect hypotheses and converge on a unique meaning.

For a trivial example consider the utterances “Naomi laughs” and “Naomi eats cookies”. They would have word symbol sets and semantic expressions as follows:

$$\begin{aligned} \{naomi, laughs\} &\mapsto \text{LAUGH}'naomi' \\ \{naomi, eats, cookies\} &\mapsto \text{EAT}'cookie'naomi' \end{aligned}$$

From these two utterances it is possible to ascertain that the meaning associated with the word symbol *naomi* must be **naomi'** since it is the only semantic element that is common to both utterances.



**Covering Constraints:** the idea of covering constraints is essentially the reverse of cross-situational learning. The idea requires that the semantic expression representing a complete utterance is built up only from the semantic expressions relating to words within that utterance, i.e. it doesn't contain any external semantic information.

Consider the situation where the semantic mapping for all but one of the word symbols is known. The semantic expression associated with the final word symbol is necessarily what is left over when all the known semantic expressions are removed from the expression representing the entire utterance.

Consider the example "Grinch hates Xmas". If the following is already known:

$$\begin{aligned} \{grinch, hates, Xmas\} &\mapsto \mathbf{HATE'xmas'grinch'} \\ grinch &\mapsto \mathbf{grinch'} \\ hate &\mapsto \mathbf{HATE'xy} \end{aligned}$$

Then the necessary conclusion is:

$$xmas \mapsto \mathbf{xmas'}$$

**Principle of Exclusivity:** the principle of exclusivity becomes useful when word-meaning mappings have already been acquired. The principle is based on the work of Berwick [3], requiring that each word in an utterance contributes a non-overlapping portion of the meaning.

For instance, given the utterance *dinah likes milk* with the hypothesised meaning **LIKE'milk'dinah'**, and the previous knowledge that the word symbol *dinah* maps to **dinah'** and that *milk* maps to **milk'**. Then *like* must map to **LIKE'xy** rather than **LIKE'milk'y**, or **LIKE'xdinah'**, or even **LIKE'milk'dinah'** because these latter semantic constituents would overlap with the constituents associated with *dinah* and *milk*.

**Constraining Hypotheses with Partial Knowledge:** cross situational learning and covering constraints are most useful if the correct semantic expression is known. In the situation where there are several semantic hypotheses, the semantic learner first attempts to reduce the number before applying these techniques.

Hypotheses can be constrained by removing all those that are impossible given what has already been learnt. To show how this works, imagine the learner has heard the utterance "Mice like cheese" and hypothesised the following semantic expressions:

$$\mathbf{LIKE'cheese'mice'} \tag{4.23}$$

$$\mathbf{MADEOF'cheese'moon'} \tag{4.24}$$

$$\mathbf{MADEOF'cake'moon'} \tag{4.25}$$

If it has already been established that *cheese* maps to **cheese'** then 4.25 can be ruled out as a possible meaning since it doesn't contain the necessary semantic expression. Hypothesis 4.24, however, can not be ruled out. The learning algorithm attempts to learn from all remaining hypotheses. If all semantic hypotheses are ruled out then the learner assumes that one of the words in the utterance has multiple senses.

In an ideal situation a child will hypothesise the correct meaning for every utterance heard. However, assigning a meaning is not straight-forward; there is unlikely to be only one obvious candidate. When the correct meaning is not hypothesised, error has been introduced. An error of this type introduces a false association between word and meaning. To combat this problem a statistical error handler can be adopted: a confidence score can be assigned to word meanings according to their frequency and consistency. Word meanings whose confidence scores fell below a threshold value can be systematically pruned.

#### 4.4.2 Forming Augmented Strings

To recap, for each utterance heard the learner receives an input stream of word tokens paired with possible semantic hypotheses. For example, on hearing the utterance “Dinah drinks milk” the learner may receive the pairing: ( $\{dinah, drinks, milk\}$ , **DRINK’milk’dinah’**). The semantic learner attempts to learn the mapping between word tokens and semantic symbols, building a lexicon containing the meaning associated with each word sense; this is achieved by using cross-situational techniques. The learning system then can create augmented strings by using the Principle of Categorial Type Transparency; allowing basic syntactic information to be inferred from the semantic type and thus producing augmented strings. Remember that, from the semantic expression (**DRINK’ milk’ dinah’**) we know that **DRINK’** is a two argument predicate. Hence, the syntactic category of “drink” will also take two arguments. This knowledge can be represented in a skeleton syntactic category as  $A|B|C$  where  $A, B, C \in Tp$  and  $|$  is a variable over  $\backslash$  and  $/$ . An innate mapping is assumed from semantic entities to primitive types. In terms of a real learner this primitive-type-mapping is equivalent to having an innate ability to recognize groups of entities linked by some common theme; and then labelling all the entities in that group with the same mental tag. This can’t be too far off what children must actually do. For instance, it seems probable that children are innately aware of the concept of an object [71] and might therefore label books, tables and chairs with the same object tag.

In summary, to form augmented strings from the pairing ( $\{dinah, drinks, milk\}$ , **DRINK’milk’dinah’**) a learner has to:

1. segment the utterance on word boundaries using the speech perception system;

*dinah, drinks, milk*

2. hypothesise a semantic expression for the utterance using the conceptual system;

**DRINK’milk’dinah’**

3. map parts of the semantic expression to word tokens using the semantic learner;

$\{dinah, drinks, milk\}$	$\mapsto$	<b>DRINK’milk’dinah’</b>
<i>dinah</i>	$\mapsto$	<b>dinah’</b>
<i>drinks</i>	$\mapsto$	<b>DRINK’xy</b>
<i>milk</i>	$\mapsto$	<b>milk’</b>

4. use the Principle of Categorial Type Transparency and knowledge of primitive types to assign skeleton categories to words; thus forming an augmented string.

$$\begin{array}{lcl}
\textit{dinah} & \mapsto & np \\
\textit{drinks} & \mapsto & \langle A \mid B \rangle \mid C \\
\textit{milk} & \mapsto & np
\end{array}$$

where  $A, B, C \in Tp$  (the set of all categorial grammar types—primitive or otherwise) and  $\mid$  is a variable over  $\backslash$  and  $/$ .

$$\begin{array}{ccc}
\textit{dinah} & \textit{drinks} & \textit{milk} \\
\mid & \mid & \mid \\
np & \langle A \mid B \rangle \mid C & np
\end{array}$$

Note in this case we could have directly inferred the mapping  $\textit{drinks} \mapsto \langle A \mid np \rangle \mid np$  since the arguments to **DRINK'** were semantic entities which map directly to primitive types. We use the mapping  $\textit{drinks} \mapsto \langle A \mid B \rangle \mid C$  here (and in the following chapters) to illustrate the more general cases where the type of the arguments are unknown.

Augmented strings are simply a method of representing how the search space of parses is constrained by the unordered semantic representation. The next chapter will describe how concepts from the previous categorial grammar learners of Buszkowski and Waldron/Villavicencio may be updated and combined to form a new categorial grammar learner that can learn from real data; using augmented strings to constrain the search space of possible parses by representing the information provided in the associated semantic form.



# Chapter 5

## The Categorical Grammar Learner

Section 4.2.2 of Chapter 4 detailed Buszkowski’s algorithm [17] to learn rigid classic-categorical grammars from a set of functor-argument structures. Kanazawa [49] modified this algorithm to learn  $k$ -valued categorical grammars by introducing partial unification. Further, he showed that  $k$ -valued categorical grammars may also be learnt from a set of strings. Simple modification to these algorithms allowed the input to be a stream rather than a set (in line with Gold’s learning model). In this Chapter (Section 5.1) we will adapt this algorithm to learn from augmented strings, which (as discussed in Section 4.3) are structures that embody the constraints placed on possible parses by the semantic representation of a string. Section 5.2 details further alterations that must be made to the learner in order for it to learn from real data. These improvements to Buszkowski’s learner are inspired by the work of Villavicencio’s parameter learner [104] (see 4.2.1 for a summary). Section 5.3 provides a step-by-step example of the operation of the Categorical Grammar Learner.

### 5.1 Learning from Augmented Strings

As described in Section 4.2.2, there are three main sections to a general algorithm for learning a  $k$ -valued classic-categorical grammar from a stream of simple strings:

**Step a—Form Search-Space:** Form a set  $\mathcal{S}$  of all possible functor-argument structures to describe the new string.

**Step b—Hypothesise Grammars:** Iterate through  $\mathcal{S}$  assigning types to each functor-argument structure. Unify (using partial unification) the assigned types with the current grammar. If unification fails, then fail, else add possible grammar to grammar set,  $\mathbf{G}$ .

**Step c—Select Grammar:** Choose one grammar from set  $\mathbf{G}$ .

The complexity of the algorithm is critically dependent on the size of  $\mathcal{S}$  since we must iterate over this set in **step b**. When learning from a stream of functor-argument structures (as Buszkowski originally suggested),  $\mathcal{S}$  had a cardinality of 1; when learning from strings of length  $n$  it will have a cardinality of  $8^{n-1}$ .

To understand this let’s consider **step b** in more detail. To learn from a stream of simple strings, types must be assigned to every possible functor-argument structure of the current string. As explained in Section 4.3.5 a string of length  $n$  will produce a number of functor-argument

structures bounded by  $8^{n-1}$ . Therefore, for strings of length  $n$ , the learner has to make around  $8^{n-1}$  iterations of **step b**.

Section 4.3.5 showed that simple strings carry minimum information content and therefore produce the maximum possible number of functor-argument structures as possible parses. Using other types of sentence objects as input to the categorial grammar learner would reduce the size of  $\mathcal{S}$  and thus reduce the complexity of the algorithm. For general case we can rewrite *step a* of the algorithm as:

**Step a—Form Search-Space:** Form a set  $\mathcal{S}$  of all possible functor-argument structures to describe the new sentence object.

In general, note that by expressing the input to a learner in terms of sentence objects it is possible to contextualise the complexity of the problem the learner is solving.

Section 4.3 postulated that augmented strings carry a more cognitively realistic data content than simple strings. Augmented strings were word tokens “*augmented*” by some extra syntactic content which could be derived from semantics. By learning from augmented strings the search-space is reduced by a constant factor, whose size is dependent on the extra syntactic information provided with the string (see Section 4.3 for a discussion).

The basic algorithm for learning from a stream of augmented strings is similar to that for learning from a stream of simple strings; the difference being that the search-space (the size of set  $\mathcal{S}$ ) is smaller. As mentioned in the previous Section 4.2.2, Kanazawa proved that  $k$ -valued classic categorial grammars may be learnt from strings. Since augmented strings carry the same (and more) information it is also possible to learn  $k$ -valued classic categorial grammars from augmented strings.

## 5.2 Improving the Categorial Grammar Learner

Natural language can not be modelled using the rule of function application alone (as discussed in Section 4.1.2). This section discusses improvements to the categorial grammar learners that make it possible to learn from real language examples. Adding greater range to the describable language will obviously increase the hypothesis-space of allowed grammars (referred to as  $\Omega$  in Gold’s language Model—Chapter 3). Steps are taken to reduce complexity; for instance, rule selection is guided by a set of heuristics. In addition, some duplicated effort in the unification step is removed by looking up word types in the lexicon before assigning types. A memory module is introduced to help in the procedure of selecting one grammar from the many produced. Kanawaza’s method of selecting a grammar is exponential to the size of the grammars; the method suggested here is linear—involving a series of validations based on the current state of the memory. Additionally, the statistical methods used to maintain integrity of the Memory Module ensure that the learner is robust to noise in the input data. The improvements to the categorial grammar learner are discussed with relevance to augmented strings, as this the most appropriate type of sentence object for learning problems involving real data.

### 5.2.1 Introducing a Memory Module

The Memory Module is used as an aid to the learner in selecting one grammar from the many produced. The categorial grammar learner can be entirely functional without its memory; gram-

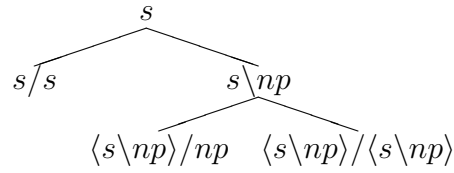


Figure 5.1: A fragment of a categorial type hierarchy

grams could be selected by the same method as Kanazawa. However, the use of a memory significantly speeds the process of selecting one grammar from the many hypothesised. The method Kanazawa uses to select a grammar is exponential to the size of the grammars (see Section 4.2.2). The method used here is linear: every item in each grammar’s lexicon is checked for validity with respect to the current state of the memory.

Inspired by Villavicencio’s categorial and word-order parameters (see Section 4.2.1), the Memory Module records details of two distinct features; there is a type memory and a word-order memory. There is no interaction between the two parts of the memory.

### Type Memory

The lexicon for a language contains a finite subset of all possible types (a subset of  $Tp$ ), the size of which depends on the language—Pullum [82] suggests that for English the lexical functional categories never need more than five arguments and that these are needed only in a limited number of cases (such as for the verb *bet* in the sentence *I bet you five pounds for England to win*).

Unlike Villavicencio’s parameter learner (which is restricted to 89 categories) the Categorial Grammar Learner is completely unrestricted in the categories it is allowed to hypothesise for a word. The type memory is used to keep track of which types have been inferred from the stream of language examples thus far. The memory then facilitates grammar selection by placing constraints on the hypothesised set of grammars,  $G$ , based on its current content.

Following the principles of Villavicencio’s UG, the type memory is structured as a hierarchy. Recall that complex types of a categorial grammar are a combination of simpler types: thus, types may be arranged in a hierarchy with more complex types inheriting from simpler ones. Consider a categorial grammar with primitive types  $Pr = \{s, np\}$ . Figure 5.1 shows a fragment of a possible hierarchy.

A type is set to ACTIVE within the Type Memory if it has been successfully inferred by the categorial grammar learner: let  $T_{ACTIVE} = \{all\ ACTIVE\ types\}$  and let all primitive types be ACTIVE by default  $Pr \subset T_{ACTIVE}$ . A type  $j$  is considered to be the direct descendant of another type,  $k$ , if  $j$  has one extra argument than  $k$  and that argument is already ACTIVE in the hierarchy (i.e.  $argument \in T_{ACTIVE}$ ). Originally only the primitive types are ACTIVE. Types are deemed POSSIBLE if they are the direct descendant of an ACTIVE type. All other types are INACTIVE (see Figure 5.2).

The grammar set returned by the learner,  $G$ , can be reduced in size by excluding grammars that contain INACTIVE types. Let the hierarchy above,  $H_1$ , be the current state of the Type Memory and assume that the learner has returned grammar set  $G = \{G_1, G_2\}$ .

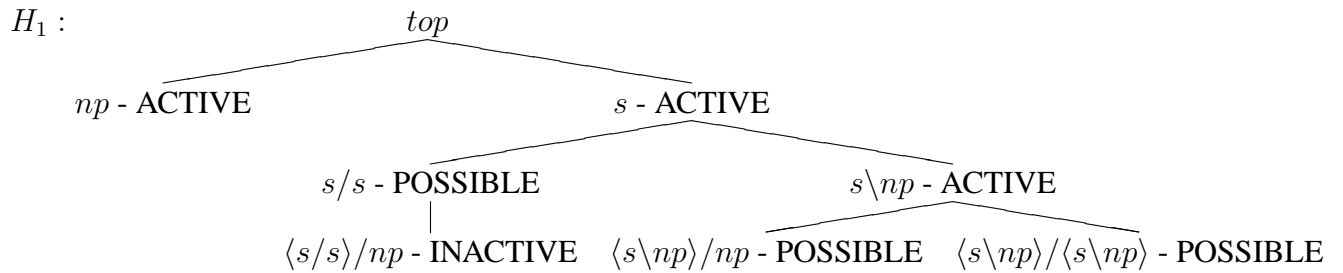


Figure 5.2: An example state of the Type Memory

$G_1 :$  word1  $\rightarrow$   $np$   
word2  $\rightarrow$   $s\np$   
word3  $\rightarrow$   $(s\np)/np$   
 $\vdots$

$G_2 :$  word1  $\rightarrow$   $np$   
word2  $\rightarrow$   $s\np$   
word3  $\rightarrow$   $(s/np)\np$   
 $\vdots$

We can exclude grammar  $G_2$  because the type  $(s/np)\np$  is inactive.

### Word Order Memory

The Word Order Memory keeps track of the underlying order in which constituents appear. The information it stores is extracted directly from semantic content associated with the input stream. We assume that a child associates a role (or argument type) with each argument of a predication. For instance, let input string “Dinah likes Alice” be associated with the semantic representation (**LIKES’alice’dinah’**); in this case we assume that the child knows that **dinah’** is the actor and **alice’** is the undergoer. The Word Order Memory would “remember” that the actor was found on the left of the word associated with the predication and the undergoer was found on the right.

This might be represented in the Word Order Memory as:

$W_1 :$  UNDERGOER-DIRECTION: /  
ACTOR-DIRECTION: \  
ARG1-DIRECTION: |  
 $\vdots$   
GENERAL-DIRECTION |

where / indicates that the argument is generally found on the right, \  
indicates that the argument is generally found on the left and | indicates that a directional tendency is yet to be found. The



directions of arguments within the memory are maintained statistically. The mechanism for this is explained in Section 5.2.2 below. GENERAL-DIRECTION keeps track of the overall directional tendency.

Now, let  $W_1$  be the current state of the Word Order Memory. Assume that the learner has been exposed to a sentence containing *word1 word2*, with a semantic representation that indicates that *word1* is the actor of *word2*. The learner has returned the grammar set  $\mathbf{G} = \{G_1, G_2\}$

$$\begin{array}{l}
 G_1 : \text{word1} \rightarrow np \\
 \quad \text{word2} \rightarrow s \backslash np \\
 \quad \quad \quad \vdots \\
 \\
 G_2 : \text{word1} \rightarrow np \\
 \quad \text{word2} \rightarrow (s \backslash np) / np \\
 \quad \quad \quad \vdots
 \end{array}$$

Grammar  $G_1$  is chosen over  $G_2$  because the type  $s \backslash np$  takes arguments from the left, which is the general preference for the location of *actors* in this language.

## 5.2.2 Setting the Memory and Memory Integrity

In order to combat noisy data a statistical method is used to maintain confidence in the settings of the Memory Module. For every POSSIBLE and ACTIVE category  $C$  a count is kept,  $n(C)$ , which is the number of times that category has been hypothesised since it became POSSIBLE. Now let  $N = \sum_C n(C)$  and  $N_i$  be the sum of all  $n(C)$  for  $C$  that occur at level  $i$  in the hierarchy; let  $\epsilon$  be a threshold value between 0 and 1. If at the end of the current iteration any of the POSSIBLE categories occurring in level  $i$  satisfy  $\frac{n(C)}{N_i} > \epsilon$  then they become ACTIVE. If any of the ACTIVE categories at level  $i$  satisfy  $\frac{n(C)}{N_i} < \epsilon$  then they revert to POSSIBLE and all of their children become INACTIVE. Hence ACTIVE tags are only assigned when sufficient evidence has been accumulated, i.e. once the associated probability reaches a threshold value. By employing this method, it becomes unlikely for memory settings to fluctuate as the consequence of an erroneous utterance. The Word Order Memory operates similarly but with both a low and high threshold to determine when to switch between | and \ or | and /.

**Type Memory:** an ACTIVE tag will only become set once enough evidence has been accumulated; i.e. once a threshold value is reached.

**Word Order Memory:** each direction tag is associated with a value between 0 and 1; the tag will remain as | unless it exceeds or subceeds a threshold.

## 5.2.3 Reducing Duplicated Effort

Consider the strings *mice scare elephants* and *cats scare mice*. Since the use of *scare* is identical in each string, it is a wasted effort for the categorial grammar learner to infer the type of *scare*

from both. If the learner is accurate, the type of *scare* will be inferred correctly the first time it is encountered.

In order to reduce duplicated effort in type assignment and unification, the words of the current string may be looked up in the current grammar ( $G_i$ ) and assigned directly if appropriate:

Let the current grammar be:

$$\begin{aligned} G_i : \text{you} &\rightarrow np \\ \text{eat} &\rightarrow s \backslash np \end{aligned}$$

Consider the input string *lions eat* with semantic representation **EAT'lion'**. This gives the following augmented string:

$$\begin{array}{cc} \text{Lions} & \text{eat} \\ | & | \\ np & A | B \end{array}$$

Looking up *eat* in the lexicon will return the type  $s \backslash np$ . The *skeleton type*  $A | B$  unifies with  $s \backslash np$  so it is directly assigned.

$$\begin{array}{cc} \text{Lions} & \text{eat} \\ | & | \\ np & s \backslash np \end{array}$$

If the skeleton type does not unify with any lexical entry then no assignment is made; the type must be inferred. If there is more than one lexical entry which will unify with the skeleton type then the most likely (according to the Type Memory) is chosen. On parse failure, the next most likely type is selected—if no more types are available then the original skeleton type is used.

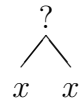
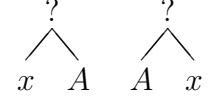

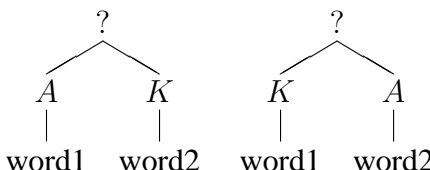
The benefit of this methodology is to build on previous knowledge; as alluded to in the study of subcategorization frames in child speech and discussion of Brown's stages (see Section 2.2.1 of Chapter 2). Assigning types can greatly reduce the search-space (the number of possible functor-argument structures) and also place constraints upon how the rules might be used.

## 5.2.4 Using Additional Rules

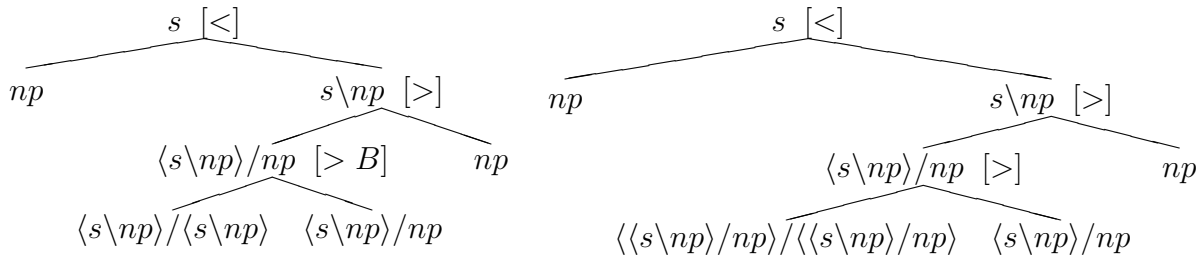
Another improvement to the basic categorial grammar learner is to add the rules of function composition and Generalised Weak Permutation (GWP); which are required to capture non-constituent co-ordination and relative clauses etc. The motivation for selecting these particular rules is primarily based on the corpus we use for testing this learner. Villavicencio [104] built a grammar to describe a section of the Sachs corpus of CHILDES [63] using the rules of application, composition and GWP; we shall evaluate performance of our learner against this grammar. Furthermore, use of function composition is standard for capturing non-constituent co-ordination. In general, we have no particular preference for using GWP over type raising in order to capture co-ordinate structures and unbounded dependencies. However, see Chapter 4 in Section 4.1.3 for a discussion of why GWP is preferable for this learning problem. Note that use of GWP is restricted to use at the leaves of a derivation.

The algorithm is updated by selecting rules at each binary branch node according to a set of heuristics. The following table lists the heuristics:  $x$  represents a primitive type ( $x \in Pr$ ),  $A$

represents a non-primitive type ( $A \in Tp \wedge A \notin Pr$ ),  $K$  represents a fully specified non-primitive type at a leaf node.

Scenario:	Action:
(1) 	Fail.
(2) 	Apply function application. On failure, consider heuristic 4. If heuristic 4 is not applicable, then fail.
(3) 	Apply function composition. On failure, apply function application. If application fails, then consider heuristic 4.
(4) 	If $K$ is a fully specified non-primitive type and is situated at a leaf node then allow Generalised Weak Permutation. If permutation fails then fail.

Since we always know something about the type of each leaf node, heuristics are always applied from the leaves upwards. A failure percolates down the parse tree to the nearest node where a *next choice* option still exists. When there are no next choice options left, the whole parse fails. Heuristic (3) allows function composition to proceed in preference to function application. This heuristic is necessary for the scenario when both composition and application will lead to valid parses but the parse using application produces an over-generation.



Heuristic (4) stops the blow up in complexity that would normally accompany the addition of the rule for Generalised Weak Permutation. The rule is used only as a last resort and may only be applied at the leaf nodes.

The basic algorithm has now been updated as follows:

**Step a—Form Search-Space:** Look up the string's words in the current grammar ( $G_i$ ). Make assignments if appropriate (see Section 5.2.3). Form a set ( $\mathcal{S}$ ) of all functor-argument structures to describe the new augmented string.

**Step b—Hypothesise Grammars:** For each  $s_c \in \mathcal{S}$ :

**Step 1:** Assign types to the structure:

- (A) Assign sentence type to root node.
- (B) (a) Infer types for remaining nodes using rules of function application, function composition and Generalised Weak Permutation in accordance with the heuristics.

(b) If (a) fails then, if there are more possible type assignments (including the skeleton type), repeat from **Step a** using the next most likely assignments, else exit.

**Step 2:** Create new grammar,  $G_{i(c)}$ , by adding types at leaf nodes to the sets of types in the current lexicon or by adding a new entry where necessary.

**Step 3:** Unify the sets of types in lexical entries of  $G_{i(c)}$ .

**Step 4:** If  $G_{i(c)}$  is unique (i.e. not a duplicate of another possible grammar), then add it to the set  $G$ .

**Step c—Select Grammar:** Select one grammar from  $G$  using the Type Memory and the Word Order Memory (as explained in Section 5.2.1). Update the Memory Module settings.

### 5.3 Categorical Grammar Learner—Worked Example

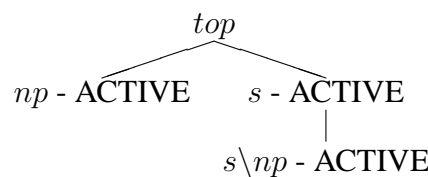
This Chapter is best illustrated by example. For the purpose of illustration a non-noisy context is assumed (no thresholds); consequently Memory Module settings may be altered on the evidence from a single input. The improved categorical grammar learner will be tracked as it learns from three strings: *lions eat*; *you eat biscuits*; and *lions might eat you*.

Let the current grammar  $G_i$ , current Type Memory  $H_i$  and current Word Order Memory  $W_i$  be as follows:

$G_i$ :

eat  $\rightarrow$   $s \backslash np$   
 you  $\rightarrow$   $np$

$H_i$ :



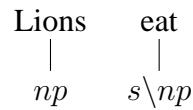
$W_i$ :

ACTOR-DIRECTION:  $\backslash$   
 GENERAL-DIRECTION:  $/$

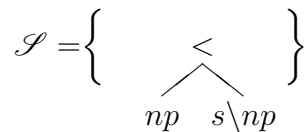
The learner now encounters  $s_i = \textit{lions eat}$ , with associated semantic content **EAT'lion'**, where **lion'** is the actor of **EAT'**. The augmented string will be:

Lions      eat  
 |          |  
 np      A | B

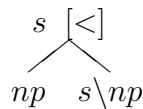
**Step a—Form Search-Space:** The word *eat* is in the lexicon and its type,  $s \backslash np$ , unifies with  $A \mid B$  giving:



The presence of the primitive type,  $np$ , forces backward application; consequently there is only one possible functor-argument structure ( $s_a \in \mathcal{S}$ ):



**Step b—Hypothesise Grammars** Using heuristic (2) we arrive at the following valid parse tree and grammar:



$$\begin{array}{l} G_{i(a)} : \text{eat} \rightarrow s \backslash np \\ \text{lions} \rightarrow np \\ \text{you} \rightarrow np \end{array}$$

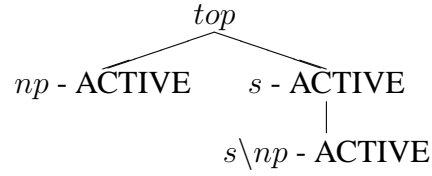
There are no unknowns in the grammar so there is no need to unify.

**Step c—Select Grammar**  $G_{i(a)}$  is the only grammar in  $\mathbf{G}$ .  $G_{i(a)}$  is selected to become the new current grammar,  $G_{i+1}$ ; none of its types are INACTIVE in the Type Memory. The Memory Module settings are recalculated as described. The counts for the ACTIVE tag  $s \backslash np$  is incremented, as is the “leftwardness” of the ACTOR-DIRECTION and the GENERAL-DIRECTION. The “rightwardness” of the GENERAL-DIRECTION is contradicted but for this example we shall say it is not enough to switch the setting; this would mean that the current utterance has not provided enough evidence to outweigh the previously accumulated evidence for this setting.

$G_{i+1}$ :

$$\begin{array}{l} \text{eat} \rightarrow s \backslash np \\ \text{lions} \rightarrow np \\ \text{you} \rightarrow np \end{array}$$

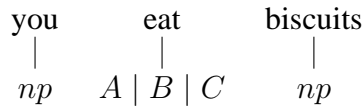
$H_{i+1}$ :



$W_{i+1}$ :

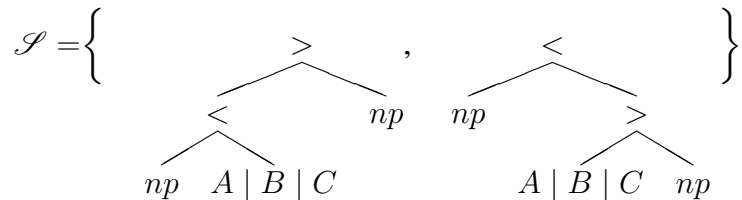
ACTOR-DIRECTION:  $\backslash$   
GENERAL-DIRECTION:  $/$

The next string to be encountered is *you eat biscuits*, with associated semantic content **EAT’biscuit’you’**, where **you’** is the actor of **EAT’** and **biscuit’** is the undergoer. The augmented string will be:

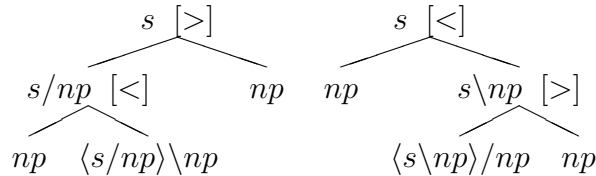


**Step a—Form Search-Space:** The word *eat* is in the lexicon,  $G_{i+1}$ , but its type,  $s\np$ , does not unify with  $(A | B) | C$ .

Due to the presence of two primitive types there are only two possible functor-argument structures,  $s_a, s_b \in \mathcal{S}$ :



**Step b—Hypothesise Grammars** Using heuristic (2) we arrive at the following valid parse trees and grammars:



$G_{i+1(a)}$  : biscuits  $\rightarrow$   $np$   
eat  $\rightarrow$   $s\np, (s/np)\np$   
lions  $\rightarrow$   $np$   
you  $\rightarrow$   $np$

$G_{i+1(b)}$  : biscuits  $\rightarrow np$   
           eat  $\rightarrow s \backslash np, (s \backslash np) / np$   
           lions  $\rightarrow np$   
           you  $\rightarrow np$

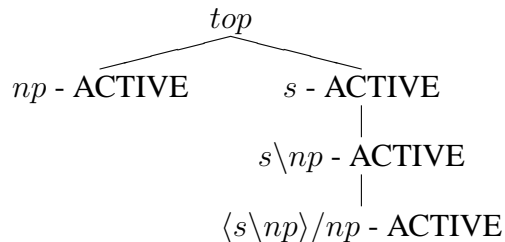
There are no unknowns in the grammars so there is no need to unify.

**Step c—Select Grammar**  $G_{i+1(b)}$  is selected as the new current grammar,  $G_{i+2}$ ; the type  $(s/np) \backslash np$  in grammar  $G_{i+1(a)}$  is INACTIVE in the Type Memory where as type  $(s \backslash np) / np$  (from  $G_{i+1(b)}$ ) is POSSIBLE. The memory module settings are recalculated: the tag for  $(s \backslash np) / np$  is set to ACTIVE; the UNDERGOER-DIRECTION is activated; the “leftwardness” of the ACTOR-DIRECTION is reinforced; and the “rightwardness” of the GENERAL-DIRECTION is both contradicted and reinforced, thus remaining constant.

$G_{i+2}$ :

biscuits  $\rightarrow np$   
           eat  $\rightarrow s \backslash np, (s \backslash np) / np$   
           lions  $\rightarrow np$   
           you  $\rightarrow np$

$H_{i+2}$ :



$W_{i+2}$ :

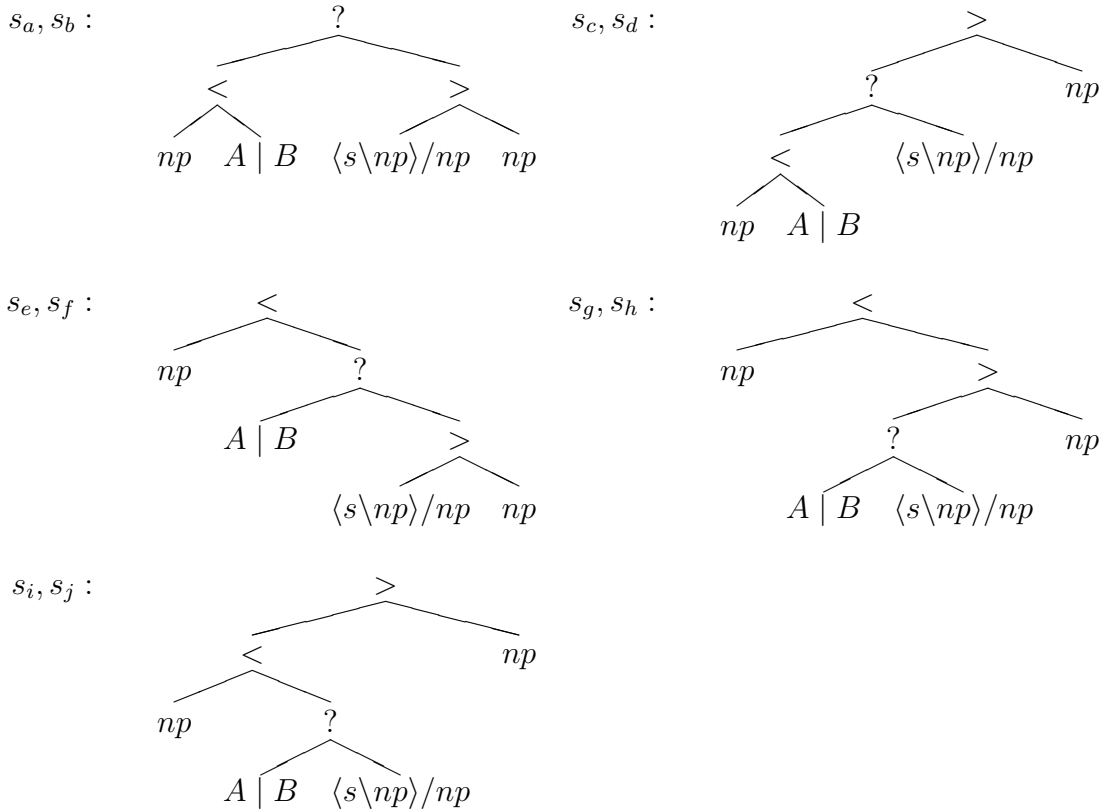
ACTOR-DIRECTION:       \  
 UNDERGOER-DIRECTION:  /  
 GENERAL-DIRECTION:     /

The final string to be encountered is *lions might eat you*, with associated semantic content **MIGHT’(EAT’you’lion’)**, where **EAT’** is the argument of **MIGHT’**. The augmented string will be:

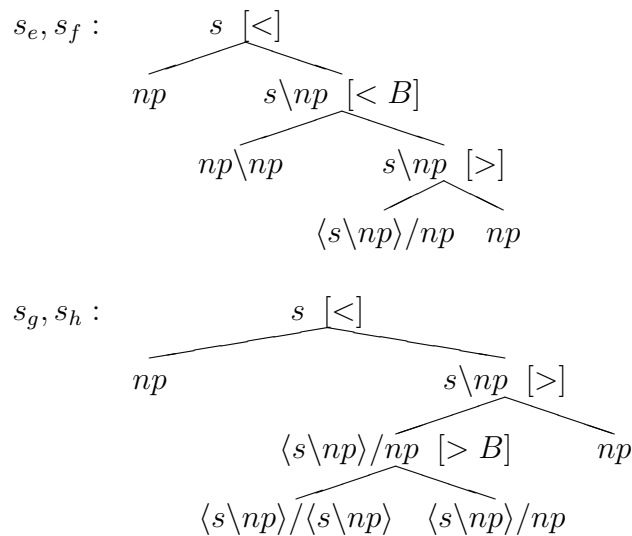
          lions    might       eat       you  
           |        |        |        |  
           np     A | B    A | B | C   np

**Step a—Form Search-Space:** The word *eat* is in the lexicon ( $G_{i+2}$ ) and one of its types,  $\langle s \backslash np \rangle / np$ , unifies with  $(A \mid B) \mid C$ .

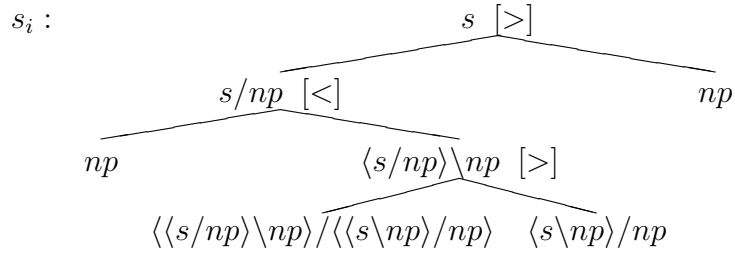
There are 10 possible functor-argument structures in  $\mathcal{S}$ . The ? symbol is being used as shorthand for  $>$  or  $<$ :



**Step b—Hypothesise Grammars**  $s_a, s_b, s_c, s_d$  and  $s_j$  fail to parse.  $s_e, s_f, s_g$ , and  $s_h$  succeed, making use of both heuristics (2) and (3).  $s_i$  succeeds using only heuristic (2).







Giving three unique grammars:

$G_{i+2(e)} :$  biscuits  $\rightarrow np$   
 eat  $\rightarrow s\np, (s/np)\np$   
 lions  $\rightarrow np$   
 might  $\rightarrow np\np$   
 you  $\rightarrow np$

$G_{i+2(g)} :$  biscuits  $\rightarrow np$   
 eat  $\rightarrow s\np, (s\np)/np$   
 lions  $\rightarrow np$   
 might  $\rightarrow (s\np)/(s\np)$   
 you  $\rightarrow np$

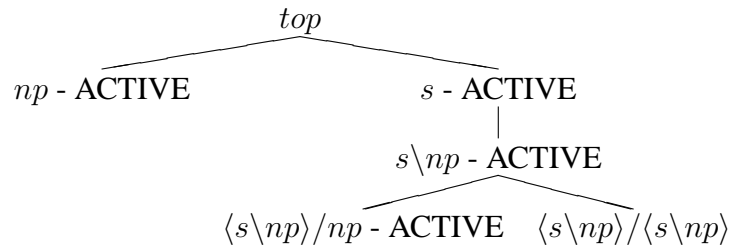
$G_{i+2(i)} :$  biscuits  $\rightarrow np$   
 eat  $\rightarrow s\np, (s\np)/np$   
 lions  $\rightarrow np$   
 might  $\rightarrow ((s/np)\np)/((s\np)/np)$   
 you  $\rightarrow np$

**Step c—Select Grammar**  $G_{i+2(g)}$  is selected as the new current grammar,  $G_{i+3}$ . This is because i) the type  $((s/np)\np)/((s\np)/np)$  from grammar  $G_{i+2(i)}$  is INACTIVE in the Type Memory; ii) without any memory data relating specifically to the type of argument that *might* takes, grammar  $G_{i+2(g)}$  is chosen over  $G_{i+2(e)}$  since the type  $(s\np)/(s\np)$  takes arguments from the right, which is the general preference of the language. The Memory Module settings are recalculated: the tag for  $(s\np)/(s\np)$  is set to ACTIVE; the ARG1-DIRECTION is activated (since ARG1 is the type of argument taken by **MIGHT**); and the “rightwardness” of the GENERAL-DIRECTION is reinforced.

$G_{i+3}$ :

biscuits  $\rightarrow np$   
 eat  $\rightarrow s \backslash np, (s \backslash np) / np$   
 lions  $\rightarrow np$   
 might  $\rightarrow (s \backslash np) / (s \backslash np)$   
 you  $\rightarrow np$

$H_{i+3}$ :



$W_{i+3}$ :

ACTOR-DIRECTION:        \  
 UNDERGOER-DIRECTION:    /  
 ARG1-DIRECTION:        /  
 GENERAL-DIRECTION:      /

## 5.4 Categorical Grammar Learner Summary

This chapter has described a Categorical Grammar Learner that can learn from streams. In order that it may describe real languages, the learner operates on syntactic categories using the rules of function application, function composition and Generalised Weak Permutation. This introduces complexity over a learner that just uses the rules of function application. In order to reduce complexity, heuristics are employed to guide rule selection. The introduction of a Memory Module serves two major functions: the first is to aid in the selection of one grammar from the many hypothesised; the second is to make the learner robust to the noise that is present in real linguistic input (see Chapter 6). The Memory Module comprises of two sections; the Type Memory, which records the syntactic categories that have been learnt and how often they have been seen; and the Word Order Memory, which records the directional tendencies of the language. By expressing the Type Memory as a hierarchical structure, constraints are placed on the syntactic types that the learner is allowed to hypothesise; a type may only be learnt if its direct parent has been learnt. Thus, the learner builds knowledge incrementally (as alluded to in Chapter 2).

# Chapter 6

## Evaluation of Model

In this chapter the Categorical Grammar Learner (CGL) will first be evaluated with respect to other parameter based learners. In Section 6.1 we compare the characteristics of the CGL with general parametric learners. In particular the influence of and differences to the Waldron/Villavicencio Learning System are discussed. Section 6.2 then looks at the efficiency of the CGL when learning within an ideal (noiseless) environment. Its performance is evaluated and compared to the Triggering Learning Algorithm and the Structural Triggers Learner within Gibson and Wexler's 3-parameter grammar-space. In Section 6.3 the real world validity of the model will be demonstrated with experimentation to show that the CGL can learn from real (noisy) data. Finally in Section 6.4 the model's developmental compatibility will be discussed with reference to Brown's stages and the subcategorization frame acquisition experiments that were detailed in Chapter 2.

### 6.1 Comparison to Previous Parameter Based Learners

The properties of the parameters used by the CGL are as follows: parameters are lexical; parameters are organised hierarchically; parameter setting is statistical.

**Lexical Parameters:** The CGL employs parameter setting as a means to acquire a lexicon; differing from other parametric learners (such as the Triggering Learning Algorithm (TLA) [39] and the Structural Triggers Learner (STL) [38], [90]), which acquire general syntactic information rather than the syntactic properties associated with individual words.<sup>1</sup>

The categorial grammar parameters of the CGL are concerned with defining the set of syntactic categories present in the language of the environment. Converging on the correct set aids acquisition by constraining the learner's hypothesised syntactic categories for an unknown word. A parameter (with value of either ACTIVE, INACTIVE or POSSIBLE) is associated with every possible syntactic category to indicate whether the learner considers the category to be part of the target grammar.

Some previous parametric learners (TLA and STL) have been primarily concerned with overall syntactic phenomena rather than the syntactic properties of individual words. Movement parameters (such as the *V2* parameter of the TLA) may be captured by the

---

<sup>1</sup>The concept of lexical parameters and the lexical-linking of parameters is to be attributed to Borer [8].

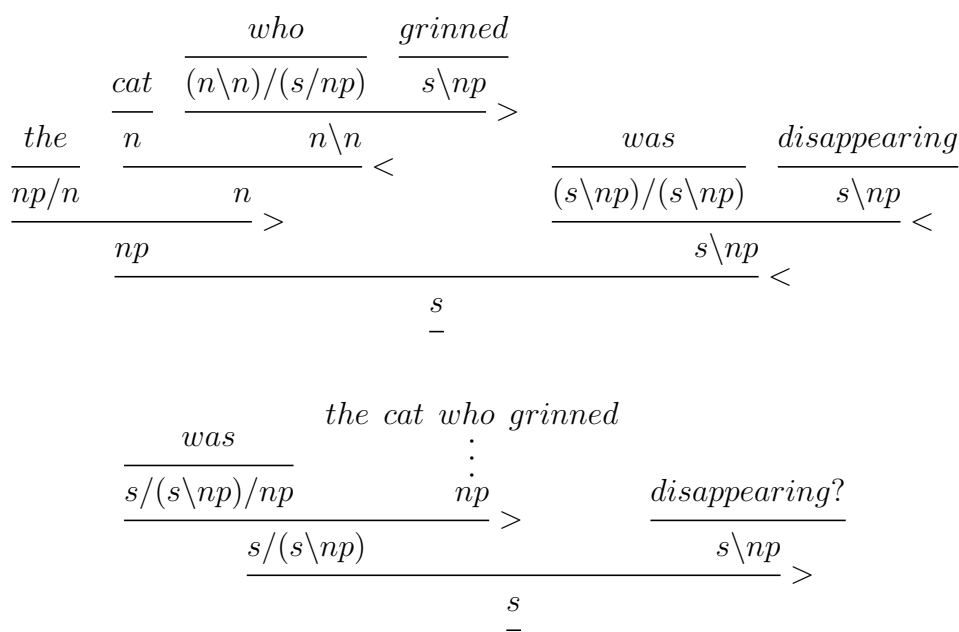


Figure 6.1: Illustration of the interrogative “was the cat who grinned disappearing?”.

CGL using multiple lexical entries. For instance, Dutch and German word order is captured by assuming that verbs in these languages have two categories, one determining main-clause order and the other subordinate-clause orders. Figure 6.1 illustrates how an interrogative may be derived by using multiple lexical entries. The word “was” has two entries in the lexicon; one determining auxiliary form and the other interrogative form.

**Hierarchical Parameters:** The complex syntactic categories of a categorial grammar are a subcategorization of simpler categories; consequently categories may be arranged in a hierarchy with more complex categories inheriting from simpler ones. Figure 6.2 shows a fragment of a possible hierarchy. This hierarchical organization of parameters provides the learner with several benefits. The hierarchy can enforce an order on learning; for instance in the CGL presented here, a constraint is imposed such that a parent parameter must be acquired before a child parameter (for example, in Figure 6.2, the learner must acquire intransitive verbs before transitive verbs may be hypothesised). Another possible function of hierarchical parameters is that values may be inherited as a method of acquisition. Such a CGL was implemented by Villavicencio [104].

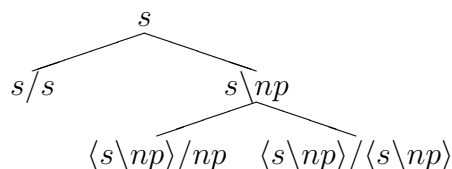


Figure 6.2: Partial hierarchy of syntactic categories

**Statistical Parameter Setting:** The CGL uses a statistical method to track the relative frequen-

cies of parameter-setting-utterances in the input. Such an approach sets parameters only if there is enough accumulated evidence. This represents a compromise between two extremes: implementations of the TLA are memoryless allowing parameter values to oscillate; while some implementations of the STL set a parameter once, for all time.<sup>2</sup> The CGL uses a threshold value to indicate when enough evidence has been accumulated. An alternative method would be to maintain a probability distribution over parameters as in Yang's Variational Learner [109]. However, this method does not seem appropriate in this case since parameters (possible type categories) are not predefined before learning commences.

A further difference between the CGL and most parametric learners is that it induces its grammar from string/semantic representation pairs (by means of an augmented string) rather than from a simple string alone. However, this has similarities with the very recent work of Zettlemoyer and Collins [112] in that both systems infer a combinatory categorial grammar from string/semantic representation pairs; although Zettlemoyer and Collins' CCG uses type-raising instead of permutation. The basic principles of the two learners are very similar; the difference is in the application and implementation. For instance, rather than use a memory module to select between hypothesised grammars, Zettlemoyer and Collins use dynamic programming to create a probability distribution over parses. Furthermore, Zettlemoyer and Collins use predefined trigger rules to hypothesise 1 of 8 possible syntactic categories. The CGL, on the other hand, is not constrained by a set of possible categories; only in the order that it acquires them. All said, the systems are trying to solve two different problems. Zettlemoyer and Collins are mapping natural language interfaces to database queries, whereas the CGL attempts to be cognitively plausible and developmentally compatible with human learning.

### 6.1.1 Influence of and Differences to Waldron/Villavicencio Learning System

The Waldron/Villavicencio Learning System (see Section 4.2.1) provides the inspiration for the Memory Module of the CGL. Villavicencio defined the Principles of a Universal Grammar to be a subset (cardinality 89) of all the possible categorial grammar categories that take up to five arguments. These syntactic categories are arranged in a hierarchy so that child types can inherit syntactic/semantic information from their parents. The parameters of this Universal Grammar are embedded within the hierarchy. For instance, each category has a categorial-parameter (e.g. *intransitive-parameter*) that will take a Boolean value. If the category is active in the current grammar then this attribute is set to *true*; otherwise it is *false*. A categorial parameter can be set to *true* if its associated trigger has been detected and if its direct parent in its group hierarchy is also true. The CGL adopts this type of category hierarchy and also the criteria for category activation. However, the major difference between Villavicencio's Universal Grammar parameters and the CGL Memory Module is that Villavicencio predefines parameters (and syntactic categories) before learning commences; this approach requires an increase of innate knowledge. The CGL Memory Module requires no predefinition; it is built during acquisition as a consequence of the categories acquired and of the arguments observed in semantic expressions. Furthermore, in Waldron/Villavicencio's system, the current grammar plays no part in constraining hypothesised syntactic categories; the Memory Module of the CGL, however, con-

---

<sup>2</sup>Although there is one version of the STL (the Guessing STL) that does employ a statistical method [37]

strains hypotheses to aid the speed of acquisition. In fact Waldron/Villavicencio are solving a slightly different problem. They present a system for learning the settings of the parameters of a Universal Grammar. The acquired grammar can then be used to parse and produce language. In order to acquire the parameter settings, the learner has to first use Waldron’s syntactic learner to hypothesise categories for the utterance heard. The mechanisms for acquisition are distinct to those for utilising the grammar being acquired. Contrastively, in the CGL, a single set of rules are required for acquisition, parsing and production. Acquisition occurs “naturally” as a consequence of trying to parse the sentence as an adult learner would do. As a consequence the CGL has no requirement for Villavicencio’s *Trigger Identification Model*. However, note here another similarity; the incremental learning nature of both of the systems reduces the notion of a trigger to “a string that contains only a small amount of unknown grammatical information”. Computationally, the Waldron/Villavicencio’s learner displays inefficiencies over the CGL. Waldron’s syntactic learner is burdened with the complexity of learning from simple strings. All possible syntactic categories are discovered by his learner; invalid categories are only later excluded by the *Valid Category Assignment Detection Module*. In the case of the CGL (which learns from augmented strings created from the associated semantic representations), categories that are not compatible with a word’s associated semantic form are never hypothesised in the first place.

### 6.1.2 Possible Problems with the Principle of Categorial Type Transparency

Both the Waldron/Villavicencio System and the CGL make crucial use of the *Principle of Categorial Type Transparency*: Villavicencio uses this principle to filter invalid categories in the *Valid Category Assignment Detection Module*; the CGL uses the principle to create augmented strings. It is commonly argued that the use of this principle could potentially cause problems for the CGL. The line of thought here is that it is not always the case that the number of syntactic arguments for a word is the same as the semantic arity. For example, consider the following sentence “*Ian seems to be happy*”. The word *seems* in this sentence has two syntactic arguments *Ian* and *to be happy* but its semantic arity is one; having a semantic representation something like **SEEM’(HAPPY’ian’)**.

However, these constructions are not a problem for the CGL nor for the Principle itself since it only states a functional relationship between semantic type and syntactic category. Figure 6.3 shows how the CGL learns the syntactic category for *seems*. In Step 1, the skeleton category is formed from the semantics: **SEEM’x**  $\rightarrow$   $A \mid B$  where  $A$  and  $B$  are allowed to be any ACTIVE syntactic categories. In Step 2, function application is used to infer the type of  $B$ ; also the parent node (the result of the function application) is labelled  $A$ . At Step 3, the CGL utilises function application again to infer that the type of  $A$  must be  $s \setminus np$ . This type percolates down the tree in Step 4 giving the syntactic type of *seems* as  $(s \setminus np) / (s \setminus np)$  or simply  $s \setminus np / (s \setminus np)$  since the categories are left associative. Thus the CGL is able to deal with the problem elegantly since the only constraints placed on the categories  $A$  and  $B$  were that they were ACTIVE; the CGL insists on an argument being a primitive type only if its semantic type is an entity like **ian’**.

In general, this mode of operation of a learning system, whereby it uses the semantic properties of words as a cue to their syntactic category, is referred to as *semantic bootstrapping* [74]. The theory requires that semantic information is available in order for the syntactic learning process to begin. Currently, the CGL can not learn anything about a word’s syntax until it knows about its semantics. However, this doesn’t imply that it is impossible to learn syntax before

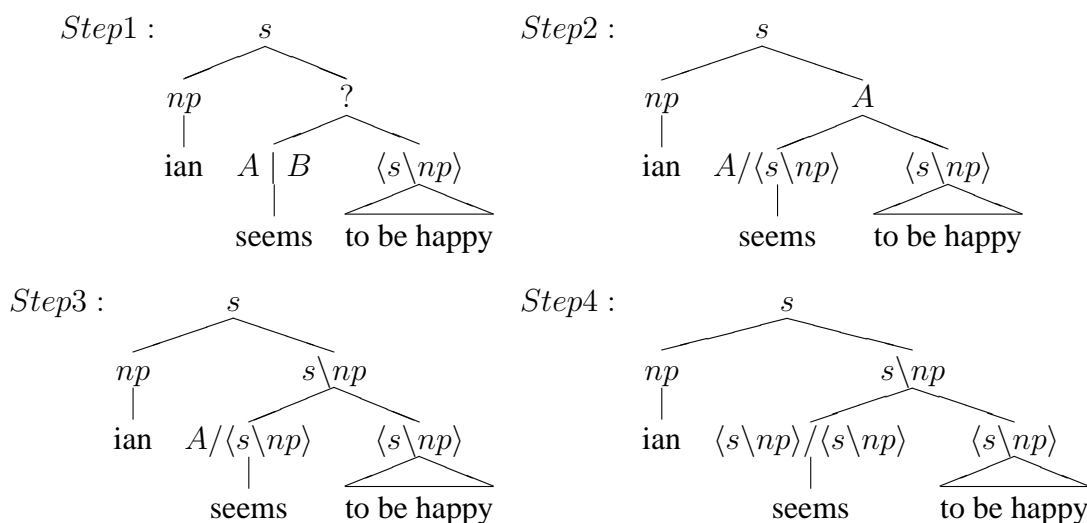


Figure 6.3: Derivation of *Ian seems to be happy*.

semantics. Consider the case where the syntactic category of every word but one is already known; the category of the new word could often be inferred by inspection of the parse tree without reference to any semantic information. This would be a simple extension to the CGL.

## 6.2 Learning in an Ideal Environment

The CGL is *formally sufficient* (can be formally demonstrated to acquire language) for languages defined by a classic categorial grammar. Kanazawa proved that  $k$ -valued classic categorial grammars may be learnt from strings. The augmented strings which this learner learns from carry the same (and more) information; hence it is also possible to learn  $k$ -valued classic categorial grammars from augmented strings.

In order to learn a  $k$ -valued classic categorial grammar from a stream of simple strings, is it necessary to investigate every possible functor-argument structure derivable from every string. For a string of length  $n$  the number of functor-argument structures is bounded by  $8^{n-1}$ . For augmented strings there will be somewhat less functor-argument structures; for instance, the introduction of  $m$  primitive types will reduce the number of trees by a factor of at least  $2^m$ . The number of functor-argument structures is further reduced by the employment of the Memory Module that filters functor-argument structures based on previous evidence. However, at the same time the complexity is increased by the introduction of the rules of function composition and Generalised Weak Permutation. The following investigates the efficiency of this learner in comparison to the TLA and STL on Gibson and Wexler's three parameter grammar-space.

The English-like language of the three-parameter system studied by Gibson and Wexler has the parameter settings and associated unembedded surface-strings as shown in Figure 6.4. For this task we assume that the surface-strings of the English-like language are independent and identically distributed in the input to the learner.

SPECIFIER	COMPLEMENT	V2
0 ( <i>Left</i> )	1 ( <i>Right</i> )	0 ( <i>off</i> )

1. Subj Verb
2. Subj Verb Obj
3. Subj Verb Obj Obj
4. Subj Aux Verb
5. Subj Aux Verb Obj
6. Subj Aux Verb Obj Obj
7. Adv Subj Verb
8. Adv Subj Verb Obj
9. Adv Subj Verb Obj Obj
10. Adv Subj Aux Verb
11. Adv Subj Aux Verb Obj
12. Adv Subj Aux Verb Obj Obj

Figure 6.4: Parameter settings and surface-strings of Gibson and Wexler’s English-like language.

### Efficiency of Triggering Learning Algorithm

For the TLA to be successful it must converge to the correct parameter settings of the English-like language (see Figure 6.4). Berwick and Niyogi [4] modelled the TLA as a Markov process (see Figure 6.5). Circles represent possible grammars (a configuration of parameter settings). The target grammar lies at the centre of the structure. Arrows represent the possible transitions between grammars. Note that the TLA is constrained to only allow movement between grammars that differ by one parameter value. The probability of moving between Grammar  $G_i$  and Grammar  $G_j$  is a measure of the number of target surface-strings that are in  $G_j$  but not  $G_i$ , normalised by the total number of target surface-strings as well as the number of alternate grammars the learner can move to. For example the probability of moving from Grammar 3 to Grammar 7 is  $2/12 * 1/3 = 1/18$  since there are 2 target surface-strings allowed by Grammar 7 that are not allowed by Grammar 3 out of a possible of 12 and three grammars that differ from Grammar 3 by one parameter value.

Using this model it is possible to calculate the probability of converging to the target from each starting grammar and the expected number of steps before convergence. Consider starting from Grammar 3, after the process finishes looping it has a  $3/5$  probability of moving to Grammar 4 (from which it will never converge) and a  $2/5$  probability of moving to Grammar 7 (from which it will definitely converge), therefore there is a 40% probability of converging to the target grammar when starting at Grammar 3.

Let  $S_n$  be the expected number of steps from state  $n$  to the target state. For starting grammars 6, 7 and 8, which definitely converge, we know:

$$S_6 = 1 + \frac{5}{6}S_6 \quad (6.1)$$

$$S_7 = 1 + \frac{2}{3}S_7 + \frac{1}{18}S_8 \quad (6.2)$$



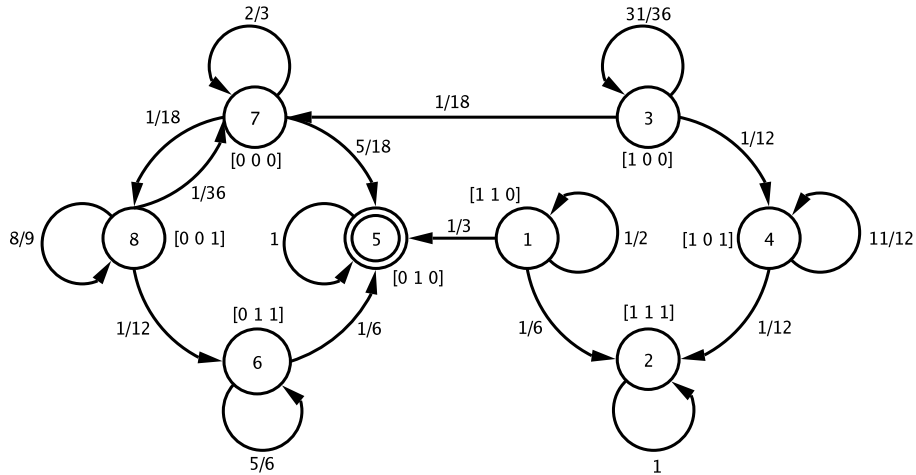


Figure 6.5: Gibson and Wexler's TLA as a Markov structure.

$$S_8 = 1 + \frac{1}{12}S_6 + \frac{1}{36}S_7 + \frac{8}{9}S_8 \quad (6.3)$$

and for the times when we do converge from grammars 3 and 1 we can expect:

$$S_1 = 1 + \frac{3}{5}S_1 \quad (6.4)$$

$$S_3 = 1 + \frac{31}{33}S_3 + \frac{2}{33}S_7 \quad (6.5)$$

Figure 6.6 shows the probability of convergence and expected number of steps to convergence for each of the starting grammars.

Initial Language	Initial Grammar	Prob. of Converging	Expected no. of Steps
VOS -V2	110	0.66	2.50
VOS +V2	111	0.00	n/a
OVS -V2	100	0.40	21.98
OVS +V2	101	0.00	n/a
SVO -V2	010	1.00	0.00
SVO +V2	011	1.00	6.00
SOV -V2	000	1.00	5.47
SOV +V2	001	1.00	14.87

Figure 6.6: Probability and expected number of steps to convergence from each starting grammar to an English-like grammar (SVO -V2) when using the TLA.

The expected number of steps to convergence ranges from infinity (for starting grammars 2 and

4) down to 2.5 for Grammar 1. If the distribution over the starting grammars is uniform then the overall probability of converging is the sum of the probabilities of converging from each state divided by the total number of states:

$$\frac{1.00 + 1.00 + 1.00 + 1.00 + 0.40 + 0.66}{8} = 0.63 \quad (6.6)$$

and the expected number of steps given that you converge is the weighted average of the number of steps from each possibly converging state:

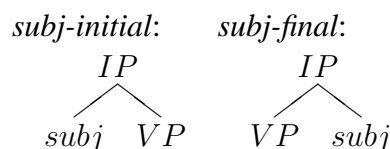
$$\frac{5.47 + 14.87 + 6 + 21.98 \times 0.4 + 2.5 \times 0.66}{1.00 + 1.00 + 1.00 + 1.00 + 0.40 + 0.66} = 7.26 \quad (6.7)$$

### Efficiency of Structural Triggers Learner

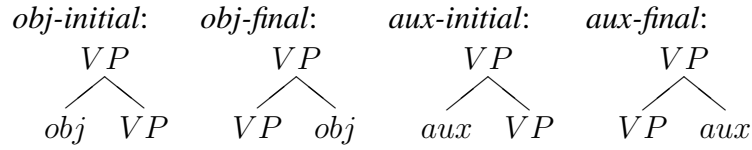
The STL does not define parameters in the same manner as the TLA. Rather, each parameter is a schematic treelet that can be used within a parse tree to derive a successful parse. The Universal Grammar consists of all the treelets that are required to parse all languages. For the STL to converge on the correct grammar it must acquire the subset of treelets needed for parsing the target language; in this case the English-like language. Treelets may only be acquired if they are found to contribute to an unambiguous parse and once learnt can not be removed from the subset. On receiving new input, the STL first attempts to parse using the current set of treelets. If this parse succeeds then the subset remains unchanged. If the parse is unsuccessful then a parse is attempted using all the treelets in the Universal Grammar. If a choice point is discovered during parsing then no treelets are acquired. Otherwise, if an unambiguous parse is found, the treelets involved in that parse are added to the subset.

A set of Universal Treelets that can describe languages in Gibson and Wexler's 3-parameter-space are outlined below. We consider the treelets required to express each parameter:

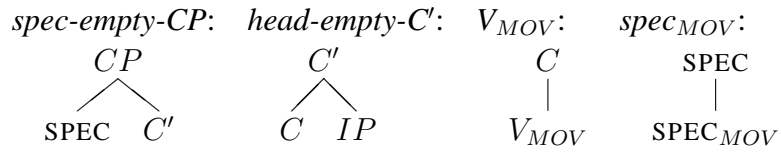
**SPECIFIER:** this parameter determines whether specifiers occur in initial or final position. In terms of Gibson and Wexler's surface-strings, this is equivalent to whether *subj* occurs before or after *verb*. Expressed as treelets in X-bar theory [47], this parameter dictates the position of a specifier in an *IP* (Inflectional Phrase). This parameter might also have determined the position of specifiers in *CPs* (Complementizer Phrases), however, in the language descriptions provided by Gibson and Wexler the specifier of the *CP* is fixed in the initial position; this is the reason why *adv* always occurs in the leftmost position of their surface-strings. The treelets required to express this parameter are as follows:



**COMPLEMENT:** this parameter determines whether complements occur in initial or final position. In terms of Gibson and Wexler's surface strings, whether *obj* and *aux* occur on the left of *verb* or to the right. This single parameter defines the behaviour of both objects and auxiliaries (i.e. you can not have *aux* occurring on the right of the verb but the *obj* on the left. The treelets required to express this parameter are:



**V2:** this parameter determines whether the finite verb in the string should be moved to the second position. In Gibson and Wexler’s surface-strings the finite verb is expressed as either *verb* or *aux*. If an *aux* is present it is this that is moved to the second position if V2 is activated, otherwise the *verb* is moved. Accompanying the verb movement either *subj*, *obj* or *adv* is moved to the first position in the string. In terms of X-bar theory the finite verb is moved to become the head of the *CP* and either *subj*, *obj* or *adv* is moved to the specifier. To describe this parameter we require the following treelets:



Here  $V_{MOV}$  and  $SPEC_{MOV}$  indicate the position of the finite verb and specifier after movement.

In addition to the treelets required to express the parameters we will need the following treelets to express the full range of surface-string in Gibson and Wexler’s languages:

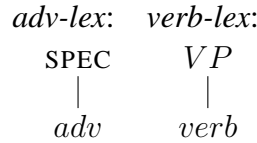


Figure 6.7 shows two possible derivations for the surface-string *subj verb obj*. The first derivation could occur in a language that is specifier initial, complement final and has no V2 movement. The second derivation could occur in a language that is specifier final, complement initial and has V2. In the second derivation *verb* is moved to  $V_{MOV}$  and *subj* to  $spec_{MOV}$  to give the surface-string.

In order to parse the English-like language the STL must acquire the subset of treelets  $\{subj-initial, obj-final, aux-initial, adv-lex, verb-lex, spec-empty-CP, head-empty-C'\}$ . The Markov model in Figure 6.8 illustrates the operation of the STL when the 12 input sentences are uniformly distributed. The states represent the subset of treelets that have been learnt; in the starting state  $s$  this subset is empty. From this model the expected number of steps for convergence can be calculated. Let  $S_x$  be the expected number of steps to converge from the state  $x$  to the final state  $d$ . We know that  $S_d = 0$ . From Figure 6.8 we see that the following equations also hold:

$$\begin{aligned} S_b &= 1 + \frac{9}{12}S_b \\ S_c &= 1 + \frac{8}{12}S_c \\ S_a &= 1 + \frac{7}{12}S_a + \frac{2}{12}S_b + \frac{1}{12}S_c \\ S_s &= 1 + \frac{6}{12}S_s + \frac{1}{12}S_a + \frac{2}{12}S_b + \frac{1}{12}S_c \end{aligned}$$

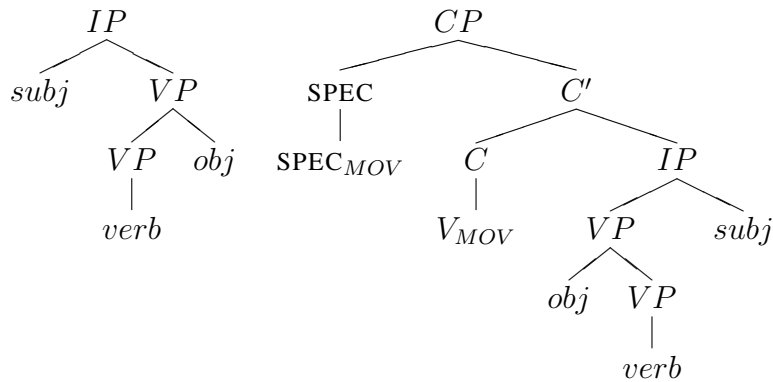


Figure 6.7: Alternative derivations for the surface-string *subj verb obj*.

These equations can be simply solved in sequence to give  $S_b = 4$ ,  $S_c = 3$ ,  $S_a = \frac{23}{5}$  and lastly,  $S_s = \frac{23}{5} = 4.6$  which is the expected number of steps from the initial state  $s$  to the final state  $d$ .

### Efficiency of Categorical Grammar Learner

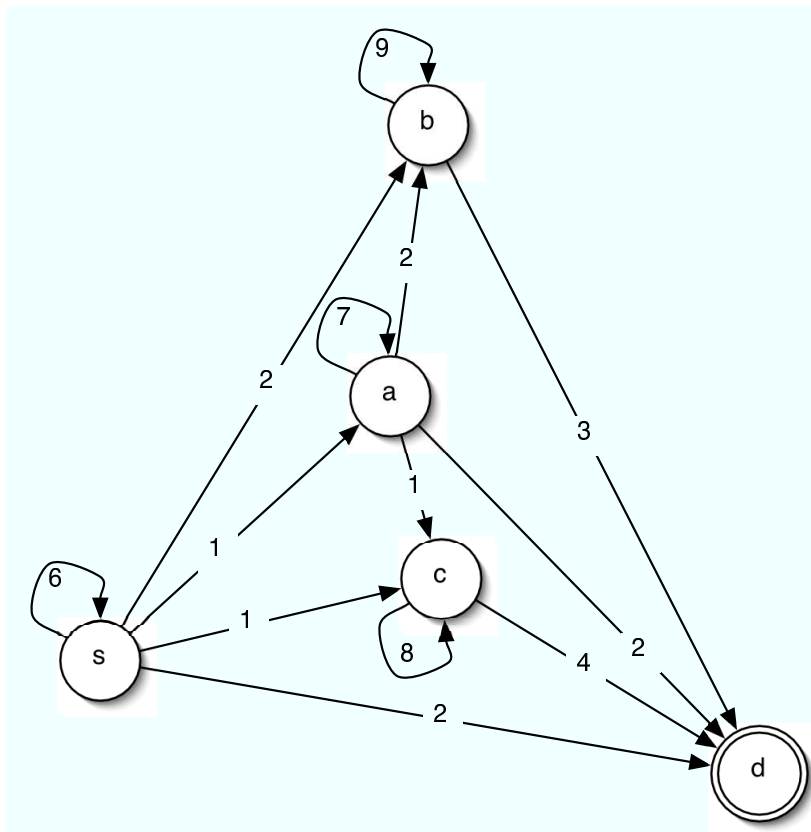
The input data to the CGL system would normally be simple-string/semantic representation pairs from which augmented strings would be derived. However, the only data available for learning from in this experiment are Gibson and Wexler's surface-strings, which consist of the word types *subj*, *obj*, *verb*, *aux* and *adj* (Figure 6.4). Since we have neither simple strings nor an associated semantic representation available to us we have to assume a mapping from semantic categories to word types in order to create the augmented strings we require. For example, given surface-string 1 (*Subj Verb*) the mappings  $Verb \mapsto \mathbf{VERB}' \mathbf{x}$  and  $Subj \mapsto \mathbf{subj}'$  are assumed. By the Principle of Categorical Type Transparency, these semantic forms provide *Verb* with a skeleton syntactic category of the form  $A|B$  (where  $A$  and  $B$  are unknown syntactic categories and  $|$  is an operator over  $\backslash$  and  $/$ ) and *Subj* with the primitive syntactic category that is called  $np$ .

The criteria for success for the CGL when acquiring Gibson and Wexler's English-like language is a lexicon containing the following (where  $s$ ,  $np$  are primitive categories which are innate to the learner):<sup>3</sup>

- Adv**  $s/s$
- Aux**  $(s \backslash np)/(s \backslash np)$
- Obj**  $np$
- Subj**  $np$
- Aux**  $(s \backslash np)/(s \backslash np)$
- Verb**  $s \backslash np$   
 $(s \backslash np)/np$   
 $((s \backslash np)/np)/np$

During the learning process the CGL will have constructed a type hierarchy in the Type Memory by setting appropriate categorial parameters to ACTIVE (see Figure 6.9). The learner will have

<sup>3</sup>Note that the lexicon would usually contain orthographic entries for the words in the language rather than word type entries.



- state s - { }
- state a - { *spec-empty-CP, head-empty-C', adv-lex, verb-lex, subj-initial* }
- state b - { *spec-empty-CP, head-empty-C', adv-lex, verb-lex, subj-initial, obj-final* }
- state c - { *spec-empty-CP, head-empty-C', adv-lex, verb-lex, subj-initial, aux-initial* }
- state d - { *spec-empty-CP, head-empty-C', adv-lex, verb-lex, subj-initial, obj-final, aux-initial* }

Note that all transition probabilities have an understood denominator of 12.

Figure 6.8: The STL as a Markov structure.

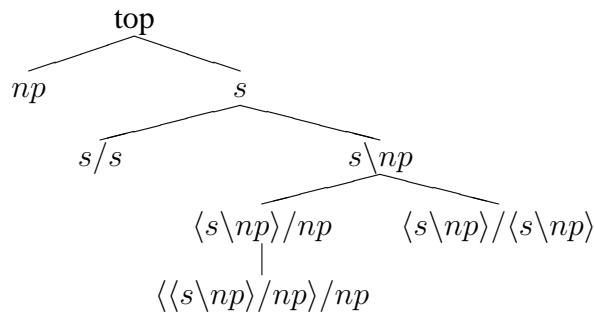


Figure 6.9: Category hierarchy required to parse Gibson and Wexler’s English-like language.

also updated the Word Order Memory, setting parameters to  $\backslash$  or  $/$ . The Memory Module is used during the learning process to constrain hypothesised syntactic categories. For this task setting the Word Order Memory becomes trivial and its role in constraining hypotheses is negligible; consequently, the rest of our argument will relate to categorial parameters only. Parameters are all originally set INACTIVE. Since the input is noiseless the switching threshold is set such that parameters may be set ACTIVE upon the evidence from one surface-string.

It is a requirement of the CGL that the parent-types of hypothesised syntax categories are ACTIVE before those categories themselves can become ACTIVE. Thus, the learner is not allowed to hypothesise the syntactic category for a transitive verb  $((s\np)/np)$  before it has learnt the category for an intransitive verb  $(s\np)$ ; additionally, it is usually not possible to derive a word’s full syntactic category (i.e. without any remaining unknowns) unless it is the only new word in the clause.

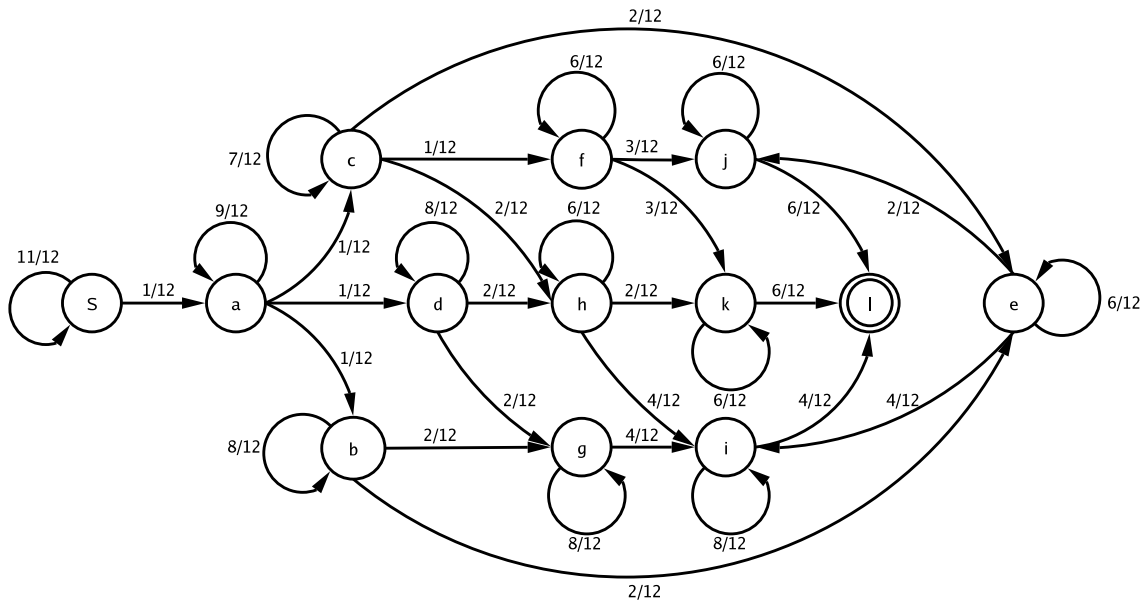
As a consequence of these issues, the order in which the surface-strings appear to the learner affects the speed of acquisition. For instance, the learner prefers to see the surface-string *Subj Verb* before *Subj Verb Obj* so that it can acquire the maximum information without wasting any strings. For the English-type language described by Gibson and Wexler the learner can optimally acquire the whole lexicon after seeing only 5 surface-strings (one string needed for each new complex syntactic category to be learnt). However, the strings appear to the learner in a random order so it is necessary to calculate the expected number of strings (or steps) before convergence.

The learner must necessarily see the string *Subj Verb* before it can learn any other information. With 12 surface-strings the probability of seeing *Subj Verb* is  $1/12$  and the expected number of strings before it is seen is 12. The learner can now learn from 3 surface-strings: *Subj Verb Obj*, *Subj Aux Verb* and *Adv Subj Verb*. Figure 6.10 shows a Markov structure of the process. From the model we can calculate the expected number of steps to converge to be 24.53.

### Comparison of Efficiency

To summarise, the TLA, STL and CGL were compared for efficiency (expected number of steps to convergence) when acquiring the English-type grammar of the three-parameter space studied by Gibson and Wexler. The TLA only converged 63% of the time but on the occasions that it did converge, the expected number of steps was given by 7.26. In a noiseless environment both the STL and CGL are guaranteed to converge; the expected number of steps for the STL and CGL were 4.6 and 24.53 respectively.

In terms of the number of steps, the STL appears to greatly out perform the CGL. However,



- state S- $\{\}$
- state a- $\{s \setminus np\}$
- state b- $\{s \setminus np, s/s\}$
- state c- $\{s \setminus np, (s \setminus np)/np\}$
- state d- $\{s \setminus np, (s \setminus np)/(s \setminus np)\}$
- state e- $\{s \setminus np, s/s, (s \setminus np)/np\}$
- state f- $\{s \setminus np, (s \setminus np)/np, ((s \setminus np)/np)/np\}$
- state g- $\{s \setminus np, (s \setminus np)/(s \setminus np), s/s\}$
- state h- $\{s \setminus np, (s \setminus np)/(s \setminus np), (s \setminus np)/np\}$
- state i- $\{s \setminus np, s/s, (s \setminus np)/np, (s \setminus np)/(s \setminus np)\}$
- state j- $\{s \setminus np, s/s, (s \setminus np)/np, ((s \setminus np)/np)/np\}$
- state k- $\{s \setminus np, (s \setminus np)/np, ((s \setminus np)/np)/np, (s \setminus np)/(s \setminus np)\}$
- state l- $\{s \setminus np, (s \setminus np)/np, ((s \setminus np)/np)/np, (s \setminus np)/(s \setminus np), s/s\}$

Figure 6.10: The CGL as a Markov structure.

analysis of the complexity of each step shows that the CGL will converge faster than the STL on average. For a sentence of length  $n$  the CGL examines at most  $2^{n-1}X_n$  parse trees; where  $X_n$  is the number of binary branching trees with  $n$  leaves (see Chapter 4). The STL can only learn from unambiguous parses. To be certain there is no ambiguity present all combinations of treelets must be checked for a given input. With a Universal Grammar containing  $T$  treelets the STL must check  $T^n X_{n+1}$  parse trees.<sup>4</sup> So, even if the STL reaches the target state within a single step, it must have checked  $T^n X_{n+1}$  parse trees. As long as  $T > 2$ , for a given grammar-space, the CGL outperforms the STL as  $n$  increases.

The CGL learns incrementally; the hypothesis-space from which it can select possible syntactic categories expands dynamically as a consequence of the hierarchical structure of parameters. The further a category is located down the hierarchy the longer it will take to be learnt; the CGL needs to see at least as many sentences as the category has arguments. Fortunately, since real language is hypothesised to never contain constituents with more than 5 arguments [82], we expect the hierarchical structure to be quite bushy; so this does not constitute a great problem.

As a further consequence of incremental learning, the speed at which the CGL acquires categories increases over time. For instance, in the starting state there is only a 1/12 probability of learning from surface-strings, whereas in state  $k$  (when all but one category has been acquired) there is a 1/2 probability. It is likely that with a more complex learning task the benefits of this incremental approach will outweigh the slow starting costs. A related work on the effects of incremental learning on STL performance [89] draws similar conclusions. Furthermore it should be noted, that in real world examples the start-up cost for the CGL (i.e. only being able to learn from one string at first) is likely to be less burdening. This is because in real language there is not a uniform distribution over sentence types; the sentence form *Subj Verb*, for example, is very common. Furthermore, the STL can learn nothing from these very common sentence forms since they have more than one possible derivation from the treelets.

As a final point, the CGL can be made robust to noise by increasing the probability threshold at which a parameter may be set to ACTIVE; neither the TLA or the STL have a mechanism for coping with noisy data.

## 6.3 Learning from Real Data

A learning system has been implemented to learn from real data. The system is composed of three modules: a semantics learning module (which creates augmented string from free-form semantic representation), a syntax learning module (the CGL) and a memory module (see Figure 6.11). For each utterance heard the learner receives an input stream of word tokens paired with possible semantic hypotheses. For example, on hearing the utterance “Dinah drinks milk” the learner may receive the pairing: ( $\{dinah, drinks, milk\}$ , **DRINK’milk’dinah**). The semantic module attempts to learn the mapping between word tokens and semantic symbols, building a lexicon containing the meaning associated with each word sense; this is achieved by using cross-situational techniques (see Section 4.4 of Chapter 4). The learning system then links the semantic module and syntactic module by using the Principle of Categorical Type Transparency; allowing basic syntactic information to be inferred from the semantic type and representing this as augmented strings (Section 4.4.2). The syntactic module attempts to learn syntactic categories from augmented strings (as described in Chapter 5). The Memory Module (also de-

<sup>4</sup>The STL needs to check all trees of  $n + 1$  leaves because of the addition of the  $V2$ -movement treelets rule.



scribed in Chapter 5 in Section 5.2.1) records the current state of the hypothesis-space. The syntactic module refers to this information to place constraints upon which syntactic categories may be learnt and also as a means of being robust to noise.

### 6.3.1 Evaluation of the Learning System

The Sachs Corpus of the CHILDES database [63] was used as input to the learning system. This corpus contains a selection of interactions between a child and her parents from the age of one year one month to five years one month. The corpus was preprocessed so that the child's sentences were excluded; only the parents' sentences are given as input to the system. Also, all phonological annotations and hesitations have been removed. The corpus is annotated such that all utterances are associated with their semantic representation(s).

A Unification-Based Generalised Categorical Grammar has been created by Villavicencio [104] to describe 2000 utterances of this corpus. The grammar uses the rules of application, composition and Generalised Weak Permutation (described in Section 4.1). Evaluation was carried out with respect to this grammar which provides both a semantic and syntactic category for every item in its lexicon.<sup>5</sup>

A problem with using Villavicencio's grammar for evaluation is that it uses lexical rules to deal with some linguistic phenomena. The CGL, however, has no equivalent of lexical rules, producing instead multiple lexical entries to describe the same phenomena. For an example, refer back to Section 6.1 which gives a discussion of dealing with *V2* movement using multiple entries. Also, Figure 6.12 illustrates how the interrogatives may be formed using the unary rule INV (as in Villavicencio's grammar) or multiple lexical entries (as in the CGL). For evaluation, a mapping had to be produced from the categories within lexical entries of Villavicencio's grammar to equivalent categories within this Learning System. For each item in Villavicencio's lexicon we enumerated all the lexical rules that could apply to that item. Possible categories for the item were then generated by applying each of the lexical rules. If by applying a lexical rule, we yielded a result that itself could be the argument of a lexical rule, we applied that rule and generated yet another category. This process was continued until either a closed set of categories was generated or (rarely) a limit on categories was reached (e.g. 20 syntactic categories per item).

A point to note in this section is that evaluation against another grammar has its problems; for instance, Villavicencio's grammar does not describe ellipsis or interjections which have been included in the input. An alternative evaluation might be to demonstrate that the acquired grammar is capable of producing the utterances produced by the child of this corpora. In fact, we might hope to demonstrate that the grammar describes a superset of the productions; with linguistic competence exceeding linguistic performance. However, such an evaluation is impractical since we do not have access to all the spoken utterances that the child has ever heard.

### 6.3.2 Noise Introduced by Indeterminacy of Meaning

If a child is to ever use language in a purposeful manner she must not only determine which utterances belong to the language but also determine what they mean. Ideally a child will hypothesise the correct meaning for every utterance it hears. However, assigning a meaning is not straight-forward; there is unlikely to be only one obvious candidate. When the correct

---

<sup>5</sup>Note that the following evaluation experiments have been previously published as [20].

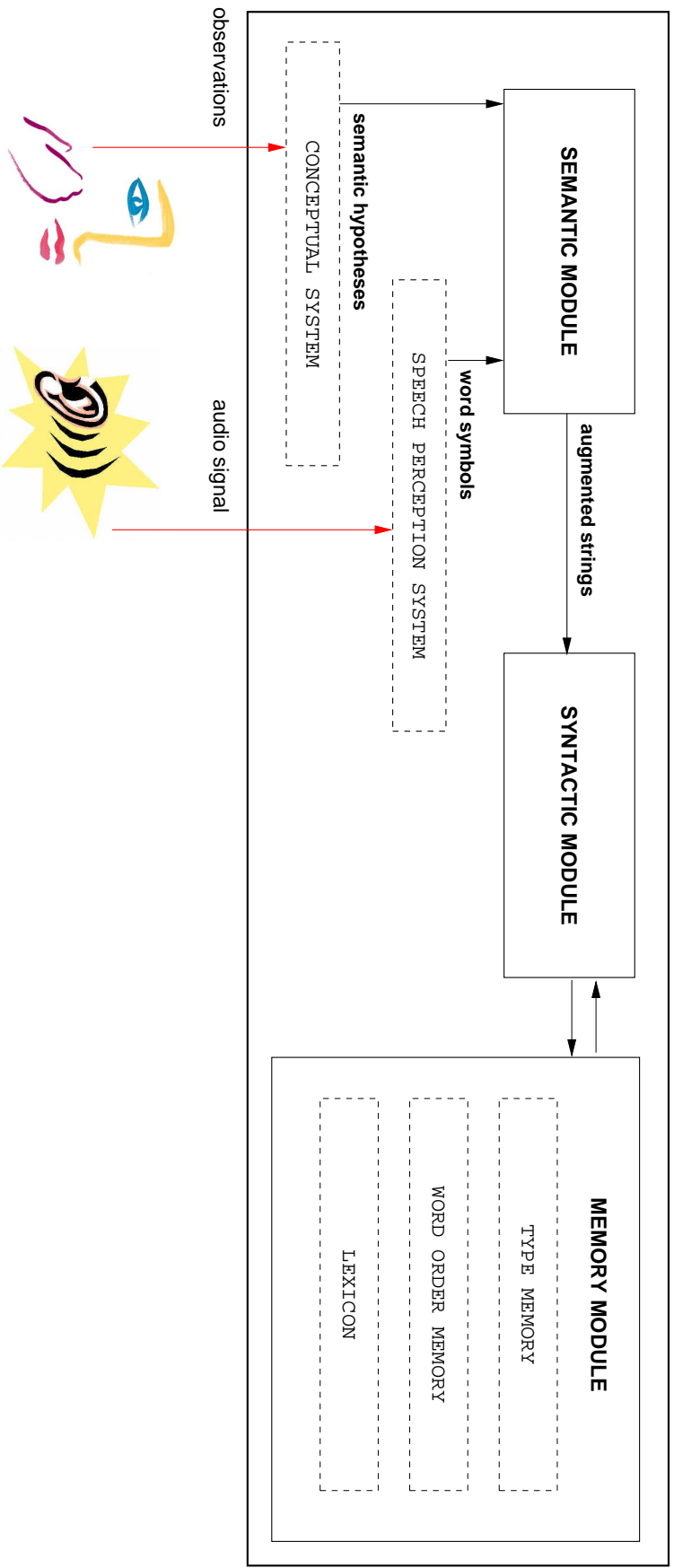


Figure 6.11: The Categorical Grammar Learner

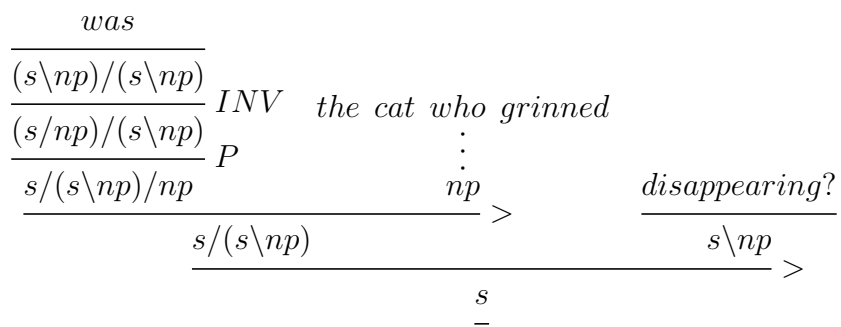
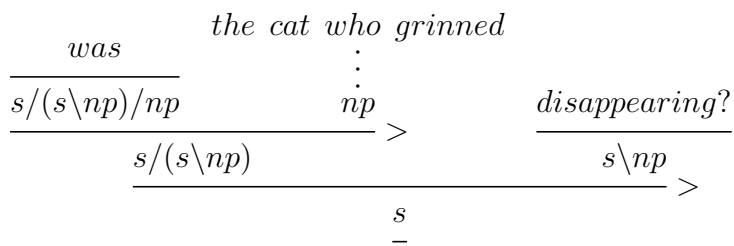
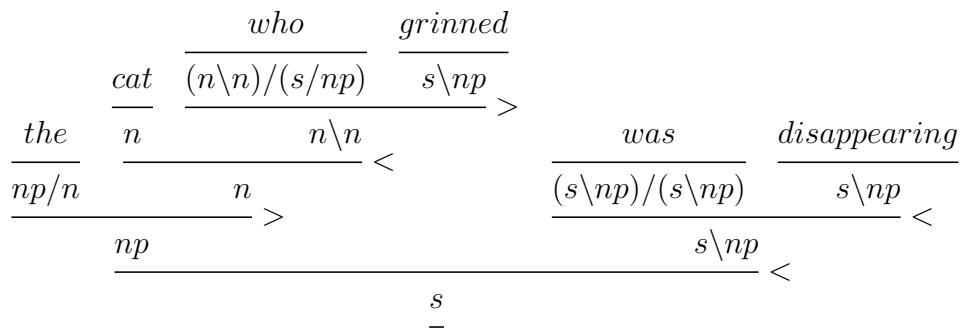


Figure 6.12: Illustration of the use of unary rules versus multiple lexical entries.

meaning is not hypothesised, error has been introduced. An error of this type introduces a false association between word and meaning within the semantics module of our learning system. To combat this problem within our learning system a statistical error handling methods were adopted (see Sections 4.4.1 and 5.2.2).

We investigated the robustness of our learning system under increasing levels of indeterminacy of meaning. This was achieved by associating utterances from the corpus with a set of possible semantic meanings (rather than just their single correct meaning). Indeterminacy could be increased by increasing the size of this associated meaning set and also by not including the actual meaning within the set.

**Experiment 1a:** The learner was run with increasing numbers of semantic hypotheses per utterance. The extra hypotheses were chosen randomly and the correct semantic expression was always present in the set. Hypotheses sets of sizes 2, 3, 5, 10 and 20 were used.

**Experiment 1b:** The learner was run with some utterances being completely mismatched with semantic hypotheses (i.e. the correct hypothesis was not present amongst the set).

### Experiment 1a.

Input utterances were associated with semantic hypotheses sets rather than just the correct meaning. The extra hypotheses were chosen randomly and the correct semantic expression was always present in the set. Hypotheses sets of sizes 2, 3, 5, 10 and 20 were used. The learner was run several times for each size of set. Recall remained fairly constant regardless of the number of hypotheses. The precision also remained very high moving only as low as 93% for one of the runs with a hypothesis set of size 20.

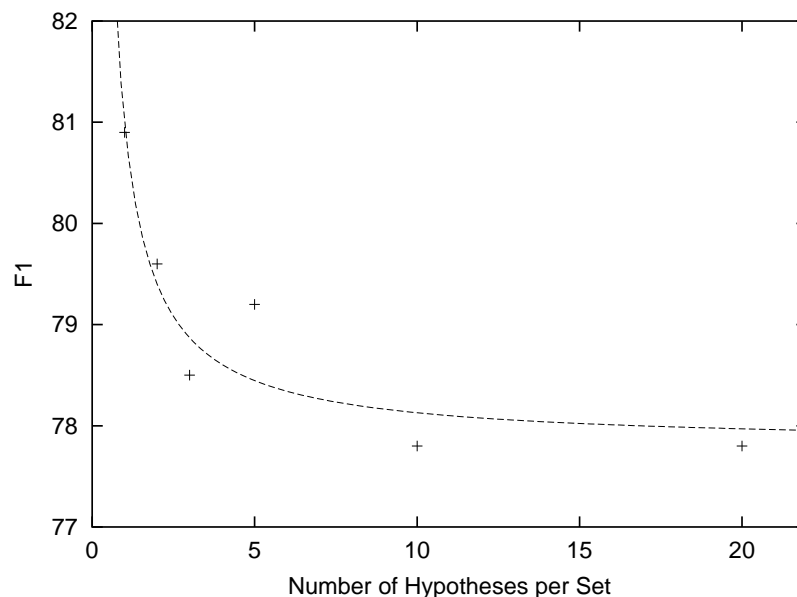


Figure 6.13: Experiment 1a: graph of size of semantic hypothesis set vs.  $F_1$

From Figure 6.13: as the number of hypotheses increases the  $F_1$  of the learner decreases, tending towards a steady state. The results can be interpreted as follows:

The confidence values associated with the word-meaning pairs of extremely common words in the corpus (such as “kitty”) become very high, very quickly. This means that many incorrect hypotheses in the hypothesis set can be ruled out (using the method of constraining hypotheses explained above). Once this starts to happen the problem rapidly reduces to that of one with a smaller starting hypothesis set. This accounts for the levelling off in  $F_1$ . The precision remains high for all hypothesis set sizes due to the statistical nature of the confidence factor; since the surplus hypotheses (those extra to the correct hypothesis) were always chosen at random, the real meaning of the word eventually emerges as the most likely.

### **Experiment 1b.**

The learner was run with some utterances being completely mismatched with semantic hypotheses (i.e. the correct hypothesis was not present amongst the set). This is analogous to the case where the child was not able to understand the meaning of the utterance from its observations. This experiment was investigated in a more qualitative fashion since the results were found to be highly dependent on the utterances that were chosen to be mismatched. If many utterances were chosen that contained infrequently occurring words then the recall would suffer a severe drop. There is a clear reason for this result. The distribution of words in the corpus is Zipfian. Most words appear very infrequently (over 250 words appear just once and more than 125 appear twice). In the original experiment (where only the correct hypothesis was paired with the meaning) 36% of words could be learnt with only one exposure. This capability is useless if a word that appears only once in the corpus is paired with an incorrect hypothesis. In such a situation the word will never be learnt. This highlights the issue that statistical learning methods are largely ineffective when data is extremely sparse.

### **6.3.3 Noise Introduce by Indeterminacy in Parameter Setting**

It is claimed that children rarely mis-set syntactic parameters [108]. However, proposed methods of parameter setting, such as the Trigger Learning Algorithm (TLA) [39] or the Structural Trigger Learner (STL) [38], allow a parameter to be updated on the basis of evidence from one utterance. This type of approach can only work if a child exclusively receives positive evidence; otherwise it is possible that an error may cause a parameter to be set incorrectly and there may never be a chance to reset it (with the consequence that the correct grammar cannot be learnt). The problem is even worse for this learning system, which learns incrementally, since an incorrect setting in the Memory Module can lead to an entire sub-branch of the defining category tree being incorrect also.

A solution is to use a statistical method that tracks relative frequencies (as described in Section 5.2.2). Such an approach sets the Memory Module to the values that are most likely given all the accumulated evidence. For this learning system, the syntactic categories within the Type Module can be considered ternary valued (ACTIVE, INACTIVE, POSSIBLE). To start with, all categories may be labelled INACTIVE, although an initial bias can be set by assigning some categories to be ACTIVE or POSSIBLE before learning commences. Evidence from input utterances will either enforce the current parameter settings or negate them. Categories become ACTIVE in the hierarchy as soon sufficient evidence has been accumulated, i.e. once their relative frequency reaches a threshold value. By employing this method, it becomes unlikely for categories to become ACTIVE as the consequence of an erroneous utterance (unlike for parameters in the deterministic methods mentioned above).

For this experiment, however, the Word Order Memory is investigated; in particular errors due to misclassification of thematic role. For illustration consider the following example utterance from the Sachs corpus: “he likes fish”. A child listening to this utterance may be unsure who is doing the liking and who is being liked. Semantically, she could consider there to be two possibilities **LIKE’fish’he’** or **LIKE’he’fish’**. In the first case we have the actor on the left of the verb and the undergoer on the right. In the second case we have the reverse. An error occurs when an English speaking child hypothesises the latter interpretation of the utterance.

In order to investigate this phenomenon, the learner was exposed to increasing amounts of misinterpreted thematic roles (from 0% up to 50% of all occurrences). This was achieved by randomly selecting the appropriate number of utterances and reversing the roles in their semantic representation. Again, the Sachs’ corpus was used as input. The order of constituents was recorded in the Word Order Memory; the actor direction in **ACTOR-DIRECTION** and the undergoer direction in **UNDERGOER-DIRECTION**. Originally both were set to |. The thresholds for setting direction parameters in the memory were set at 0.25 and 0.75: i.e. if the relative frequency of the forward direction (/) reached 0.75 then the parameter was set to /; and if the relative frequency dropped to 0.25 the parameter was set to \. The criteria for success is for the learner to set **ACTOR-DIRECTION** to \ and **UNDERGOER-DIRECTION** to / despite the noise due to misinterpreted thematic role.

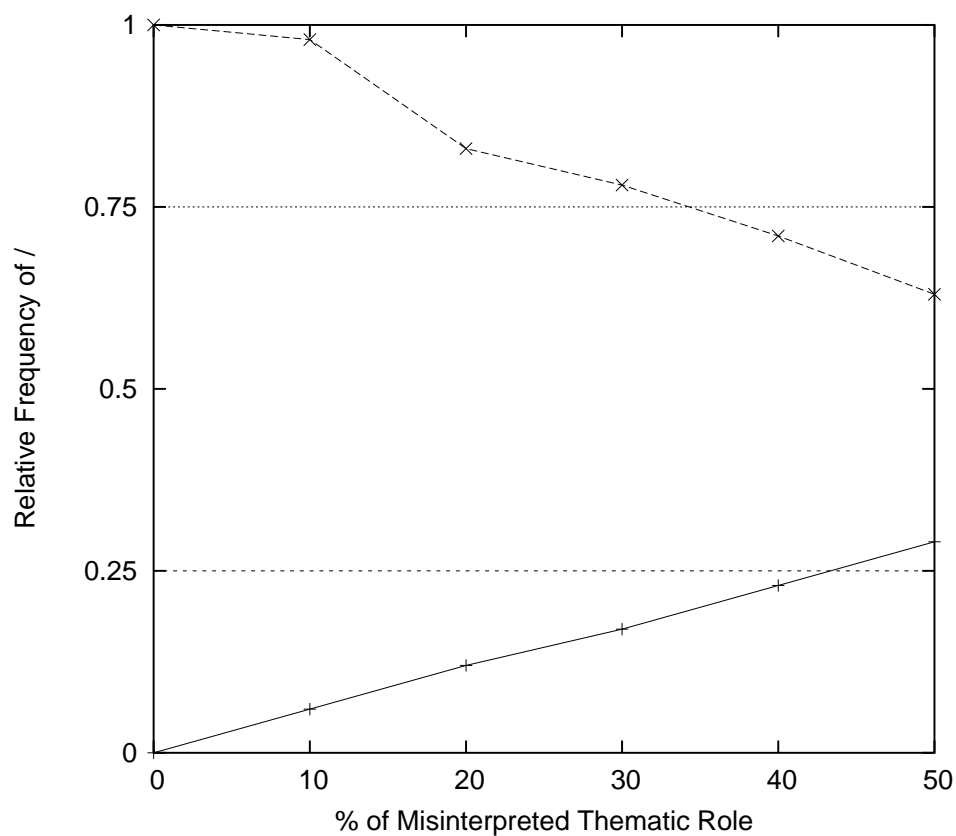
Figure 6.14 shows the relative frequency of / for **ACTOR-DIRECTION** and **UNDERGOER-DIRECTION** in the Word Order Memory at the end of learning. With misinterpretation of thematic roles at 0% both parameters easily obtain their target value while at 50% the relative frequency for neither parameter is great enough to be set. Notice that the line representing **ACTOR-DIRECTION** does not deviate far outside the target  $< 0.25$  zone. This is because of the presence of intransitive verbs; with no undergoer to confuse with the actor, every time an intransitive verb is seen the \ frequency is increased—this counteracts the effect of some of the thematic role errors.

With thematic role error at 50% we might expect the relative frequency of the / in **UNDERGOER-DIRECTION** to be 0.5. However, this is not the case. There are two reasons for this: first, the category  $s/np \setminus np$  can not be learnt (and hence the / frequency incremented) unless the category  $s/np$  has been learnt. This category is not learnt as readily as the intransitive category ( $s \setminus np$ ); it is mainly associated with imperatives or noisy incomplete utterances. The second reason is that once the **ACTOR-DIRECTION** is set, grammars which agree with this setting (i.e. with undergoer direction /) are selected in preference.

These results show that by using statistical error handling this learning system can not only learn from real data but is also robust to errors introduced by indeterminacy in parameter settings.

## 6.4 Developmental Compatibility

So far it has been demonstrated that the CGL is efficient compared to other parametric learners and that it may learn from real data. The following section will discuss the developmental compatibility of the CGL; i.e. can the model give an explanation for children’s language production? The CGL learns incrementally: interaction with the Memory Module ensures that simple syntactic categories are acquired before more complex categories. To investigate the compatibility of such a model with a real learner, the acquired subcategorization frames in CHILDES2 have been examined to see if they show evidence of incremental learning (where CHILDES2 is the child speech corpus investigated in Chapter 2). Figure 6.15 shows all the SCFs that appear at



%	ACTOR-DIRECTION	UNDERGOER-DIRECTION	OUTCOME
0	0.00 (\)	1.00 (/)	SUCCESS
10	0.06 (\)	0.98 (/)	SUCCESS
20	0.12 (\)	0.83 (/)	SUCCESS
30	0.17 (\)	0.78 (/)	SUCCESS
40	0.23 (\)	0.71 ( )	FAILURE
50	0.29 ( )	0.63 ( )	FAILURE

Figure 6.14: Graph of relative frequency of / against % of misinterpreted thematic roles; × indicates UNDERGOER, + indicates ACTOR.

least 100 times in CHILDES2. All of the SCFs directly inherit from each other. This does not prove that children learn incrementally but it does support the idea; we would run into problems if children were producing syntactic categories that could not be attached to the hierarchy.

Referring back to Brown's Stages we see that in Stage 1 the child first produces two-word utterances of the form *agent+action* or *action+object*. This is followed by production of declarative statements of the form *subject+verb+object* and then by the introduction of preposition phrases. In terms of our hierarchy this may be represented as shown in Figure 6.16. It is important to notice that all of the verb types acquired in Stage 1 can be drawn as a continuous hierarchy; there are no detached categories. Furthermore children learnt the categories further down the hierarchy after acquiring those simpler categories nearer the root. Thus, from Brown's Stage 1 we see encouraging evidence that the CGL is adequately modelling incremental learning.

In Brown's Stage 2 (Figure 6.17) children start using preliminary auxiliary categories like "wanna", possessives and simple *what*, *where* and *why* questions. Note that the two new verbal categories can directly inherit from categories in the hierarchy representing Stage 1 (Figure 6.16) and that the category for the possessive inherits directly from *np* which is another assumed primitive like *s*.

By Stage 3 the child is using modal verbs  $\langle s \setminus np \rangle / \langle s \setminus np \rangle$ , quantifiers  $\langle s \setminus np \rangle \setminus \langle s \setminus np \rangle$  and has started to ask *who* and *how* questions  $\langle np / np \rangle / \langle s \setminus np \rangle$ . Again these categories are directly inherited from the current hierarchy. Stages 4 and 5 mainly involve acquisition of inflections and an understanding of turn taking, which are not modelled by the CGL.

In general, evidence from both Brown's Stages and the CHILDES2 child-speech corpus suggest that the incremental learning method of the CGL is not at odds with real child development.

## 6.5 Summary

In the first section of this chapter we demonstrated that the CGL is more efficient than previous parametric learners (the TLA and the STL) when learning an English-like language under ideal conditions in Gibson and Wexler's 3-parameter-space. We also discussed differences and similarities of the CGL to Waldron/Villavicencio Learning System, which is a parametric system that also learns a categorial grammar and has inspired the Memory Module of the CGL. We found that the CGL requires less innate knowledge than this previous system since it learns from augmented strings (rather than strings) and uses a Memory Module to constrain hypotheses. Section 6.3 of this chapter demonstrated that the CGL can learn from real (noisy) data and is also robust to ambiguity of parameter setting. The final section has discussed the validity of the incremental learning method adopted by the CGL with relevance to a child-speech corpus and Brown's Stages; it was found that the incremental method is at least consistent with observed productions.



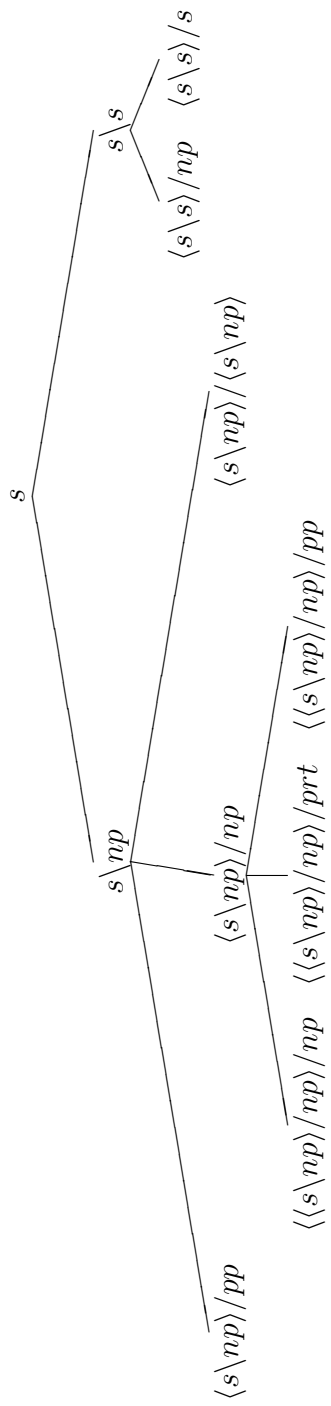


Figure 6.15: Category hierarchy of all SCFs that appears at least 100 times in CHILDES2.

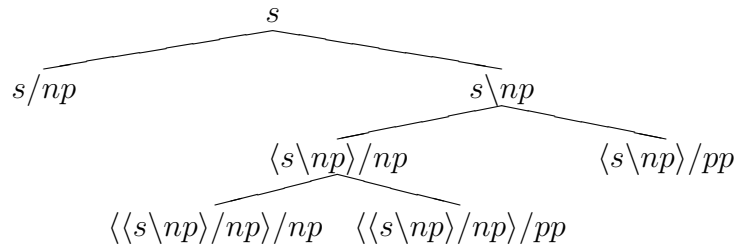


Figure 6.16: The syntactic categories involved in Brown Stage 1

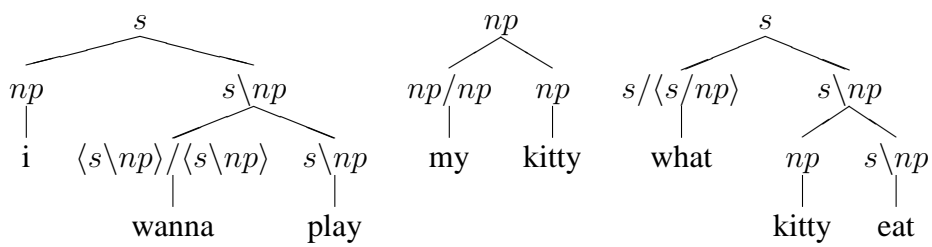


Figure 6.17: The syntactic constructions that have been mastered by Brown Stage 2.

# Chapter 7

## Conclusions

This work investigates a computational model of first language acquisition; the Categorical Grammar Learner or CGL. The CGL is neither pure nativist or pure empiricist in its ideology; rather it lies on the middle ground between the two extremes. The model assumes some innate knowledge but also relies on language examples to guide the learning process. In particular, the model assumes knowledge of the rules of a Generalised Categorical Grammar; those of function application, function composition and Generalised Weak Permutation. Other concepts assumed by the CGL are not specific to syntactic acquisition. For instance, the awareness of the primitive type that has been called *np* could stem from the ability to recognise objects; *np* being merely a label for a class of such objects. Whether these concepts are innate or developed is not argued here; although for this particular example, evidence leans towards development [71]. Either way, any linguistic universals that are present before language learning commences can be considered to be the Principles of a Chomskian Universal Grammar.

In general, a premise for using a Principles and Parameters approach to modelling is that, without constraining the hypothesis-space, learning is too complex. The CGL does not strictly adhere to the Principles and Parameters ideology in that parameters are not rigidly defined before learning commences; rather it makes use of a Memory Module that is built dynamically and is used to constrain hypothesised syntactic categories. However, in terms of efficiency, we have seen that the CGL out performs both the TLA and the STL on a simple learning task in Gibson and Wexler's three parameter domain. Furthermore, the utilisation of the Memory Model ensures that the CGL is both more efficient and requires less innate knowledge than a previous categorial grammar learner implemented by Waldron and Villavicencio.

In fact, the design of this Memory Module is key in the CGL; especially the Type Memory hierarchy. The CGL does not learn a syntactic category until its direct parent in the type hierarchy has been learnt. Thus the CGL learns incrementally, which is compatible with the child development studies of Brown and also those of this work. A consequence of this incremental learning is that the operation of the CGL relies on initially receiving short and simple sentences. Corpora studies presented here have shown that child-directed speech is syntactically simpler and less diverse than speech between adults; containing a similar distribution of SCFs to child speech. Thus, child-directed speech facilitates the operation of the CGL.

The Word Order Memory also plays a key role in constraining hypotheses; if an argument direction has been set to either / or \ then syntactic categories that are compatible over arguments of the same type are selected in preference. A consequence of this is that directionality in the language as a whole is governed by the directionality properties of the very frequent categories.

For instance, a language that is VSO will have its GENERAL-DIRECTION set to / in the Word Order Memory; consequently the CGL prefers / (arguments on the right) to \ (arguments on the left) and is therefore likely to exhibit prepositions in preference to postpositions. This agrees with work by Hawkins on Implicational Universals [44].

As discussed in Chapter 3, a learner can deal with errors if it is allowed to choose the most likely hypothesis in preference to one that incorporates every piece of evidence seen so far. This suggests that some statistical retention of the data is necessary if a learner is going to cope with errors. This is not to suggest that a child should consciously be counting events, but perhaps rather that the brain has some capacity to store this information without any effort on behalf of the learner. A possible method for doing this is to use a connectionist model; however, such an approach is infeasible since it requires an enormous amount of data and computation for learning. Models that follow a more nativist theme (the Principles and Parameters learners) potentially require much less data since the hypothesis-space is constrained. However, to eliminate the possibility of reaching a local maxima these parametric models must also have some mechanism for recognising parametric ambiguity. Hence, the Memory Module of the CGL is statistical in its nature of operation; it relies on the recurrence of a linguistic phenomenon in order to “keep it in memory”. The ability of the CGL to learn from real (noisy) data (and to learn from some parametrically ambiguous data) demonstrates the statistical utility of the Memory Module.

For input the CGL receives augmented strings. The information content of sentence objects (simple strings, augmented strings, unlabelled structures and functor-argument structures) has been discussed. The concept of an augmented string (a string augmented with some basic syntactic information) has been proposed as a sensible starting point for learning, since it is a cognitively plausible object that also benefits from carrying more information than a simple string. To form an augmented string from a simple string, syntactic information is extracted from the semantic types of the words using the *Principle of Categorical Type Transparency*. An augmented string is thus a representation of the constraints placed on the search-space of possible parses due to the semantics associated with the string. A possible extension would be to also include prosodic information in the augmented strings. Such an extension would allow for clause structure and question types to be identified in English; which could help in constraining hypothesised categories, especially for longer input strings.

Currently, the CGL can not learn anything about a word’s syntax until it knows about its semantics. However, this doesn’t imply that it is impossible to learn syntax before semantics. Consider the case where the syntactic category of every word but one is already known; the category of the new word could often be inferred by inspection of the parse tree without reference to any semantic information. This would be a simple extension to the CGL. Gleitman asserts that there are some verbs for which the semantics can not possibly be learnt without resorting to their subcategorization frame. This is what she calls *syntactic bootstrapping* ([41] and [36]). Pinker on the other hand says this information is interesting “like a puzzle” and therefore potentially useful to clever children/adults but is not essential. If Gleitman is correct and it is impossible to ascertain the meaning of some words without first resolving their syntactic category, then it will be essential to provide feedback within the learning system from the CGL back to the semantic module.

The crucial and interesting elements of the CGL are the concepts of augmented strings and a Memory Module (the Type Memory and Word Order Memory): augmented strings have been shown to be cognitively plausible sentence objects that one can sensibly learn from; and regard-

ing the Memory Module, it seems that the incremental learning enforced by the Type Memory together with the directional tendencies enforced by the Word Order Memory will be able to explain some *Implicational Universals* (i.e the idea that if a language has a property  $P$  then it also exhibits property  $Q$ ). By making types ACTIVE in the Type Memory or by setting direction preferences in the Word Order Memory, the language becomes constrained in certain ways. Properties set in the Memory Module early on in learning affect the properties that are available to a language later on; thus we have an “if  $P$  then  $Q$  effect”. This matter will be investigated as future work



# Appendix A

## SCF Classification

The following is a list of all 163 verb subcategorization frames employed by Briscoe and Carroll's SCF acquisition system (RASP). The SCFs were merged from the SCFs found in the COMLEX and ANLT syntax dictionaries and about 30 more SCFs were added by examining unclassifiable patterns of corpus examples.

Most of the following records contain 4 lines of information. The first is the COMLEX SCF name together with the frequency with which that SCF appeared in ANLT. The second line gives the frame specification using ANLT notation. The third gives a tagged example sentence from corpus data where the SCF occurs. The final line gives the SCF specification according to the grammar employed by RASP.

For entries after the 117<sup>th</sup> there are only three entries per record – these are SCFs which do not appear in COMLEX, thus the first line is in fact the frame specification in ANLT.

1. ADJP / 93  
(SUBCAT SC\_AP, SUBTYPE EQUI) / XTAG: Tnx0Va1  
his\_AT reputation\_NN1 sank\_VVD low\_JJ  
(VSUBCAT AP)
2. ADJP-PRED-RS / 15  
(SUBCAT SC\_AP, SUBTYPE RAIS) / XTAG: Tnx0Ax1  
he\_NP1 appears\_VVZ crazy\_JJ / distressed\_VVN  
(VSUBCAT AP) / (VSUBCAT VPPRT)
3. ADVP / 64  
(SUBCAT ADVP)  
he\_NP1 meant\_VVD well\_RP  
(VSUBCAT NONE, PRT +) well
4. ADVP-PRED-RS / 0 (in vppp)  
(SUBCAT ADVP, SUBTYPE RAIS)  
He\_NP1 seems\_VVZ well\_RP  
(VSUBCAT NONE, PRT +) well
5. AS-NP / 0 (in vppp with PRT 1 = end)  
(SUBCAT SC\_NP, SUBTYPE EQUI, PREP as)  
i\_NP1 worked\_VVZ as\_CSA an\_AT1 apprentice\_NN1 cook\_NN1

(VSUBCAT PP) as

6. EXTRAP-NP-S / 58  
(SUBCAT NP\_SFIN, SUBTYPE EXTRAP, AGR N2[NFORM IT])  
it\_PPH1 annoys\_VVZ them\_PPHO2 that\_CST she\_PPHS1 left\_VVD  
it (VSUBCAT NP\_SCOMP) \* \*\_VVZ/D/G
7. S-SUBJ-NP-OBJ / 58  
(SUBCAT NP\_SFIN, SUBTYPE EXTRAP, AGR S[FIN +]) / XTAG:Ts0Vnx1  
that\_CST she\_PPHS1 left\_VVD annoys\_VVZ them\_PPHO2  
\*\_VVD/Z/G (VSUBCAT NP)
8. TO-INF-SUBJ-NP-OBJ / 56  
(SUBCAT OC\_INF, SUBTYPE EQU\_EXTRAP, AGR VP[FIN -])  
to\_TO read\_VV0 pleases\_VVZ them\_PPHO2  
\*\_VV0 (VSUBCAT NP)
9. EXTRAP-TO-INF / 4  
(SUBCAT VPINF, SUBTYPE EXTRAP, AGR N2[NFORM IT])  
it\_PPH1 remains\_VVZ to\_TO find\_VV0 a\_AT1 cure\_NN1  
IT (VSUBCAT VPINF)
10. EXTRAP-FOR-TO-INF / 0 (not in vppp)  
(SUBCAT SINF, SUBTYPE EXTRAP, AGR N2[NFORM IT])  
it\_PPH1 remains\_VVZ for\_IF us\_PPHO2 to\_TO find\_VV0 a\_AT1 cure\_NN1  
IT (VSUBCAT PP\_VPINF) for (PSUBCAT NP)
11. EXTRAP-NP-TO-INF / 56  
(SUBCAT OC\_INF, SUBTYPE EQU\_EXTRAP, AGR N2[NFORM IT])  
it\_PPH1 pleases\_VVZ them\_PPHO2 to\_TO find\_VV0 a\_AT1 cure\_NN1  
IT (VSUBCAT SINF)
12. EXTRAP-TO-NP-S / 5 (4 without EXTRAP)  
(SUBCAT PP\_SFIN, SUBTYPE EXTRAP, PFORM to, AGR N2[NFORM IT])  
it\_PPH1 matters\_VVZ to\_II them\_PPHO2 that\_CST she\_PPHS1 left\_VVD  
IT (VSUBCAT PP\_SCOMP) to (PSUBCAT NP) \* \*\_VVZ/D/G
13. EXTRAP-TO-NP-TO-INF / 1  
(SUBCAT PP\_VPINF, SUBTYPE EXTRAP, PFORM to)  
it\_PPH1 occurred\_VVD to\_II them\_PPHO2 to\_TO watch\_VV0  
IT (VSUBCAT PP\_VPINF) to (PSUBCAT NP)
14. S-SUBJ-TO-NP-OBJ / 5  
(SUBCAT PP\_SFIN, SUBTYPE EXTRAP, AGR S[FIN +])  
that\_CST she\_PPHS1 left\_VVD matters\_VVZ to\_II them\_PPHO2  
\*\_VVD/G/Z (VSUBCAT PP) to (PSUBCAT NP)
15. FOR-TO-INF / 17



- (SUBCAT SINF)  
iPPHS1 prefer\_VV0 for\_IF her\_PPHO1 to\_TO do\_VV0 it\_PPH1  
(VSUBCAT PP\_VPINF) FOR (PSUBCAT NP)
16. HOW-S / 155 (combined with other wh comps)  
(SUBCAT WHS)  
he\_PPHS1 asked\_VVD how\_RGQ she\_PPHS1 did\_VDD it\_PPH1  
(VSUBCAT PP) HOW/WHY/WHERE/WHEN (PSUBCAT SFIN)
17. HOW-TO-INF / 100 (combined with other wh comps)  
(SUBCAT WHVP)  
he\_PPHS1 explained\_VVD how\_RGQ to\_TO do\_VV0 it\_PPH1  
(VSUBCAT PP) HOW/WHERE/WHEN (PSUBCAT VPINF)
18. INF-AC / ??  
ANLT gap (SUBCAT VC\_BSE)  
he\_PPHS1 helped\_VVD bake\_VV0 the\_AT cake\_NN1  
(VSUBCAT VPBSE)
19. ING-NP-OMIT / 242  
(SUBCAT SC\_ING, SUBTYPE EQUI)  
his\_AT hair\_NN1 needs\_VVZ combing\_VVG  
(VSUBCAT VPING)
20. ING-SC/BE-ING-SC / 21  
(SUBCAT SC\_ING, SUBTYPE RAIS)  
she\_PPHS1 stopped\_VVD smoking\_VVG  
(VSUBCAT VPING)
21. ING-AC / ??  
ANLT gap (SUBCAT VC\_ING)  
she\_PPHS1 discussed\_VVD writing\_VVG novels\_NN2  
(VSUBCAT VPING)
22. INTRANS / 2985  
(SUBCAT NULL)  
he\_PPHS1 went\_VVD  
(VSUBCAT NONE)
23. INTRANS-RECIP(SUBJ-PL/COORD) / ??  
(SUBCAT NULL)  
They\_PPHS2 met\_VVD  
\*\_PP/NN\*2 (VSUBCAT NONE)  
John\_NP1 and\_CC her\_AT brother\_NN1 met\_VVD  
\*\_CC (VSUBCAT NONE) \*\*\*
24. NP / 5281  
(SUBCAT NP) / XTAG:Tnx0Vnx1  
he\_PPHS1 loved\_VVD her\_PPHO1

- (VSUBCAT NP)
25. NP-ADJP / 113  
(SUBCAT OC\_AP, SUBTYPE EQUI)  
he\_PPHS1 painted\_VVD the\_AT car\_NN1 black\_JJ  
(VSUBCAT NP\_AP)
  26. NP-ADJP-PRED / 46  
(SUBCAT OC\_AP, SUBTYPE RAIS) / XTAG:Tnx0Vs1  
she\_PPHS1 considered\_VVD him\_PPHO1 foolish\_JJ  
(VSUBCAT NP\_AP)
  27. NP-ADVP / 9  
(SUBCAT NP\_ADVP)  
he\_PPHS1 put\_VVD it\_PPH1 there\_RL  
(VSUBCAT NP, PRT +) \* there
  28. NP-ADVP-PRED / 281 (with PPs)  
(SUBCAT NP\_LOC) / XTAG:Tnx0Vs1  
they\_PPHS2 mistakenly\_RA thought\_VVD him\_PPHO1 here\_RL  
(VSUBCAT NP, PRT +) here
  29. NP-AS-NP / 3  
(SUBCAT SC\_NP\_NP, SUBTYPE RAIS, PREP as)  
iPPHS1 sent\_VVD him\_PPHO1 as\_CSA a\_AT1 messenger\_NN1  
(VSUBCAT NP\_PP) (PFORM AS)
  30. NP-AS-NP-SC / 3  
(SUBCAT SC\_NP\_NP, SUBTYPE RAIS, PREP as)  
she\_PPHS1 served\_VVD the\_AT firm\_NN1 as\_CSA a\_AT1 researcher\_NN1  
(VSUBCAT NP\_PP) (PFORM AS)
  31. NP-FOR-NP / 90  
(SUBCAT NP\_PP, SUBTYPE DMOVT, PFORM for)  
she\_PPHS1 bought\_VVD a\_AT1 book\_NN1 for\_IF him\_PPHO1  
(VSUBCAT NP\_PP) (PFORM FOR)
  32. NP-INF / 11  
(SUBCAT OC\_BSE, SUBTYPE RAIS) / XTAG:Tnx0Vs1  
he\_PPHS1 made\_VVD her\_PPHO1 sing\_VV0  
(VSUBCAT SCOMP) \*\_VV0
  33. NP-INF-OC / 17  
(SUBCAT OC\_BSE, SUBTYPE EQUI)  
he\_PPHS1 helped\_VVD her\_PP\$ bake\_VV0 the\_AT cake\_NN1  
(VSUBCAT SCOMP) \*\_VV0
  34. NP-ING / 28  
(SUBCAT OC\_ING, SUBTYPE RAIS) / XTAG:Tnx0Vs1

- iPPHS1 kept\_VVD them\_PPHO2 laughing\_VVG  
(VSUBCAT SING)
35. NP-ING-OC / 45  
(SUBCAT OC\_ING, SUBTYPE EQUI)  
iPPHS1 caught\_VVD him\_PPHO1 stealing\_VVG  
(VSUBCAT SING)
36. NP-ING-SC / ??  
ANLT gap: real complement?  
he\_PPHS1 combed\_VVD the\_AT woods\_NN2 looking\_VVG for\_IF  
her\_PPHO1  
(VSUBCAT SING)
37. NP-NP / 231  
(SUBCAT NP\_NP) / XTAG:Tnx0Vnx1nx2  
she\_PPHS1 asked\_VVD him\_PPHO1 his\_AT name\_NN1  
(VSUBCAT NP\_NP)
38. NP-NP-PRED / 38  
(SUBCAT OC\_NP, SUBTYPE EQUI) / XTAG:Tnx0Vs1  
they\_PPHS2 appointed\_VVD him\_PPHO1 professor\_NN1  
(VSUBCAT NP\_NP)
39. NP-P-ING / 2  
(SUBCAT OC\_PP\_ING, PFORM from, SUBTYPE PVERB\_OR, ORDER  
POSTNP)  
iPPHS1 prevented\_VVD her\_PPHO1 from\_II leaving\_VVG  
(VSUBCAT NP\_PP) from (PSUBCAT VPING)
40. NP-P-ING-OC / 31  
(SUBCAT OC\_PP\_ING, PFORM, SUBTYPE PVERB\_OE, ORDER  
POSTNP)  
iPPHS1 accused\_VVD her\_PPHO1 of\_IO murdering\_VVG her\_AT hus-  
band\_NN1  
(VSUBCAT SING, PRT +) of  
(VSUBCAT NP\_PP) \* (PSUBCAT VPING)
41. NP-P-ING-SC / ??  
Gap in ANLT scheme, shld be: (SUBCAT SC\_PP\_ING, PRT, ORDER  
POSTNP)  
he\_PPHS1 wasted\_VVD time\_NNT1 on\_II fussing\_VVG with\_IW his\_AT  
hair\_NN1  
(VSUBCAT NP\_PP) on (PSUBCAT VPING)
42. NP-P-ING-AC / ??  
Gap in ANLT scheme (SUBCAT VC\_PP\_ING)  
he\_PPHS1 told\_VVD her\_PPHO1 about\_II climbing\_VVG the\_AT moun-  
tain\_NN1

- (VSUBCAT NP\_PP) about (PSUBCAT VPING)
43. NP-P-NP-ING / ??  
 ANLT gap (SUBCAT NP\_PP\_SING)  
 he\_PPHS1 attributed\_VVD his\_AT failure\_NN1 to\_II noone\_NP1 buying\_VVG  
 his\_AT books\_NN2  
 (VSUBCAT NP\_PP) to (PSUBCAT SING)
44. NP-P-POSSING / ??  
 ANLT gap (SUBCAT NP\_PP\_SING)  
 They\_PPHS2 asked\_VVD him\_PPHO1 about\_II his\_PPHO1 participat-  
 ing\_VVG in\_II the\_AT conference\_NN1  
 (VSUBCAT NP\_PP) about (PSUBCAT SING)
45. NP-P-WH-S / 0 (not in vppp, and below)  
 (SUBCAT NP\_WHS, PREP)  
 they\_PPHS2 made\_VVD a\_AT1 great\_JJ fuss\_NN1 about\_II whether\_CSW  
 they\_PPHS2 should\_VM participate\_VV0  
 (VSUBCAT NP\_PP) about (PSUBCAT PP) whether (PSUBCAT SFIN)
46. NP-P-WHAT-S / 0  
 (SUBCAT NP\_WHS, PREP)  
 they\_PPHS2 made\_VVD a\_AT1 great\_JJ fuss\_NN1 about\_II what\_DDQ  
 they\_PPHS2 should\_VM do\_VV0  
 (VSUBCAT NP\_WHPP) about (PSUBCAT WHS)
47. NP-P-WHAT-TO-INF / 0  
 (SUBCAT NP\_WHVP, PREP)  
 they\_PPHS2 made\_VVD a\_AT1 great\_JJ fuss\_NN1 about\_II what\_DDQ to\_TO  
 do\_VV0  
 (VSUBCAT NP\_WHPP) about (PSUBCAT NP)
48. NP-P-WH-TO-INF / 0  
 (SUBCAT NP\_WHS, PREP)  
 they\_PPHS2 made\_VVD a\_AT1 great\_JJ fuss\_NN1 about\_II whether\_CSW  
 to\_TO go\_VV0  
 (VSUBCAT NP\_PP) about (PSUBCAT PP) whether (PSUBCAT VPINF)
49. NP-PP / 2010  
 (SUBCAT NP\_PP, PFORM, SUBTYPE NONE/PVERB?) /  
 XTAG:Tnx0Vnx1pnx2  
 she\_PPHS1 added\_VVD the\_AT flowers\_NN2 to\_II the\_AT bouquet\_NN1  
 (VSUBCAT NP\_PP) to
50. NP-PP-PRED / 2010/50??  
 (SUBCAT NP\_PP, PFORM of, SUBTYPE NONE, PRD +)  
 iPPHS1 considered\_VVD that\_AT problem\_NN1 of\_IO little\_JJ concern\_NN1  
 (VSUBCAT NP\_PPOF)

51. NP-PRED-RS / 12  
 (SUBCAT SC\_NP, SUBTYPE RAIS)  
 he\_PPHS1 seemed\_VVD a\_AT1 fool\_NN  
 (VSUBCAT NP)
52. NP-S / 33  
 (SUBCAT NP\_SFIN, SUBTYPE NONE) / XTAG:Tnx0Vnx1s2  
 he\_PPHS1 told\_VVD the\_AT audience\_NN1 that\_CST he\_PPHS1 was\_VBZ  
 leaving\_VVG  
 (VSUBCAT NP\_SCOMP) \*\_\*\_VVZ/D/G
53. NP-TO-INF-OC / 189  
 (SUBCAT OC\_INF, SUBTYPE EQUI)  
 iPPHS1 advised\_VVD Mary\_NP1 to\_TO go\_VV0  
 (VSUBCAT SINF)
54. NP-TO-INF-SC / 1  
 (SUBCAT SC\_NP\_INF, SUBTYPE EQUI)  
 John\_NP1 promised\_VVD Mary\_NP1 to\_TO resign\_VV0  
 (VSUBCAT SINF)
55. NP-TO-INF-VC / ??  
 ANLT gap  
 they\_PPHS2 badgered\_VVD him\_PPHO1 to\_TO go\_VV0  
 (VSUBCAT SINF)
56. NP-TO-NP / 105  
 (SUBCAT NP\_PP, PFORM to, SUBTYPE DMOVT) / XTAG:Tnx0Vnx1Pnx2  
 he\_PPHS1 gave\_VVD a\_AT1 big\_JJ kiss\_NN1 to\_II his\_AT mother\_NN1  
 (VSUBCAT NP\_PP) to
57. NP-TOBE / 88  
 (SUBCAT OC\_INF, SUBTYPE RAIS)  
 iPPHS1 found\_VVD him\_PPHO1 to\_TO be\_VB0 a\_AT1 good\_JJ doctor\_NN1  
 (VSUBCAT SINF) BE
58. NP-VEN-NP-OMIT / 3  
 (SUBCAT OC\_PASS, SUBTYPE EQUI/RAISING)  
 he\_PPHS1 wanted\_VVD the\_AT children\_NN2 found\_VVN  
 (VSUBCAT SCOMP) \*\_VVN
59. NP-WH-S / 12  
 (SUBCAT NP\_WHS)  
 they\_PPHS2 asked\_VVD him\_PPHO1 whether\_CSW he\_PPHS1 was\_VBZ go-  
 ing\_VVG  
 (VSUBCAT NP\_PP) WHETHER/IF (PSUBCAT SFIN)
60. NP-WHAT-S / 12  
 (SUBCAT NP\_WHS)

- they\_PPHS2 asked\_VVD him\_PPHO1 what\_DDQ he\_PPHS1 was\_VBZ do-  
ing\_VVG  
(VSUBCAT NP\_SCOMP) S(WH +)
61. NP-WH-TO-INF / 12  
(SUBCAT NP\_WHVP)  
he\_PPHS1 asked\_VVD him\_PPHO1 whether\_CSW to\_TO clean\_VV0 the\_AT  
house\_NN1  
(VSUBCAT NP\_PP) WHETHER (PSUBCAT VPINF)
62. NP-WHAT-TO-INF / 12  
(SUBCAT NP\_WHVP)  
he\_PPHS1 asked\_VVD him\_PPHO1 what\_DDQ to\_TO do\_VV0  
(VSUBCAT NP\_NP) \* WHAT/WHO/WHICH
63. P-ING-SC / 100  
(SUBCAT SC\_ING, SUBTYPE EQUI, PREP)  
they\_PPHS2 failed\_VVD in\_II attempting\_VVG the\_AT climb\_NN1  
(VSUBCAT PP) in (PSUBCAT VPING)
64. P-ING-AC / ??  
ANLT gap (SUBCAT VC\_ING, PRT)  
they\_PPHS2 disapproved\_VVD of\_IO attempting\_VVG the\_AT climb\_NN1  
(VSUBCAT VPING, PRT +) of  
they\_PPHS2 argued\_VVD about\_II attempting\_VVG the\_AT climb\_NN1  
(VSUBCAT PP) about (PSUBCAT VPING)
65. P-NP-ING / 8  
(SUBCAT OC\_PP\_ING, PFORM @p, SUBTYPE PVERB\_OR/OE, ORDER  
PRENP)  
they\_PPHS2 worried\_VVD about\_II him\_PPHO1 drinking\_VVG  
(VSUBCAT PP) about (PSUBCAT SING)
66. P-NP-TO-INF(-SC) / 6  
(SUBCAT SC\_PP\_INF, PFORM @p, SUBTYPE EQUI)  
he\_PPHS1 conspired\_VVD with\_IW them\_PPHO2 to\_TO do\_VV0 it\_PPH1  
(VSUBCAT PP\_VPINF) with (PSUBCAT NP)
67. P-NP-TO-INF-OC / 29  
(SUBCAT OC\_PP\_INF, PFORM @p, SUBTYPE PVERB\_OE/OR/EQUI)  
he\_PPHS1 beckoned\_VVD to\_II him\_PPHO1 to\_TO come\_VV0  
(VSUBCAT PP\_VPINF) to (PSUBCAT NP)
68. P-NP-TO-INF-VC / ??  
ANLT gap  
she\_PPHS1 appealed\_VVD to\_II him\_PPHO1 to\_TO go\_VV0  
she\_PPHS1 appealed\_VVD to\_II him\_PPHO1 to\_TO be\_VB0 freed\_JJ  
(VSUBCAT PP\_VPINF) to (PSUBCAT NP)

69. P-POSSING / 10  
 (SUBCAT OC\_PP\_ING, PFORM @p, SUBTYPE PVERB\_OR, ORDER PRENP)  
 they\_PP\$2 argued\_VVD about\_II his\_PP\$ coming\_VVG  
 (VSUBCAT PP) about (PSUBCAT SING)
70. P-WH-S / 37  
 (SUBCAT WHS, PRT/PREP @p)  
 he\_PP\$1 thought\_VVD about\_II whether\_CS\$ he\_PP\$1 wanted\_VVD  
 to\_TO go\_VV0  
 (VSUBCAT PP) about (PSUBCAT PP) WHETHER/IF (PSUBCAT SFIN)
71. P-WHAT-S / 37  
 (SUBCAT WHS, PRT/PREP @p)  
 he\_PP\$1 thought\_VVD about\_II what\_DDQ he\_PP\$1 wanted\_VVD  
 (VSUBCAT WHPP) about (PSUBCAT WHS)
72. P-WH-TO-INF / 27  
 (SUBCAT WHVP, PREP @p)  
 he\_PP\$1 thought\_VVD about\_II whether\_CS\$ to\_TO go\_VV0  
 (VSUBCAT PP) about (PSUBCAT PP) whether (PSUBCAT VPINF)
73. P-WHAT-TO-INF / 27  
 (SUBCAT WHVP, PREP @p)  
 he\_PP\$1 thought\_VVD about\_II what\_DDQ to\_TO do\_VV0  
 (VSUBCAT WHPP) about
74. PART / 3219  
 (SUBCAT NULL, PRT) / XTAG:Tnx0Vpl  
 she\_PP\$1 gave\_VVD up\_RL  
 (VSUBCAT NONE, PRT +) up  
 she\_PP\$1 gave\_VVD up\_II  
 (VSUBCAT PP) up (PSUBCAT NONE)
75. PART-ING-SC / 7  
 (SUBCAT SC\_ING, SUBTYPE EQUI, PRT/PREP)  
 he\_PP\$1 ruled\_VVD out\_II paying\_VVG her\_AT debts\_NN2  
 (VSUBCAT PP) out (PSUBCAT VPING)  
 he\_PP\$1 ruled\_VVD out\_RP paying\_VVG her\_AT debts\_NN2  
 (VSUBCAT VPING, PRT +) out
76. PART-NP/NP-PART / 2134  
 (SUBCAT NP, PRT) (ORDER FREE) / XTAG:Tnx0Vplnx1  
 iPP\$1 looked\_VVD up\_RL the\_AT entry\_NN1  
 (VSUBCAT NP, PRT +) up \*  
 iPP\$1 looked\_VVD the\_AT entry\_NN1 up\_RL  
 (VSUBCAT NP, PRT +) \* up
77. PART-NP-PP / 312

- (SUBCAT NP\_PP, PFORM, PRT, SUBTYPE NONE/PVERB?) (ORDER FREE)  
iPPHS1 separated\_VVD out\_II the\_AT three\_JJ boys\_NN2 from\_II the\_AT crowd\_NN1  
(VSUBCAT PP\_PP) out (PSUBCAT NP) from (PSUBCAT NP)  
iPPHS1 separated\_VVD out\_RL the\_AT three\_JJ boys\_NN2 from\_II the\_AT crowd\_NN1  
(VSUBCAT NP\_PP, PRT +) out from (PSUBCAT NP)
78. PART-PP / 234  
(SUBCAT PP, PFORM, PRT, SUBTYPE PVERB)  
she\_PPHS1 looked\_VVD in\_II on\_II her\_AT friend\_NN1  
(VSUBCAT PP) in (PSUBCAT PP) on (PSUBCAT NP)  
she\_PPHS1 looked\_VVD in\_RL on\_II her\_AT friend\_NN1  
(VSUBCAT PP, PRT +) in on (PSUBCAT NP)
79. PART-WH-S / 20  
(SUBCAT WHS, PRT, SUBTYPE NONE)  
they\_PPHS2 figured\_VVD out\_II whether\_CSW she\_PPHS1 had\_VHD n't\_XX done\_VVD her\_AT job\_NN1  
(VSUBCAT PP) out (PSUBCAT PP) WHETHER/IF (PSUBCAT SFIN)  
they\_PPHS2 figured\_VVD out\_RP whether\_CSW she\_PPHS1 had\_VHD n't\_XX done\_VVD her\_AT job\_NN1  
(VSUBCAT PP, PRT +) out WHETHER/IF (PSUBCAT SFIN)
80. PART-WHAT-S / 20  
(SUBCAT WHS, PRT, SUBTYPE NONE)  
they\_PPHS2 figured\_VVD out\_II what\_DDQ she\_PPHS1 had\_VHD n't\_XX done\_VVD  
(VSUBCAT WHPP) out (PSUBCAT WHS)  
they\_PPHS2 figured\_VVD out\_RP what\_DDQ she\_PPHS1 had\_VHD n't\_XX done\_VVD  
(VSUBCAT SCOMP, PRT +) out S(WH +)
81. PART-WH-TO-INF / 22  
(SUBCAT WHVP, PRT, SUBTYPE NONE)  
they\_PPHS2 figured\_VVD out\_II whether\_CSW to\_TO go\_VV0  
(VSUBCAT PP) out (PSUBCAT PP) whether (PSUBCAT VPINF)  
they\_PPHS2 figured\_VVD out\_RP whether\_CSW to\_TO go\_VV0  
(VSUBCAT PP, PRT +) out whether (PSUBCAT VPINF)
82. PART-WHAT-TO-INF / 22  
(SUBCAT WHVP, PRT, SUBTYPE NONE)  
they\_PPHS2 figured\_VVD out\_II what\_DDQ to\_TO do\_VV0  
(VSUBCAT WHPP) out (PSUBCAT NP)  
they\_PPHS2 figured\_VVD out\_RP what\_DDQ to\_TO do\_VV0  
(VSUBCAT NP, PRT +) WHAT/WHICH/WHO
83. PART-THAT-S / 48



- (SUBCAT SFIN, PRT, SUBTYPE NONE)  
they\_PPHS2 figured\_VVD out\_II that\_CST she\_PPHS1 had\_VHD n't\_XX  
done\_VVD her\_AT job\_NN1  
(VSUBCAT PP\_SCOMP) out (PSUBCAT NONE) \*\_VVG/Z/D  
they\_PPHS2 figured\_VVD out\_RP that\_CST she\_PPHS1 had\_VHD n't\_XX  
done\_VVD her\_AT job\_NN1  
(VSUBCAT SCOMP, PRT +) out \*\_VVG/Z/D
84. POSSING / 27  
(SUBCAT OC\_ING, SUBTYPE RAIS)  
he\_PPHS1 dismissed\_VVD their\_PP\$ writing\_VVG novels\_NN2  
(VSUBCAT SING)
85. POSSING-PP / ??  
ANLT gap (SUBCAT OC\_ING\_PP)  
she\_PPHS1 attributed\_VVD his\_PP\$ drinking\_VVG too\_RA much\_RA to\_II  
his\_AT anxiety\_NN1  
(VSUBCAT SING\_PP) to (PSUBCAT NP)
86. ING-PP / ??  
ANLT gap  
they\_PPHS2 limited\_VVD smoking\_VVG a\_AT pipe\_NN1 to\_II the\_AT  
lounge\_NN1  
(VSUBCAT VPING\_PP) to (PSUBCAT NP)
87. PP / 2465 (366 LOC)  
(SUBCAT PP/LOC, PFORM, SUBTYPE NONE/PVERB) /  
XTAG:Tnx0Vpnx1  
they\_PPHS2 apologized\_VVD to\_II him\_PPHO1  
(VSUBCAT PP) to (PSUBCAT NP)
88. PP-FOR-TO-INF / 1  
(SUBCAT PP\_SINF, PFORM)  
they\_PPHS2 contracted\_VVD with\_IW him\_PPHO1 for\_IF the\_AT man\_NN1  
to\_TO go\_VV0  
(VSUBCAT PP\_PP) with (PSUBCAT NP) for (PSUBCAT SINF)
89. PP-HOW-S / 7  
(SUBCAT PP\_WHS, PFORM)  
he\_PPHS1 explained\_VVD to\_II her\_PPHO1 how\_RGQ she\_PPHS1 did\_VDD  
it\_PPH1  
(VSUBCAT PP\_PP) to (PSUBCAT NP) HOW/WHY/WHERE/WHEN  
(PSUBCAT SFIN)
90. PP-HOW-TO-INF / 3  
(SUBCAT PP\_WHVP, PFORM)  
he\_PPHS1 explained\_VVD to\_II them\_PPHO2 how\_RGQ to\_TO do\_VV0  
it\_PPH1

- (VSUBCAT PP\_PP) to (PSUBCAT NP) HOW/WHERE/WHEN (PSUBCAT VPINF)
91. PP-P-WH-S / ??  
 Gap in ANLT scheme: (SUBCAT PP\_WHS, PFORM, PRT)  
 iPPHS1 agreed\_VVD with\_IW him\_PPHO1 about\_II whether\_CSW he\_PPHS1  
 should\_VM kill\_VV0 the\_AT peasants\_NN2  
 (VSUBCAT PP\_PP) with (PSUBCAT NP) about (PSUBCAT PP) WHETHER  
 (PSUBCAT SFIN)
92. PP-P-WHAT-S / ??  
 Gap in ANLT scheme  
 iPPHS1 agreed\_VVD with\_IW him\_PPHO1 about\_II what\_DDQ he\_PPHS1  
 should\_VM do\_VV0  
 (VSUBCAT PP\_WHPP) with (PSUBCAT NP) about (PSUBCAT WHS)
93. PP-P-WHAT-TO-INF / ??  
 Gap in ANLT scheme  
 iPPHS1 agreed\_VVD with\_IW him\_PPHO1 about\_II what\_DDQ to\_TO  
 do\_VV0  
 (VSUBCAT PP\_WHPP) with (PSUBCAT NP) about (PSUBCAT NP)
94. PP-P-WH-TO-INF / ??  
 Gap in ANLT scheme  
 iPPHS1 agreed\_VVD with\_IW him\_PPHO1 about\_II whether\_CSW to\_TO  
 go\_VV0  
 (VSUBCAT PP\_PP) with (PSUBCAT NP) about (PSUBCAT PP) whether  
 (PSUBCAT VPINF)
95. PP-PP / 64 (22 PVERB)  
 (SUBCAT PP\_PP)  
 they\_PPHS2 flew\_VVD from\_II London\_NP1 to\_II Rome\_NP1  
 (VSUBCAT PP\_PP) from (PSUBCAT NP) to (PSUBCAT NP)
96. PP-PRED-RS / 0 (not in vppp)  
 (SUBCAT PP, SUBTYPE RAIS)  
 the\_AT matter\_NN1 seems\_VVZ in\_II dispute\_NN1  
 (VSUBCAT PP) in (PSUBCAT NP)
97. PP-THAT-S / 22  
 (SUBCAT PP\_SF, SUBTYPE NONE, PFORM)  
 they\_PPHS2 admitted\_VVD to\_II the\_AT authorities\_NN2 that\_CST  
 they\_PPHS2 had\_VHD entered\_VVD illegally\_RA  
 (VSUBCAT PP\_SCOMP) to (PSUBCAT NP) \*\_VVD/Z/G
98. PP-THAT-S-SUBJUNCT / 2  
 (SUBCAT PP\_SBSE, PFORM, S[BSE, that])  
 they\_PPHS2 suggested\_VVD to\_II him\_PPHO1 that\_CST he\_PPHS1 go\_VV0  
 (VSUBCAT PP\_SCOMP) to (PSUBCAT NP) \*\_VV0

99. PP-TO-INF-RS / 1  
 (SUBCAT SC\_PP\_INF, SUBTYPE RAIS, PFORM, VP[to])  
 he\_PPHS1 appeared\_VVD to\_II her\_PPHO1 to\_TO be\_VB0 ill\_JJ  
 (VSUBCAT PP\_VPINF) to (PSUBCAT NP) BE
100. PP-WH-S / 7  
 (SUBCAT PP\_WHS, PFORM)  
 they\_PPHS2 asked\_VVD about\_II everybody\_NP1 whether\_CSW they\_PPHS2  
 had\_VHD enrolled\_VVN  
 (VSUBCAT PP\_PP) about (PSUBCAT NP) WHETHER/IF (PSUBCAT SFIN)
101. PP-WHAT-S / 7  
 (SUBCAT PP\_WHS, PFORM)  
 they\_PPHS2 asked\_VVD about\_II everybody\_NP1 what\_DDQ they\_PPHS2  
 had\_VHD done\_VVN  
 (VSUBCAT PP\_WHS) about (PSUBCAT NP)
102. PP-WH-TO-INF / 3  
 (SUBCAT PP\_WHVP)  
 they\_PPHS2 deduced\_VVD from\_II kim\_NP1 whether\_CSW to\_TO go\_VV0  
 (VSUBCAT PP\_PP) from (PSUBCAT NP) whether (PSUBCAT VPINF)
103. PP-WHAT-TO-INF / 3  
 (SUBCAT PP\_WHVP)  
 they\_PPHS2 deduced\_VVD from\_II kim\_NP1 what\_DDQ to\_TO do\_VV0  
 (VSUBCAT PP\_WHVP) from (PSUBCAT NP) WHAT/WHO/WHICH
104. S / 296  
 (SUBCAT SFIN, SUBTYPE NONE) / XTAG:Tnx0Vs1  
 they\_PPHS2 thought\_VVD that\_CST he\_PPHS1 was\_VBZ always\_RA late\_JJ  
 (VSUBCAT SCOMP) \*\_VVD/Z/G
105. S-SUBJ-S-OBJ / 9  
 (SUBCAT SFIN, SUBTYPE EXTRAP, AGR S[FIN -])  
 for\_IF him\_PPHO1 to\_TO report\_VV0 the\_AT theft\_NN1 indicates\_VVD  
 that\_CST he\_PPHS1 was\_VBZ n't\_XX guilty\_JJ  
 \*\_VV0 (VSUBCAT SCOMP) \*\_VVD/Z/G
106. S-SUBJUNCT / 27  
 (SUBCAT SBSE)  
 She\_PPHS1 demanded\_VVD that\_CST he\_PPHS1 leave\_VV0 immedi-  
 ately\_RA  
 (VSUBCAT SCOMP) \*\_VV0
107. SEEM-S / 9  
 (SUBCAT SFIN, SUBTYPE EXTRAP, AGR N2[NFORM IT])  
 it\_PPH1 seems\_VVZ that\_CST they\_PPHS2 left\_VVD  
 IT (VSUBCAT SCOMP) \*\_VVD/Z/G

108. SEEM-TO-NP-S / 1  
 (SUBCAT PP\_SFIN, SUBTYPE EXTRAP, PFORM, AGR N2[NFORM IT])  
 it\_PPH1 seems\_VVZ to\_II her\_PPHO1 that\_CST they\_PPHS2 were\_VBDR  
 wrong\_JJ  
 IT (VSUBCAT PP\_SCOMP) to (PSUBCAT NP) \*\_VVD/Z/G
109. THAT-S / 296 (with 104)  
 (SUBCAT SFIN, SUBTYPE NONE) / XTAG:Tnx0Vs1  
 he\_PPHS1 complained\_VVD that\_CST they\_PPHS2 were\_VBDR com-  
 ing\_VVG  
 (VSUBCAT SCOMP) \*\_VVD/Z/G
110. TO-INF-AC / ??  
 ANLT gap (SUBCAT VC\_INF)  
 He\_PPHS1 helped\_VVD to\_TO save\_VV0 the\_AT child\_NN1  
 (VSUBCAT VPINF)
111. TO-INF-RS / 27  
 (SUBCAT SC\_INF, SUBTYPE RAIS)  
 he\_PPHS1 seemed\_VVD to\_TO come\_VV0  
 (VSUBCAT VPINF) be
112. TO-INF-SC / 179  
 (SUBCAT SC\_INF, SUBTYPE EQUI)  
 iPPHS1 wanted\_VVD to\_TO come\_VV0  
 (VSUBCAT VPINF)
113. WH-S / 133  
 (SUBCAT WHS) / XTAG:Tnx0Vs1  
 he\_PPHS1 asked\_VVD whether\_CSW he\_PPHS1 should\_VM come\_VV0  
 (VSUBCAT PP) WHETHER/IF (PSUBCAT SFIN)
114. WHAT-S / 133  
 (SUBCAT WHS) / XTAG:Tnx0Vs1  
 he\_PPHS1 asked\_VVD what\_DDQ he\_PPHS1 should\_VM do\_VV0  
 (VSUBCAT SCOMP) S(WH +)
115. WH-TO-INF / 78  
 (SUBCAT WHVP) / XTAG:Tnx0Vs1  
 he\_PPHS1 asked\_VVD whether\_CSW to\_TO clean\_VV0 the\_AT house\_NN1  
 (VSUBCAT PP) whether (PSUBCAT VPINF)
116. WHAT-TO-INF / 78  
 (SUBCAT WHVP) / XTAG:Tnx0Vs1  
 he\_PPHS1 asked\_VVD what\_DDQ to\_TO do\_VV0  
 (VSUBCAT NP) WHAT/WHO/WHICH
117. NP-NP-up / 45

- (SUBCAT NP\_NP, PRT)  
i\_PPHS1 opened\_VVD him\_PPHO1 up\_RP a\_AT new\_JJ bank\_NN1 ac-  
count\_NN1  
(VSUBCAT NP\_NP, PRT +) up
118. XTAG:Light-verbs (various classes) / ??  
he\_PPHS1 made\_VVD comments\_NN2 on\_II the\_AT paper\_NN1  
(VSUBCAT NP\_PP) (make comments) on (PSUBCAT NP)
119. (SUBCAT PP/LOC / PFORM, PRT, SUBTYPE NONE) / 881 (LOC 45)  
he\_PPHS1 breaks\_VVZ away\_RP from\_II the\_AT abbey\_NN1  
(VSUBCAT PP, PRT +) away from (PSUBCAT NP)
120. (SUBCAT NP\_PP / PFORM, PRT, SUBTYPE DMOVT) / 25  
he\_PPHS1 brought\_VVD a\_AT book\_NN1 back\_RP for\_IF me\_PPHO1  
(VSUBCAT NP\_PP, PRT +) back for (PSUBCAT NP)
121. (SUBCAT PP\_PP / PFORM, PRT) / 3  
he\_PPHS1 came\_VVD down\_RP on\_II him\_PPHO1 for\_IF his\_AT bad\_JJ be-  
haviour\_NN1  
(VSUBCAT PP\_PP, PRT +) down on (PSUBCAT NP) for (PSUBCAT NP)
122. (SUBCAT NP\_PP\_PP, PFORM) / 16  
he\_PPHS1 turned\_VVD it\_PPHO1 from\_II a\_AT disaster\_NN1 into\_II a\_AT  
victory\_NN1  
(VSUBCAT NP\_PP\_PP) from (PSUBCAT NP) into (PSUBCAT NP)
123. (SUBCAT MP) / 29  
it\_PPHS1 cost\_VVD ten\_MC pounds\_NNU2  
(VSUBCAT NP) \_NNU/(NTYPE MEAS)  
v\_np\_non\_trans\_le (but cf v\_expl\_it\_subj\_np\_np\_cp\_inf\_le)
124. (SUBCAT NP\_MP) / 6  
it\_PPHS1 cost\_VVD him\_PPHO1 ten\_MC pounds\_NNU2  
(VSUBCAT NP\_NP) \_NNU/(NTYPE MEAS)
125. (SUBCAT NP\_MP-back) / 1  
it\_PPHS1 set\_VVD him\_PPHO1 back\_RP ten\_MC pounds\_NNU2  
(VSUBCAT NP\_NP, PRT +) back \_NNU/(NTYPE MEAS)
126. (SUBCAT ADL) / 13  
he\_PPHS1 came\_VVD off\_RP badly\_RP  
(VSUBCAT NONE, PRT +) off (...PRT +) badly
127. (SUBCAT ADV\_PP / PFORM) / 2  
things\_NN2 augur\_VV0 well\_RP for\_IF him\_PPHO1  
(VSUBCAT PP, PRT +) well for (PSUBCAT NP)
128. (SUBCAT SFIN, AGR N2[NFORM IT], PRT) / 3

- it\_PPHS1 turns\_VVZ out\_RP that\_CST he\_PPHS1 did\_VVD it\_PPHO1  
IT (VSUBCAT SCOMP, PRT +) out \*\_VVD/Z/G
129. (SUBCAT SFIN, AGR S[FIN +], SUBTYPE EXTRAP) / 9  
that\_CST he\_PPHS1 came\_VVD matters\_VVZ  
\*\_VVD/G/Z (VSUBCAT NONE)
130. (SUBCAT NP\_SFIN, SUBTYPE NONE, PRT) / 4  
he\_PPHS1 had\_VVD her\_PPHO1 on\_RP that\_CST he\_PPHO1 attended\_VVD  
(VSUBCAT NP\_SCOMP, PRT +) on \*\_VVD/Z/G
131. (SUBCAT PP\_SFIN, SUBTYPE NONE, PRT) / 4  
she\_PPHS1 gets\_VVZ through\_RP to\_II him\_PPHO1 that\_CST he\_PPHS1  
came\_VVD  
(VSUBCAT PP\_SCOMP, PRT +) through to (PSUBCAT NP) \*\_VVD/Z/G
132. (SUBCAT NP\_NP\_SFIN) / 4  
he\_PPHS1 bet\_VVD her\_PPHO1 ten\_MC pounds\_NNU2 that\_CST he\_PPHS1  
came\_VVD  
(VSUBCAT NP\_NP\_SCOMP) \_NNU\*/(NTYPE MEAS) \*\_VVD/Z/G
133. (SUBCAT NP\_SBSE) / 1  
he\_PPHS1 petitioned\_VVD them\_PPHO2 that\_CST he\_PPHS1 be\_VB0  
freed\_VVN  
(VSUBCAT NP\_SCOMP) \*\*\_VB0
134. (SUBCAT IT\_WHS, SUBTYPE IF, AGR N2[NFORM IT]) / 1  
i\_PPHS1 would\_VM appreciate\_VV0 it\_PPHO1 if\_CF he\_PPHS1 came\_VVD  
(VSUBCAT NP\_PP) if (PSUBCAT SFIN)
135. (SUBCAT PP\_WHS, PFORM, AGR N2[NFORM IT]) / 1  
it\_PPHS1 dawned\_VVD on\_II him\_PPHO1 what\_DDQ he\_PPHS1 should\_VM  
do\_VV0  
IT (VSUBCAT PP\_WHS) on (PSUBCAT NP)
136. (SUBCAT SC\_NP, PRT, SUBTYPE RAIS/EQUI, PRD +) / 2  
he\_PPHS1 turned\_VVD out\_RP a\_AT fool\_NN1  
(VSUBCAT NP, PRT +) out
137. (SUBCAT SC\_AP, PRT, SUBTYPE EQUI/RAIS) / 22 (RAIS 3)  
he\_PPHS1 started\_VVD out\_RP poor\_JJ  
(VSUBCAT AP, PRT +) out  
he\_PPHS1 started\_VVD out\_II poor\_JJ  
(VSUBCAT PP\_AP) out (PSUBCAT NONE)
138. (SUBCAT SC\_INF, PRT, SUBTYPE RAIS) / 6  
he\_PPHS1 turned\_VVD out\_RP to\_TO be\_VB0 a\_AT crook\_NN1  
(VSUBCAT VPINF, PRT +) out BE  
he\_PPHS1 turned\_VVD out\_II to\_TO be\_VB0 a\_AT crook\_NN1

- (VSUBCAT PP\_VPINF) out (PSUBCAT NONE) BE
139. (SUBCAT SC\_INF, PRT, SUBTYPE EQUI) / 12  
 he\_PPHS1 set\_VVD out\_RP to\_TO win\_VV0  
 (VSUBCAT VPINF, PRT +) out  
 he\_PPHS1 set\_VVD out\_II to\_TO win\_VV0  
 (VSUBCAT PP\_VPINF) out (PSUBCAT NONE)
140. (SUBCAT SC\_ING, PREP, PRT, SUBTYPE EQUI) / 32  
 he\_PPHS1 got\_VVD around\_RP to\_II leaving\_VVG  
 (VSUBCAT PP, PRT +) around to (PSUBCAT VPING)
141. (SUBCAT SC\_PASS, SUBTYPE RAIS) / 4  
 he\_PPHS1 got\_VVD given\_VVN a\_AT book\_NN1  
 (VSUBCAT VPPRT)
142. (SUBCAT SC\_BSE, SUBTYPE EQUI) / 3  
 he\_PPHS1 dared\_VVD dance\_VV0  
 (VSUBCAT VPBSE)
143. (SUBCAT SC\_NP\_AP, SUBTYPE RAIS, PREP as) / 3  
 he\_PPHS1 strikes\_VVZ me\_PPHO1 as\_CSA foolish\_JJ  
 (VSUBCAT NP\_PP) AS (PSUBCAT AP)
144. (SUBCAT OC\_NP, SUBTYPE RAIS) / 35  
 he\_PPHS1 considers\_VVZ Fido\_NP1 a\_AT fool\_NN1  
 (VSUBCAT NP\_NP)
145. (SUBCAT OC\_AP, SUBTYPE RAIS, PRT) / 3  
 he\_PPHS1 makes\_VVD him\_PPHO1 out\_RP crazy\_JJ  
 (VSUBCAT NP\_AP, PRT +) out
146. (SUBCAT OC\_AP, SUBTYPE EQUI, PRT) / 4  
 he\_PPHS1 sands\_VVZ it\_PPHO1 down\_RP smooth\_JJ  
 (VSUBCAT NP\_AP, PRT +) down
147. (SUBCAT OC\_AP, SUBTYPE EQUI, PREP as) / 5  
 he\_PPHS1 condemned\_VVD him\_PPHO1 as\_CSA stupid\_JJ  
 (VSUBCAT NP\_PP) AS (PSUBCAT AP)
148. (SUBCAT OC\_AP, SUBTYPE EQUI, PREP as, PRT) / 6  
 he\_PPHS1 put\_VVD him\_PPHO1 down\_RP as\_CSA stupid\_JJ  
 (VSUBCAT NP\_PP, PRT +) down AS (PSUBCAT AP)
149. (SUBCAT OC\_INF, SUBTYPE RAIS, PRT) / 3  
 he\_PPHS1 made\_VVD him\_PPHO1 out\_RP to\_TO be\_VV0 crazy\_JJ  
 (VSUBCAT SINF, PRT +) out BE
150. (SUBCAT OC\_INF, SUBTYPE EQUI, PRT) / 19

- he\_PPHS1 spurred\_VVD him\_PPHO1 on\_RP to\_TO try\_VV0  
(VSUBCAT SINF, PRT +) on
151. (SUBCAT OC\_PP\_INF, SUBTYPE PVERB\_OE, PFORM, PRT) / 6  
he\_PPHS1 kept\_VVD on\_RP at\_II him\_PPHO1 to\_TO join\_VV0  
(VSUBCAT PP\_VPINF, PRT +) on at (PSUBCAT NP)
152. (SUBCAT OC\_PP\_ING, SUBTYPE PVERB\_OE, PFORM, PRT) / 4  
he\_PPHS1 talked\_VVD him\_PPHO1 around\_RP into\_II leaving\_VVG  
(VSUBCAT NP\_PP, PRT +) around into (PSUBCAT VPING)
153. (SUBCAT OC\_PP\_BSE, PFORM, SUBTYPE PVERB\_OE) / 1  
he\_PPHS1 looked\_VVD at\_II him\_PPHO1 leave\_VV0  
(VSUBCAT PP\_SCOMP) at (PSUBCAT NONE) \*\_VV0
154. (SUBCAT VPINF, SUBTYPE EXTRAP, AGR VP[FIN-]) / 4  
to\_TO see\_VV0 them\_PPHO2 hurts\_VVZ  
\_VV0 (VSUBCAT NONE)
155. (SUBCAT NP\_ADL) / 39  
he\_PPHS1 stood\_VVD it\_PPHO1 alone\_RL  
(VSUBCAT NP, PRT +) \*\*\_RL/A/P
156. NP-HOW-S / ?  
he\_PPHS1 asked\_VVD him\_PPHO1 how\_RGQ he\_PPHS1 came\_VVD  
(VSUBCAT NP\_PP) HOW/WHY/WHERE/WHEN (PSUBCAT SFIN)
157. NP-FOR-TO-INF / ?  
he\_PPHS1 gave\_VVD money\_NN2 for\_IF him\_PPHO1 to\_TO go\_VV0  
(VSUBCAT NP\_PP FOR (PSUBCAT SINF))
158. IT-PASS-SFIN / ?  
it\_PPHS1 is\_VBZ believed\_VVN that\_CST he\_PPHS1 came\_VVD  
IT PASS (VSUBCAT SCOMP)
159. AS-IF-SFIN / ?  
he\_PPHS1 seems\_VVZ as\_CS if\_CS he\_PPHS1 is\_VBZ clever\_JJ  
(VSUBCAT PP) AS (PSUBCAT PP) IF (PSUBCAT SFIN)
160. ADL)  
it\_PPHS1 carves\_VVZ easily\_RP  
(VSUBCAT NONE) \*\_RP/A
161. SC\_NP SUBTYPE EQUI)  
he\_PPHS1 felt\_VVD a\_AT fool\_NN1  
(VSUBCAT NP)
162. AS-VPPRT  
he\_PPHS1 accepted\_VVD him\_PPHO1 as\_II/CSA associated\_VVN



(VSUBCAT NP\_PP) AS (PSUBCAT VPPRT)

163. AS-VPING

he\_PPHS1 accepted\_VVD him\_PPHO1 as\_II/CSA being\_VBG normal\_JJ  
(VSUBCAT NP\_PP) AS (PSUBCAT VPING)



## Appendix B

# Verb Distributions in Adult Speech vs. Child Directed Speech

The following is a table of the top 100 most frequent verbs found in the BNC and the CHILDES1 corpora.

Rank	BNC	n	CHILDES1	n
1	<i>get</i>	5000+	<i>go</i>	5000+
2	<i>go</i>	5000+	<i>be</i>	5000+
3	<i>say</i>	5000+	<i>do</i>	5000+
4	<i>be</i>	5000+	<i>see</i>	4200
5	<i>know</i>	5000+	<i>put</i>	4037
6	<i>do</i>	5000+	<i>get</i>	4018
7	<i>think</i>	4074	<i>want</i>	3411
8	<i>see</i>	2852	<i>can</i>	3409
9	<i>like</i>	2827	<i>let</i>	2771
10	<i>can</i>	2710	<i>look</i>	2585
11	<i>come</i>	2602	<i>think</i>	2280
12	<i>want</i>	2148	<i>like</i>	2038
13	<i>mean</i>	2078	<i>know</i>	1768
14	<i>look</i>	1930	<i>say</i>	1755
15	<i>put</i>	1776	<i>come</i>	1693
16	<i>take</i>	1443	<i>make</i>	1692
17	<i>tell</i>	1122	<i>okay</i>	1593
18	<i>make</i>	1092	<i>take</i>	1356
19	<i>use</i>	1016	<i>eat</i>	1172
20	<i>will</i>	1007	<i>give</i>	990
21	<i>give</i>	920	<i>play</i>	944
22	<i>buy</i>	590	<i>tell</i>	860
23	<i>leave</i>	548	<i>find</i>	661
24	<i>keep</i>	545	<i>happen</i>	581
25	<i>pay</i>	543	<i>sit</i>	580
26	<i>let</i>	536	<i>read</i>	571
27	<i>remember</i>	517	<i>remember</i>	563
28	<i>work</i>	495	<i>try</i>	556
29	<i>suppose</i>	489	<i>fall</i>	546
30	<i>play</i>	477	<i>will</i>	537

31	<i>talk</i>	475	<i>need</i>	531
32	<i>ask</i>	469	<i>hold</i>	527
33	<i>find</i>	464	<i>turn</i>	492
34	<i>start</i>	445	<i>call</i>	439
35	<i>need</i>	443	<i>talk</i>	426
36	<i>call</i>	431	<i>thank</i>	408
37	<i>try</i>	430	<i>show</i>	404
38	<i>eat</i>	394	<i>wait</i>	395
39	<i>hear</i>	370	<i>bring</i>	389
40	<i>stop</i>	345	<i>mean</i>	379
41	<i>sit</i>	342	<i>sleep</i>	369
42	<i>turn</i>	301	<i>build</i>	367
43	<i>feel</i>	299	<i>wear</i>	363
44	<i>wait</i>	297	<i>watch</i>	308
45	<i>bring</i>	286	<i>help</i>	289
46	<i>run</i>	274	<i>fit</i>	288
47	<i>live</i>	271	<i>use</i>	286
48	<i>walk</i>	263	<i>drink</i>	284
49	<i>watch</i>	260	<i>throw</i>	268
50	<i>seem</i>	254	<i>pull</i>	266
51	<i>pick</i>	248	<i>fix</i>	260
52	<i>love</i>	247	<i>ride</i>	255
53	<i>happen</i>	246	<i>leave</i>	254
54	<i>mind</i>	244	<i>break</i>	253
55	<i>send</i>	241	<i>pick</i>	248
56	<i>move</i>	238	<i>keep</i>	244
57	<i>write</i>	236	<i>open</i>	243
58	<i>finish</i>	232	<i>stay</i>	243
59	<i>show</i>	226	<i>draw</i>	242
60	<i>ring</i>	226	<i>hurt</i>	235
61	<i>wonder</i>	224	<i>stand</i>	231
62	<i>forget</i>	224	<i>stick</i>	225
63	<i>sell</i>	209	<i>push</i>	221
64	<i>bother</i>	200	<i>hear</i>	212
65	<i>okay</i>	198	<i>feed</i>	195
66	<i>speak</i>	191	<i>finish</i>	182
67	<i>stick</i>	188	<i>move</i>	177
68	<i>cut</i>	187	<i>pretend</i>	175
69	<i>stand</i>	182	<i>work</i>	170
70	<i>change</i>	182	<i>close</i>	170
71	<i>read</i>	182	<i>buy</i>	169
72	<i>stay</i>	178	<i>catch</i>	168
73	<i>lose</i>	174	<i>run</i>	166
74	<i>thank</i>	172	<i>hum</i>	159
75	<i>listen</i>	160	<i>cry</i>	159
76	<i>win</i>	157	<i>hit</i>	158
77	<i>help</i>	149	<i>ask</i>	158
78	<i>drive</i>	148	<i>walk</i>	152
79	<i>open</i>	146	<i>wash</i>	151

80	<i>throw</i>	145	<i>hang</i>	148
81	<i>draw</i>	142	<i>sing</i>	143
82	<i>reckon</i>	140	<i>bite</i>	143
83	<i>break</i>	140	<i>bet</i>	140
84	<i>fuck</i>	139	<i>jump</i>	139
85	<i>round</i>	138	<i>feel</i>	137
86	<i>hope</i>	136	<i>blow</i>	134
87	<i>bet</i>	135	<i>listen</i>	132
88	<i>wear</i>	132	<i>fly</i>	130
89	<i>believe</i>	132	<i>guess</i>	127
90	<i>pull</i>	121	<i>cut</i>	126
91	<i>drop</i>	120	<i>live</i>	124
92	<i>hate</i>	119	<i>start</i>	121
93	<i>fall</i>	118	<i>stop</i>	121
94	<i>matter</i>	116	<i>knock</i>	121
95	<i>expect</i>	115	<i>drive</i>	120
96	<i>meet</i>	114	<i>brush</i>	119
97	<i>sort</i>	114	<i>roll</i>	114
98	<i>spend</i>	113	<i>cook</i>	112
99	<i>hold</i>	112	<i>touch</i>	112



# Appendix C

## Full SCF Data in Adult Speech vs. Child Directed Speech

For each of the following 104 verbs, the following table gives pairs of numbers for the BNC and the CHILDES1 corpora. The pairs of numbers represent the fact that a particular verb occurred with a particular SCF a given number of times. For example, the first pair, 24 84, listed in the table represents the fact that the verb “ask” occurred with SCF number 24 a total of 84 times in the BNC.

	<b>ask</b>
BNC:	24 84, 22 27, 37 10, 53 5, 26 4, 104 4, 106 4, 52 3, 59 2, 132 2, 133 2, 156 2, 47 1, 49 1, 62 1, 74 1, 76 1, 113 1, 147 1, 154 1
CHILDES1:	24 86, 22 28, 37 19, 87 6, 106 5, 104 4, 52 2, 59 2, 7 1, 8 1, 23 1, 49 1, 53 1, 62 1, 112 1, 129 1
	<b>bet</b>
BNC:	104 44, 22 28, 24 25, 106 24, 26 9, 37 5, 25 1, 35 1, 52 1
CHILDES1:	106 66, 104 43, 22 17, 24 8, 133 2, 1 1, 52 1
	<b>break</b>
BNC:	24 49, 22 33, 76 6, 37 3, 23 2, 49 2, 154 2, 1 1, 52 1, 78 1, 87 1, 104 1, 106 1, 113 1, 133 1
CHILDES1:	24 72, 22 38, 106 5, 37 3, 142 3, 8 2, 76 2, 123 2, 1 1, 50 1, 52 1, 87 1, 104 1, 133 1
	<b>bring</b>
BNC:	24 108, 76 53, 22 26, 37 26, 106 11, 49 6, 2 1, 25 1, 53 1, 62 1, 74 1, 87 1, 104 1, 129 1, 142 1
CHILDES1:	24 112, 76 90, 37 29, 22 13, 106 8, 1 3, 104 2, 112 2, 83 1, 116 1, 123 1
	<b>build</b>
BNC:	24 31, 22 17, 37 7, 49 3, 87 3, 76 2, 23 1, 26 1, 52 1, 74 1, 129 1, 154 1
CHILDES1:	24 64, 22 17, 87 3, 106 3, 37 2, 25 1, 49 1, 104 1
	<b>buy</b>
BNC:	24 63, 22 31, 37 28, 49 9, 106 9, 104 5, 52 3, 87 3, 7 2, 123 2, 147 2, 25 1, 35 1, 50 1, 53 1, 74 1, 75 1, 116 1, 117 1, 133 1, 142 1, 156 1
CHILDES1:	24 52, 22 33, 37 28, 87 19, 49 16, 106 4, 104 2, 7 1, 23 1, 50 1, 52 1, 62 1, 77 1, 89 1, 116 1, 122 1, 133 1
	<b>call</b>
BNC:	24 171, 22 63, 37 50, 106 9, 104 8, 1 5, 87 5, 76 3, 7 2, 52 2, 74 2, 154 2, 2 1, 15 1, 23 1, 25 1, 26 1, 113 1, 117 1, 124 1, 132 1, 142 1, 158 1

CHILDES1: 24 146, 37 109, 22 89, 106 6, 104 3, 7 2, 62 2, 76 2, 2 1, 25 1, 35 1, 52 1, 116 1, 158 1  
**can**  
 BNC: 22 43, 24 13, 106 11, 37 2, 19 1, 26 1, 52 1, 87 1, 112 1, 133 1  
 CHILDES1: 24 23, 22 20, 106 17, 104 5, 37 4, 52 4, 16 1, 117 1, 133 1  
**carry**  
 BNC: 24 25, 22 10, 37 8, 76 3, 106 2, 7 1, 8 1, 154 1  
 CHILDES1: 24 40, 22 27, 76 2, 37 1, 129 1  
**catch**  
 BNC: 24 41, 22 14, 49 7, 37 5, 104 4, 35 3, 52 2, 53 2, 76 2, 87 2, 1 1, 23 1, 25 1, 156 1  
 CHILDES1: 24 48, 22 17, 37 7, 104 7, 123 4, 8 3, 106 3, 1 1, 49 1, 52 1, 87 1, 129 1, 134 1  
**change**  
 BNC: 24 32, 22 27, 37 2, 104 2, 7 1, 75 1, 107 1, 133 1, 142 1  
 CHILDES1: 24 57, 37 4, 22 3, 76 3, 106 3, 53 1, 154 1  
**check**  
 BNC: 24 15, 22 11, 37 4, 104 3, 106 3, 133 3, 87 2, 26 1, 76 1, 83 1, 112 1, 156 1  
 CHILDES1: 24 32, 76 5, 22 3, 50 1, 57 1, 133 1  
**clean**  
 BNC: 22 22, 24 16, 76 5, 106 5, 37 4, 49 2, 104 2, 26 1, 52 1, 87 1, 107 1, 132 1, 133 1, 142 1  
 CHILDES1: 24 18, 22 8, 76 7, 37 4, 49 2, 52 2, 133 2, 25 1, 87 1  
**close**  
 BNC: 24 19, 22 8, 37 2, 49 2, 52 2, 106 2, 26 1, 76 1, 87 1, 104 1, 156 1  
 CHILDES1: 24 26, 22 16, 106 8, 104 1  
**come**  
 BNC: 22 535, 87 66, 24 62, 106 25, 112 18, 104 17, 74 13, 1 10, 23 9, 123 9, 142 9, 37 8, 129 6, 16 4, 19 4, 52 4, 133 4, 49 2, 78 2, 117 2, 137 2, 7 1, 8 1, 14 1, 25 1, 26 1, 50 1, 76 1, 77 1, 97 1, 113 1  
 CHILDES1: 22 590, 106 110, 24 60, 87 54, 142 22, 112 21, 74 17, 1 12, 19 8, 23 8, 52 8, 37 7, 123 7, 133 7, 2 4, 76 4, 12 2, 83 2, 26 1, 49 1, 97 1, 116 1, 132 1  
**cook**  
 BNC: 22 17, 24 17, 37 6, 104 6, 87 5, 49 4, 50 2, 1 1, 52 1, 78 1, 83 1  
 CHILDES1: 24 20, 22 13, 49 7, 87 4, 142 3, 104 2, 106 2, 133 2, 7 1, 37 1, 129 1  
**cover**  
 BNC: 24 19, 22 13, 76 7, 87 6, 49 5, 23 1, 37 1, 106 1  
 CHILDES1: 24 33, 76 9, 22 7, 37 2, 7 1, 49 1, 106 1  
**cut**  
 BNC: 24 30, 22 14, 76 14, 49 6, 37 5, 106 5, 25 4, 23 3, 117 3, 52 2, 104 2, 1 1, 26 1, 53 1, 74 1, 107 1, 116 1, 132 1, 133 1, 146 1  
 CHILDES1: 24 50, 22 17, 76 13, 49 8, 37 5, 87 4, 106 4, 7 2, 120 2, 133 2, 142 2, 154 2, 19 1, 47 1, 52 1, 59 1, 117 1, 158 1  
**do**  
 BNC: 22 149, 24 93, 106 47, 37 17, 87 16, 142 14, 49 11, 104 10, 129 10, 23 4, 116 4, 7 3, 50 3, 52 3, 76 3, 133 3, 154 3, 1 2, 107 2, 2 1, 14 1, 26 1, 74 1, 147 1, 156 1  
 CHILDES1: 22 131, 24 102, 106 93, 87 22, 142 13, 129 9, 104 4, 116 4, 52 3, 1 2, 23 2, 111 2, 113 2, 154 2, 14 1, 19 1, 74 1, 123 1, 133 1  
**draw**



BNC: 24 56, 22 47, 37 6, 87 5, 106 4, 76 2, 104 2, 129 2, 1 1, 8 1, 23 1, 133 1  
 CHILDES1: 24 61, 22 44, 87 6, 37 4, 52 4, 106 3, 7 1, 49 1, 76 1, 104 1, 123 1, 133 1, 142  
 1  
**drink**

BNC: 24 38, 22 23, 37 9, 106 5, 133 4, 132 2, 26 1, 47 1, 50 1, 52 1, 59 1, 74 1, 87 1  
 CHILDES1: 24 53, 22 16, 37 10, 26 3, 52 3, 106 2, 23 1, 87 1, 133 1, 142 1  
**drive**

BNC: 22 47, 24 28, 87 7, 37 2, 104 2, 106 2, 154 2, 1 1, 14 1, 19 1, 25 1, 75 1, 113  
 1, 129 1  
 CHILDES1: 24 49, 22 39, 87 4, 37 3, 1 1, 14 1, 23 1, 25 1, 76 1, 83 1, 116 1  
**drop**

BNC: 24 20, 22 12, 76 7, 37 2, 52 2, 1 1, 77 1, 87 1, 104 1, 142 1  
 CHILDES1: 24 34, 22 7, 76 4, 106 3, 104 2, 8 1, 25 1, 62 1, 87 1, 116 1, 133 1  
**eat**

BNC: 24 151, 22 114, 37 46, 106 24, 104 11, 1 6, 50 5, 76 4, 133 4, 124 2, 129 2,  
 132 2, 142 2, 2 1, 8 1, 23 1, 25 1, 35 1, 52 1, 97 1, 122 1, 134 1, 153 1, 154 1,  
 156 1  
 CHILDES1: 24 175, 22 105, 37 40, 87 17, 104 16, 106 11, 49 10, 129 6, 50 5, 76 5, 25 3, 7  
 2, 77 2, 116 2, 142 2, 1 1, 75 1, 132 1, 133 1  
**fall**

BNC: 22 28, 1 11, 87 11, 24 3, 129 3, 74 2, 106 2, 9 1, 104 1  
 CHILDES1: 22 32, 87 6, 106 6, 2 2, 24 1, 104 1  
**feed**

BNC: 24 14, 22 3, 1 2, 37 2, 49 2, 7 1, 8 1, 52 1, 87 1, 104 1, 132 1  
 CHILDES1: 24 38, 22 8, 37 2, 49 1, 104 1, 117 1  
**feel**

BNC: 1 39, 22 39, 24 21, 106 18, 104 7, 26 4, 25 3, 2 2, 87 2, 75 1  
 CHILDES1: 1 53, 22 47, 24 18, 106 14, 116 2, 137 2, 26 1, 37 1, 104 1  
**find**

BNC: 24 195, 22 70, 104 26, 106 24, 37 21, 25 14, 26 8, 1 5, 76 4, 50 3, 7 2, 112 2,  
 133 2, 16 1, 23 1, 52 1, 117 1, 123 1, 129 1, 134 1, 142 1, 156 1  
 CHILDES1: 24 303, 22 63, 106 30, 37 28, 104 18, 1 2, 50 2, 62 2, 74 2, 76 2, 133 2, 16 1,  
 117 1  
**finish**

BNC: 22 56, 24 46, 37 12, 106 11, 19 6, 87 5, 76 4, 49 3, 104 2, 156 2, 12 1, 74 1, 98  
 1, 133 1, 158 1  
 CHILDES1: 24 58, 19 33, 22 26, 37 10, 87 9, 49 6, 104 5, 25 2, 26 2, 52 2, 106 2, 133 2, 1  
 1, 142 1  
**fit**

BNC: 22 23, 24 16, 76 4, 37 3, 129 2, 2 1, 74 1, 87 1, 106 1, 123 1, 132 1  
 CHILDES1: 22 21, 24 8, 87 3, 76 2, 106 2, 123 2, 107 1, 113 1, 154 1  
**forget**

BNC: 22 28, 24 23, 104 11, 87 5, 112 5, 52 3, 23 2, 37 2, 83 2, 106 2, 16 1, 26 1, 35  
 1, 49 1, 50 1, 76 1, 105 1, 133 1, 154 1  
 CHILDES1: 24 32, 22 24, 112 17, 104 5, 106 5, 52 3, 37 2, 49 2, 87 2, 8 1, 25 1  
**get**

BNC: 24 1322, 22 903, 37 308, 1 141, 76 136, 2 131, 106 88, 104 75, 49 73, 25 41, 50 32, 53 27, 117 27, 87 25, 112 24, 26 21, 52 16, 123 16, 132 16, 35 15, 7 13, 133 12, 23 10, 19 7, 77 7, 142 7, 129 6, 8 5, 74 4, 124 4, 154 4, 59 3, 62 3, 86 3, 98 3, 122 3, 147 3, 116 2, 146 2, 158 2, 14 1, 16 1, 40 1, 43 1, 85 1, 97 1, 107 1, 111 1, 120 1, 148 1, 150 1, 156 1

CHILDES1: 24 1703, 22 587, 1 283, 76 157, 106 147, 2 122, 112 102, 104 72, 25 19, 7 18, 123 17, 142 17, 50 14, 133 13, 52 10, 26 9, 116 9, 117 8, 23 6, 154 5, 8 4, 62 3, 74 3, 129 3, 132 3, 9 2, 19 2, 53 2, 107 2, 111 2, 153 2, 75 1, 83 1, 103 1

**give**

BNC: 37 440, 24 283, 76 29, 106 25, 7 4, 117 4, 132 4, 8 3, 19 3, 120 2, 123 2, 133 2, 25 1, 53 1, 62 1, 107 1

CHILDES1: 37 450, 24 355, 76 27, 106 21, 1 11, 7 6, 117 5, 35 2, 62 2, 120 2, 116 1, 132 1, 133 1, 142 1

**go**

BNC: 22 2181, 24 353, 112 329, 87 221, 1 130, 106 74, 74 37, 23 35, 142 21, 9 18, 19 18, 111 16, 129 13, 25 9, 26 9, 123 9, 76 7, 132 6, 154 6, 75 4, 107 4, 133 4, 2 3, 67 3, 97 3, 113 3, 16 2, 53 2, 86 2, 43 1, 46 1, 83 1, 124 1, 137 1, 150 1

CHILDES1: 22 2265, 112 991, 106 211, 142 87, 9 34, 74 28, 1 27, 111 21, 19 20, 104 19, 129 19, 23 18, 2 10, 76 7, 123 7, 116 6, 14 5, 154 4, 62 3, 107 3, 6 1, 16 1, 86 1, 105 1, 113 1, 153 1, 158 1

**hang**

BNC: 22 18, 24 9, 37 5, 87 5, 76 4, 106 3, 104 2, 75 1, 133 1

CHILDES1: 22 23, 24 21, 87 10, 76 6, 106 5, 26 4, 74 4, 37 3, 104 2, 142 2, 129 1

**happen**

BNC: 22 142, 24 24, 87 13, 106 10, 104 8, 49 6, 74 4, 112 4, 1 1, 26 1, 52 1, 59 1, 107 1, 111 1, 113 1, 129 1, 153 1

CHILDES1: 22 195, 87 10, 74 5, 104 4, 49 2, 9 1, 23 1, 24 1, 76 1, 106 1, 153 1

**hear**

BNC: 24 73, 22 48, 104 23, 106 16, 37 7, 87 4, 49 3, 35 2, 76 2, 7 1, 8 1, 19 1, 23 1, 26 1, 52 1, 97 1, 113 1, 116 1, 117 1, 122 1, 133 1

CHILDES1: 24 117, 106 33, 22 24, 104 13, 37 4, 52 2, 117 2, 156 2, 1 1, 23 1, 62 1, 105 1, 123 1

**help**

BNC: 24 59, 22 37, 106 18, 37 7, 76 5, 104 3, 112 2, 19 1, 49 1, 50 1, 52 1, 53 1, 129 1, 133 1

CHILDES1: 24 81, 106 34, 22 22, 26 1, 37 1, 49 1, 76 1

**hit**

BNC: 24 36, 22 4, 104 3, 76 2, 7 1, 16 1, 25 1, 37 1, 117 1, 129 1

CHILDES1: 24 34, 22 10, 106 4, 1 1, 8 1, 76 1, 104 1

**hold**

BNC: 24 50, 22 10, 76 9, 37 8, 87 3, 106 2, 123 2, 7 1, 8 1, 25 1, 52 1, 107 1, 133 1, 142 1

CHILDES1: 24 72, 22 7, 76 7, 106 5, 123 4, 87 2, 104 2, 133 2, 142 2, 37 1, 132 1

**keep**

BNC: 24 80, 19 48, 22 34, 76 16, 104 15, 106 12, 25 9, 1 7, 49 3, 7 1, 26 1, 35 1, 112 1

CHILDES1: 24 85, 19 72, 76 19, 106 17, 104 16, 25 9, 22 5, 1 3, 7 2, 49 2, 86 2, 8 1, 37 1, 40 1, 50 1, 75 1, 123 1

**knock**

BNC: 24 20, 22 12, 76 12, 87 9, 49 3, 106 3, 37 2, 26 1, 74 1, 132 1, 156 1  
 CHILDES1: 76 37, 24 23, 22 12, 106 6, 87 5, 25 1  
**know**

BNC: 22 1044, 24 204, 104 197, 106 142, 26 44, 37 30, 116 14, 154 9, 23 7, 113 7, 129 6, 16 5, 1 4, 133 3, 153 3, 142 2, 7 1, 35 1, 49 1, 103 1, 105 1, 123 1, 139 1, 159 1  
 CHILDES1: 22 647, 104 499, 106 167, 24 159, 116 107, 26 40, 37 22, 113 12, 139 8, 17 7, 133 5, 154 5, 50 4, 87 3, 1 2, 23 2, 129 2, 7 1, 8 1, 35 1, 49 1, 62 1, 78 1, 137 1, 142 1, 147 1, 153 1  
**laugh**

BNC: 22 42, 87 6, 24 5, 106 3, 26 2, 104 2, 113 2, 23 1, 37 1, 74 1, 83 1, 101 1, 133 1  
 CHILDES1: 22 53, 24 6, 87 2, 104 2, 1 1, 37 1, 142 1, 153 1  
**leave**

BNC: 24 119, 22 52, 37 14, 106 11, 49 10, 76 6, 1 4, 25 3, 104 3, 132 3, 26 2, 52 2, 85 2, 87 2, 150 2, 16 1, 23 1, 35 1, 50 1, 98 1, 129 1, 133 1  
 CHILDES1: 24 141, 22 38, 76 22, 106 17, 49 9, 37 5, 104 5, 8 2, 25 2, 123 2, 87 1, 107 1, 129 1  
**let**

BNC: 106 388, 24 86, 22 25, 76 16, 52 2, 142 2, 7 1, 117 1  
 CHILDES1: 106 501, 24 13, 37 13, 22 11  
**like**

BNC: 24 792, 22 733, 106 99, 112 74, 104 69, 19 25, 23 19, 1 18, 76 16, 26 14, 129 12, 7 11, 8 9, 154 8, 133 7, 142 7, 25 5, 83 5, 123 5, 2 3, 35 3, 107 3, 105 2, 111 2, 116 2, 124 2, 82 1, 98 1, 113 1  
 CHILDES1: 24 994, 112 404, 22 382, 106 55, 104 28, 19 18, 111 13, 1 8, 52 6, 154 6, 8 5, 7 4, 25 4, 50 4, 116 4, 133 4, 23 3, 123 3, 35 2, 107 2, 142 2, 156 2, 62 1, 129 1  
**listen**

BNC: 22 52, 87 38, 74 3, 97 3, 104 3, 106 3, 112 3, 24 2, 52 2, 16 1, 26 1, 53 1, 83 1, 98 1, 113 1  
 CHILDES1: 87 45, 22 37, 106 13, 24 4, 74 4, 104 4, 67 2, 77 2, 97 2, 23 1, 37 1, 52 1, 89 1, 112 1, 113 1, 133 1, 142 1  
**live**

BNC: 22 32, 87 28, 24 7, 104 3, 74 2, 106 2, 133 2, 16 1, 98 1  
 CHILDES1: 22 48, 87 25, 24 5, 133 2, 106 1  
**look**

BNC: 22 670, 87 386, 1 232, 24 105, 106 101, 104 67, 26 17, 74 16, 37 13, 159 13, 16 10, 25 10, 76 10, 23 8, 75 7, 142 7, 116 5, 129 5, 154 5, 107 4, 14 3, 49 3, 133 3, 69 2, 123 2, 2 1, 7 1, 8 1, 59 1, 82 1, 137 1  
 CHILDES1: 87 612, 22 437, 104 157, 106 147, 1 85, 24 64, 52 22, 26 16, 74 16, 16 9, 142 9, 37 8, 69 5, 133 5, 154 5, 14 4, 23 4, 116 4, 129 3, 25 1, 35 1, 75 1, 76 1, 77 1, 78 1, 98 1, 101 1, 107 1  
**lose**

BNC: 24 28, 22 17, 37 6, 50 2, 87 2, 106 2, 2 1, 25 1, 26 1, 52 1, 117 1  
 CHILDES1: 24 38, 22 12, 104 4, 50 2, 142 2, 1 1, 8 1, 37 1, 113 1, 123 1  
**love**

BNC: 22 44, 24 43, 52 4, 104 4, 112 4, 19 3, 37 3, 106 2, 23 1, 25 1, 76 1, 107 1  
 CHILDES1: 24 78, 22 11, 112 6, 37 4, 104 4, 106 4, 1 1, 2 1, 132 1, 133 1

**make**

BNC: 24 476, 22 138, 106 132, 37 84, 25 35, 1 33, 104 32, 76 23, 7 7, 117 6, 23 4, 123 4, 132 4, 26 3, 8 2, 53 2, 77 2, 133 2, 35 1, 40 1, 52 1, 62 1, 105 1, 107 1, 112 1, 113 1, 129 1, 142 1, 147 1, 150 1, 154 1, 156 1  
CHILDES1: 24 550, 106 214, 22 151, 37 34, 25 30, 76 12, 104 11, 7 10, 1 5, 8 3, 49 3, 62 3, 116 3, 87 2, 105 2, 123 2, 132 2, 26 1, 35 1, 52 1, 107 1, 117 1, 133 1

**mean**

BNC: 22 175, 24 48, 104 47, 106 42, 26 14, 87 5, 112 5, 113 4, 49 3, 52 3, 57 3, 142 2, 19 1, 59 1, 75 1, 83 1, 133 1, 153 1  
CHILDES1: 22 158, 24 67, 104 39, 106 30, 87 16, 112 9, 142 7, 1 5, 49 5, 83 5, 52 3, 26 2, 107 2, 113 2, 16 1, 23 1, 35 1, 57 1, 75 1, 97 1, 133 1, 147 1, 154 1, 156 1

**miss**

BNC: 24 54, 22 12, 37 7, 104 3, 8 1, 76 1, 123 1  
CHILDES1: 24 49, 22 21, 37 2, 104 2, 106 2, 49 1, 107 1, 123 1

**move**

BNC: 24 50, 22 48, 87 8, 76 6, 37 5, 117 3, 106 2, 7 1, 23 1, 49 1, 74 1, 104 1, 120 1, 123 1, 129 1, 133 1, 142 1  
CHILDES1: 24 61, 22 51, 76 25, 106 6, 37 3, 87 2, 104 2, 117 2, 7 1, 16 1, 62 1, 74 1, 116 1, 129 1, 133 1, 153 1

**need**

BNC: 22 109, 24 108, 112 90, 37 31, 106 17, 19 11, 49 10, 104 10, 52 6, 133 6, 35 3, 50 3, 53 3, 8 2, 9 2, 76 2, 132 2, 147 2, 1 1, 7 1, 16 1, 23 1, 25 1, 57 1, 67 1, 75 1, 86 1, 97 1, 111 1, 117 1, 123 1, 124 1, 154 1, 156 1  
CHILDES1: 24 153, 112 109, 22 60, 37 44, 49 19, 53 9, 106 7, 76 6, 104 5, 111 4, 19 3, 23 2, 25 2, 50 2, 133 2, 59 1, 116 1, 123 1, 129 1, 154 1

**okay**

BNC: 22 139, 24 21, 106 10, 104 5, 1 1, 7 1, 9 1, 23 1, 26 1, 87 1, 103 1, 113 1, 129 1, 142 1, 154 1, 156 1  
CHILDES1: 22 188, 106 2, 142 2, 24 1, 77 1

**open**

BNC: 24 66, 22 42, 37 5, 104 5, 76 4, 106 3, 8 1, 25 1, 49 1, 83 1, 117 1, 123 1, 129 1, 142 1, 154 1  
CHILDES1: 24 73, 22 42, 37 7, 106 3, 133 3, 76 2, 123 2, 50 1, 52 1, 104 1, 116 1, 142 1

**pick**

BNC: 76 62, 24 56, 22 23, 37 17, 106 8, 49 6, 104 6, 117 5, 87 4, 52 3, 146 2, 1 1, 40 1, 77 1, 107 1, 132 1, 133 1  
CHILDES1: 76 60, 24 28, 106 24, 22 14, 37 6, 49 6, 52 6, 77 6, 117 4, 87 3, 7 2, 50 2, 133 2, 2 1, 104 1, 113 1, 124 1, 129 1, 132 1

**play**

BNC: 22 194, 24 132, 87 28, 76 7, 104 7, 117 6, 106 5, 7 3, 23 3, 129 3, 1 2, 26 2, 50 2, 19 1, 25 1, 52 1, 62 1, 74 1, 75 1, 98 1, 107 1, 123 1, 133 1, 142 1  
CHILDES1: 22 159, 87 100, 24 70, 106 11, 1 6, 23 4, 76 2, 104 2, 113 2, 142 2, 26 1, 62 1, 74 1, 83 1, 133 1

**please**

BNC: 22 36, 24 10, 106 5, 142 3, 52 1, 75 1, 104 1, 113 1  
CHILDES1: 22 33, 106 17, 24 3, 35 1, 49 1, 87 1, 104 1, 142 1

**pull**

BNC: 24 45, 76 24, 22 11, 117 4, 37 3, 87 2, 7 1, 8 1, 19 1, 26 1, 106 1  
CHILDES1: 24 55, 76 30, 106 12, 117 5, 8 4, 22 4, 37 1, 87 1, 104 1, 132 1

**push**  
 BNC: 24 21, 76 12, 22 8, 37 7, 49 5, 106 5, 87 3, 117 2, 1 1, 7 1, 25 1, 40 1, 112 1, 123 1, 132 1, 133 1, 142 1, 150 1  
 CHILDES1: 24 27, 76 14, 22 9, 106 9, 37 3, 117 3, 133 3, 1 2, 87 2, 104 2, 25 1, 49 1

**put**  
 BNC: 24 702, 76 376, 22 214, 49 88, 37 75, 106 47, 104 24, 87 23, 117 16, 123 12, 7 10, 50 10, 25 7, 77 6, 52 4, 74 4, 133 4, 8 3, 23 3, 26 3, 40 2, 120 2, 122 2, 132 2, 142 2, 147 2, 16 1, 35 1, 53 1, 59 1, 75 1, 83 1, 89 1, 113 1, 116 1, 124 1, 129 1, 148 1, 156 1  
 CHILDES1: 24 749, 76 541, 106 72, 117 27, 104 15, 77 12, 156 10, 7 5, 132 5, 87 3, 123 3, 129 3, 1 2, 50 2, 52 2, 142 2, 2 1, 8 1, 74 1, 124 1, 133 1

**read**  
 BNC: 24 80, 22 59, 106 6, 76 4, 104 4, 123 3, 37 2, 87 2, 129 2, 7 1, 19 1, 23 1, 52 1, 103 1, 116 1, 117 1  
 CHILDES1: 24 89, 22 37, 37 35, 106 4, 104 3, 87 2, 116 2, 23 1, 35 1, 52 1, 83 1, 117 1, 129 1

**remember**  
 BNC: 22 156, 24 115, 104 67, 37 23, 19 20, 83 19, 106 19, 52 16, 26 8, 35 8, 112 5, 23 4, 133 4, 156 3, 53 2, 113 2, 116 2, 153 2, 1 1, 2 1, 7 1, 8 1, 16 1, 25 1, 43 1, 75 1, 98 1, 107 1, 129 1, 147 1  
 CHILDES1: 22 190, 104 87, 24 85, 83 24, 106 22, 52 16, 16 13, 37 10, 49 10, 112 7, 87 6, 19 4, 23 4, 129 4, 50 3, 133 3, 153 3, 154 3, 53 2, 142 2, 1 1, 35 1, 97 1, 156 1

**roll**  
 BNC: 24 21, 22 11, 106 5, 53 2, 76 2, 87 2, 142 2, 19 1, 26 1, 49 1, 59 1, 97 1, 104 1  
 CHILDES1: 24 29, 22 4, 49 3, 87 3, 76 2, 106 2, 117 2, 146 2, 25 1, 37 1, 52 1, 83 1, 122 1, 133 1

**run**  
 BNC: 22 41, 24 29, 87 14, 76 7, 37 3, 106 3, 1 1, 49 1, 153 1  
 CHILDES1: 22 67, 87 12, 24 7, 1 3, 74 1, 76 1, 104 1, 106 1, 129 1

**say**  
 BNC: 22 680, 24 323, 104 255, 106 254, 26 56, 52 24, 23 14, 133 14, 116 13, 1 12, 129 6, 142 5, 7 3, 107 3, 8 1, 103 1, 113 1, 123 1, 124 1  
 CHILDES1: 24 643, 22 617, 106 211, 104 133, 1 38, 23 11, 116 10, 133 10, 19 8, 107 7, 26 6, 129 6, 142 5, 2 3, 7 2, 123 2, 62 1, 113 1, 153 1, 154 1

**see**  
 BNC: 22 1018, 24 909, 104 259, 106 225, 37 112, 26 54, 49 33, 116 17, 113 16, 16 15, 23 13, 35 9, 25 8, 129 8, 133 8, 1 6, 123 5, 7 4, 142 4, 154 4, 8 3, 59 3, 98 3, 19 2, 47 2, 112 2, 156 2, 2 1, 62 1, 83 1, 105 1, 124 1, 147 1, 153 1  
 CHILDES1: 22 1123, 24 929, 104 308, 106 247, 113 24, 76 20, 26 17, 16 12, 52 8, 116 7, 1 6, 133 6, 62 5, 154 5, 23 4, 35 4, 74 4, 25 3, 123 3, 129 3, 7 2, 8 2, 107 2, 137 2, 142 2, 153 2

**send**  
 BNC: 24 48, 22 15, 37 13, 76 11, 132 2, 8 1, 25 1, 49 1, 77 1, 117 1  
 CHILDES1: 24 44, 37 27, 22 24, 49 2, 50 2, 104 2, 1 1, 76 1, 77 1, 123 1

**set**  
 BNC: 24 14, 22 9, 49 6, 37 4, 76 2, 87 2, 7 1, 52 1, 77 1  
 CHILDES1: 24 13, 76 12, 22 2, 49 2, 106 2, 25 1, 87 1

**show**

BNC: 24 108, 37 36, 22 26, 104 10, 106 5, 49 4, 76 4, 87 4, 132 3, 1 2, 8 2, 62 2, 52 1, 107 1, 156 1  
CHILDES1: 24 143, 37 40, 62 7, 22 6, 52 4, 106 4, 104 3, 49 1, 59 1, 156 1  
**sing**

BNC: 24 25, 22 22, 37 3, 87 3, 106 3, 76 2, 104 2, 23 1, 26 1, 49 1, 142 1  
CHILDES1: 22 19, 24 18, 87 9, 37 4, 49 4, 104 3, 112 2, 133 2, 74 1, 103 1, 106 1, 137 1, 142 1  
**sit**

BNC: 22 141, 87 27, 24 10, 104 5, 74 4, 19 3, 106 3, 76 2, 1 1, 23 1, 40 1, 67 1, 69 1  
CHILDES1: 22 122, 87 37, 106 4, 19 3, 142 2, 23 1, 74 1, 104 1, 129 1, 154 1  
**sleep**

BNC: 22 51, 87 30, 24 5, 106 4, 104 2, 142 2, 14 1, 23 1, 25 1, 50 1, 77 1, 129 1  
CHILDES1: 22 61, 24 20, 87 16, 106 3, 1 2, 104 2, 52 1, 77 1, 83 1, 116 1  
**sound**

BNC: 1 14, 22 13, 106 8, 87 4, 7 2, 24 2, 37 2, 12 1, 49 1, 97 1, 104 1, 142 1, 154 1, 159 1  
CHILDES1: 1 16, 106 10, 22 9, 142 7, 87 2, 23 1, 49 1, 75 1, 103 1  
**stand**

BNC: 22 60, 24 29, 87 15, 49 3, 106 3, 7 2, 19 2, 37 2, 52 2, 104 2, 1 1, 2 1, 14 1, 23 1, 74 1, 86 1  
CHILDES1: 22 34, 76 19, 87 12, 24 9, 106 6, 104 4, 123 1, 133 1, 142 1  
**start**

BNC: 24 36, 22 35, 19 19, 112 6, 1 2, 37 2, 104 2, 9 1, 53 1, 76 1, 87 1, 106 1  
CHILDES1: 22 43, 24 21, 19 12, 112 8, 106 5, 52 2, 87 2, 104 2, 9 1, 76 1, 111 1, 142 1  
**stay**

BNC: 22 96, 87 23, 24 8, 106 6, 104 5, 1 3, 23 1, 37 1, 74 1, 76 1, 97 1, 113 1, 123 1, 154 1  
CHILDES1: 22 91, 1 11, 106 9, 87 7, 24 6, 26 2, 35 2, 52 2, 129 2, 7 1, 19 1, 37 1, 62 1  
**stick**

BNC: 22 40, 24 39, 87 18, 37 11, 49 11, 106 5, 52 3, 76 3, 2 2, 14 2, 107 2, 156 2, 19 1, 35 1, 62 1, 74 1, 104 1, 112 1, 122 1, 129 1, 153 1, 154 1, 158 1  
CHILDES1: 24 45, 22 43, 49 22, 87 22, 37 6, 52 5, 74 5, 158 3, 26 2, 106 2, 132 2, 133 2, 104 1, 107 1, 129 1  
**stop**

BNC: 22 48, 24 31, 19 10, 35 6, 37 5, 106 3, 1 2, 104 2, 87 1, 116 1, 123 1, 133 1, 142 1  
CHILDES1: 22 48, 24 46, 19 13, 106 7, 6 3, 142 3, 104 1  
**suppose**

BNC: 22 28, 24 16, 104 13, 53 12, 106 11, 112 7, 26 3, 52 2, 57 2, 111 2, 37 1, 49 1, 133 1, 142 1  
CHILDES1: 112 35, 53 27, 22 13, 24 9, 9 4, 57 4, 106 4, 74 2, 111 2, 23 1, 26 1  
**take**

BNC: 24 662, 76 174, 22 168, 37 71, 106 39, 104 23, 87 17, 49 16, 117 15, 1 9, 7 6, 77 5, 132 4, 9 3, 53 3, 112 3, 6 2, 11 2, 23 2, 26 2, 59 2, 116 2, 8 1, 19 1, 25 1, 67 1, 83 1, 98 1, 113 1, 120 1, 123 1, 124 1, 129 1, 133 1, 142 1, 156 1  
CHILDES1: 24 694, 76 312, 22 85, 106 40, 37 25, 104 20, 7 14, 117 12, 49 10, 87 8, 1 4, 8 4, 77 4, 116 3, 133 3, 142 2, 23 1, 53 1, 59 1, 107 1, 129 1, 154 1, 156 1  
**talk**

BNC: 22 164, 87 102, 24 25, 1 12, 74 7, 23 3, 123 3, 16 2, 75 2, 129 2, 14 1, 25 1, 76 1, 97 1, 101 1, 106 1, 154 1

CHILDES1: 22 156, 87 137, 74 25, 1 8, 106 8, 75 2, 104 2, 14 1, 142 1, 153 1

**tell**

BNC: 24 511, 37 85, 52 48, 104 29, 62 26, 106 17, 133 14, 132 8, 7 7, 49 6, 53 6, 76 6, 8 3, 26 3, 116 3, 6 2, 59 2, 117 2, 142 2, 1 1, 23 1, 50 1, 74 1, 83 1, 87 1, 107 1, 112 1, 124 1, 129 1, 156 1

CHILDES1: 24 585, 37 122, 104 23, 62 21, 106 21, 52 17, 133 6, 1 4, 7 2, 8 1, 25 1, 35 1, 40 1, 50 1, 76 1, 83 1, 129 1, 139 1, 153 1, 156 1

**thank**

BNC: 24 123, 22 17, 37 8, 7 3, 52 3, 106 3, 49 2, 104 2, 25 1, 75 1, 133 1

CHILDES1: 24 149, 22 9, 7 2, 106 2, 23 1, 49 1

**think**

BNC: 22 651, 104 611, 106 356, 24 350, 26 113, 133 13, 23 8, 1 6, 116 4, 154 4, 113 3, 19 2, 105 2, 123 2, 7 1, 35 1, 62 1, 67 1, 107 1, 111 1, 129 1, 142 1

CHILDES1: 104 870, 22 547, 106 543, 26 70, 35 5, 23 4, 129 3, 142 3, 2 2, 25 2, 116 2, 133 2, 154 2, 1 1, 8 1, 62 1

**throw**

BNC: 24 47, 76 37, 22 19, 37 7, 49 6, 106 2, 132 2, 26 1, 87 1, 103 1

CHILDES1: 24 59, 22 33, 37 10, 76 9, 8 8, 106 8, 49 6, 62 2, 154 2, 1 1, 133 1

**touch**

BNC: 24 34, 22 7, 37 4, 104 3, 8 2, 76 1, 123 1, 133 1

CHILDES1: 24 29, 106 10, 22 4, 8 2, 105 2, 133 2, 142 2, 35 1, 37 1, 154 1

**try**

BNC: 112 142, 22 109, 24 62, 106 13, 76 7, 1 5, 19 5, 104 5, 111 3, 7 1, 9 1, 35 1, 50 1, 83 1, 117 1, 132 1, 133 1

CHILDES1: 24 103, 112 78, 22 75, 37 49, 106 25, 19 20, 76 7, 9 3, 116 2, 1 1, 49 1, 50 1, 52 1, 104 1, 117 1, 132 1, 133 1

**turn**

BNC: 24 57, 76 56, 22 33, 1 10, 106 10, 37 7, 117 5, 2 4, 87 3, 104 3, 74 2, 120 2, 8 1, 23 1, 26 1, 49 1, 62 1, 75 1, 123 1, 133 1, 142 1, 150 1

CHILDES1: 24 118, 22 48, 37 31, 106 27, 76 26, 117 6, 104 5, 1 2, 7 2, 6 1, 8 1, 49 1, 52 1, 132 1

**understand**

BNC: 22 24, 24 24, 106 6, 37 2, 1 1, 23 1, 26 1

CHILDES1: 22 36, 24 23, 129 3, 106 2, 7 1, 116 1

**use**

BNC: 24 77, 22 74, 112 56, 106 8, 111 8, 37 7, 74 4, 75 4, 9 3, 87 3, 23 2, 40 2, 53 2, 104 2, 98 1, 116 1, 117 1, 133 1

CHILDES1: 24 139, 22 68, 112 28, 87 6, 106 5, 111 3, 104 2, 123 2, 8 1, 23 1, 53 1, 83 1, 147 1

**wait**

BNC: 22 103, 87 56, 24 43, 106 14, 112 12, 142 7, 23 5, 52 4, 15 3, 37 3, 75 2, 9 1, 74 1, 78 1, 83 1, 98 1, 103 1, 107 1, 132 1, 133 1, 154 1

CHILDES1: 24 92, 106 74, 22 52, 87 17, 104 10, 142 6, 112 5, 52 4, 26 3, 83 3, 133 3, 129 2, 7 1, 23 1, 117 1

**walk**

BNC: 22 60, 24 18, 87 12, 49 3, 106 3, 1 2, 23 2, 37 2, 74 2, 142 2, 59 1, 76 1, 78 1, 104 1, 129 1

CHILDES1: 22 69, 87 25, 24 11, 1 1, 23 1, 76 1, 104 1, 106 1  
           **want**  
           BNC: 24 860, 112 519, 22 508, 106 53, 37 52, 104 41, 76 24, 111 18, 19 17, 53 14,  
               87 10, 142 10, 1 8, 52 7, 49 6, 133 6, 8 5, 26 5, 35 4, 154 4, 7 3, 9 3, 23 3, 25  
               3, 116 3, 123 3, 132 3, 105 2, 117 2, 59 1, 75 1, 107 1, 113 1  
 CHILDES1: 24 848, 112 797, 22 237, 106 141, 76 22, 53 21, 111 17, 1 15, 116 8, 142 8,  
               50 7, 7 6, 9 6, 35 5, 133 4, 57 2, 8 1, 11 1, 14 1, 77 1  
           **wash**  
           BNC: 24 35, 22 15, 37 6, 76 3, 106 3, 133 3, 104 2, 2 1, 26 1, 49 1, 52 1, 87 1, 153  
               1, 156 1  
 CHILDES1: 24 64, 22 5, 37 5, 49 3, 7 2, 76 2, 106 2, 52 1, 87 1, 133 1, 142 1  
           **watch**  
           BNC: 24 143, 22 58, 106 19, 37 11, 104 9, 7 3, 8 2, 35 2, 23 1, 26 1, 76 1, 116 1, 123  
               1, 129 1  
 CHILDES1: 22 98, 24 84, 106 26, 104 16, 1 2, 7 2, 76 2, 116 2, 25 1, 26 1, 107 1, 133 1,  
               142 1  
           **wear**  
           BNC: 24 43, 22 24, 37 18, 49 7, 106 7, 1 5, 133 4, 87 3, 7 1, 23 1, 25 1, 52 1, 76 1,  
               77 1, 83 1, 104 1, 147 1  
 CHILDES1: 24 49, 22 40, 87 9, 37 7, 104 6, 49 4, 106 4, 6 1, 23 1, 52 1, 53 1, 117 1, 132  
               1, 133 1  
           **will**  
           BNC: 22 39, 24 17, 106 8, 104 3, 37 2, 2 1, 19 1, 26 1, 78 1, 87 1, 111 1, 142 1  
 CHILDES1: 22 34, 24 15, 106 11, 116 2, 16 1, 37 1, 113 1, 133 1, 153 1  
           **wonder**  
           BNC: 22 17, 104 16, 106 8, 113 8, 87 6, 16 5, 116 3, 17 1, 23 1, 37 1, 115 1  
 CHILDES1: 104 38, 113 10, 22 7, 106 7, 24 4, 16 2, 37 1  
           **work**  
           BNC: 22 93, 24 18, 87 14, 106 7, 37 4, 76 4, 1 1, 7 1, 16 1, 23 1, 26 1, 49 1, 104 1,  
               113 1, 142 1, 154 1  
 CHILDES1: 22 114, 87 10, 1 6, 24 6, 129 6, 23 3, 106 3, 37 1, 142 1



## Appendix D

# SCF Distributions in Adult Speech vs. Child Directed Speech

The following table gives the total frequencies of the SCFs that occurred for the 104 selected verbs in the BNC and the CHILDES1 corpora. The first column is the SCF number, the second is the SCF name, the third is the frequency in the BNC and the fourth is the frequency in CHILDES1.

N0.	Type (example)	BNC	CHILDES1
1	ADJP (his reputation sank low)	774	660
2	ADJP-PRED-RS (he appears crazy / distressed)	158	147
6	EXTRAP-NP-S (it annoys them that she left)	4	6
7	S-SUBJ-NP-OBJ (that she left annoys them )	114	97
8	TO-INF-SUBJ-NP-OBJ (to read pleases them)	57	52
9	EXTRAP-TO-INF (it remains to find a cure)	34	51
11	EXTRAP-NP-TO-INF (it pleases them to find a cure)	2	1
12	EXTRAP-TO-NP-S (it matters to them that she left)	2	2
14	S-SUBJ-TO-NP-OBJ (that she left matters to them)	12	13
15	FOR-TO-INF (i prefer for her to do it)	4	0
16	HOW-S (he asked how she did it)	54	42
17	HOW-TO-INF (he explained how to do it)	1	7
19	ING-NP-OMIT (his hair needs combing)	218	220
22	INTRANS	13677	11599

	(he went)		
23	INTRANS-RECIP(SUBJ-PL/COORD) (they met)	194	93
24	NP (he loved her)	12050	13042
25	NP-ADJP (he painted the car black)	176	93
26	NP-ADJP-PRED (she considered him foolish)	429	186
35	NP-ING-OC (i caught him stealing)	68	35
37	NP-NP (she asked him his name)	1895	1323
40	NP-P-ING-OC (i accused her of murdering her husband)	9	2
43	NP-P-NP-ING (he attributed his failure to noone buying his books)	3	0
46	NP-P-WHAT-S (they made a great fuss about what they should do)	1	0
47	NP-P-WHAT-TO-INF (they made a great fuss about what to do)	4	1
49	NP-PP (she added the flowers to the bouquet)	396	180
50	NP-PP-PRED (i considered that problem of little concern)	71	56
52	NP-S (he told the audience that he was leaving)	201	157
53	NP-TO-INF-OC (i advised ary to go)	90	66
57	NP-TOBE (i found him to be a good doctor)	6	8
59	NP-WH-S (they asked him whether he was going)	20	6
62	NP-WHAT-TO-INF (he asked him what to do)	40	61
67	P-NP-TO-INF-OC (he beckoned to him to come)	8	2
69	P-POSSING (they argued about his coming)	3	5
74	PART (she gave up)	124	122
75	PART-ING-SC (he ruled out paying her debts)	34	11
76	PART-NP/NP-PART (i looked up the entry)	1272	1568
77	PART-NP-PP (i separated out the three boys from the crowd)	27	32
78	PART-PP (she looked in on her friend)	7	2

82	PART-WHAT-TO-INF (they figured out what to do)	2	0
83	PART-THAT-S (they figured out that she hadn't done her job)	40	43
85	POSSING-PP (she attributed his drinking too much to his anxiety)	3	0
86	ING-PP (they limited smoking a pipe to the lounge)	7	3
87	PP (they apologized to him)	1358	1374
89	PP-HOW-S (he explained to her how she did it)	1	2
97	PP-THAT-S (they admitted to the authorities that they had entered illegally)	15	5
98	PP-THAT-S-SUBJUNCT (they suggested to him that he go)	16	1
101	PP-WHAT-S (they asked about everybody what they had done)	2	1
103	PP-WHAT-TO-INF (they deduced from kim what to do )	6	3
104	S (they thought that he was always late)	2183	2595
105	S-SUBJ-S-OBJ (for me to report the theft shows that i am guilty)	10	6
106	S-SUBJUNCT (he demanded that he leave immediately)	2689	3729
107	SEEM-S (it seems that they left)	36	28
111	TO-INF-RS (he seemed to come)	54	65
112	TO-INF-SC (i wanted to come)	1319	2623
113	WH-S (he asked whether he should come)	63	59
115	WH-TO-INF (he asked whether to clean the house)	1	0
116	WHAT-TO-INF (he asked what to do)	82	192
117	NP-NP-up (i opened him up a new bank account)	115	85
120	(SUBCAT NP-PP / PFORM, PRT, SUBTYPE DMOVT) (he bought a book back for me)	9	4
122	(SUBCAT NP-PP-PP, PFORM) (he turned it from a disaster into a victory)	9	2
123	(SUBCAT MP) (it cost ten pounds)	96	72
124	(SUBCAT NP-MP) (it cost him ten pounds)	16	2
129	(SUBCAT SFIN, AGR S[FIN +], SUBTYPE EXTRAP)	112	90

	(that he came matters)		
132	(SUBCAT NP-NP-SFIN) (he bet her ten pounds that he came)	75	22
133	(SUBCAT NP-SBSE) (he petitioned them that he be freed)	163	128
134	(SUBCAT IT-WHS, SUBTYPE IF, AGR N2[NFORM IT]) (i would appreciate it if he came)	2	1
137	(SUBCAT SC-AP, PRT, SUBTYPE EQUI/RAIS) (he started out poor)	4	6
139	(SUBCAT SC-INF, PRT, SUBTYPE EQUI) (he set out to win)	1	9
142	(SUBCAT SC-BSE, SUBTYPE EQUI) (he dared dance)	138	237
146	(SUBCAT OC-AP, SUBTYPE EQUI, PRT) (he sands it down smooth)	5	2
147	(SUBCAT OC-AP, SUBTYPE EQUI, PREP) as (he condemned him as stupid)	15	3
148	(SUBCAT OC-AP, SUBTYPE EQUI, PREP as, PRT) (he put him down as stupid)	2	0
150	(SUBCAT OC-INF, SUBTYPE EQUI, PRT) (he spurred him on to try)	7	0
153	(SUBCAT OC-PP-BSE, PFORM, SUBTYPE PVERB-OE) (he looked at him leave)	12	16
154	(SUBCAT VPINF, SUBTYPE EXTRAP, AGR VP[FIN-]) (to see them hurts)	68	49
156	NP-HOW-S (he asked him how he came)	28	19
158	IT-PASS-SFIN (it is believed that he came)	5	6
159	AS-IF-SFIN (he seems as if he is clever)	15	0

# Appendix E

## SCF Distribution in Child Speech

The following table gives the total frequencies of SCFs occurring in studied verbs in the CHILDES2 corpus. They are ranked according to frequency. The first column is the SCF number, the second is the SCF name and the third is the frequency with which it occurred in CHILDES2.

NO.	Type (example)	CHILDES2
24	NP (he loved her)	7904
22	INTRANS (he went)	5547
106	S-SUBJUNCT (he demanded that he leave immediately)	1980
76	PART-NP/NP-PART (i looked up the entry)	801
37	NP-NP (she asked him his name)	740
87	PP (they apologized to him)	736
112	TO-INF-SC (i wanted to come)	646
104	S (they thought that he was always late)	430
1	ADJP (his reputation sank low)	231
49	NP-PP (she added the flowers to the bouquet)	121
7	S-SUBJ-NP-OBJ (that she left annoys them )	112
142	(SUBCAT SC-BSE, SUBTYPE EQUI) (he dared dance)	111
123	(SUBCAT MP) (it cost ten pounds)	81
133	(SUBCAT NP-SBSE) (he petitioned them that he be freed)	64
2	ADJP-PRED-RS (he appears crazy / distressed)	58

19	ING-NP-OMIT (his hair needs combing)	56
25	NP-ADJP (he painted the car black)	55
129	(SUBCAT SFIN, AGR S[FIN +], SUBTYPE EXTRAP) (that he came matters)	54
116	WHAT-TO-INF (he asked what to do)	40
52	NP-S (he told the audience that he was leaving)	40
53	NP-TO-INF-OC (i advised ary to go)	39
74	PART (she gave up)	37
23	INTRANS-RECIP(SUBJ-PL/COORD) (hey met )	33
117	NP-NP-up (i opened him up a new bank account)	32
26	NP-ADJP-PRED (she considered him foolish)	26
132	(SUBCAT NP-NP-SFIN) (he bet her ten pounds that he came)	21
8	TO-INF-SUBJ-NP-OBJ (to read pleases them)	17
50	NP-PP-PRED (i considered that problem of little concern)	16
111	TO-INF-RS (he seemed to come)	14
35	NP-ING-OC (i caught him stealing)	12
130	(SUBCAT NP-SFIN, SUBTYPE NONE, PRT) (he had her on that he attended)	9
154	(SUBCAT VPINF, SUBTYPE EXTRAP, AGR VP[FIN-]) (to see them hurts)	9
16	HOW-S (he asked how she did it)	8
14	S-SUBJ-TO-NP-OBJ (that she left matters to them)	8
9	EXTRAP-TO-INF (it remains to find a cure)	8
77	PART-NP-PP (i separated out the three boys from the crowd)	7
69	P-POSSING (they argued about his coming)	5
17	HOW-TO-INF (he explained how to do it)	4
156	NP-HOW-S (he asked him how he came)	4
83	PART-THAT-S	4

	(they figured out that she had n't done her job)	
113	WH-S (he asked whether he should come)	4
97	PP-THAT-S (they admitted to the authorities that they had entered illegally)	4
153	(SUBCAT OC-PP-BSE, PFORM, SUBTYPE PVERB-OE) (he looked at him leave)	4
62	NP-WHAT-TO-INF (he asked him what to do)	4
139	(SUBCAT SC-INF, PRT, SUBTYPE EQUI) (he set out to win)	3
120	(SUBCAT NP-PP / PFORM, PRT, SUBTYPE DMOVT) (he brought a book back for me)	3
57	NP-TOBE (i found him to be a good doctor)	3
124	(SUBCAT NP-MP) (it cost him ten pounds)	3
40	NP-P-ING-OC (i accused her of murdering her husband)	2
75	PART-ING-SC (he ruled out paying her debts)	1
59	NP-WH-S (they asked him whether he was going)	1
150	(SUBCAT OC-INF, SUBTYPE EQUI, PRT) (he spurred him on to try)	1
107	SEEM-S (it seems that they left)	1
158	IT-PASS-SFIN (it is believed that he came)	1
105	S-SUBJ-S-OBJ (for him to report the theft indicates that he was n't guilty)	1
6	EXTRAP-NP-S (it annoys them that she left)	1
15	FOR-TO-INF (i prefer for her to do it)	1
47	NP-P-WHAT-TO-INF (they made a great fuss about what to do)	1





# Bibliography

- [1] P Adriaans. *Language learning from a categorial perspective*. University of Amsterdam, academish proefschrift edition, 1992.
- [2] F Baader and J Siekmann. Unification theory. In C Hogger D Gabbay and J Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, pages 41–125. Oxford University Press, Oxford, UK, 1994.
- [3] R Berwick. Learning word meaning from examples. In *Proceedings of the English International Joint Conference on Artificial Intelligence*, pages 459–461, Karlsruhe, Germany, 1983. IJCAI–83.
- [4] R Berwick and P Niyogi. Learning from triggers. *Linguistic Inquiry*, 27(4):605–622, 1996.
- [5] D Bickerton. The language bioprogram hypothesis. *The Behavioral and Brain Sciences*, 7(2):173–222, 1984.
- [6] M Bishop and K Mogford, editors. *Language Development in Exceptional Circumstances*. Psychology Press, Hove, UK, 1988.
- [7] B Boguraev, T Briscoe, J Carroll, D Carter, and C Grover. The derivation of a grammatically-indexed lexicon from the Longman Dictionary of Contemporary English. In *25th Annual Meeting of the Association for Computational Linguistics*, pages 193–200, Stanford, CA, 1987. ACL.
- [8] H Borer. *Parametric Syntax: Case Studies in Semitic and Romance Languages*. Foris, Dordrecht, Holland, 1984.
- [9] M Brent. Speech segmentation and word discovery: A computational perspective. *Trends in Cognitive Science*, 3:294–301, 1999.
- [10] E Briscoe and J Carroll. Automatic extraction of subcategorization from corpora. In *5th ACL Conference on Applied Natural Language Processing*, pages 356–363, Washington, DC, 1997. ACL.
- [11] E Briscoe and J Carroll. Robust accurate statistical annotation of general text. In *Third International Conference on Language Resources and Evaluation*, pages 1499–1504, Las Palmas, Canary Islands, 2002. LREC.
- [12] T Briscoe. The acquisition of grammar in an evolving population of language agents. *Machine Intelligence*, 16, 1999.

- [13] R Brown. *A first Language: the early stages*. Harvard University Press, Cambridge, MA, 1973.
- [14] R Brown and C Hanlon. Derivational complexity and the order of acquisition of child speech. In J Hayes, editor, *Cognition and the Development of Language*. Wiley, New York, NY, 1970.
- [15] R Bush and F Mosteller. A mathematical model for simple learning. *Psychological Review*, 68:313–23, 1951.
- [16] W Buszkowski. Discovery procedures for categorial grammars. In E Klein and J van Benthem, editors, *Categories, Polymorphism and Unification*. University of Amsterdam, Amsterdam, Holland, 1987.
- [17] W Buszkowski. Solvable problems for classic categorial grammars. *Bulletin of the Polish Academy of Sciences: Mathematics*, 35:373–382, 1987.
- [18] W Buszkowski and G Penn. Categorial grammars determined from linguistic data by unification. *Studia Logica*, 49:431–454, 1990.
- [19] P Buttery. Learning from annotated corpora. Master’s thesis, University of Cambridge, 2001.
- [20] P Buttery and T Briscoe. The significance of errors to parametric models of language acquisition. Technical Report SS-04-05, American Association of Artificial Intelligence AAAI, March 2004.
- [21] P Buttery and A Korhonen. Large-scale analysis of verb subcategorization differences between child directed speech and adult speech. In *Interdisciplinary Workshop on the Identification and Representation of Verb Features and Verb Classes*, Saarbrücken, DE, 2005.
- [22] N Chomsky. *Aspects of the Theory of Syntax*. MIT Press, 1965.
- [23] N Chomsky. *Rules and Representations*. Columbia University Press, New York, NY, 1980.
- [24] N Chomsky. *Lectures on Government and Binding*. Foris Publications, Dordrecht, Holland, 1981.
- [25] E Clark. The principle of contrast: A constraint on language acquisition. In B MacWhinney, editor, *Mechanisms of language acquisition*. Erlbaum, Hillsdale, NJ, 1987.
- [26] R Clark. Papers on learnability and natural selection. *Technical Reports in Computational Linguistics*, 1, 1990. Universite de Geneve.
- [27] R Clark. The selection of syntactic knowledge. *Language Acquisition*, 2(2):83–149, 1992.
- [28] R Clark. Finitude, boundedness and complexity: learnability and the study of first language acquisition. In B Lust *et al.*, editor, *Syntactic Theory and First Language Acquisition: Cross Linguistic Perspectives*. Lawrence Erlbaum, Mahwah, NJ, 1994.

- [29] S Clark and J Curran. Log-linear models for wide-coverage CCG parsing. In *the SIGDAT Conference on Empirical Methods in Natural Language Processing*, Sapporo, Japan, 2003. EMNLP '03.
- [30] S Curtiss. *Genie: a Psycholinguistic Study of a Modern-day "Wild Child"*. Academic Press, New York, NY, 1977.
- [31] M Demetras. Changes in parents' conversational responses: a function of grammatical development. In *1989 Annual Convention of The American Speech-Language-Hearing Association*, St Louis, MO, 1989. ASHA.
- [32] M Demetras. Working parents' conversational responses to their two-year-old sons. University of Arizona, Ms., 1989.
- [33] E Dresher and J Kaye. A computational learning model for metrical phonology. *Cognition*, 34:137–95, 1990.
- [34] J Elman. Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning*, 7:195–225, 1991.
- [35] Ethnologue. Languages of the world, 14th edition. SIL International, 2004. <http://www.ethnologue.com/>.
- [36] C Fisher, G Hall, S Rakowitz, and L Gleitman. When it is better to receive than to give: syntactic and conceptual constraints on vocabulary growth. *Lingua*, 92(1–4):333–375, April 1994.
- [37] J Fodor. Parsing to learn. *Journal of Psycholinguistic Research*, 27(3):339–374, 1998.
- [38] J Fodor. Unambiguous triggers. *Linguistic Inquiry*, 29(1):1–36, 1998.
- [39] E Gibson and K Wexler. Triggers. *Linguistic Inquiry*, 25(3):407–454, 1994.
- [40] J Gleason. The child's learning of English morphology. *Word*, 14:150–177, 1958.
- [41] L Gleitman. The structural sources of verb meaning. *Language Acquisition*, 1:3–55, 1990.
- [42] E Gold. Language identification in the limit. *Information and Control*, 10(5):447–474, 1967.
- [43] R Grishman, C Macleod, and A Meyers. Complex syntax: building a computational lexicon. In *International Conference on Computational Linguistics*, pages 268–272, Kyoto, JP, 1994. COLING.
- [44] J Hawkins. Explaining language universals. In J Hawkins, editor, *Explaining Language Universals*, pages 1–28. Basil Blackwell, Oxford, UK, 1988.
- [45] R Higginson. *Fixing-assimilation in language acquisition*. PhD thesis, Washington State University, 1985.

- [46] D Ingram. *First Language Acquisition*. Cambridge University Press, Cambridge, UK, 1989.
- [47] R Jackendoff. *X'Syntax: A study of Phrase Structure*. MIT Press, Cambridge, MA, 1977.
- [48] R Jackendoff. *Semantic Structures*. MIT Press, Cambridge, MA, 1990.
- [49] M Kanazawa. *Learnable classes of categorial grammars*. CSLI Publications & folli, Standford, CA, 1998.
- [50] R Kayne. *Parameters and Universals*. Oxford University Press, Oxford, UK, 2000.
- [51] A Korhonen. *Subcategorization Acquisition*. PhD thesis, University of Cambridge, 2002. Thesis published as Technical Report UCAM-CL-TR-530.
- [52] A Korhonen and Y Krymolowski. On the robustness of entropy-based similarity measures in evaluation of subcategorization acquisition systems. In *6th Conference on Natural Language Learning*, pages 91–97, Taipei, Taiwan, 2002. CoNLL-02.
- [53] A Kroch. Reflexes of grammar in patterns of language change. *Journal of Language Variation and Change*, 1:199–244, 1989.
- [54] B Landau and L Gleitman. *Language and Experience: evidence from the blind child*. Harvard University Press, Cambridge, MA, 1985.
- [55] L Lee. Measures of distributional similarity. In *37th Annual Meeting of the Association of Computational Linguistics*, pages 25–32, College Park, MD, 1999. ACL.
- [56] G Leech. 100 million words of English: the British National Corpus. *Language Research*, 28(1):1–13, 1992.
- [57] J Legate. Was the argument that was made empirical? Cambridge, ma, Massachusetts Institute of Technology MIT, 1999.
- [58] E Lenneberg. *Biological Foundations of Language*. Wiley Press, New York, NY, 1967.
- [59] B Levin. *English Verb Classes and Alternations*. Chicago University Press, Chicago, IL, 1993.
- [60] D Lightfoot. *Principles of Diachronic Syntax*. Cambridge University Press, Cambridge, UK, 1979.
- [61] J Lin. Divergence measures based on the Shannon-entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, 1991.
- [62] J Macnamara. *Names for Things: A Study of Human Learning*. MIT Press, Cambridge, MA., 1982.
- [63] B MacWhinney. *The CHILDES project: Tools for analysing talk*. Lawerance ErlBaum Associates, Hillsdale, NJ, second edition, 1995.
- [64] P Matthews. *Syntax*. Cambridge University Press, Cambridge, UK, 1981.

- [65] J McClelland and A Kawamoto. Mechanisms of sentence processing: Assigning roles to constituents of sentences. In D Rumelhart and J McClelland, editors, *Parallel Distributed Processing*, volume II, pages 318–362. MIT Press, Cambridge, MA, 1986.
- [66] A Meyers *et al.* Standardization of the complement adjunct distinction. New York University, Ms., 1994.
- [67] L Naigles. Children use syntax to learn verb meanings. *Journal of Child Language*, 17:357–374, 1990.
- [68] E Newport. Reduced input in the acquisition of signed languages: contributions to the study of creolization. In M DeGraff, editor, *Language Creation and Language Change*, pages 161–178. MIT Press, Cambridge, MA, 1999.
- [69] M Nice. Length of sentences as a criterion of child’s progress in speech. *Journal of Educational Psychology*, 16:370–9, 1925.
- [70] M Osbourne and E Briscoe. Learning stochastic categorial grammars. In *35th annual meeting of the Association of Computational Linguistics, CoNLL97 workshop*, Madrid, Spain, 1997. ACL.
- [71] J Piaget. *The construction of reality in the child*. Basic Books, New York, NY, 1954.
- [72] J Pine. The language of primary caregivers. In C. Gallaway and B. Richards, editors, *Input interaction and language acquisition*, pages 13–37. Cambridge University Press, Cambridge, UK, 1994.
- [73] S Pinker. *Language Learnability and Language Development*. Harvard University Press, Cambridge, MA, 1984.
- [74] S Pinker. The bootstrapping problem in language acquisition. In B MacWhinney, editor, *Mechanisms of language acquisition*. Erlbaum, Hillsdale, NJ, 1987.
- [75] S Pinker. *Learnability and Cognition*. MIT Press, Cambridge, MA, 1989.
- [76] S Pinker. How could a child use syntax to learn verb semantics. *Lingua*, 92(1–4):377–410, April 1994.
- [77] S Pinker. *The Language Instinct: How the Mind Creates Language*. Harper Collins, New York, NY, 1994.
- [78] S Pinker. *Words and Rules: The Ingredients of Language*. Basic Books, New York, NY, 1999.
- [79] S Pinker and A Prince. On language and connectionism: Analysis of a parallel distributed processing model of language acquisition. *Cognition*, 23:73–193, 1988.
- [80] C Pollard and I Sag. *Information-based syntax and semantics*, volume 1 of *CSLI Lecture Notes 13*. Centre for the Study of Language and Information, Stanford, CA, 1987.
- [81] K Post. *The language learning environment of laterborns in a rural Florida community*. PhD thesis, Harvard University, 1992.

- [82] G Pullum. How many possible human languages are there? *Linguistic Inquiry*, 14:447–467, 1983.
- [83] J Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.
- [84] D Roland and D Jurafsky. How verb subcategorization frequencies are affected by corpus choice. In *Proceedings of the 36th Annual Meeting of the Association of Computational Linguistics and the International Conference of Computational Linguistics*, pages 1117–1121, Montreal, Canada, 1998. COLING-ACL'98.
- [85] D Roland, D Jurafsky, L Menn, S Gahl, E Elder, and C Riddoch. Verb subcategorization frequency differences between business-news and balanced corpora. In *The 38th Annual Meeting of the Association of Computational Linguistics workshop on Comparing Corpora*, pages 28–34, Hong Kong, China, 2000. ACL.
- [86] D Rumelhart and J McClelland. *Computational Models of Cognition and Perception*, chapter On Learning Past Tenses of English Verbs, Chapt 18, pages 216–271. MIT Press, Cambridge, MA, 1986.
- [87] J Sachs. Talking about the there and then: The emergence of displaced reference in parent-child discourse. In K Nelson, editor, *Children's Language*, volume 4. Lawrence Erlbaum Associates, Hillsdale, NJ, 1983.
- [88] J Sachs and M Johnson. Language development in a hearing child of deaf parents. In von Raffler-Engel W and Lebrun Y, editors, *Baby Talk and Infant Speech*. Swets and Zeitlinger, Amsterdam, Holland, 1976.
- [89] W Sakas. *Ambiguity and the Computational Feasibility of Syntax Acquisition*. PhD thesis, City University of New York, 2000.
- [90] W Sakas and J Fodor. The structural triggers learner. In S Bertolo, editor, *Language Acquisition and Learnability*, chapter 5. Cambridge University Press, Cambridge, UK, 2001.
- [91] B Schieffelin. The acquisition of Kaluli. In D Slobin, editor, *The cross-linguistic study of language acquisition*, volume 1. Lawrence Erlbaum, Hillsdale, NJ, 1985.
- [92] B Schieffelin and E Ochs. Cultural perspectives on the transition from pre-linguistic to linguistic communication. In R Golinkoff, editor, *The transition from pre-linguistic to linguistic communication*. Lawrence Erlbaum., Hillsdale, NJ, 1983.
- [93] J Siskind. A computational study of cross situational techniques for learning word-to-meaning mappings. *Cognition*, 61(1–2):39–91, Nov/Oct 1996.
- [94] C Snow. Conversations with children. In P Fletcher and M Garman, editors, *Language Acquisition*, pages 363–375. Cambridge University Press, New York, NY, 2nd edition, 1986.
- [95] H Somers. On the validity of the complement-adjunct distinction in valency grammar. *Linguistics*, 22:507–520, 1984.

- [96] C Spearman. The proof and measurement of association between two things. *American Journal of Psychology*, 15:72–101, 1904.
- [97] M Steedman. Type-raising and directionality in combinatory grammar. In *26th Meeting of the Association for Computational Linguistics*, pages 71–78, Berkeley, CA, 1991. ACL.
- [98] M Steedman. *The Syntactic Process*. MIT Press/Bradford Books, Cambridge, MA, 2000.
- [99] P Suppes. The semantics of children’s language. *American Psychologist*, 29:103–114, 1974.
- [100] R Tang and R Mooney. Using multiple clause constructors in inductive logic programming for semantic parsing. In *12th European Conference on Machine Learning*, pages 466–477, Freiburg, Germany, 2001. ECML–2001.
- [101] A Thompson and R Mooney. Acquiring word-meaning mappings for natural language interface. *Journal of Artificial Intelligence*, 18, 2002.
- [102] V Valian. Logical and psychological constraints on the acquisition of syntax. In L Frazier and J Villiers, editors, *Language Processing and Language Acquisition*, pages 119–145. Kluwer, Dordrecht, Holland, 1990.
- [103] L Valiant. A theory of the learnable. *Communications of the ACM*, pages 1134–1142, 1984.
- [104] A Villavicencio. *The acquisition of a unification-based generalised categorial grammar*. PhD thesis, University of Cambridge, 2002.
- [105] B Waldron. Learning grammar from corpora. Master’s thesis, Cambridge University, 1999.
- [106] A Warren-Leubecker. *Sex differences in speech to children*. PhD thesis, Georgia Institute of Technology, 1982.
- [107] S Watkinson and S Manandhar. Unsupervised lexical learning with categorial grammars using the LLL corpus. In J Cussens and S Dzeroski, editors, *Learning Language in Logic*, pages 127–142. Springer, 2000.
- [108] K Wexler. Very early parameter setting and the unique checking constraint: A new explanation of the optional infinitive stage. *Lingua*, 106:23–79, 1998.
- [109] C Yang. *Knowledge and Learning in Natural Language*. Oxford University Press, Oxford, UK, 2002.
- [110] M Zelle and R Mooney. Learning semantic grammars with constructive inductive logic programming. In *11th National Conference on Artificial Intelligence*, pages 817–822, Washington, DC, 1993. AAAI–93.
- [111] M Zelle and R Mooney. Learning to parse database queries using inductive logic programming. In *14th National Conference on Artificial Intelligence*, pages 1050–1055, Portland, OR, 1996. AAAI–96.

- [112] L Zettlemoyer and M Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *21st Conference on Uncertainty in Artificial Intelligence*, Edinburgh, UK, 2005. UAI-05.