Carnegie Mellon University
Research Showcase @ CMU

Department of Psychology | Dietrich College of Humanities and Social Sciences

1-1-2009

Enriching CHILDES for Morphosyntactic Analysis

Brian MacWhinney
Carnegie Mellon University, macw@cmu.edu

Follow this and additional works at: http://repository.cmu.edu/psychology

Recommended Citation
MacWhinney, Brian, "Enriching CHILDES for Morphosyntactic Analysis" (2009). *Department of Psychology.* Paper 175.
http://repository.cmu.edu/psychology/175

This Book Chapter is brought to you for free and open access by the Dietrich College of Humanities and Social Sciences at Research Showcase @ CMU. It has been accepted for inclusion in Department of Psychology by an authorized administrator of Research Showcase @ CMU. For more information, please contact research-showcase@andrew.cmu.edu.

Enriching CHILDES for Morphosyntactic Analysis

Brian MacWhinney

Carnegie Mellon University

1. Introduction

The modern study of child language development owes much to the methodological and conceptual advances introduced by Brown (1973). In his study of the language development of Adam, Eve, and Sarah, Roger Brown focused on a variety of core measurement issues, such as acquisition sequence, growth curves, morpheme inventories, productivity analysis, grammar formulation, and sampling methodology. The basic question that Brown was trying to answer was how one could use transcripts of interactions between children and adults to test theoretical claims regarding the child's learning of grammar. Like many other child language researchers, Brown considered the utterances produced by children to be a remarkably rich data source for testing theoretical claims. At the same time, Brown realized that one needed to specify a highly systematic methodology for collecting and analyzing these spontaneous productions.

Language acquisition theory has advanced in many ways since Brown (1973), but we are still dealing with many of the same basic methodological issues he confronted. Elaborating on Brown's approach, researchers have formulated increasingly reliable methods for measuring the growth of grammar, or morphosyntax, in the child. These new approaches serve to extend Brown's vision into the modern world of computers

and computational linguistics. New methods for tagging parts of speech and grammatical relations now open up new and more powerful ways of testing hypotheses and models regarding children's language learning.

The current paper examines a particular approach to morphosyntactic analysis that has been elaborated in the context of the CHILDES (Child Language Data Exchange System) database. Readers unfamiliar with this database and its role in child language acquisition research may find it useful to download and study the materials (manuals, programs, and database) that are available for free over the web at http://childes.psy.cmu.edu. However, before doing this, users should read the "Ground Rules" for proper usage of the system. This database now contains over 44 million spoken words from 28 different languages. In fact, CHILDES is the largest corpus of conversational spoken language data currently in existence. In terms of size, the next largest collection of conversational data is the British National Corpus with 5 million words. What makes CHILDES a single corpus is the fact that all of the data in the system are consistently coded using a single transcript format called CHAT. Moreover, for several languages, all of the corpora have been tagged for part of speech using an automatic tagging program called MOR.

When Catherine Snow and I proposed the formation of the CHILDES database in 1984, we envisioned the construction of a large corpus base would allow child language researchers to improve the empirical grounding of their analyses. In fact, the overwhelming majority of new studies of the development of grammatical production rely on the programs and data in the CHILDES database. In 2002, we conducted a review of articles based on the use of the database and found that more

than 2000 articles had used the data and/or programs. The fact that CHILDES has had this effect on the field is enormously gratifying to all of us who have worked to build the database. At the same time, the quality and size of the database constitutes a testimony to the collegiality of the many researchers in child language who have contributed their data for the use of future generations.

For the future, our goal is to build on these successful uses of the database to promote even more high quality transcription, analysis, and research. In order to move in this direction, it is important for the research community to understand why we have devoted so much attention to the improvement of morphosyntactic coding in CHILDES. To communicate effectively regarding this new morphosyntactic coding, we need to address the interests of three very different types of readers. Some readers are already very familiar with CHILDES and have perhaps already worked with the development and use of tools like MOR and POST. For these readers, this chapter is designed to highlight problematic areas in morphosyntactic coding and areas of new activity. It is perhaps a good idea to warn this group of readers that there have been major improvements in the programs and database over the last ten years. As a result, commands that worked with an earlier version of the programs will no longer work in the same way. It is a good idea for all active researchers to use this chapter as a way of refreshing their understanding of the CHILDES and TalkBank tools

A second group of readers will have extensive background in computational methods, but little familiarity with the CHILDES corpus. For these readers, this chapter is an introduction to the possibilities that CHILDES offers for the development of new computational approaches and analyses. Finally, for child language researchers who

are not yet familiar with the use of CHILDES for studying grammatical development, this chapter should be approached as an introduction to the possibilities that are now available. Readers in this last group will find some of the sections rather technical. Beginning users do not need to master all of these technical details at once. Instead, they should just approach the chapter as an introduction to possible modes of analysis that they may wish to use some time in the future.

Before embarking on our review of computational tools in CHILDES, it is helpful to review briefly the ways in which researchers have come to use transcripts to study morphosyntactic development. When Brown collected his corpora back in the 1960s, the application of generative grammar to language development was in its infancy. However, throughout the 1980s and 1990s (Chomsky & Lasnik, 1993), linguistic theory developed increasingly specific proposals about how the facts of child language learning could illuminate the shape of Universal Grammar. At the same time, learning theorists were developing increasingly powerful methods for extracting linguistic patterns from input data. Some of these new methods relied on distributed neural networks (Rumelhart & McClelland, 1986), but others focused more on the ways in which children can pick up a wide variety of patterns in terms of relative cue validity (MacWhinney, 1987).

These two very different research traditions have each assigned a pivotal role to the acquisition of morphosyntax in illuminating core issues in learning and development. Generativist theories have emphasized issues such as: the role of triggers in the early setting of a parameter for subject omission (Hyams & Wexler, 1993), evidence for advanced early syntactic competence (Wexler, 1998), evidence for early absence

functional categories that attach to the IP node (Radford, 1990), the role of optional infinitives in normal and disordered acquisition (Rice, 1997), and the child's ability to process syntax without any exposure to relevant data (Crain, 1991). Generativists have sometimes been criticized for paying inadequate attention to the empirical patterns of distribution in children's productions. However, work by researchers, such as Stromswold (1994), van Kampen (1998), and Meisel (1986), demonstrates the important role that transcript data can play in evaluating alternative generative accounts.

Learning theorists have placed an even greater emphasis on the use of transcripts for understanding morphosyntactic development. Neural network models have shown how cue validities can determine the sequence of acquisition for both morphological (MacWhinney & Leinbach, 1991; MacWhinney, Leinbach, Taraban, & McDonald, 1989; Plunkett & Marchman, 1991) and syntactic (Elman, 1993; Mintz, Newport, & Bever, 2002; Siskind, 1999) development. This work derives further support from a broad movement within linguistics toward a focus on data-driven models (Bybee & Hopper, 2001) for understanding language learning and structure. These accounts formulate accounts that view constructions (Tomasello, 2003) and item-based patterns (MacWhinney, 1975) as the loci for statistical learning.

2. Analysis by Transcript Scanning

Although the CHILDES Project has succeeded in many ways, it has not yet provided a complete set of computational linguistic tools for the study of morphosyntactic development. In order to conduct serious corpus-based research on the development of morphosyntax, users will want to supplement corpora with tags that identify the

morphological and syntactic status of every morpheme in the corpus. Without these tags, researchers who want to track the development of specific word forms or syntactic structures are forced to work with a methodology that is not much more advanced than that used by Brown in the 1960s. In those days, researchers looking for the occurrence of a particular morphosyntactic structure, such as auxiliary fronting in yes-no questions, would have to simply scan through entire transcripts and mark occurrences in the margins of the paper copy using a red pencil. With the advent of the personal computer in the 1980s, the marks in the margins were replaced by codes entered on a %syn (syntactic structure) or %mor (morphological analysis with parts of speech) coding tier. However, it was still necessary to pour over the full transcripts line by line to locate occurrences of the relevant target forms.

3. Analysis by Lexical Tracking

If a researcher is clever, there are ways to convince the computer to help out in this exhausting process of transcript scanning. An easy first step is to download the CLAN programs from the CHILDES website at http://childes.psy.cmu.edu. These programs provide several methods for tracing patterns within and between words. For example, if you are interested in studying the learning of English verb morphology, you can create a file containing all the irregular past tense verbs of English, as listed in the CHILDES manual. After typing all of these words into a file and then naming that file something like irreg.cut, you can use the CLAN program called KWAL with the +s@irreg.cut switch to locate all the occurrences of irregular past tense forms. Or, if you only want a frequency count, you can run FREQ with the same switch to get a frequency count of the various forms and the overall frequency of irregulars.

Although this type of search is very helpful, you will also want to be able to search for

overregularizations and overmarkings such as "*ranned", "*runned", "*goed", or "*jumpeded". Unless these are already specially marked in the transcripts, the only way to locate these forms is to create an even bigger list with all possible overmarkings. This is possible for the common irregular overmarkings, but doing this for all overmarked regulars, such as "*playeded", is not really possible. Finally, you also want to locate all correctly marked regular verbs. Here, again, making the search list is a difficult matter. You can search for all words ending in –ed, but you will have to cull out from this list forms like "bed", "moped", and "sled". A good illustration of research based on generalized lexical searches of this type can be found in the study of English verb learning by Marcus et al. (1992).

Or, to take another example, suppose you would like to trace the learning of auxiliary fronting in yes-no questions. For this, you would need to create a list of possible English auxiliaries to be included in a file called aux.cut. Using this, you could easily find all sentences with auxiliaries and then write out these sentences to a file for further analysis. However, only a minority of these sentences will involve yes-no questions. Thus, to further sharpen your analysis, you would want to further limit the search to sentences in which the auxiliary begins the utterance. To do this, you would need to dig carefully through the electronic version of the CHILDES manual to find the ways in which to use the COMBO program to compose search strings that include markers for the beginnings and ends of sentences. Also, you may wish to separate out sentences in which the auxiliary is moved to follow a wh-word. Here, again, you can compose a complicated COMBO search string that looks for a list of possible initial interrogative or "wh" words, followed by a list of possible auxiliaries. Although such searches are possible, they tend to be difficult, slow, and prone to error. Clearly, it

would be better if the searches could examine not strings of words, but rather strings of morphosyntactic categories. For example, we would be able to trace sentences with initial wh-words followed by auxiliaries by just looking for the pattern of "int + aux". However, in order to perform such searches, we must first tag our corpora for the relevant morphosyntactic features. The current article explains how this is done.

4. Measures of Morphosyntactic Development

The tagging of corpora is crucial not only for research on morphosyntactic development, but also for the development of automatic ways of evaluating children's level of grammatical development. Let us consider four common measures of grammatical development: MLU, VOCD, DSS, and IPSyn.

MLU. The computation of the mean length of utterance (MLU) is usually governed by a set of nine detailed criteria specified by Brown (1973). These nine criteria are discussed in detail in the CLAN manual (http://childes.psy.cmu.edu/manuals/clan.pdf) under the heading for the MLU program. When Brown applied these criteria to his typewritten corpora in the 1960s, he and his assistants actually looked at every word in the transcript and applied each of the nine criteria to each word in each sentence in order to compute MLU. Of course, they figured out how to do this very quickly, since they all had important dissertations to write. Four decades later, relying on computerized corpora and search algorithms, we can compute MLU automatically in seconds. However, to do this accurately, we must first code the data in ways that allow the computer to apply the criteria correctly. The SALT program (Miller & Chapman, 1983) achieves this effect by dividing words directly into their component morphemes on the main line, as in "boot-s" or "boot-3S" for "boots". Earlier versions

of CHILDES followed this same method, but we soon discovered that transcribers were not applying these codes consistently. Instead, they were producing high levels of error that impacted not only the computation of MLU, but also the overall value of the database. Beginning in 1998, we removed all main line segmentations from the database and began instead to rely on the computation of MLU from the %mor line, as we will discuss in detail later. To further control accurate computation of MLU, we rely on symbols like &text, xxx, and certain postcodes to exclude material from the count. Researchers wishing to compute MLU in words, rather than morphemes, can perform this computation from either the main line or the %mor line.

VOCD. Beginning in 1997 (Malvern & Richards, 1997) and culminating with a book-length publication in 2004 (Malvern, Richards, Chipere, & Purán, 2004), Malvern and colleagues introduced a new measure called VOCD (VOCabulary Diversity) as a replacement for the earlier concept of a type/token ratio (TTR). The TTR is a simple ratio of the types of words user by a speaker in a transcript over the total number of words in the transcript. For example, if the child uses 30 different words and produces a total output of 120 words, then the TTR is 30/120 or .25. The problem with the TTR is that it is too sensitive to sample size. Small transcripts often have inaccurately high TTR ratios, simply because they are not big enough to allow for word repetitions. VOCD corrects this problem statistically for all but the smallest samples (for details, see the CLAN manual). Like MLU, one can compute VOCD either from the main line or the %mor line. However, the goal of both TTR and VOCD is to measure lexical diversity. For such analyses, it is not appropriate to treat variant inflected forms of the same base as different. To avoid this problem, one can

now compute VOCD from the %mor line using the +m switch to control the filtering of affixes.

DSS. A third common measure of morphosyntactic development is the DSS (Developmental Sentence Score) from Lee (1974). This score is computed through a checklist system for tracking children's acquisition of a variety of syntactic structures, ranging from tag questions to auxiliary placement. By summing scores across a wide range of structures by hand, researchers can compute an overall developmental sentence score or DSS. However, using the %mor line in CHAT files and the CLAN DSS program, it is now possible to compute DSS automatically. After automatic running of DSS, there may be a few remaining codes that will have to be judged by eye. This requires a second "interactive" pass where these remaining issues are resolved. The DSS measure has also been adapted for use in Japanese, where it forms an important part of the Kokusai project, organized by Susanne Miyata.

IPSyn. Scarborough (1990) introduced another checklist measure of syntactic growth called IPSyn (Index of Productive Syntax). Although IPSyn overlaps with DSS in many dimensions, it is easier for human coders to compute and it places a bit more emphasis on syntactic, rather than morphological structures. IPSyn has gained much popularity in recent years, being used in at least 70 published studies. Unlike the DSS, correct coding of IPSyn requires attention to not only the parts of speech of lexical items, but also the grammatical relation between words. In order to automate the computation of IPSyn, Sagae, Lavie, & MacWhinney (2005) built a program that uses information on the %syn and %mor lines in a CHAT file or a collection of CHAT files. Sagae et al. were able to show that the accuracy rate for this automatic

IPSyn is very close to the 95% accuracy level achieved by the best human coders and significantly better than the lexically-based method developed used in the Computerized Profiling system (Long & Channell, 2001).

5. Generative Frameworks

We have discussed the ways in which researchers have used CHILDES corpora to study the acquisition of particular constructions and the ways in which they use corpora to compute measures of morphosyntactic development. There is a third approach to morphosyntactic analysis that is important to the field, but currently unsupported by the CLAN programs. This is the use of the database to construct and validate full generative grammars. Here, the term "generative" refers not just to grammars in the tradition of Chomsky (1957; 1965; 1995), but rather to any computational system that manages to generate all the sentences that would be accepted by the child and none that would be outside the child's range of production. A generative grammar of this type can function as both a generator and a recognizer or parser. As a generator, it would produce sentences of the types found in the CHILDES database. As a recognizer, it would assign grammatical relations to the words in the actual utterances produced by children in the CHILDES database. Wintner, MacWhinney, & Lavie (2007) provide a first attempt to provide a grammar at this level of detail, using HPSG-like type hierarchies. However, a generative grammar could be implemented through a variety of formalisms, including rewrite rules (Culicover & Jackendoff, 2005), item-based patterns (MacWhinney, 2005), constraint hierarchies (Sells, 2001), neural networks (Miikkulainen & Mayberry, 1999), or parser-generators (Hausser, 1999). In each case, the goal would be to use

the grammar to generate or parse all of the sentences in the transcripts without producing or recognizing sentences that would not be possible in the child's idiolect.

In the typical case, we would want to construct these grammars on the basis of induction from samples of the data. For grammars based on minimalist theory, the induction could be based on parameter setting, either discrete (Hyams, 1986) or driven by statistical analysis (Buttery, 2004; Pearl, 2005). For grammars based on statistical learning, the induction might involve unsupervised linking of words without tags (Edelman, Solan, Horn, & Ruppin, 2004; Stolcke & Omohundro, 1994). For item-based grammar learning systems (Gobet & Pine, 1997; Pine & Lieven, 1997), learning involves tracking local binary pair combinations. For all of these systems, it would be "cheating" to rely on the categories and grammatical relations found in the %mor and %syn lines. However, once the induction is completed, it is crucial to use this information as the "gold standard" for evaluation. For example, the most recent CoNNL (Computational Natural Language Learning) Shared Task competition includes a component that focuses on the construction of parsers for the English corpora in the CHILDES database. In this shared task, parsing systems compete both in terms of their ability to learn on labelled (tagged) and unlabelled (untagged) data.

In summary then, we have seen that a wide range of approaches to morphosyntactic development, measurement, and grammar induction all depend heavily on annotated information regarding part of speech and grammatical relations. Because of the centrality of these codes to the research community, we have devoted years of work to building up systems for automatic coding of morphosyntax. This paper explains the

current state of this work and reviews the many detailed coding decisions that are needed to achieve accurate tagging of child language corpora.

6. Analysis based on automatic morphosyntactic coding

So far, our analysis has examined how researchers can use the database through transcript scanning and lexical scanning. However, often these methods are inadequate for addressing broader and more complex issues such as detailed syntactic analysis or the comparisons and evaluations of full generative frameworks. To address these more complex issues, the CHILDES system now provides full support for analysis based on automatic morphosyntactic coding. The core programs used in this work are MOR, POST, and GRASP.

The initial morphological tagging of the CHILDES database relies on the application of the MOR program. Running MOR on a CHAT file is easy. In the simplest case, it involves nothing much more than a one-line command. However, before discussing the mechanics of MOR, let us take a look at what it produces. To give an example of the results of a MOR analysis for English, consider this sentence from eve15.cha in Roger Brown's corpus for Eve.

*CHI: oop I spilled it .

%mor: int|loop pro|I v|spill-PAST pro|it .

Here, the main line gives the child's production and the %mor line gives the part of speech for each word, along with the morphological analysis of affixes, such as the past tense mark (-PAST) on the verb. The %mor lines in these files were not created by hand. To produce them, we ran the MOR program, using the MOR grammar for English, which can be downloaded from http://childes.psy.cmu.edu/morgrams/. \

The command for running MOR, which is described in detail in the CLAN manual, is nothing more in this case than “mor *.cha”. After running MOR, the file looks like this:

*CHI: oop I spilled it .

%mor: int|loop pro|I part|spill-PERF^v|spill-PAST pro|it .

Notice that the word “spilled” is initially ambiguous between the past tense and participle readings. To resolve such ambiguities, we run a program called POST. Running POST for English is also simple. The command is just “post *.cha” After POST has been run, the sentence is then “disambiguated.” Using this disambiguated form, we can then run the GRASP program, which is currently a separate program available from the CHILDES website, to create the representation given in the %xsyn line below:

*CHI: oop I spilled it .

%mor: int|loop pro|I v|spill-PAST pro|it .

%xsyn: 1|3|JCT 2|3|SUBJ 3|0|ROOT 4|3|OBJ 5|3|PUNCT

In this %xsyn line, we see that the second word “I” is related to the verb (“spilled”) through the grammatical relation (GR) of Subject. The fourth word “it” is related to the verb through the grammatical relation of Object.

Using GRASP, we have recently inserted dependency grammar tags for all of these grammatical relations in the Eve corpus. In tests run on the Eve corpus, 94% of the tags were assigned accurately (Sagae, Davis, Lavie, MacWhinney, & Wintner, 2007). A further test of GRASP on the Adam corpus also yielded an accuracy level of 94%. For both of these corpora, grammatical relations were mistagged 6% of the time. It is

likely that, over time, this level of accuracy will improve, although we would never expect 100% accuracy for any tagging program. In fact, only a few human taggers can achieve 94% accuracy in their first pass tagging of a corpus.

The work of building MOR, POST, and GRASP has been supported by a number of people. Mitzi Morris built MOR in 1997, using design specifications from Roland Hausser. Since 2000, Leonid Spektor has extended MOR in many ways. Christophe Parisse built POST and POSTTRAIN (Parisse & Le Normand, 2000) and continues to maintain and refine them. Kenji Sagae built GRASP as a part of his dissertation work for the Language Technologies Institute at Carnegie Mellon University (Sagae, MacWhinney, & Lavie, 2004a, 2004b). GRASP was then applied in detail to the Eve and Adam corpus by Eric Davis and Shuly Wintner.

These initial experiments with GRASP and computational modelling of grammatical development in the CHILDES corpora underscore the increasing importance of methods from computational linguistics for the analysis of child language data. Together with statistical computational analyses (Edelman, Solan, Horn, & Ruppin, 2004) and neural network analyses (Li, Zhao, & MacWhinney, 2007), we should expect to see increasing input from computational linguistics, as the morphosyntactic tagging of the CHILDES database becomes increasingly refined and accurate.

6.1. MOR and FST

We have now constructed MOR grammars for 10 languages, including Cantonese, Dutch, English, French, German, Hebrew, Japanese, Italian, Putonghua, and Spanish. These grammars can be downloaded from http://childes.psy.cmu.edu/morgrams/ The grammars for Dutch and German are still in the development phase and have not yet

been applied systematically to CHILDES corpora. However, the grammars for the other seven languages are all well advanced and have been used to process all or most of the transcripts for the relevant languages. In this section, we will focus on the construction of the MOR grammar for English, taking up the issue of MOR for other languages in the next section.

Before looking closely at MOR, it may be a good idea to situate the status of this program within the larger framework of computational approaches to morphological tagging. This is an area where there has been serious computational and linguistic work for several decades, much of it culminating in the formulation of an approach based on the use of finite state transition (FST) networks based on the two-level morphology of Koskenniemi (1983), but later modified to fit into the general framework of finite state automata. Software called XFST for building and running FSM (finite state morphology) taggers has been made available from Xerox-Parc (http://www.fsmbook.com) and a detailed description of this work was published by Beasley & Karttunen (2003). Although this framework is extremely solid in computational terms, it has several limitations for use in the context of the CHILDES Project.

1. Although XFST has been used to develop taggers for many languages, nearly all of this work is unavailable for general academic use, either because it is based on commercial applications or because it is subject to various licensing restrictions.
2. The FSM framework does not have facilities that allow it to take into consideration the various special codings needed for spoken language transcription in CHAT.

3. XFST systems tend to overgenerate and it is not always easy to see how to constrain this overgeneration.
4. Learning to build taggers in XFST requires good training in programming. Building MOR taggers requires careful attention to linguistic theory, feeding-bleeding relations, and categories, but no special ability in programming.
5. As Wintner (2007) argues, the greater memory capacity and speed of modern computers makes it feasible to approach morphological analysis through generation (the approach taken in MOR), rather than being limited to recognition (the approach taken in XFST).
6. MOR invokes a clearer separation of the lexical resources from the analysis programs than does XFST. This is quite important for maintenance and extension of the analyzer.
7. XFST does not provide as clear debugging information as does MOR.

For these various reasons, we have decided to continue our development of MOR grammars for the languages in CHILDES, rather than shifting to work with the more prominent and better-documented XFST system.

6.2. Understanding MOR

When MOR runs, it breaks up morphemes into their component parts. In a relatively analytic language like English, many words require no analysis at all. However, even in English, a word like "coworkers" can be seen to contain four component morphemes, including the prefix "co", the stem, the agential suffix, and the plural. For this form, MOR will produce the analysis: co#n:v|work-AGT-PL. This representation uses the symbols # and – to separate the four different morphemes. Here, the prefix stands at the beginning of the analysis, followed by the stem (n|work), and the two suffixes. In general, stems always have the form of a part of speech

category, such as “n” for noun, followed by the vertical bar and then a statement of the stem’s lexical form.

In order to understand the functioning of the MOR grammar for English, the best place to begin is with a tour of the files inside the /english folder that you can download from the server. At the top level, you will see these files:

1. ar.cut – These are the rules that generate allomorphic variants from stems.
2. cr.cut – These are the rules that specify the possible combinations of morphemes going from left to right in an English word.
3. debug.cdc – This file holds the complete trace of an analysis of a given word by MOR. It always holds the results of the most recent analysis. It is mostly useful for people who are developing new ar.cut or cr.cut files as a way of tracing out or debugging problems with these rules.
4. docs – This is a folder containing a file of instructions on how to train POST and a list of tags and categories used in the English grammar.
5. eng.db – This is a file used by POST and should be left untouched.
6. ex.cut – This file includes analyses that are being “overgenerated” by MOR and should simply be filtered out or excluded whenever they occur.
7. lex – This is the heart of the MOR grammar. We will examine it in greater detail below.
8. posttags.cut – This is a list of the grammatical tags of English that should be included whenever running POST.
9. sf.cut – This file tells MOR how to deal with words that end with certain special form markers such as @b for babbling.

10. traintags.cut – This is a list of the tags that are used by POSTTRAIN when it creates the eng.db database.

(When examining these files and others, please note that the exact shapes of the files, the word listings, and the rules are sure to change over time. We recommend that users open up these various files to understand their contents. However, over time, the contents will diverge more and more from the names and examples given here. Still, it should be possible to trace through the same basic principles, even given these inevitable changes.)

We will study the ar.cut, cr.cut, and sf.cut files in more detail later. For now, let us take a look at the many files contained inside the /lex folder. Here, we find 72 files that break out the possible words of English into different files for each specific part of speech or compound structure. Because these distinctions are so important to the correct transcription of child language and the correct running of MOR, it is worthwhile to consider the contents of each of these various files. As the following table shows, about half of these word types involve different part of speech configurations within compounds. This analysis of compounds into their part of speech components is intended to further study of the child's learning of compounds as well as to provide good information regarding the part of speech of the whole. The name of the compound files indicates their composition. For example the name adj+n+adj.cut indicates compounds with a noun followed by an adjective (n+adj) whose overall function is that of an adjective. In English, the part of speech of a compound is usually the same as that of the last component of the compound.

File	Function	Example
0affix.cut	prefixes and suffixes	see expanded list below
0uk.cut	terms local to the UK	fave, doofer, sixpence
adj-dup.cut	baby talk doubles	nice+nice, pink+pink
adj-ir.cut	irregular adjectives	better, furthest
adj-kidy.cut	adjectives with babytalk –y	bunchy, eaty, crawly
adj.cut	regular adjectives	tall, redundant
adj+adj+adj.cut	compounds	half+hearted, hot+crossed
adj+adj+adj(on).cut	compounds	super+duper, easy+peasy
adj+n+adj.cut	compounds	dog+eared, stir+crazy
adj+v+prep+n.cut	compounds	pay+per+view
adj+v+v.cut	compounds	make+believe, see+through
adv-int.cut	intensifying adverbs	really, plumb, quite
adv-loc.cut	locative adverbs	north, upstairs
adv-tem.cut	temporal adverbs	tomorrow, tonight, anytime
adv.cut	regular adverbs	ajar, fast, mostly
adv+adj+adv.cut	compounds	half+off, slant+wise
adv+adj+n.cut	compounds	half+way, off+shore
adv+n+prep+n.cut	compounds	face+to+face
auxil.cut	auxiliaries and modals	should, can, are
co-cant.cut	Cantonese bilngual forms	wo, wai, la
co-voc.cut	vocative communicators	honey, dear, sir
co.cut	regular communicators	blah, bybye, gah, no
conj.cut	conjunctions	and, although, because
det.cut	deictic determiners	this, that, the,
fil.cut	fillers	um, uh, er
int-rhymes.cut	rhymes as interjections	fee_figh_foe_fum
int.cut	interjections	farewell, boohoo, hubba
int+int+int.cut	compounds	good+afternoon
int+int+int+int.cut	compounds	ready+set+go
n-abbrev.cut	abbreviations	c_d, t_v, w_c
n-baby.cut	babytalk forms	passie, wawa, booboo
n-dashed.cut	non-compound combinations	cul_de_sac, seven_up
n-dup.cut	duplicate nouns	cow+cow, chick_chick
n-ir.cut	irregular nouns	children, cacti, teeth
n-loan.cut	loan words	goyim, amigo, smuck
n-pluraletant.cut	nouns with no singular	golashes, kinesics, scissors
n.cut	regular nouns	dog, corner, window
n+adj+n.cut	compounds	big+shot, cutie+pie
n+adj+v+adj.cut	compounds	merry+go+round
n+n+conj+n.cut	compounds	four+by+four, dot+to+dot
n+n+n-on.cut	compounds	quack+duck, moo+cow
n+n+n.cut	compounds	candy+bar, foot+race
n+n+novel.cut	compounds	children+bed, dog+fish
n+n+prep+det+n.cut	compounds	corn+on+the+cob
n+on+on-baby.cut	compounds	wee+wee, meow+meow
n+v+x+n.cut	compounds	jump+over+hand

n+v+n.cut	compounds	squirm+worm, snap+bead
n+v+ptl.cut	compounds	chin+up, hide+out
num-ord.cut	ordinals	fourth, thirteenth
num.cut	cardinals	five, twenty
on.cut	onomatopoeia	boom, choo_choo
on+on+on.cut	compounds	cluck+cluck, knock+knock
prep.cut	prepositions	under, minus
pro.cut	pronouns	he, nobody, himself
ptl.cut	verbal particle	up, about, on
quan.cut	quantifier	some, all, only, most
small.cut	assorted forms	not, to, xxx, yyy
v-baby.cut	baby verbs	wee, poo
v-clit.cut	cliticized forms	gonna, looka
v-dup.cut	verb duplications	eat+eat, drip+drip
v-ir.cut	irregular verbs	came, beset, slept
v.cut	regular verbs	run, take, remember
v+adj+v.cut	compounds	deep+fry, tippy+toe
v+n+v.cut	compounds	bunny+hop, sleep+walk
v+v+conj+v.cut	compounds	hide+and+seek
wh.cut	interrogatives	which, how, why
zero.cut	omitted words	0know, 0conj, 0is

The construction of these lexicon files involves a variety of decisions. Here are some of the most important issues to consider.

1. Words may often appear in several files. For example, virtually every noun in English can also function as a verb. However, when this function is indicated by a suffix, as in “milking” the noun can be recognized as a verb through a process of morphological derivation contained in a rule in the cr.cut file. In such cases, it is not necessary to list the word as a verb. Of course, this process fails for unmarked verbs. However, it is generally not a good idea to represent all nouns as verbs, since this tends to overgenerate ambiguity. Instead, it is possible to use the POSTMORTEM program to detect cases where nouns are functioning as bare verbs.
2. If a word can be analyzed morphologically, it should not be given a full listing. For example, since “coworker” can be analyzed by MOR into three morphemes as co#n:v|work-AGT, it should not be separately listed in the n.cut

file. If it is, then POST will not be able to distinguish co#n:v|work-AGT from n|coworker.

3. In the zero.cut file, possible omitted words are listed without the preceding 0. For example, there is an entry for “conj” and “the”. However, in the transcript, these would be represented as “0conj” and “0the”.
4. It is always best to use spaces to break up word sequences that are really just combinations of words. For example, instead of transcribing 1964 as “nineteen+sixty+four”, “nineteen-sixty-four”, or “nineteen_sixty_four”, it is best to transcribe simply as “nineteen sixty four”. This principle is particularly important for Chinese, where there is a tendency to underutilize spaces, since Chinese itself is written without spaces.
5. For most languages that use Roman characters, you can rely on capitalization to force MOR to treat words as proper nouns. To understand this, take a look at the forms in the sf.cut file at the top of the MOR directory. These various entries tell MOR how to process forms like k@l for the letter “k” or John_Paul_Jones for the famous admiral. The symbol \c indicates that a form is capitalized and the symbol \l indicates that it is lowercase.
6. Deciding how to represent compounds is a difficult matter. See the discussion in the next section.

6.3. Compounds and Complex Forms

The initial formulations of CHAT that were published in 1991 (1991), 1995 (1995), and 2000 (2000) specified no guidelines for the annotation of compounds. They only stipulated that compounds should be represented with a plus. When MOR saw a word with a plus, it simply tagged it as a noun compound. This was a big mistake, since many of the compounds tagged in this way were not common nouns. Instead, they

included verbs, adjectives, proper nouns, idioms, greetings, onomatopoeia, and many other nonstandard forms. Unfortunately, once this genie had been let out of the bottle, it was very difficult to convince it to go back in. To solve this problem, we had to shift from blanket recognition of compounds to an exhaustive listing of the actual forms of possible compounds. The result of this shift is that we have now created many special compound files such as n+n+n.cut or v+n+v.cut. Fixing the forms in the database to correspond to this new, tighter standard was a huge job, perhaps even more tedious than that involved in removing main line morphemicization from the corpus. However, now that we have a full analysis of compounds, there is a much more accurate analysis of children's learning of these forms.

In the current system, compounds are listed in the lexical files according to both their overall part of speech (X-bar) and the parts of speech of their components. However, there are seven types of complex word combinations that should not be treated as compounds.

1. **Underscored words**. The n-dashed.cut file includes 40 forms that resemble compounds, but are best viewed as units with non-morphemic components. For example, kool_aid and band_aid are not really combinations of morphemes, although they clearly have two components. The same is true for hi_fi and coca_cola. In general, MOR and CLAN pay little attention to the underscore character, so it can be used as needed when a plus for compounding is not appropriate. The underscore mark is particularly useful for representing the combinations of words found in proper nouns such as John_Paul_Jones, Columbia_University, or The_Beauty_and_the_Beast. As long as these words are capitalized, they do not need to be included in the

MOR lexicon, since all capitalized words are taken as proper nouns in English. However, these forms cannot contain pluses, since compounds are not proper nouns. And please be careful not to overuse this form.

2. **Separate words**. Many noun-noun combinations in English should just be written out as separate words. An example would be "faucet stem assembly rubber gasket holder". We don't want to write this as "Faucet_stem_assembly_rubber_gasket_holder" or "faucet_stem_assembly_rubber_gasket_holder" or even "faucet+stem+assembly+rubber+gasket+holder". It is worth noting here that German treats all such forms as single words. This means that different conventions have to be adopted for German in order to avoid the need for exhaustive listing of the infinite number of German compound nouns.
3. **Spelling sequences**. Sequences of letter names such as "O-U-T" for the spelling of "out" are transcribed with the suffix @k, as in out@k.
4. **Acronyms**. Forms such as FBI are transcribed with underscores, as in F_B_I. Presence of the initial capital letter tells MOR to treat F_B_I as a proper noun. This same format is used for non-proper abbreviations such as c_d or d_v_d.
5. **Products**. Coming up with good forms for commercial products such as Coca-Cola is tricky. Because of the need to ban the use of the dash on the main line, we have avoided the use of the dash in these names. It is clear that they should not be compounds, as in coca+cola, and compounds cannot be capitalized, so Coca+Cola is not possible. This leaves us with the option of either coca_cola or Coca_Cola. The option coca_cola seems best, since this is not really a proper noun.

6. **Interjections**. The choice between underscoring, compounding, and writing as single words is particularly tricky for interjections and set phrases. A careful study of files such as co-voc.cut, co.cut, n-dashed.cut, n-abbrev.cut, int-rhymes.cut, int.cut, inti+int+int.cut, and int+int+int.cut will show how difficult it is to apply these distinctions consistently. We continue to sharpen these distinctions, so the best way to trace these categories is to scan through the relevant files to see the principles that are being used to separate forms into these various types.
7. **Babbling and word play**. In earlier versions of CHAT and MOR, transcribers often represent sequences of babbling or word play syllables as compounds. This was done mostly because the plus provides a nice way of separating out the separate syllables in these productions. In order to make it clear that these separations are simply marked for purposes of syllabification, we now ask transcribers to use forms such as ba^ba^ga^ga@wp or choo^bung^choo^bung@o to represent these patterns.

The introduction of this more precise system for transcription of complex forms opens up additional options for programs like MLU, KWAL, FREQ, and GRASP. For MLU, compounds will be counted as single words, unless the plus sign is added to the morpheme delimiter set using the +b+ option switch. For GRASP, processing of compounds only needs to look at the part of speech of the compound as a whole, since the internal composition of the compound is not relevant to the syntax. Additionally, forms such as "faucet handle valve washer assembly" do not need to be treated as compounds, since GRASP can learn to treat sequences of nouns as complex phrases header by the final noun.

6.4. Lemmatization

Researchers are often interested in computing frequency profiles that are computed using lemmas or root forms, rather inflected forms. For example, they may want to treat "dropped" as an instance of the use of the lemma "drop." In order to perform these types of computations, the KWAL and FREQ programs provide a series of options that allow users to refer to various parts of complex structures in the %mor line. This system recognizes the following structures on the %mor line:

Element	Symbol	Example	Representation	Part
prefix	#	unwinding	un#v|wind-PROG	un#
stem	r	unwinding	un#v|wind-PROG	wind
suffix	-	unwinding	un#v|wind-PROG	PROG
fusion	&	unwound	un#v|wind&PAST	PAST
translation	=	gato	n|gato=cat	cat
other	o	-	-	-

To illustrate the use of these symbols, let us look at several possible commands. All of these commands take the form: freq +t%mor -t* filename.cha. However, in addition, they add the +s switches that are given in the second column. In these commands, the asterisk is used to distinguish across forms in the frequency count and the % sign is used to combine across forms.

Function	String
All stems with their parts of speech, merge the rest	+s@"r+*,|+*,o+%"
Only verbs	+s@"|+v"
All forms of the stem "go"	+s@"r+go"
The different parts of speech of the stem "go"	+s@"r+go,|+*,o+%"
The stem "go" only when it is a verb	+s@"r+go,|+v,o+%"
All stems, merge the rest	+s@"r+*,o+%"

Of these various forms, the last one given above would be the one required for conducting a frequency count based on lemmas or stems alone. Essentially CLAN

breaks every element on %mor tier into its individual components and then matches either literal strings or wild cards provided by the user to each component.

6.5. Errors and Replacements

Transcriptions on the main line have to serve two, sometimes conflicting (Edwards, 1992), functions. On the one hand, they need to represent the form of the speech as actually produced. On the other hand, they need to provide input that can be used for morphosyntactic analysis. When words are pronounced in their standard form, these two functions are in alignment. However, when words are pronounced with phonological or morphological errors, it is important to separate out the actual production from the morphological target. This can be done through a system of main line tagging of errors. This system largely replaces the coding of errors on a separate %err line, although that form is still available, if needed. The form of the newer system is illustrated here:

CHI: him [case] ated [: ate] [* +ed-sup] a f(l)ower and a pun [: bun].

For the first error, there is no need to provide a replacement, since MOR can process "him" as a pronoun. However, since the second error is not a real word form, the replacement is necessary in order to tell MOR how to process the form. The third error is just an omission of "l" from the cluster and the final error is a mispronunciation of the initial consonant. Phonological errors are not coded here, since that level of analysis is best conducted inside the Phon program (Rose et al., 2005).

7. Using MOR with a New Corpus

Because the English MOR grammar is stable and robust, the work of analyzing a new corpus seldom involves changes to the rules in the ar.cut or cr.cut files. However, a

new English corpus is still likely to need extensive lexical clean up before it is fully recognized by MOR. The unrecognized words can be identified quickly by running this command:

```
mor +xl *.cha
```

This command will go through a collection of files and output a single file "mini lexicon" of unrecognized words. The output is given the name of the first file in the collection. After this command finishes, open up the file and you will see all the words not recognized by MOR. There are several typical reasons for a word not being recognized:

1. It is misspelled.
2. The word should be preceded by an ampersand (&) to block look up through MOR. Specifically, incomplete words should be transcribed as &text so that the ampersand character can block MOR look up. Similarly, sounds like laughing can be transcribed as &=laughs to achieve the same effect.
3. The word should have been transcribed with a special form marker, as in bobo@o or bo^bo@o for onomatopoeia. It is impossible to list all possible onomatopoeic forms in the MOR lexicon, so the @o marker solves this problem by telling MOR how to treat the form. This approach will be needed for other special forms, such as babbling, word play, and so on.
4. The word was transcribed in "eye-dialect" to represent phonological reductions. When this is done, there are two basic ways to allow MOR to achieve correct lookup. If the word can be transcribed with parentheses for the missing material, as in "(be)cause", then MOR will be happy. This method is particularly useful in Spanish and German. Alternatively, if there is a sound

substitution, then you can transcribe using the [: text] replacement method, as in "pittie [: kittie]".

5. You should treat the word as a proper noun by capitalizing the first letter. This method works for many languages, but not in German where all nouns are capitalized and not in Asian languages, since those languages do not have systems for capitalization.
6. The word should be treated as a compound, as discussed in the previous section.
7. The stem is in MOR, but the inflected form is not recognized. In this case, it is possible that one of the analytic rules of MOR is not working. These problems can be reported to me at macw@cmu.edu.
8. The stem or word is missing from MOR. In that case, you can create a file called something like 0add.cut in the /lex folder of the MOR grammar. Once you have accumulated a collection of such words, you can email them to me for permanent addition to the lexicon.

Some of these forms can be corrected during the initial process of transcription by running CHECK. However, others will not be evident until you run the mor +xl command and get a list of unrecognized words. In order to correct these forms, there are basically two possible tools. The first is the KWAL program built in to CLAN. Let us say that your filename.ulx.cex list of unrecognized words has the form "cuaght" as a misspelling of "caught." Let us further imagine that you have a single collection of 80 files in one folder. To correct this error, just type this command into the Commands window:

kwal *.cha +scuaght

KWAL will then send input to your screen as it goes through the 80 files. There may be no more than one case of this misspelling in the whole collection. You will see this as the output scrolls by. If necessary, just scroll back in the CLAN Output window to find the error and then triple click to go to the spot of the error and then retype the word correctly.

For errors that are not too frequent, this method works fairly well. However, if you have made some error consistently and frequently, you may need stronger methods. Perhaps you transcribed "byebye" as "bye+bye" as many as 60 times. In this case, you could use the CHSTRING program to fix this, but a better method would involve the use of a powerful Programmer's Editor system such as BBEdit for the Mac or Epsilon for Windows. Any system you use must include an ability to process Regular Expressions (RegExp) and to operate smoothly across whole directories at a time. However, let me give a word of warning about the use of more powerful editors. When using these systems, particularly at first, you may make some mistakes. Always make sure that you keep a backup copy of your entire folder before each major replacement command that you issue.

Once you have succeeded in reducing the context of the minilex to zero, you are ready to run a final pass of MOR. After that, if there is a .db file in the MOR grammar for your language, you can run POST to disambiguate your file. After disambiguation, you should run CHECK again. There may be some errors if POST was not able to disambiguate everything. In that case, you would either need to fix MOR or else just

use CLAN's disambiguate tier function (escape-2) to finish the final stages of disambiguation.

8. Affixes and Control Features

At this point, it is probably a good idea to warn beginning users of CLAN and MOR that the remaining sections in this paper deal with a variety of increasingly difficult technical aspects of programs, coding, and analysis. As a result, beginning users will probably want to just scan these remaining sections and return to them when they have become more thoroughly acquainted with the use of CLAN.

To complete our tour of the MOR lexicon for English, we will take a brief look at the 0affix.cut file, as well some additional control features in the other lexical files. The responsibility of processing inflectional and derivational morphology is divided across these three files. Let us first look at a few entries in the 0affix.cut file.

1. This file begins with a list of prefixes such as "mis" and "semi" that attach either to nouns or verbs. Each prefix also has a permission feature, such as [allow mis]. This feature only comes into play when a noun or verb in n.cut or v.cut also has the feature [pre no]. For example, the verb "test" has the feature [pre no] included in order to block prefixing with "de-" to produce "detest" which is not a derivational form of "test". At the same time, we want to permit prefixing with "re-", the entry for "test" has [pre no][allow re]. Then, when the relevant rule in cr.cut sees a verb following "re-" it checks for a match in the [allow] feature and allows the attachment in this case.
2. Next we see some derivational suffixes such as diminutive –ie or agential –er. Unlike the prefixes, these suffixes often change the spelling of the stem by

dropping silent e or doubling final consonants. The ar.cut file controls this process, and the [allo x] features listed there control the selection of the correct form of the suffix.

3. Each suffix is represented by a grammatical category in parentheses. These categories are taken from a typologically valid list given in the CHAT Manual.
4. Each suffix specifies the grammatical category of the form that will result after its attachment. For suffixes that change the part of speech, this is given in the scat, as in [scat adj:n]. Prefixes do not change parts of speech, so they are simply listed as [scat pfx] and use the [pcat x] feature to specify the shape of the forms to which they can attach.
5. The long list of suffixes concludes with a list of cliticized auxiliaries and reduced main verbs. These forms are represented in English as contractions. Many of these forms are multiply ambiguous and it will be the job of POST to choose the correct reading from among the various alternatives.

Outside of the 0affix.cut file, in the various other *.cut lexical files, there are several control features that specify how stems should be treated. One important set includes the [comp x+x] features for compounds. These features control how compounds will be unpacked for formatting on the %mor line. Irregular adjectives in adj-ir.cut have features specifying their degree as comparative or superlative. Irregular nouns have features controlling the use of the plural. Irregular verbs have features controlling consonant doubling [gg +] and the formation of the perfect tense.

9. MOR for Bilingual Corpora

It is now possible to use MOR and POST to process bilingual corpora. The first application of this method has been to the transcripts collected by Virginia Yip and Stephen Matthews from Cantonese-English bilingual children in Hong Kong. In these corpora, parents, caretakers, and children often switch back and forth between the two languages. In order to tell MOR which grammar to use for which utterances, each sentence must be clearly identified for language. It turns out that this is not too difficult to do. First, by the nature of the goals of the study and the people conversing with the child, certain files are typically biased toward one language or the other. In the YipMatthews corpus, English is the default language in folders such as SophieEng or TimEng and Cantonese is the default in folders such as SophieCan and TimCan. Within the English files, the postcode [+ cant] is placed at the end of utterances that are primarily in Cantonese. If single Cantonese words are used inside English utterances, they are marked with the special form marker @s:c. For the files that are primarily in Cantonese, the opposite pattern is used. In those files, English sentences are marked as [+ eng] and English words inside Cantonese are marked by @s:e.

To minimize cross-language listing, it was also helpful to create easy ways of representing words that were shared between languages. This was particularly important for the names of family members or relation names. For example, the Cantonese form 姐姐 for "big sister" can be written in English as Zeze, so that this form can be processed correctly as a proper noun address term. Similarly, Cantonese has borrowed a set of English salutations such as "byebye" and "sorry" which are simply added directly to the Cantonese grammar in the co-eng.cut file.

Once these various adaptations and markings are completed, it is then possible to run MOR in two passes on the corpus. For the English corpora, the steps are:

1. Set the MOR library to English and run: mor -s”<+ cant>” *.cha +1
2. Disambiguate the results with: post *.cha +1
3. Run CHECK to check for problems.
4. Set the MOR library to Cantonese and run: mor +s”<+ cant>” *.cha +1
5. Disambiguate the results with: post +dcant.db *.cha +1
6. Run CHECK to check for problems.

To illustrate the result of this process, here is a representative snippet from the te951130.cha file in the /TimEng folder. Note that the default language here is English and that sentences in Cantonese are explicitly marked as [+ can].

*LIN: where is grandma first, tell me ?

%mor: adv:wh|where v|be n|grandma adv|first v|tell pro|me ?

*LIN: well, what's this ?

%mor: co|well pro:wh|what~v|be pro:dem|this ?

*CHI: xxx 呢 個 唔 夠 架 . [+ can]

%mor: unk|xxx det|ni1=this cl|go3=cl neg|m4=not adv|gau3=enough sfp|gaa3=sfp . [+ can]

*LIN: 呢 個 唔 夠 . [+ can]

%mor: det|ni1=this cl|go3=cl neg|m4=not adv|gau3=enough . [+ can]

*LIN: <what does it mean> [>] ?

%mor: pro:wh|what v:aux|do pro|it v|mean ?

Currently, this type of analysis is possible whenever MOR grammars exist for both languages, as would be the case for Japanese-English, Spanish-French, Putonghua-Cantonese, or Italian-Chinese bilinguals.

10. Training POST

The English POST disambiguator currently achieves over 95% correct disambiguation. We have not yet computed the levels of accuracy for the other disambiguators. However, the levels may be a bit better for inflectional languages like Spanish or Italian. In order to train the POST disambiguator, we first had to create a hand-annotated training set for each language. We created this corpus through a process of bootstrapping. Here is the sequence of basic steps in training.

1. First run MOR on a small corpus and used the escape-2 hand disambiguation process to disambiguate.
2. Then rename the %mor line in the corpus to %trn.
3. Run MOR again to create a separate %mor line.
4. Run POSTTRAIN with this command: posttrain +ttraintags.cut +c +o0errors.cut +x *.cha
5. This will create a new eng.db database.
6. You then need to go through the 0errors.cut file line by line to eliminate each mismatch between your %trn line and the codes of the %mor line. Mismatches arise primarily from changes made to the MOR codes in between runs of MOR.
7. Disambiguate the MOR line with: post +deng.db +tposttags.cut *.cha +1
8. Compare the results of POST with your hand disambiguation using: trnfix *.cha

In order to perform careful comparison using trnfix, you can set your *.trn.cex files into CA font and run longtier *.cha +1. This will show clearly the differences between the %trn and %mor lines. Sometimes the %trn will be at fault and sometimes %mor will be at fault. You can only fix the %trn line. To fix the %mor results, you just have to keep on compiling more training data by iterating the above process. As a rule of thumb, you eventually want to have at least 5000 utterances in your training corpus. However, a corpus with 1000 utterances will be useful initially.

11. Difficult decisions

During work in constructing the training corpus for POSTTRAIN, you will eventually bump into some areas of English grammar where the distinction between parts of speech is difficult to make without careful specification of detailed criteria. We can identify three areas that are particularly problematic in terms of their subsequent effects on GR (grammatical relation) identification:

1. **Adverb vs. preposition vs. particle**. The words “about”, “across”, “after”, “away”, “back”, “down”, “in”, “off”, “on”, “out”, “over”, and “up” belong to three categories: ADVerb, PREPosition and ParTicLe. To annotate them correctly, we apply the following criteria. First, a preposition must have a prepositional object. Second, a preposition forms a constituent with its noun phrase object, and hence is more closely bound to its object than an adverb or a particle. Third, prepositional phrases can be fronted, whereas the noun phrases that happen to follow adverbs or particles cannot. Fourth, a manner adverb can be placed between the verb and a preposition, but not between a verb and a particle. To distinguish between an adverb and a particle, the

meaning of the head verb is considered. If the meaning of the verb and the target word, taken together, cannot be predicted from the meanings of the verb and the target word separately, then the target word is a particle. In all other cases it is an adverb.

2. **Verb vs. auxiliary**. Distinguishing between Verb and AUXiliary is especially tricky for the verbs "be", "do" and "have". The following tests can be applied. First, if the target word is accompanied by a nonfinite verb in the same clause, it is an auxiliary, as in "I have had enough" or "I do not like eggs". Another test that works for these examples is fronting. In interrogative sentences, the auxiliary is moved to the beginning of the clause, as in "Have I had enough?" and "Do I like eggs?" whereas main verbs do not move. In verb-participle constructions headed by the verb "be", if the participle is in the progressive tense ("John is smiling"), then the head verb is labeled as an AUXiliary, otherwise it is a Verb ("John is happy").
3. **Communicator vs. Interjection vs. Locative adverbs**. COmmunicators can be hard to distinguish from interjections, and locative adverbs, especially at the beginning of a sentence. Consider a sentence such as "There you are" where "there" could be interpreted as either specifying a location or as providing an attentional focus, much like French voilà. The convention we have adopted is that CO must modify an entire sentence, so if a word appears by itself, it cannot be a CO. For example, utterances that begin with "here" or "there" without a following break are labelled as ADVerb. However, if these words appear at the beginning of a sentence and are followed by a break or pause, then they are labelled CO. Additionally, for lack of a better label, in here/there you are/go, here or there are labelled CO. Interjections, such as

"oh+my+goodness" are often transcribed at the beginning of sentences as if they behaved like communicators. However, they might better be considered as sentence fragments in their own right.

12. Building MOR Grammars

So far, this discussion of the MOR grammar for English has avoided an examination of the ar.cut and cr.cut files. It is true that users of English MOR will seldom need to tinker with these files. However, serious students of morphosyntax need to understand how MOR and POST operate. In order to do this, they have to understand how the ar.cut and cr.cut files work. Fortunately, for English at least, these rule files are not too complex. The relative simplicity of English morphology is reflected in the fact that the ar.cut file for English has only 391 lines, whereas the same file for Spanish has 3172 lines. In English, the main patterns involve consonant doubling, silent –e, changes of y to i, and irregulars like "knives" or "leaves." The rules use the spelling of final consonants and vowels to predict these various allomorphic variations. Variables such as $V or $C are set up at the beginning of the file to refer to vowels and consonants and then the rules use these variables to describe alternative lexical patterns and the shapes of allomorphs. For example the rule for consonant doubling takes this shape:

```
LEX-ENTRY:
LEXSURF = $O$V$C
LEXCAT = [scat v], ![tense OR past perf], ![gem no]  % to block putting
ALLO:
ALLOSURF = $O$V$C$C
ALLOCAT = LEXCAT, ADD [allo vHb]
```

ALLO:

ALLOSURF = LEXSURF

ALLOCAT = LEXCAT, ADD [allo vHa]

Here, the string OV$C characterizes verbs like “bat” that end with vowels followed by consonants. The first allo will produce words like “batting” or “batter” and the second will give a stem for “bats” or “bat”. A complete list of allomorphy types for English is given in the file engcats.cdc in the /docs folder in the MOR grammar.

When a user types the “mor” command to CLAN, the program loads up all the *.cut files in the lexicon and then passes each lexical form past the rules of the ar.cut file. The rules in the ar.cut file are strictly ordered. If a form matches a rule, that rule fires and the allomorphs it produces are encoded into a lexical tree based on a “trie” structure. Then MOR moves on to the next lexical form, without considering any additional rules. This means that it is important to place more specific cases before more general cases in a standard bleeding relation. There is no “feeding” relation in the ar.cut file, since each form is shipped over to the tree structure after matching.

The other “core” file in a MOR grammar is the cr.cut file that contains the rules that specify pathways through possible words. The basic idea of crules or concatenation or continuation rules is taken from Hausser’s (1999) left-associative grammar which specifies the shape of possible “continuations” as a parser moves from left to right through a word. Unlike the rules of the ar.cut file, the rules in the cr.cut file are not ordered. Instead, they work through a “feeding” relation. MOR goes through a candidate word from left to right to match up the current sequence with forms in the lexical trie TREE?? structure. When a match is made, the categories of the current

form become a part of the STARTCAT. If the STARTCAT matches up with the STARTCAT of one of the rules in cr.cut, as well as satisfying some additional matching conditions specified in the rule, then that rule fires. The result of this firing is to change the shape of the STARTCAT and to then thread processing into some additional rules. For example, let us consider the processing of the verb "reconsidering." Here, the first rule to fire is the specific-vpfx-start rule which matches the fact that "re-" has the feature [scat pfx] and [pcat v]. This initial recognition of the prefix then threads into the specific-vpfx-verb rule that requires the next item have the feature [scat v]. This rule has the feature CTYPE # which serves to introduce the # sign into the final tagging to produce re#part|consider-PROG. After the verb "consider" is accepted, the RULEPACKAGE tells MOR to move on to three other rules: v-conj, n:v-deriv, and adj:v-deriv. Each of these rules can be viewed as a separate thread out of the specific-vpfx-verb rule. At this point in processing the word, the remaining orthographic material is "-ing". Looking at the 0affix.cut file, we see that "ing" has three entries: [scat part], [scat v:n], and [scat n:gerund]. One of the pathways at this point leads through the v-conj rule. Within v-conj, only the fourth clause fires, since that clause matches [scat part]. This clause can lead on to three further threads, but, since there is no further orthographic material, there is no NEXTCAT for these rules. Therefore, this thread then goes on to the end rules and outputs the first successful parse of "reconsidering." The second thread from the specific-vpfx-verb rule leads to the n:v-deriv rule. This rule accepts the reading of "ing" as [scat n:gerund] to produce the second reading of "reconsidering". Finally, MOR traces the third thread from the specific-vpfx-verb rule which leads to adj:v-deriv. This route produces no matches, so processing terminates with this result:

Result: re#part|consider-PROG^re#n:gerund|consider-GERUND

Later, POST will work to choose between these two possible readings of "reconsidering" on the basis of the syntactic context. As we noted earlier, when "reconsidering" follows an auxiliary ("is eating") or when it functions adjectivally ("an eating binge"), it is treated as a participle. However, when it appears as the head of an NP ("eating is good for you"), it is treated as a gerund. Categories and processes of this type can be modified to match up with the requirements of the GRASP program to be discussed below.

The process of building ar.cut and cr.cut files for a new language involves a slow iteration of lexicon building with rule building. During this process, and throughout work with development of MOR, it is often helpful to use MOR in its interactive mode by typing: mor +xi . When using MOR in this mode, there are several additional options that become available in the CLAN Output window. They are:

word - analyze this word

:q quit- exit program

:c print out current set of crules

:d display application of a rules.

:l re-load rules and lexicon files

:h help - print this message

If you type in a word, such as "dog" or "perro," MOR will try to analyze it and give you its component morphemes. If all is well, you can move on the next word. If it is not, you need to change your rules or the lexicon. You can stay within CLAN and just open these using the Editor. After you save your changes, use :l to reload and retest the word again.

The problem with building up a MOR grammar one word at a time like this is that changes that favour the analysis of one word can break the analysis of other words. To make sure that this is not happening, it is important to have a collection of test words that you continually monitor using mor +xl. One approach to this is just to have a growing set of transcripts or utterances that can be analyzed. Another approach is to have a systematic target set configured not as sentences but as transcripts with one word in each sentence. An example of this approach can be found in the /verbi folder in the Italian MOR grammar. This folder has one file for each of the 106 verbal paradigms of the Berlitz Italian Verb Handbook (2005). That handbook gives the full paradigm of one "leading" verb for each conjugational type. We then typed all of the relevant forms into CHAT files. Then, as we built up the ar.cut file for Italian, we designed allo types using features that matched the numbers in the Handbook. In the end, things become a bit more complex in Spanish, Italian, and French.

1. The initial rules of the ar.cut file for these languages specify the most limited and lexically-bound patterns by listing almost the full stem, as in $Xdice for verbs like "dicere", "predicere" or "benedicere" which all behave similarly, or "nuoce" which is the only verb of its type.
2. Further in the rule list, verbs are listed through a general phonology, but often limited to the presence of a lexical tag such as [type 16] that indicates verb membership in a conjugational class.
3. Within the rule for each verb type, the grammar specifies up to 12 stem allomorph types. Some of these have the same surface phonology. However, to match up properly across the paradigm, it is important to generate this full set. Once this basic grid is determined, it is easy to add new rules for each

additional conjugational type by a process of cut-and-paste followed by local modifications.

4. Where possible, the rules are left in an order that corresponds to the order of the conjugational numbers of the Berlitz Handbook. However, when this order interferes with rule bleeding, it is changed.
5. Perhaps the biggest conceptual challenge is the formulation of a good set of [allo x] tags for the paradigm. The current Italian grammar mixes together tags like [allo vv] that are defined on phonological grounds and tags like [allo vpart] that are defined on paradigmatic grounds. A more systematic analysis would probably use a somewhat larger set of tags to cover all tense-aspect-mood slots and use the phonological tags as a secondary overlay on the basic semantic tags.
6. Although verbs are the major challenge in Romance languages, it is also important to manage verbal clitics and noun and adjectives plurals. In the end, all nouns must be listed with gender information. Nouns that have both masculine and feminine forms are listed with the feature [anim yes] that allows the ar.cut file to generate both sets of allomorphs.
7. Spanish has additional complexities involving the placement of stress marks for infinitives and imperatives with suffixed clitics, such as dámelo. Italian has additional complications for forms such as “nello” and the various pronominal and clitic forms.

To begin the process, start working with the sample “minMOR” grammars available from the net. These files should allow you to build up a lexicon of uninflected stems. Try to build up separate files for each of the parts of speech in your language. As you

start to feel comfortable with this, you should begin to add affixes. To do this, you need to create a lexicon file for affixes, such as affix.cut. Using the technique found in unification grammars, you want to set up categories and allos for these affixes that will allow them to match up with the right stems when the crules fire. For example, you might want to assign [scat nsfx] to the noun plural suffix in order to emphasize the fact that it should attach to nouns. And you could give the designation [allo mdim] to the masculine diminutive suffix -ito in Spanish in order to make sure that it only attaches to masculine stems and produces a masculine output.

As you progress with your work, continually check each new rule change by entering :l (colon followed by "l" for load) into the CLAN Output window and then testing some crucial words. If you have changed something in a way that produces a syntactic violation, you will learn this immediately and be able to change it back. If you find that a method fails, you should first rethink your logic. Consider these factors:

1. Arules are strictly ordered. Maybe you have placed a general case before a specific case.
2. Crules depend on direction from the RULEPACKAGES statement. Perhaps you are not reaching the rule that needs to fire.
3. There has to be a START and END rule for each part of speech. If you are getting too many entries for a word, maybe you have started it twice. Alternatively, you may have created too many allomorphs with the arules.
4. Possibly, you form is not satisfying the requirements of the end rules. If it doesn't these rules will not "let it out."

5. If you have a MATCHCAT allos statement, all allos must match. The operation DEL [allo] deletes all allos and you must add back any you want to keep.
6. Make sure that you understand the use of variable notation and pattern matching symbols for specifying the surface form in the arules.

However, sometimes it is not clear why a method is not working. In this case, you will want to check the application of the crules using the :c option in the CLAN Output window. You then need to trace through the firing of the rules. The most important information is often at the end of this output.

If the stem itself is not being recognized, you will need to also trace the operation of the arules. To do this, you should either use the +e option in standard MOR or else the :d option in interactive MOR. The latter is probably the most useful. To use this option, you should create a directory called testlex with a single file with the words you are working with. Then run: mor +xi +ltestlex
Once this runs, type :d and then :l and the output of the arules for this test lexicon will go to debug.cdc. Use your editor to open that file and try to trace what is happening there.

As you progress with the construction of rules and the enlargement of the lexicon, you can tackle whole corpora. At this point you will occasionally run the +xl analysis. Then you take the problems noted by +xl and use them as the basis for repeated testing using the +xi switch and repeated reloading of the rules as you improve them. As you build up your rule sets, you will want to annotate them fully using comments preceded by the % symbol.

13. Chinese MOR

In comparison to the morphologies of languages like Italian or Japanese, the development of MOR grammars for Putonghua or Cantonese is much simpler. This is because these languages have essentially no affixes. The few exceptions to this are the four forms listed in the 0affix.cut file for Putonghua. There are no suffixes for Cantonese at all. In addition, both Cantonese and Putonghua have a single rule that produces diminutive reduplications for nouns and verbs. For adjectives, the pattern is more complex and is listed for each possible lexical form.

Although Putonghua and Cantonese have few suffixes, they have very productive systems of compounding. However, because Chinese characters are written with spaces to separate words, there is no systematic tradition for lemmatization of Chinese compounds. One current trend tends to include adjectives with nouns as compounds, forming single words from combinations such as “good boy” or “train station.” Of course, Chinese has many true compounds, such as 图书馆 "tu2shu1guan3" for “library” or 椭圆形"tuo3yuan2xing2" for “oval.” Within the verbs, there is a tendency to treat combination of serial verbs such as 上去 "shang4qu4" “up go” as units, perhaps under the influence of translations from English. However, the meanings in such cases are fully combinatorial. Figuring out how to list true compounds without adding superfluous compounds remains a major task for work on Chinese. The basic criteria here should be the same as in other languages. Word sequences should not be listed as single words if the meaning is fully predicted from the combination of the separate pieces and if there are no processes of allomorphy triggered by the combination.

The other major challenge in Chinese is the specification of part of speech tags. Currently available lexicons use a wide variety of tags deriving from different grammatical analyses. Often adjectives are treated as verbs. It is likely that this is done because Chinese deletes the copula. Without a copula to serve as the predicate, the adjective is then promoted to the status of a full verb. However, a clearer treatment of the relevant syntax would treat sentences with missing copulas as representing topic + comment structures. In that analysis, adjectives would simply function as adjectives. Similar issues arise with the listing of adjectives as adverbs. Here again, part of speech categorization is being driven by the selection of a particular syntactic analysis. Despite these various problems with part of speech categorization, we have managed to construct Chinese lexicons and training corpora that can be successfully used to achieve automatic disambiguation with POST at a high level of accuracy.

14. GRASP

After finishing tagging with MOR and POST, researchers will want to run the GRASP program (also called MEGRASP in the version on the web) to create a %xsyn line with tagged grammatical relations. GRASP produces labelled dependency structures for CHILDES transcript data. The system uses the 29 relations summarized in this table:

GR	Definition	Example
SUBJ	nonclausal subj	Mary saw a movie.
CSUBJ	finite clausal subject	That Mary screamed scared John.
XSUBJ	nonfinite clausal subject	Eating vegetables is important.
OBJ	direct object	Mary saw a movie.
OBJ2	indirect object	Mary gave John a book.
IOBJ	required prepositional phrase	Mary gave a book to John.
COMP	finite clausal verb complement	I think that Mary saw a movie.

XCOMP	nonfinite clausal verb complement	Mary likes watching movies.
PRED	predicate nominal or adjective	Mary is a student.
CPRED	predicate finite clausal complement	The problem is that Mary sings.
XPRED	predicate nonfinite clausal comp.	My goal is to win the race.
JCT	PP or adv as adjunct Head is v, adj, or adv	Mary spoke clearly. Mary spoke at the meeting Mary spoke very clearly.
CJCT	finite clause as adjunct	Mary left after she heard the news.
XJCT	nonfinite clause as adjunct	Mary left after hearing the news.
MOD	nonclausal modifier	Mary saw a red car.
CMOD	finite clausal modifier	The car that bumped me was red.
XMOD	nonfinite clausal modifier	The car driving by is red.
AUX	auxiliary of a verb or modal	Mary has seen many movies.
NEG	verbal negation	I am not eating cake.
DET	determiner of a noun (art, poss pro)	The students ate that cake.
POBJ	object of a preposition	Mary saw the book on her desk.
PTL	verb particle	Mary put off the meeting.
CPZR	complementizer	I think that Mary left.
COM	communicator	Okay, you can go.
INF	infinitival	Mary wants to go.
VOC	vocative	Mary, you look lovely.
TAG	tag questions	That is good, isn't it?
COORD	coordination, conj is the head	Mary likes cats and dogs.
ROOT	relation between verb and left wall	Mary saw Jim last week.

When these GRs are applied to a string of words in files, they yield a labeled dependency structure. Here is an example from the Eve corpus.

*COL: do we have everything ?

%mor: v:aux|do pro|we v|have pro:indef|everything ?

%xsyn: 1|3|AUX 2|3|SUBJ 3|0|ROOT 4|3|OBJ 5|3|PUNCT

The relations given on the %xsyn line can be reformatted into a graph structure in which all of the elements depend on the verb (item #3) and the verb itself is attached to the root (item #0). Or to take a slightly more complex example:

*MOT: well it's already out # isn't it ?

%mor: co|well pro|it~v|be&3S adv|already adv|out v|be&3S~neg|not pro|it ?

%xsyn: 1|3|COM 2|3|SUBJ 3|0|ROOT 4|5|JCT 5|3|PRED 6|3|TAG 7|6|NEG 8|6|SUBJ 9|3|PUNCT

Here, the word "out" is treated as a predicate modifying the verb and "already" is a daughter of "out." This seems correct semantically. The final words are processed as a tag question.

Currently, GRASP processing is limited to English. However, it can be extended to other languages, and we would be happy to work with colleagues on such extensions. Ongoing progress in the development of the GRASP system has been described in three recent papers (Sagae, Davis, Lavie, MacWhinney, & Wintner, 2007; Sagae, Lavie, & MacWhinney, 2005; Sagae, MacWhinney, & Lavie, 2004b). The following discussion of the current state of GRASP is taken from Sagae et al (2007).

Our most recent work began with the completion of hand-annotations for 15 of the 20 files in the Eve section of Roger Brown's corpus. These files included 18,863 fully hand-annotated utterances with 10,280 from adults and 8,563 from Eve. The utterances contained 84,226 grammatical relations and 65,363 words. The parser is highly efficient. Training on the Eve corpus takes 20 minutes and, once trained, the corpus can be parsed in 20 seconds. The parser produces correct dependency relations for 96% of the relations in the gold standard. In addition, the dependency relations are labelled with the correct GR 94% of the time. Performance was slightly better on the adult utterances with 95% correct labelling for adult GRs and 93% correct labelling for child GRs.

The parser relies on a best-first probabilistic shift-reduce algorithm, working left-to-right to find labelled dependencies one at a time. The two main data structures in the algorithm are a stack and a queue. The stack holds subtrees, and the queue holds the words in an input sentence that have not yet been assigned to subtrees. At the beginning of processing, the stack is empty and the queue holds all the words in the sentence with the first word of the sentence in the front of the queue. The parser performs two main types of actions: shift and reduce. When a shift action is taken, a word is shifted from the front of the queue, and placed on the top of the stack. When a reduce action is taken, the two top items on the top of the stack are popped, and a new item is pushed onto the stack. Depending on whether the head of the new tree is to the left or to the right of its new dependent, we call the action either shift-left or shift-right. Each tree fragment built in this way must also be given a grammatical relation label.

To extend this deterministic model to a probabilistic model, we use a best-first strategy. This involves an extension of the deterministic shift-reduce into a best-first shift-reduce algorithm based on selection of a parser action from a heap of parser actions ordered by their relative probabilities. The parser uses maximum entropy modelling (Berger, Della Pietra, & Della Pietra, 1996) to determine the actions and their probabilities. Features used in classification at any point during parsing are derived from the parser's current configuration (contents of the stack and queue) at that point. The specific features fed into the classifier include: the word and its POS tag, the shape of related dependencies, and the shape of recently applied rules.

15. Research Using the New Infrastructure

The goal of all this tagging work is to support easier analysis of morphosyntactic development and more accurate computation of automatic indices such as MLU, VOCD, DSS, and IPSyn. Currently, researchers will find that the easiest way to make use of these new tags is to use the basic CLAN search programs of KWAL, COMBO, and MODREP. For example, if you want to study the acquisition of auxiliary verbs, you can simply search the %mor line using a command like this:

```
kwal +t%mor +s"v:aux|*"  *.cha
```

If you want to study the development of compounds, you could use commands like these:

```
kwal +t%mor +s"n|+n*" *.cha
```

If you want to trace combinations of parts of speech, you can use the COMBO command. For example, this command would search for auxiliaries followed by nouns in yes-no questions:

```
combo +t%mor +s"v:aux|*^n|*"  *.cha
```

You can also use a combination of the MODREP and COMBO commands to search for cases when something has a particularly part of speech role on the %mor line and a grammatical relation role on the %xsyn line. For example, you could search in this way for pronouns that are objects of prepositions.

The %mor and %xsyn lines open up a wide variety of possibilities for increasing precision in the study of morphosyntactic development. The range of topics that can be investigated in this area is limited only by the imagination of researchers and the scopes of the relevant theories. Consider some of the following possibilities:

1. Optional infinitival errors – tags on the %mor and %syn line will allow you to identify both correct and incorrect uses of infinitives.
2. Case errors – working from the %syn structure, one can identify pronouns with and without correct case marking.
3. Fronted auxiliaries – these forms can be readily identified from the %mor line.
4. Grammatical role identification – roles can be read off the %syn line.
5. Distinguishing particular argument structures – using the %syn line, one can distinguish between "to" phrases used to mark datives and those used to mark location.
6. Locating double object constructions. The %syn line will identify structures with double objects.

Searches within the %xsyn tier have a similar logic. One can use KWAL to find basic GR types, such as complements or relative clauses. Thus, these tags would fully automate an analysis of the type found in Diessel & Tomasello (2001). For linear combinations of types, you can use COMBO. In the future, we hope to provide more powerful methods for searching syntactic structures on the %xsyn line.

16. Next Steps

Once our work with the tagging and parsing of the CHILDES database is largely completed, we need to provide tools that will allow researchers to take full advantage of this new tagged infrastructure. First, we need to construct methods for searching effectively through the dependency graphs constructed by GRASP. Consider the example of the structures examined in MacWhinney (2005). These involve sentences such as these:

1. The man who is running is coming.

2. *Is the man who running is coming?
3. Is the man who is running coming?

In his debate with Jean Piaget (Piattelli-Palmarini, 1980), Chomsky argued that children would know immediately that (2) is ungrammatical, despite the fact that they never hear sentences like (3). According to Chomsky, this means that the child's acquisition of grammar is "hopelessly underdetermined by the fragmentary evidence available." A study by Crain & Nakayama (1987) supports the idea that children are sensitive to the ungrammaticality of (2) and a corpus search in MacWhinney(2005) for sentences such as (3) in CHILDES supports Chomsky's belief that such sentences are virtually absent in the input. However, MacWhinney also found extensive evidence for slightly different, but related sentences that provide clear positive evidence for learning regarding the acceptability of (3). In searching for the relevant contexts, MacWhinney was forced to rely on searches based on the %mor tier. To do this, it was necessary to compose search strings involving AUX, WH, and other categories. Although these search patterns are effective in catching most relevant patterns, there is always a possibility that they are missing some cases. Moreover, they do not provide good categorizations of the relevant structures by grammatical relations. And that is, after all, what is involved in this debate. By constructing search methods that can look at relative clauses in different structural conditions, we will be able to understand such issues more explicitly.

Work with these tagged corpora is not limited to processing of specific constructions. It is also possible to use these new tags to explore ways of inducing grammars from corpora. As we complete the GR tagging of the database, it will be possible to evaluate alternative learning systems in terms of their ability to move through

longitudinal data and match the tags at each stage computed by GRASP. As we noted earlier, these systems could utilize perspectives from Minimalism with parameters (Buttery, 2004), HPSG (Wintner, MacWhinney, & Lavie, 2007), item-based grammars (Gobet & Pine, 1997), or statistical learning (Edelman, Solan, Horn, & Ruppin, 2004). In each case, we will be providing a level playing field for the evaluation of the abilities of these contrasting systems.

17. Conclusion

This paper has surveyed a wide variety of issues in the automatic construction of morphosyntactic tags for the CHILDES database. This discussion was targeted to three audiences: experienced CHILDES users, researchers new to CHILDES, and computational linguists. Experienced researchers doing work on morphosyntactic analysis need to understand all of these issues in great detail. Researchers who are new to the use of CHILDES data need to be aware of the various options open for analysis as they learn to use transcripts to address theoretical questions. Computational linguists can use the CHILDES database as a test bed for evaluating new methods for tagging and analyzing corpus material. Much of the material discussed in this chapter has involved issues that could be dismissed as "messy" coding details. However, in reality, making systematic decisions about the treatment of interjections, adverbs, or specific grammatical relations involve fundamental theoretical thinking about the shape of human language and the ways in which language presents itself to the child.

References

Beasley, K., & Karttunen, L. (2003). *Finite state morphology*. Stanford, CA: CSLI Publications.

Berger, A., Della Pietra, S. A., & Della Pietra, V. J. (1996). A maximum entropy approach to natural language processing. *Computational Linguistics, 22*, 39-71.

Berlitz. (2005). *Berlitz Italian verbs handbook: 2nd Edition*. New York: Berlitz.

Brown, R. (1973). *A first language: The early stages*. Cambridge, MA: Harvard.

Buttery, P. (2004). A quantitative evaluation of naturalistic models of language acquisition; the efficiency of the Triggering Learning Algorithm compared to a Categorial Grammar Learner. *Coling 2004*, 1-8.

Bybee, J., & Hopper, P. (2001). *Frequency and the emergence of linguistic structure*. Amsterdam: John Benjamins.

Chomsky, N. (1957). *Syntactic structures*. The Hague: Mouton.

Chomsky, N. (1965). *Aspects of the theory of syntax*. Cambridge, MA: MIT Press.

Chomsky, N. (1995). *The minimalist program*. Cambridge: MIT Press.

Chomsky, N., & Lasnik, H. (1993). The theory of principles and parameters. In J. Jacobs (Ed.), *Syntax: An international handbook of contemporary research* (pp. 1-32). Berlin: Walter de Gruyter.

Crain, S. (1991). Language acquisition in the absence of experience. *Behavioral and Brain Sciences, 14*, 597-611.

Crain, S., & Nakayama, M. (1987). Structure dependence in grammar formation. *Language, 63 No. 3*, 522-543.

Culicover, P., & Jackendoff, R. (2005). *Simpler syntax*. Oxford: Oxford University Press.

Diessel, H., & Tomasello, M. (2001). The acquisition of finite complement clauses in English: A corpus-based analysis. *Cognitive Linguistics, 12*, 97-141.

Edelman, S., Solan, Z., Horn, D., & Ruppin, E. (2004). Bridging computational, formal and psycholinguistic approaches to language. *Cognitive Science Society*.

Edwards, J. (1992). Computer methods in child language research: four principles for the use of archived data. *Journal of Child Language, 19*, 435-458.

Elman, J. L. (1993). Learning and development in neural networks: The importance of starting small. *Cognition, 48*, 71-99.

Gobet, F., & Pine, J. M. (1997). Modelling the acquisition of syntactic categories. In *Proceedings of the 19th Annual Meeting of the Cognitive Science Society*. Hillsdale, NJ: Lawrence Erlbaum.

Hausser, R. (1999). *Foundations of computational linguistics: Man-machine communication in natural language*. Berlin: Springer.

Hyams, N. (1986). *Language acquisition and the theory of parameters*. Dordrecht: D. Reidel.

Hyams, N., & Wexler, K. (1993). On the grammatical basis of null subjects in child language. *Linguistic Inquiry, 24*(3), 421-459.

Koskenniemi, K. (1983). *Two-level morphology: a general computational model for word-form recognition and production*. Helsinki: University of Helsinki, Department of Linguistics.

Lee, L. (1974). *Developmental Sentence Analysis*. Evanston, IL: Northwestern University Press.

Li, P., Zhao, X., & MacWhinney, B. (2007). Self-organizing processes in early lexical learning. *Cognitive Science, 31*.

Long, S., & Channell, R. (2001). Accuracy of four language analysis procedures performed automatically. *American Journal of Speech-Language Pathology, 10*, 212-225.

MacWhinney, B. (1975). Pragmatic patterns in child syntax. *Stanford Papers And Reports on Child Language Development, 10*, 153-165.
MacWhinney, B. (1987). The Competition Model. In B. MacWhinney (Ed.), *Mechanisms of language acquisition* (pp. 249-308). Hillsdale, NJ: Lawrence Erlbaum.
MacWhinney, B. (1991). *The CHILDES project: Tools for analyzing talk*. Hillsdale, NJ: Erlbaum.
MacWhinney, B. (1995). *The CHILDES Project: Tools for analyzing talk. 2nd Edition.* (Second ed.). Hillsdale, NJ: Lawrence Erlbaum.
MacWhinney, B. (2000). *The CHILDES Project: Tools for Analyzing Talk. 3rd Edition*. Mahwah, NJ: Lawrence Erlbaum Associates.
MacWhinney, B. (2005). Item-based constructions and the logical problem. *ACL 2005*, 46-54.
MacWhinney, B., & Leinbach, J. (1991). Implementations are not conceptualizations: Revising the verb learning model. *Cognition, 29*, 121-157.
MacWhinney, B., Leinbach, J., Taraban, R., & McDonald, J. (1989). Language learning: Cues or rules? *Journal of Memory and Language, 28*, 255-277.
Malvern, D. D., & Richards, B. J. (1997). A new measure of lexical diversity. In R. A. & W. A. (Eds.), *Evolving models of language* (pp. 58-71). Clevedon: Multilingual Matters.
Malvern, D. D., Richards, B. J., Chipere, N., & Purán, P. (2004). *Lexical diversity and language development*. New York: Palgrave Macmillan.
Marcus, G., Ullman, M., Pinker, S., Hollander, M., Rosen, T., & Xu, F. (1992). Overregularization in language acquisition. *Monographs of the Society for Research in Child Development, 57*(4), 1-182.
Meisel, J. (1986). Word order and case marking in early child language. Evidence from simultaneous acquisition of two first languages: French and German. *Linguistics, 24*, 123-185.
Miikkulainen, R., & Mayberry, M. R. (1999). Disambiguation and grammar as emergent soft constraints. In B. MacWhinney (Ed.), *The emergence of language* (pp. 153-176). Mahwah, NJ: Lawrence Erlbaum Associates.
Miller, J., & Chapman, R. (1983). *SALT: Systematic Analysis of Language Transcripts, User's Manual*. Madison, WI: University of Wisconsin Press.
Mintz, T. H., Newport, E. L., & Bever, T. G. (2002). The distributional structure of grammatical categories in speech to young children. *Cognitive Science, 26*, 393-424.
Parisse, C., & Le Normand, M. T. (2000). Automatic disambiguation of the morphosyntax in spoken language corpora. *Behavior Research Methods, Instruments, and Computers, 32*, 468-481.
Pearl, J. (2005). The input for syntactic acquisition: Solutions from language change modeling. *ACL 2005*, 12-18.
Piattelli-Palmarini, M. (1980). *Language and learning: the debate between Jean Piaget and Noam Chomsky*. Cambridge MA: Harvard University Press.
Pine, J. M., & Lieven, E. V. M. (1997). Slot and frame patterns and the development of the determiner category. *Applied Psycholinguistics, 18*, 123-138.
Plunkett, K., & Marchman, V. (1991). U-shaped learning and frequency effects in a multi-layered perceptron: Implications for child language acquisition. *Cognition, 38*, 43-102.
Radford, A. (1990). *Syntactic theory and the acquisition of English syntax*. Oxford: Basil Blackwell.

Rice, S. (1997). The analysis of ontogenetic trajectories: When a change in size or shape is not heterochrony. *Proceedings of the National Academy of Sciences, 94*, 907-912.

Rose, Y., MacWhinney, B., Byrne, R., Hedlund, G., Maddocks, K., O'Brien, P., et al. (2005). Introducing Phon: A Software Solution for the Study of Phonological Acquisition. In D. Bamman, T. Magnitskaia & C. Zaller (Eds.), *30th Annual Boston University Conference on Language Development* (pp. 489-500). Somerville, MA: Cascadilla Press.

Rumelhart, D. E., & McClelland, J. L. (1986). On learning the past tense of English verbs. In J. L. McClelland & D. E. Rumelhart (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition* (pp. 216-271). Cambridge: MIT Press.

Sagae, K., Davis, E., Lavie, E., MacWhinney, B., & Wintner, S. (2007). High-accuracy annotation and parsing of CHILDES transcripts. In *Proceedings of the 45th Meeting of the Association for Computational Linguistics*. Prague: ACL.

Sagae, K., Lavie, A., & MacWhinney, B. (2005). Automatic measurement of syntactic development in child language. In *Proceedings of the 43rd Meeting of the Association for Computational Linguistics* (pp. 197-204). Ann Arbor: ACL.

Sagae, K., MacWhinney, B., & Lavie, A. (2004a). Adding syntactic annotations to transcripts of parent-child dialogs. In *LREC 2004* (pp. 1815-1818). Lisbon: LREC.

Sagae, K., MacWhinney, B., & Lavie, A. (2004b). Automatic parsing of parent-child interactions. *Behavior Research Methods, Instruments, and Computers, 36*, 113-126.

Scarborough, H. S. (1990). Index of productive syntax. *Applied Psycholinguistics, 11*, 1-22.

Sells, P. (Ed.). (2001). *Formal and theoretical issues in optimality theoretic syntax*. Stanford: CSLI Publications.

Siskind, J. M. (1999). Learning word-to-meaning mappings. In J. Muure & P. Broeder (Eds.), *Cognitive Models of Language Acquisition*. Cambridge: MIT Press.

Stolcke, S. A., & Omohundro, S. (1994). Inducing probabilistic grammars by Bayesian model merging. In R. C. Carrasco & J. Oncina (Eds.), *Grammatical inference and applications*. Berlin: Springer.

Stromswold, K. (1994). Using spontaneous production data to assess syntactic development. In D. McDaniel, C. McKee & H. Cairns (Eds.), *Methods for assessing children's syntax*. Cambridge: MIT Press.

Tomasello, M. (2003). *Constructing a first language: A usage-based theory of language acquisition*. Cambridge: Harvard University Press.

van Kampen, J. (1998). Left branch extraction as operator movement: Evidence from child Dutch. In S. Powers & C. Hamman (Eds.), *The acquisition of scrambling and cliticization*. Norwell: Kluwer.

Wexler, K. (1998). Very early parameter setting and the unique checking constraint: A new explanation of the optional infinitive stage. *Lingua, 106*, 23-79.

Wintner, S. (2007). Finite-state technology as a programming environment. In A. Gelbukh (Ed.), *CICLing 2007, LNCS 4394* (pp. 97-106). Heidelberg: Springer.

Wintner, S., MacWhinney, B., & Lavie, A. (2007). Formal grammars of early language. *ACL*.