Towards a Computational Model of Early Syntactic Acquisition

Sheli Kol

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE MASTER DEGREE

University of Haifa
Faculty of Social Sciences
Department of Computer Science

February, 2011

Towards a Computational Model of Early Syntactic Acquisition

By: Sheli Kol
Supervised By: Prof. Shuly Wintner and Dr. Bracha Nir

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE MASTER DEGREE

University of Haifa
Faculty of Social Sciences
Department of Computer Science

February, 2011

Approved by: _____ Date:_____
(supervisor)

Approved by: _____ Date:_____
(supervisor)

Approved by: _____ Date:_____
(Chairman of M.A. Committee)

Contents

1 Introduction **1**

2 Computational Models of Language Learning **1**
2.1 Review of Research Methodology 2
2.1.1 The Use of Corpora in Language Learning Research 2
2.1.2 The Evaluation of Language Learning Models 4
2.2 Computational Grammar Induction 5
2.3 Cognitively Motivated Language Learning Algorithms 9

3 Implementing a Cognitively Motivated Model: The Traceback Method **12**
3.1 Re-implementation of the TBM 14
3.2 Evaluation of the TBM 17

4 A DSFA Model of Early Syntactic Acquisition **19**
4.1 Motivating Principles 20
4.2 The Language Acquisition Algorithm 22
4.3 Properties of the Algorithm 27
4.4 Examples 32

5 Results and Evaluation **34**
5.1 Clustering of Words 38
5.2 Comparison to Other Learning Systems 39

6 Conclusion **40**

Abstract

We propose a formal, computational model of early syntactic acquisition, based on stochastic finite-state automata. The model learns from adult and child utterances and generalizes them such that novel utterances can be produced. Unlike much work in computational grammar induction, our model is informed by several well-established psycholinguistic principles, which are implemented as biases in the learning process. Unlike some works in cognitive psycholinguistics, our model is mathematically well-defined and robustly evaluated. We demonstrate the ability of the model to generalize by evaluating it on several corpora of child-adult interactions.

1 Introduction

One of the most fascinating and fundamental questions in linguistics is how children acquire their native language. Much research in psycholinguistics sheds light on this puzzling process; we propose a formal, computational model of early syntactic acquisition, focusing on structures emerging in the first three years of life. We present an algorithm that inputs adult and child utterances and outputs a compact representation of the syntactic knowledge of the child exposed to the input, in the form of a highly restricted stochastic finite-state automaton.

Unlike much work in computational grammar induction, our model is informed by several well-established psycholinguistic principles, which are implemented as biases in the learning process. Unlike some works in cognitive psycholinguistics, our model is mathematically well-defined and robustly evaluated. We demonstrate the ability of the model to generalize by evaluating it on several corpora of child-adult interactions. We thus consolidate formal, empirical and cognitive approaches to language learning and harness them to better explain early stages of human language acquisition.

We review related work in Section 2, focusing on a particular method in Section 3. Then, we present our learning algorithm in Section 4. We define it mathematically and formally prove some of its properties. In Section 5 we evaluate the performance of the algorithm. We conclude with suggestions for future work.

2 Computational Models of Language Learning

Computational models of language learning are the focus of two quite different (and independent) fields of study (Adriaans and van Zaanen, 2006; Alishahi, 2011). One line of research falls under the category of *grammar induction* or, more specifically, *computational grammar inference*. Here, the goal is to devise algorithms that can learn accurate and compact models for identification of language (i.e., grammars) from finite sets of examples. Such approaches are usually not cognitively motivated, and their relevance to

child language acquisition is questionable. The other research direction is motivated by the quest for a cognitively plausible explanation of child language acquisition. Such works may use computational models to either implement cognitive theories or evaluate them (or both), but are sometimes lacking in accuracy and robust evaluation. Sections 2.2 and 2.3 survey some representative works in both paradigms. We begin, however, by reviewing the use of corpora in child language research and the issue of evaluation.

2.1 Review of Research Methodology

The task that we focus on is language learning: a learner, be it a child or a computer program, is presented with data, in the form of raw utterances. The learner's task is to generalize the data and induce a model of the grammatical utterances (in other words, a grammar). The success of the learner can be evaluated by testing its grammar on new utterances. Two aspects of the grammar can be tested: its ability to generate new utterances; and its ability to assign a valid structure to the grammatical utterances. We describe here the corpora and the evaluation methods that are used by researchers from the two fields of study mentioned above.

2.1.1 The Use of Corpora in Language Learning Research

Language learning algorithms need input data (e.g., sequence of utterances from a specified language) in order to generate the language model (the grammar) and evaluate it. For this need, grammar induction tasks standardly use the manually-annotated sentences from the Penn Tree Bank (Marcus et al., 1993), whose data are taken from the Wall Street Journal (WSJ). The Penn Tree Bank is a large annotated corpus consisting of over 4.5 million words of American English. This corpus is annotated for part-of-speech (POS) and for skeletal syntactic structure. The POS tag-set contains 36 POS tags and 12 other tags for punctuation and currency symbols (a detailed list of the tag-set is at Marcus et al. (1993), page 319). Such annotated corpora are valuable for automatic construction of statistical models and for evaluating the adequacy of parsing models.

Some grammar induction algorithms only use the sequences of POS tags from the WSJ corpus, ignoring the actual words as the training data for the language learning algorithm. Furthermore, training material in this paradigm may consist of the utterances in the WSJ whose length is less than 10 words (WSJ10), a genre that is not well-defined linguistically, and quite likely irrelevant for language acquisition investigations (Berwick, 2009).

Cognitively-motivated models of language acquisition, in contrast, typically utilize transcripts of spontaneous speech, mainly based on recordings of child-adult interactions that are collected and maintained as part of the Child Language Data Exchange System (CHILDES). The CHILDES database (MacWhinney, 2000) contains transcripts and media data that were collected from conversations between young children and their playmates and caretakers. CHILDES is the largest corpus of conversational spoken language data which is publicly available worldwide (`http://childes.psy.cmu.edu`). All of the data in the system are consistently coded using a single transcript format called CHAT. Moreover, for several languages, the corpora have been tagged for part of speech using an automatic tagging program called MOR. The output of MOR includes two lines for each utterance. The main line lists the speaker's production and the MOR line provides the part of speech of each word, along with morphological analysis of affixes, such as the past tense mark (-PAST) on the verb. The GRASP program (Sagae et al., 2010), which adds syntactic dependency relation tags, is currently also available from the CHILDES website (`http://www.cs.cmu.edu/~sagae/childesparser/`). The English section of CHILDES is now fully annotated with morphological and syntactic tags. For example:

```
*ATT:   you       did           so       .
%mor:   pro|you   aux|do&PAST   adv|so   .
%xgra:  1|2|SUBJ  2|0|AUX-ROOT  3|2|JCT  4|2|PUNCT
```

In the above example each word has part-of-speech tag (first line) and syntactic tag (second line), expressed as a labeled dependency on some other word in the same utterance.

A new source of data which contains transcripts of children's speech has been in use

by researchers lately (Dąbrowska and Lieven, 2005). This corpus contains transcribed longitudinal recordings of four children, Annie, Brian, Fraser and Eleanor. Each was recorded for 6 weeks at ages 2;0 and 3;0. This six-week period resulted in 30 hours of recordings, which were manually transcribed and annotated. This is an excellent source of high density information. Unfortunately, this corpus is not available to the research community.

2.1.2 The Evaluation of Language Learning Models

Data-driven, corpus-based natural language precessing systems must be thoroughly and robustly evaluated. Several factors make the evaluation of language learning systems difficult (van Zaanen and Geertzen, 2008). First, the training data provided to the system (i.e., the corpus used for induction) are usually limited. This is especially true when child-data are concerned, since even with high-density corpora it is assumed that the corpus reflects less than 10% of the utterances the child was exposed to during a very short period (see Rowland et al. (2008)). It is thus hard to evaluate the quality of the generalizations performed by the system. Second, while it is relatively easy to measure the ability of the model to account for new utterances, it is much harder to assess the proportion of the utterances that the model produces and are indeed grammatical.

In the computational linguistics community, language learning models are standardly evaluated using two measures adopted from Information Retrieval: *recall* and *precision*. Informally, recall measures the ability of the grammar to account for new utterances. Precision, on the other hand, measures the proportion of the selected items that the system got right; a task that has been found difficult to handle computationally and often requires either evaluation based on results obtained from alternative parsers (Berant et al. (2007)) or on human judgments (Solan et al., 2005; Brodsky et al., 2007). Within the cognitive linguistic paradigm, language learning models are usually evaluated by human judgments, rendering the evaluation both limited and subjective.

2.2 Computational Grammar Induction

In this section we describe works whose motivation is to design computational algorithms that induce grammars from data. A fundamental concept of such research is *language identification in the limit*, originally introduced by Gold (1967), and proven highly influential in subsequent works. This is a formal framework in which a learner is provided with a finite set of strings, S_+, that belong to some language, L (the positive sample) and, possibly, a finite set of strings, S_-, that do not belong to L (the negative sample). Learning is viewed as an infinite process. Each time a string from the positive sample is read, the learner provides a grammar for the language. A learner is said to have identified a language if given any set of positive (and, possibly, negative) examples of the language, the learner produces only a finite number of wrong representations, and therefore converges on the correct representation in a finite number of steps. However, the learner is not necessarily able to announce its correctness since a contrasting example to any representation could appear as a positive example in the following steps of the learning process. Gold (1967) proves that the class of regular languages is not learnable in the limit from positive examples only, but can be successfully identified from both positive and negative examples.

Following Gold (1967), Oncina and García (1992) propose a learning algorithm that identifies in the limit any regular language in polynomial time (in the size of the positive and negative samples). The input of the algorithm are fixed finite sets of positive and negative examples of an unknown regular language, L, and the output is a Deterministic Finite-state Automaton (DFA), M, such that in the limit (when $|S_+| \to \infty$ and $|S_-| \to \infty$), $L(M) = L$. The algorithm starts with a Trie (a prefix tree) which is a DFA generating the positive sample and then proceeds by trying to merge the states of this Trie. By the definition of a Trie, all the descendants of a node have a common prefix of the string associated with that node, and the root is associated with the empty string. Hence, the states in a Trie are lexicographically ordered, first by length and then by alphabetic order within every length. The algorithm takes advantage of this order and tries to merge

states with their (lexically) smaller siblings. The merge is approved only if the induced automaton rejects the negative examples. If the candidate state for merge has no siblings, this step is repeated, but now with all smaller states (i.e., in a higher level of the tree).

In this model, generalization of the positive examples is controlled by the negative sample to prevent merging of incompatible states. However, in the case of child language acquisition, the assumption is that explicit negative examples are not available (Gold, 1967). Hence, this framework cannot in itself be a plausible model of child language acquisition. However, as is well known from the language development literature, negative evidence *is* implicit in the input provided to the child, both through repetition and reinforcement (Clark and Lappin, 2011, chapter 3) and more importantly, via statistic cues: nothing prevents the child from employing a *stochastic* model of its language, using the frequency of the input utterances as biases (Clark and Lappin, 2011, chapter 5).

Indeed, to compensate for the lack of negative examples, learning algorithms that employ *stochastic* biases were proposed. The idea is that the statistical information of the positive sample can compensate for the lack of negative examples. In particular, such algorithms may create stochastic automata from positive examples. The language generated by the automaton, defined as the set of all strings accepted by the automaton with probability greater than 0, is a Stochastic Regular Language (SRL).

Carrasco and Oncina (1994) propose a learning algorithm that constructs a Deterministic Stochastic Finite-state Automaton (DSFA) from positive examples of a formal language in which the probability of every string is drawn from a well defined distribution. The proposed algorithm first builds the Prefix Tree Acceptor (PTA) with frequencies (weights) from the positive sample and assigns a probability to each transition. For each node q_i, n_i is defined as the number of paths going through q_i, $f_i(a)$ is defined as the number of arcs labeled by a outgoing from q_i, and $f_i(\#)$ is defined as the number of paths ending at node q_i. The quotient $f_i(a)/n_i$ estimates the probability of the transition from q_i labeled by a and $f_i(\#)/n_i$ estimates the probability of q_i being an accepting state.

Next, the algorithm tries to merge equivalent nodes in the PTA. Nodes are chosen to

be candidates for merge in lexicographic order. Two nodes are defined as equivalent if (1) they have similar outgoing transition probabilities for every symbol in the alphabet; (2) they have similar termination probabilities (probability that the node is accepting); and (3) the same holds for their daughters, accessible from the two nodes by same-labeled arcs (recursively). The model was evaluated on several known SRLs, and experimentally, the algorithm is efficient and needs relatively small samples in order to identify the regular language.

The learning algorithm proposed by Carrasco and Oncina (1994) performs very well in learning formal languages, since usually in formal languages the alphabet (Σ) is a small set of symbols and the strings in the language are long. In contrast, child language data are characterized by very large lexicons (the alphabet) and very short strings (the multi-word utterances). Our proposed learning algorithm is based on this algorithm; but it is adapted to reflect the properties of child language acquisition. In particular, our merge criteria are different, as they implement biases motivated by child language research.

Several works attempt to learn *context-free* languages from data (Lee, 1996). Here, also according to Gold (1967), learning a CFL from positive data only is undecidable. As in the case of regular languages, several suggested approaches use statistical models to overcome the lack of negative examples and induce Probabilistic CFGs from positive training data.

Stolcke and Omohundro (1994) propose a framework for extracting probabilistic grammars from corpora of positive examples. First, strings that are observed in the data are incorporated by adding ad-hoc rules to form an initial grammar; then, the grammar is made more concise by merging some of the rules. This work presents two incarnations of the technique, one in which the models are probabilistic context-free grammars (PCFGs), and another in which they are hidden Markov models (HMMs). In the former, rules are merged by identifying non-terminal symbols A and B if the rule A $\rightarrow$ B is in the grammar; this is the source of generalization in this model. In the latter, two HMM states are merged to a state that inherits the union of their transitions (and emission

probabilities). In both cases, the prior probabilities are optimized by minimizing their description length. Experiments with the suggested algorithms on natural language data have yielded mixed results. A fundamental problem is that available data are typically sparse relative to the complexity of the target grammars, i.e., not all constructions will be present with sufficient coverage to allow the induction of correct generalizations.

The ADIOS system (Solan et al., 2005) implements a novel algorithm that learns a complex context-free grammar (CFG) from raw data. Based on a graph representation, the algorithm performs segmentation and generalization of the input simultaneously. The system was applied to several types of data, both linguistic (including transcripts of children and their caregivers) and non-linguistic (protein sequences). The results show that ADIOS is superior to other grammar induction algorithms that can learn from raw data.

EMILE (Adriaans, 1992, 2001; Adriaans and Vervoort, 2002) is another algorithm that learns CFG from raw data. The EMILE model attempts to learn the grammatical structure of a language from positive examples, without prior knowledge of the grammar. It is based on the idea that expressions of the same (syntactic) type can be substituted in the same context, and hence it searches for clusters of expressions and contexts in the input, interpreting them as grammatical types. The model then generalizes the sample and learns rules of a context-free grammar.

We maintain that CFG (or PCFG) is too powerful a formalism for modeling early syntactic knowledge, and a much more constrained model is more fitting to the type of utterances observed in the data. In our work, learning is based on the linguistic input provided to the child, which is viewed as positive examples only. The negative result of Gold (1967) is irrelevant for our work due to the different assumptions: the learning algorithm can be biased in a way that compensates for the lack of negative examples. In the above models, the given positive samples (the training data) are first read in their entirety and then generalized. In contrast, children acquire their knowledge of language incrementally, processing the utterances they hear one at a time. Hence,

the suggested models ignore an important facet of child language acquisition, namely its on-line manner. Crucially, the above works do not give a conspicuous expression of psycholinguistic observations that attempt to determine generalizations over the course of child language acquisition (see section 4.1 for more details). Next, we survey the other research direction, namely works that are motivated by psycholinguistic observations.

2.3 Cognitively Motivated Language Learning Algorithms

We survey in this section some prominent approaches that use computational modeling either to implement and simulate or to evaluate cognitively-motivated theories of language acquisition.

In a series of works, Freudenthal et al. (2006, 2007, 2009) develop the MOSAIC (Model of Syntax Acquisition in Children) paradigm. This model takes as input corpora of transcribed child-directed speech and learns to produce as output utterances that become progressively longer as learning proceeds. The model is based on a hierarchical network in which more deeply embedded nodes represent longer utterances, and where links connect nodes to form certain generalizations. Unfortunately, the model is not described with sufficient rigor and precision that would enable its reproduction. One detail that is emphasized, however, is that the same corpus is given to the learner several times. This is a very unusual requirement in a model of child language acquisition, and indeed, according to Freudenthal et al. (2010, page 650), "MOSAIC is best viewed, not as a realistic model of the language acquisition process itself, but as one of many possible ways of implementing an utterance-final (and in the current version of the model, utterance-initial) bias in learning."

Borensztajn et al. (2008) extract a child's grammar from transcripts of child speech and examine grammatical abstraction in child language. The goal is to show that abstraction increases with age. Grammatical abstraction is defined here as the relative number of variable slots (utterances that includes variation in the same position) in the productive units (the nonterminals) of the grammar. The acquisition of constructions with variable

slots by children leads to the beginning of abstraction and category formation, and marks the beginning of grammar.

The study is conducted on the Brown (1973) corpus from the CHILDES database. This corpus contains transcribed longitudinal recordings of three children, Adam, Eve and Sarah. Each child's corpus is split into three parts of roughly equal size, representing three consecutive time periods. Then, child-directed speech, any annotation or comments and all the incomplete and interrupted sentences are removed. Only grammatical development *within* each child is compared, and not *between* them.

Tree Substitution Grammar (TSG) is used to store the multi-word syntactic primitives as well as single unit primitives. The generative components of a TSG are tree fragments of arbitrary size and depth. These fragments contain variable slots for syntactic categories, making them suitable for representing abstract constructions, or abstract rules. TSGs are used extensively in the framework of Data Oriented Parsing (DOP) (Bod, 2006), which facilitates parsing of new sentences using fragments from sentences observed in a corpus. The probabilities of the fragments are determined using the algorithm of Borensztajn and Zuidema (2007). Then, standard statistical parsing techniques can be used to find the most probable derivation of any sentence in a corpus. In this work, DOP is used as a statistical approach for discovering the constructions in child language. When the most probable derivation of a child's utterance is determined, it becomes possible to quantify the properties of the child's grammar at various stages.

The results are that the number of nodes, number of nonterminals, number of terminals, depth and other relevant quantities, all increase with age. The main conclusion is that abstraction, defined as the relative number of non-terminal leaves in multi-word utterances, increases with age.

The resulting grammars in this work are precise and testable. A main drawback of this approach, however, is that the learning model assumes that the input is morphologically and syntactically annotated, implying that abstract structural representations are a prerequisite for language acquisition. In the model we propose here, the input consists of

the utterances the child is exposed to, namely, raw data with no annotation.

Bannard et al. (2009) also extract a grammar from transcripts of child speech using a computational model. Grammars that reflect children's speech at 2 and 3 years are extracted from the corpus, and then novel utterances are generated using these grammars. They use transcriptions of 28+ hours of two children's speech; 90% of the transcripts of each child are used to produce the grammar and the remaining 10% are used for evaluation. Bannard et al. (2009) define a *concrete sign* as a single word, a part of an utterance or the whole utterance. A second kind of sign is an utterance with one or more slots, which is defined to be a *schema*. Both *concrete signs* and *schemas* can fill *slots*. Filling a slot is called *INSERT*, and this is the only operation employed by their algorithm.

The rules in the grammar (a context-free grammar over a single non-terminal symbol, X) are generated based on utterances that share lexical material. For example, assume that in the corpus the following 2 utterances occur: (i) *I have that one*, (ii) *Mommy-'s have a tiny one*. The rule $\{X \rightarrow$ X have X one $\}$ is added to the grammar since *X have X one* is defined as a *schema with slots*, and *I*, *Mummy-'s*, *that* and *a tiny* are defined as *concrete signs*.

The results of this study are that approximately 80% of the test utterances can be parsed by the resulting grammar. In more detail, the results are: 84% for Brian and 75% for Annie at age 2 and 70% for Brian and 80% for Annie at age 3. Beside coverage, in order to check how well the models predict the test data, the *perplexity* of the models is calculated. Perplexity is a measure of how well a probability distribution over a set of events (in this case, words of utterances) matches the distribution of the events seen in some data, i.e., how surprised a model is by that data. The lower the perplexity, the better the fit. The perplexity results of the output grammar in this study show that the model provides a good fit to the data. The details are sketchy, but in order to reliably compute the perplexity of a language model, huge amounts of data are needed (Jelinek et al., 1977; Bahl et al., 1983). We define a related but different measure of the fitness of

our model to test data.

In sum, each of these models, although they are cognitively motivated, is lacking in terms of modeling early child language acquisition. However, it is clear that taking into account cognitive principles is advantageous when modeling language acquisition. In this context,one question that we aim to test is to which extent a purely cognitively motivated model, one that is not designed from a computational perspective, can yield satisfactory results. For this purpose, we focus in the next section on the implementation of such a model.

3 Implementing a Cognitively Motivated Model: The Traceback Method

Several recent studies (Lieven et al. (2003), Dąbrowska and Lieven (2005), Lieven et al. (2009)) propose the *Traceback Method* as a model of early language acquisition, explaining some of the phenomena associated with children's ability to generalize previously-heard utterances and generate novel ones. This method is specified in formal terms that make it amenable to computational implementation; yet it is evaluated in a way that could benefit from better acquaintance with accepted computational evaluation techniques.

Lieven et al. (2003) lay down the principles of what is later termed "The Traceback Method" (henceforth, TBM) as a way of tracing "novel utterances in the test corpus back to strings in the main corpus from which they could have been constructed" (Bannard and Lieven, 2009)). Lieven et al. (2003) thus define five types of operations which the child can use to construct a new utterance from fragments of previously-heard linguistic material: *substitute, add-on, drop, insert*, and *rearrange*. They then use these operations to replicate previous results while focusing on a high-density corpus consisting of 5 hours of recordings per week for one child at the age of 2;00. Their findings show that only one third of the multi-word utterances of the child were novel, and three quarters of those can be accounted for by one operation only, that is, some sort of manipulation on previous

utterances.

Dąbrowska and Lieven (2005) (henceforth, D&L) identify two problems with the procedure suggested by Lieven et al. (2003): first, "the method does not provide an explicit description of the child's linguistic knowledge." In other words, no explicit model of linguistic knowledge, or *grammar*, is defined. Second, "the method is too unconstrained since the five operations defined by the authors made it possible, in principle, to derive any utterance from any string." That is, the model is *over-generating*. To overcome this problem, they propose to rely on only *two* operations: *juxtaposition* and *superimposition*, to account for the acquisition of increasingly more abstract constructions, specifically, WH questions. Again, D&L use a dense corpus, consisting of four developmental corpora for two English speaking children, Annie and Brian, each recorded for 6 weeks at the age of 2;0 and 3;0. Results show that approximately 90% of children's WH questions can be produced by this model, and in line with previous studies, that 11%-36% are direct repeats of utterances that already occurred in the main corpus (11% for Annie and 36% for Brian at age 2;0). Moreover, at the age of 2;0, the majority of both children's utterances require only one operation for a successful derivation (55% for Brian and 66% for Annie). At age 3;0, there are considerably more utterances requiring two or (especially in the case of Annie) more operations, although a large proportion (25% for Annie and 43% for Brian) of the children's WH questions can still be derived by applying a single operation.

It thus seems that the TBM is able to provide a constructivist account for the majority of the childrens' interrogative utterances on the basis of a lexically specific grammar that can be manipulated by two general operations. Importantly, this model has been recently applied to *all* child utterances in eight hours worth of speech productions of the same two children and of two additional English speaking children, with highly consistent results (Bannard and Lieven (2009), and see also Vogt and Lieven (Forthcoming)). The TBM was able to trace back between 83.1% to 95% of all child utterances, with around 25-40% of utterances constituting exact repetitions, and between 36-48% of utterances requiring

just one operation for derivation. In another recent study, Lieven et al. (2009) corroborate these results for the case in which the model is only trained on child speech.

The TBM is thus an increasingly applied method which can be used to empirically study child language acquisition from a usage-based perspective. The question, however, still remains whether this method is indeed sufficiently constrained. In this section the generative power of the TBM is computationally evaluated, under the view that computational simulations are a useful tool for systematically representing any information processing task (Feldman, 2004; Buttery, 2006; Alishahi, 2011). As such, we do not aim to examine whether the TBM is up to the task of representing the process of language acquisition more than other suggested approaches. Rather, we focus on identifying the strengths and drawbacks of the model as they emerge through a computationally supported investigation.

3.1 Re-implementation of the TBM

The version of the model specified in D&L (and, more recently, in Bannard and Lieven (2009)) particularly lends itself to such an evaluation since it is clear how to reformulate it as an algorithm.[1] As noted above, the model attempts to generate the child's novel utterances using what the child heard or said before. Given a target utterance T, the model works as follows:

1. Identify all *component units* with respect to the given target utterance, T. A component unit is defined as a string u that is a substring of the target utterance, T, and that occurs at lease twice in the learning corpus.

2. If T is available to the child as a component unit (i.e., it was produced before by either the child and/or an adult), exit: the derivation is defined by this unit.

3. Otherwise, choose the longest possible string, S, which is a substring of T and

[1] Unlike most psycholinguistic models, which are more vague and represent grammar implicitly and informally, sometimes in a very ad-hoc manner.

which was observed twice or more in the training material, with variation in the same position; this position is referred to as a *slot*. The substring S is defined as the *frame*. Then, choose the longest available component unit u for the slot; this unit is referred to as a *filler*. The derivation is defined as $Superimpose(S, u)$. The superimposition operation is constrained such that the filler must have the same semantic label as one of the occurrences of the frame in the training corpus.

4. If no frame was found, choose S to be the longest possible string which is a substring of T that is available to the child as a component unit.

5. If more words exist in T, repeat the algorithm above from step 2 for the remaining utterance, R. The derivation is then defined as $Juxtapose(Superimpose(S, u), R)$, where $Juxtapose(x, y)$ is defined as a linear concatenation of two utterances x and y according to their order in the target utterance. If no frame was found, the derivation is defined as $Juxtapose(S, R)$.

Note that the algorithm above implements the "rules" specified by D&L (and re-iterated in Bannard and Lieven (2009)), and in particular guarantees that derivations use the minimum number of operations.

As the first step of our computational evaluation, we re-implemented this algorithm, with some necessary modifications. Not having access to the original dense corpus, we settled instead for the online corpora of Brown (1973) and Suppes (1974), both available from CHILDES (MacWhinney, 2000). Similarly to the original method, each corpus was divided into 2 parts, *test* and *main*. The files in each corpus are ordered chronologically; we consider all child utterances in the last file as the test corpus, and all earlier files, along with the adult utterances in the last file, as training utterances. The size of each corpus (the number of *multi-word* utterances) is detailed in table 1 below.

In addition, all data in the corpora used for the TBM were manually annotated with semantic labels, following the assumption that children store pairings of phonological forms and semantic representations. This annotation includes seven types of tags:

Corpus	main corpus		test corpus	
	Utterances	Tokens	Utterances	Tokens
Eve	19,536	85,350	224	875
Adam	20,443	75,213	792	3,166
Sarah	6,425	23,330	106	252
Nina	38,736	175,748	458	1,632

Table 1: Size of the corpora

REFERENT, PROCESS, ATTRIBUTE, LOCATION, DIRECTION, POSSESSOR, and UTTERANCE. To approximate these semantic labels, we developed a mapping based on Part-of-Speech and dependency relation tagging available in CHILDES. These tags were produced by GRASP (Sagae et al., 2010), a dependency parser for identification of grammatical relations such as Subject and Object (among others) in child language transcripts. Our assumption is, following work both in computational cognitive science and in cognitive linguistics (e.g., Rapaport (1988); Croft (2001)) that such grammatical relations can be identified with semantic roles of the type used by the TBM. Thus, for example, the combination of the Part-of-Speech category PRONOUN with the grammatical relation SUBJECT yields a good approximation of the tag REFERENT.

The results of our re-implementation, which are summarized in Table 2, show that between 70% and 88% of the children's utterances in the test corpus can be derived using this algorithm. Our results thus show that the bulk of the target data are generated by the computer program implementing the TBM. They also show that out of the successfully derived utterances, between 24% (Eve) and 43% (Sarah) were exact repetitions ("Fixed" strings) of previously heard utterances. It is thus quite striking that even in a corpus of sparse data, about one third of all test utterances are exact repetitions of utterances that were previously heard or produced by the child. These results are compatible with those obtained in all TBM studies mentioned above (and especially in Bannard and Lieven (2009) who use this method to trace back all the utterances in the child test corpus), and can be treated as supporting evidence to the claim that children in fact learn chunks from what they hear. Table 2 also shows that most of the utterances (46%-56%) are derived

by using Superimposition while only 9%-20% are derived by Juxtaposition. In addition, similarly to the original results in all TBM studies, most of the test utterances can be derived using only one or two operations.

Corpus		Derived		Fixed		Superimpose		Juxtapose		Num. of Ops		
	size	#	%	#	%	#	%	#	%	1	2	> 2
Eve	224	155	69.19	37	23.87	87	56.13	32	20.64	95	11	12
Adam	792	675	85.22	183	27.11	312	46.22	185	27.40	362	70	60
Sarah	106	94	88.68	40	42.55	45	47.87	9	9.57	54	0	0
Nina	458	401	87.55	119	29.67	217	54.11	66	16.46	230	27	25

Table 2: Evaluation results, re-implementation of Dąbrowska and Lieven (2005)

Our results are still lower than those reported in the original TBM works. For example, Bannard and Lieven (2009) were able to generate as many as 95% of all child utterances. One main reason for this discrepancy could be that our analysis is carried out on a much sparser corpus, which makes the induction task more difficult for the system. This confirms the view that relying on a dense corpus affects the variablity of the syntactic structures in use (Demuth, 2008).[2] Moreover, our "semantic" annotation is done automatically rather than manually: we merely approximate meaning by resorting to the syntactic annotation of the data. While the annotation of the Eve corpus was for the most part done manually, the other corpora were annotated automatically, which means that many of the tags may be wrong. Of course, translating the annotations to semantic tags introduces yet another level of noise. Even with these caveats, however, our results are comparable to the original ones.

3.2 Evaluation of the TBM

In lieu of an accepted method for evaluating the precision of language learning algorithms (see section 2.1.2), we suggest a method to assess the level of over-generation of the model. To assess over-generation, we repeat the same procedure, training on the same data but evaluating on child utterances (longer than one word) *in reverse word order*. This allows

[2]Our results show a contradicting phenomenon; see section 5.

us to evaluate the ability of the model to generate presumably ungrammatical utterances.

As can be seen in Table 3, approximately 68-89% of the *reversed* multi-word utterances can be derived, indicating a serious over-generation problem.

Corpus				fixed		Superimpose		Juxtapose		OP		
	size	#	%	#	%	#	%	#	%	1op	2op	More
Eve	224	153	68.30	5	3.26	64	41.83	84	54.90	93	28	27
Adam	792	659	83.20	43	6.52	226	34.29	403	61.15	335	146	135
Sarah	106	94	88.68	8	8.51	63	67.02	23	24.47	82	4	0
Nina	458	389	84.93	22	5.65	143	36.76	227	58.35	241	72	54

Table 3: Evaluation results, reversed utterances

Thus, while the TBM is able to learn a significant portion (70-88%) of all child utterances even for a non-dense corpus, this very ability may be interpreted as evidence of the system's over-generative power; indeed, it can also produce around three quarters of what are most likely ungrammatical structures. Consider, for example, the (reversed) utterance *"Boston to go"*. This is generated by juxtaposing *"Boston"* with the component *"to go"*. A single juxtaposition operation suffices also for generating the reversed utterances *"Gloria you're"* and *"more no"*. Superimposition is used to generate *"it eat"* as an instance of the schema *"it PROCESS"*, where *"eat"* fills the PROCESS role (the original utterance, of course, is *"eat it"*.) And the use of one superimposition and one juxtaposition generated the utterance *"more buy to have"* as an instance of the schema *"PROCESS to have"*.

Several factors may contribute to this over generation. First, the TBM ignores the frequency with which utterances are presented to the learner. It thus ignores frequency effects on the entrenchment of linguistic structures. A model that is more sensitive to frequency effects may better fit the data (see section 4.1). Second, and more crucially, there is no principled way to determine the set of operations that the model consists of: Lieven et al. (1997, 2003) define a set of five operations, which are reduced to two by D&L (also replicated by Bannard and Lieven (2009)). Lieven et al. (2009) retain two operations, but the *superimposition* of D&L is here replaced by a slightly different *substitute*

operation. Finally, Vogt and Lieven (Forthcoming) use three operations (including both superimposition and substitution). Third, a major contribution to over-generation stems from the fact that the number of operations (of any kind) allowed in the derivation of any target utterance is not limited in any way. Finally, some of the operations probably need to be more constrained. While superimposition is constrained by the type of the slot, juxtaposition is not, and in particular, it allows either order of combination. This means that the child is as likely to concatenate two strings in one order as she is to use the other. Bannard and Lieven (2009) restrict the application of *add* by saying that it is "only allowed if the component unit could, in principle, go at either end of the utterance." It is unclear how this is determined (or how the child could know it.)

In conclusion, while the TBM may be a plausible model of early syntactic acquisition, its evaluation is lacking. We believe that an appropriate computational model of early language acquisition must be extremely constrained in its expressive power. In the next section, we present a much more constrained model, based on a restricted variant of finite-state automata, that can account for the type of generalizations exhibited by early language learners without resorting to the over-generalization we point to above.

4 A DSFA Model of Early Syntactic Acquisition

We present a model of early syntactic acquisition, based on deterministic stochastic finite-state automata (DSFA), whose development is influenced by well-established psycholinguistic principles detailed in section 4.1. The purpose of our study is to combine seminal cognitively-motivated child language learning hypotheses with a computational model based on computational learning theory, and demonstrate that a much less expressive formalism (here, a constrained finite-state automaton) can suffice for modeling processes of early child language acquisition. We present in section 4.2 an algorithm for learning language from data implementing these insights. Like other cognitively-motivated language learning models, the input to our algorithm is child and caregiver transcribed

utterances. The output of the algorithm is a (formal) grammar represented as a DSFA, as in the grammar induction approach; this representation of the child's syntactic knowledge is thus mathematically well defined (section 4.2). We prove some of the properties of the algorithm in section 4.3 and exemplify its operation in section 4.4. Results and evaluation are deferred to section 5.

4.1 Motivating Principles

Our algorithm follows some of the insights of Oncina and García (1992) and Carrasco and Oncina (1994), who also construct a DSFA from raw data. We, too, express generalization by successively merging DSFA states that satisfy a *merge criterion*. The main innovation of our work is that the merge criteria we employ reflect the psycholinguistic insights discussed below.

Language development typically follows a predictable sequence of stages, in increasing complexity and originality of the child's utterances (Brown, 1973; Elman, 1993). The incremental trajectory of natural language acquisition processes call for an *on-line*, *incremental* computational model which processes its input in chronological order rather than as an unordered set. Consequently, and unlike many other language learning algorithms, the DSFA we construct is dependent on the order of the data it receives.

Psycholinguistic research suggests that early language is highly constrained, that utterances are short and repetitive (Brodsky et al., 2007) and that deeply-nested structures emerge later, and even then are very constrained. In particular, the need for recursion in child language is very limited (Diessel and Tomasello, 2005; Bannard et al., 2009; Luuk and Luuk, 2011). Hence, models of early syntactic acquisition that do not allow recursive structures may perform better than more expressive models. Indeed, one of the features of our model is acyclicity.

A considerable body of psycholinguistic research has shown that children's early multiword utterances are constructed using rote-learned phrases, or *lexically based patterns*, which at some point along development evolve into less specific constructions that contain

some level of abstraction (e.g., MacWhinney (1975); Peters (1983); Lieven et al. (1997); Tomasello (2003)). Under this suggestion, children learn constructions that contain both lexically specific frames and open-ended *slots* or *schemas*, into which some category of words can be integrated. This means that before children reach linguistic productivity, they must first go through stages where most of their knowledge is restricted and non-novel. Consequently, a computational model of early language acquisition must be able to generalize previously-heard utterances and generate novel ones using rote-learned phrases and lexically based patterns.

Also, several studies (Tomasello, 2003; Diessel, 2007; Behrens, 2006) reveal that the frequency of the input (*entrenchment*) may provide an explanation for generalization. Computational models of language acquisition must therefore reflect input frequency.

Under the assumption that children's early multiword utterances are constructed using rote-learned phrases, or *lexically based patterns*, our algorithm relies on similarity of word contexts in order to find generalizations. State merging occurs in the algorithm as a result of shared lexical material with or without a slot. In more detail, if the input includes the utterances $w_1\ w_2\ w_3$, $w_1\ w_4\ w_3$ and $w_1\ w_5\ w_3\ w_6$, the assumption is that the lexically-based pattern $w_1\ SLOT\ w_3$ is created by the child and that w_2, w_4, w_5 occur in the same context and are hence in the same syntactic category. The resulting automaton reflects that by on-line merging of states that have incoming edges labeled by w_2, w_4, w_5. In this case, only 3 utterances are present, but in the general case, there may be several of them, and hence their probabilities are also taken into account. The probabilities of w_2, w_4, w_5 are actually calculated using their frequency in the input which is expressed in the model by weights associated with DSFA edges.

Slobin (1973) lists several principles that govern the way children acquire their early syntactic structures. One principle (Slobin, 1973, page 197) is that "The standard order of functor morphemes in the input is preserved in child speech". Moreover, "word order in child speech reflects word order in the input language". Clearly, then, re-ordering operations should be discouraged in a plausible model of language acquisition. These

principles are reflected in our algorithm by not allowing any reordering among words.

Both Slobin (1973) and MacWhinney (1978) claim that the child will attempt to "avoid exceptions". Specifically, this means that a child will attempt to acquire a single form to express a single function. Hence, in a probabilistic model of language acquisition, utterances with unusual lexical form should be less significant than common structures. In our model, DSFA paths reflecting more frequent input sequences acquire higher probability than paths corresponding to less frequent utterances, thereby implementing a bias towards more regular forms.

4.2 The Language Acquisition Algorithm

In this section we present the algorithm formally. We begin by defining some basic notions.

Definition 1. ***Deterministic Stochastic Finite Automaton.*** *A* DSFA *is a quadruple* (Q, Σ, P, q_0)*, consisting of an alphabet* Σ*, a finite set of states* Q*, an initial state* $q_0 \in Q$*, and a set* P *of probabilities* $p_{ij}(w)$ *giving the probability of a* w*-transition from node* q_i *to* q_j*, and of probabilities* p_j^f *giving the probability that* q_j *is a final state, such that for every node* $q_i \in Q$*,* $\sum_{q_j \in Q} \sum_{w \in \Sigma} p_{ij}(w) + p_i^f = 1$*. Furthermore, for every node* $q_i \in Q$ *and symbol* $w \in \Sigma$*, there exists at most one node* q_j *such that* $p_{ij}(w) > 0$ *(the deterministic limitation).*

Definition 2. ***The transition function.*** *As a result of the deterministic limitation of the automaton, a* transition function*,* $\delta : Q \times \Sigma \to Q$*, can be defined. This function gives the state* q_k *for a given state* q_i *and a symbol* $w \in \Sigma$*, if* $p_{ik}(w) > 0$*. In addition, we define* $\hat{\delta} : Q \times \Sigma^* \to Q$ *as follows:* $\hat{\delta}(q, \epsilon) = q$ *for every* $q \in Q$*; and* $\hat{\delta}(q, w_1 ... w_k) = q'$ *if* $\delta(q, w_1) = q''$ *and* $\hat{\delta}(q'', w_2 ... w_k) = q'$*. Below, we abuse notation by referring to* $\hat{\delta}$ *as* δ*.*

Definition 3. ***Language of a DSFA.*** *Let* M *be a DSFA, and* $w = w_1 w_2 ... w_n$ *be a string over the alphabet* Σ*.* w *is* generated *by* M *if a sequence of states,* $q_{i_0}, q_{i_1}, ..., q_{i_n}$*, exists in* Q *such that (1)* $i_0 = 0$*, (2)* $q_{i_x} = \delta(q_{i_{x-1}}, w_x)$*, for* $x = 1, ..., n$ *and (3)* $p_{i_n}^f > 0$*. As*

a result of the deterministic nature of the automaton, if such a sequence of states exists, it is unique. The probability of this sequence is $p(q_{i_0}, q_{i_1}, ..., q_{i_n}) = \prod_{j=0}^{n-1} p_{i_j,i_{j+1}} \times p^f_{i_n}$. *Consequently, the probability* $p(w)$ *is defined as* $p(w) = p(q_{i_0}, q_{i_1}, ..., q_{i_n})$. *The* language of the DSFA *is* $L(M) = \{(w, p(w)) \mid p(w) > 0\}$

The learning algorithm is presented in figure 1. Its input is a sequence of strings (the *training* or *main* corpus); its output is a DSFA.

Input: Sequence of strings (the main corpus)
Output: Stochastic DFA

Let $M = (Q, \Sigma, P, q_0)$ be the automaton including one path from $q_0 \in Q$ to $q_{end} \in Q$, representing the first utterance in the input; $final(q_{end}) \leftarrow 1$;
foreach $q \in Q$ **do** $in(q) \leftarrow 1$; $final(q) \leftarrow 0$;
foreach *edge* $(q_i, \sigma, q_j) \in M$ **do** $weight(q_i, \sigma, q_j) \leftarrow 1$;
foreach *utterance* $w = w_1\ w_2\ ...\ w_n$ *in the input* **do**
    $in(q_0) + +$;
    foreach $i \leftarrow 1$ *to* n **do**
        Let $q = \delta(q_0, w_1 ... w_{i-1})$;
        if $\delta(q, w_i)$ *is defined* **then**
            Let $q' = \delta(q, w_i)$;
            if $i \neq n$ *and* $q' = q_{end}$ **then**
                $r \leftarrow AddNode(q, w_i)$; Change (q, w_i, q') to (q, w_i, r);
                $in(r) = weight(q, w_i, q')$; $in(q_{end}) - = weight(q, w_i, q')$;
                $final(r) = weight(q, w_i, q')$; $final(q_{end}) - = weight(q, w_i, q')$;
                continue;
            $weight(q, w_i, q') + +$; $in(q') + +$;
            if $i = n$ **then** $final(q') + +$;
            if $i \neq n$ *and for some* $\sigma \in \Sigma$ *and* $r' \in Q$, $\delta(q, \sigma) = r'$ *and* $\delta(r', w_{i+1}) = r''$ *for some* $r'' \in Q$ **then** $Merge(q', r')$;
        else
            if $i \neq n$ *and for some* $w \in \Sigma$ *and* $r \in Q$, $\delta(q, w) = r$ *and* $\delta(r, w_{i+1})$ *is defined* **then**
                Add an edge (q, w_i, r) with $weight(q, w_i, r) = 1$; $in(r) + +$;
            else
                if $i \neq n$ **then** $AddNode(q, w_i)$;
                else Add an edge (q, w_i, q_{end}) to δ with $weight(q, w_i, q_{end}) = 1$;
                $in(q_{end}) + +$; $final(q_{end}) + +$;
foreach *edge* (q_i, σ, q_j) *in* M **do** $p_{i,j}(\sigma) \leftarrow (weight(q_i, \sigma, q_j)/in(q_i))$;
foreach *node* $q_i \in Q$ **do** $p^f_i \leftarrow (final(q_i)/in(q_i))$;

Algorithm 1: Language Acquisition Algorithm

As the positive instances (the training utterances) come in, each utterance is added

Input: M, a DSFA, $q \in Q$, $w \in \Sigma$
Output: New node in M
Add a fresh node r to Q;
Add an edge (q, w, r) to δ;
$weight(q, w, r) \leftarrow 1$;
$in(r) \leftarrow 1$;
$final(r) \leftarrow 0$;
return r;

Algorithm 2: Add Node

Input: M, a DSFA, and Two states q_1, q_2 to be merged in M
Output: boolean
Assumption: q_1, q_2 are siblings, i.e., for some $a, b \in \Sigma$ there exists $q \in Q$ such that $\delta(q, a) = q_1$ and $\delta(q, b) = q_2$;
foreach $a \in \Sigma$ *such that* $\delta(q_1, a)$ *and* $\delta(q_2, a)$ *are defined* **do**
    Let $q_1' = \delta(q_1, a)$ and $q_2' = \delta(q_2, a)$; Let $w_i = weight(q_1, a, q_1')$ and $w_j = weight(q_2, a, q_2')$;
    if *different*$(in(q_1), w_i, in(q_2), w_j)$ **then** return false;
    if *different*$(in(q_1'), final(q_1'), in(q_2'), final(q_2'))$ **then** return false;
    if *Not Merge*(q_1', q_2') **then** return false;
foreach *edge* (r, σ, q_2) *in* M **do** add an edge (r, σ, q_1) to M with the same weight;
foreach *edge* (q_2, σ, q) *in* M **do** add an edge (q_1, σ, q) to M with the same weight;
$in(q_1) += in(q_2)$; $final(q_1) += final(q_2)$; remove q_2 and all its incoming and outgoing edges;
return true;

Algorithm 3: Merge

to the automaton by adding its words one by one. Then, this utterance is represented in the resulting automaton by a path, leading from the initial state to a final state, whose arcs are labeled by the words that make up the given utterance. During the construction of the automaton, while each string in the input is added as a path to the automaton, for every node $q_i \in Q$ two quantities are manipulated: $in(q_i)$ is the number of strings in the input whose prefix is represented by a path leading from q_0 to q_i; and $final(q_i)$ is the

Input: $n_1 = in(q_1)$, $w_1 = weight(q_1)$ or $final(q_1)$, $n_2 = in(q_2)$, $w_2 = weight(q_2)$ or $final(q_2)$
Output: boolean
return $\left|\frac{w_1}{n_1} - \frac{w_2}{n_2}\right| > \sqrt{\frac{1}{2} log \frac{2}{\alpha}} (\frac{1}{\sqrt{n_1}} + \frac{1}{\sqrt{n_2}})$

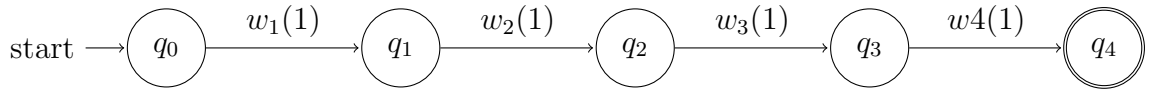
Algorithm 4: Different (Carrasco and Oncina, 1994)

number of strings in the input which are represented in the automaton by a path leading from q_0 to q_i. In addition, for every edge (q_i, σ, q_j) we define a weight, $weight(q_i, \sigma, q_j)$, as the number of strings in the input which are represented in the automaton by a path that includes the edge (q_i, σ, q_j).

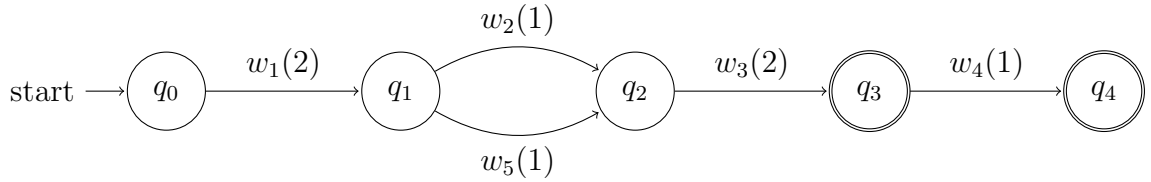
During the addition of an utterance to the automaton, successive state merging is done online according to a merge criterion that implements generalization. Merge is not done between two nodes at different levels in the automaton (different distances from q_0) to avoid the creation of cycles.

Consider algorithm 1. Utterances in the main corpus are added to the DSFA one by one (step 4). Each utterance $w = w_1 w_2 ... w_n$ is added by adding each of its words successively (step 6– 20). Adding a word w_i can result in either of three outcomes: (1) adding weight to an existing edge (labeled by w_i, step 12 in the algorithm); (2) adding an edge between two existing states (steps 16–17); or (3) adding a new edge between an existing state and a newly added state (steps 10, 19).

Outcomes (1), (2) add a new word as part of an existing path in the automaton, thereby *merging* part of the current utterance with material already represented in the automaton. We refer to this process as *online* merge. Note that outcome (2) also implements a generalization. For example, assume that the input includes two utterances: w_1 w_2 w_3 w_4 and w_1 w_5 w_3. Initially, the DSFA includes only one path q_0, q_1, q_2, q_3, q_4 which represents the first utterances in the input:



Then, when the algorithm processes the next utterance, for $i = 2$, a new edge (q_1, w_5, q_2) is added between two existing states as a result of steps 16–17 and the DFSA is now:



Indeed, in this case, online merge causes a generalization: the DSFA generates the utterances $w_1\ w_5\ w_3\ w_4$ and $w_1\ w_5\ w_3$ which did not occur in the input.

The algorithm successively maps words in the current utterance to existing paths in the automaton, starting in q_0 for the first word. The main loop of the algorithm (step 6) handles the i-th word w_i by trying to add it as an outgoing edge from state q (step 7). First, the algorithm checks if w_i matches the label of an outgoing edge from q (step 8). If it does, an online merge occurs: instead of adding the new word as a new edge, it is merged with an existing edge in the automaton. Online merge is performed locally between an edge that should represent one word in the input utterance and an existing edge in the automaton, so it explicitly ends when the input utterance ends.

Implementing even more generalizations, the algorithm also performs an *offline* merge, which is another type of state merging using the *Merge* function (step 14). Offline merge occurs between two existing paths in the automaton. Each path represents one or more utterances that have already been processed by the algorithm. When the algorithm determines that two *sister* states, q_1, q_2, are "similar"(in term of their outgoing edges, see below), offline state merging is performed: all edges that enter q_2 are changed to point to q_1, all edges that leave q_2 are changed to become outgoing edges from q_1, and q_2 is removed from the automaton (steps 7–8 in algorithm 3). Offline merge also involves recursive calls to check the descendant nodes of the two merged nodes. Hence, offline merge actually merges paths rather than single edges. Offline merge involves only a finite number of recursive calls, because when q_i, q_j are candidates for an offline merge, necessarily for some $w_1, w_2 \in \Sigma$ there exists $q \in Q$ such that $\delta(q, w_1) = q_i$ and $\delta(q, w_2) = q_j$ (conditions 8, 14 in algorithm 1). In other words, offline state merging is only done between siblings. Offline merge is performed on *similar* states: statistical similarity is verified using an algorithm called *Different* (algorithm 4) which is adapted from Carrasco and Oncina (1994). This algorithm checks the similarity between the labels and the probabilities of all outgoing edges from a given pair of nodes. When this similarity measure exceeds a given threshold, merge is licensed.

At the end of the algorithm (step 21), when each string in the main corpus is represented by a path (starting at q_0) in the output automaton, the transition probabilities are calculated. We follow Carrasco and Oncina (1994) and define the probability of an edge (q_i, σ, q_j) as its weight ($weight(q_i, \sigma, q_j)$) divided by $in(q_i)$.

4.3 Properties of the Algorithm

We now prove some properties of the algorithm. In the following discussion, let $M = \langle q, \Sigma, P, q_0 \rangle$ be the result of applying algorithm 1 to a non-empty sequence of utterances.

Lemma 1. *M is connected.*

Proof. The algorithm starts from a connected directed graph (representing the first utterance in the input) and each time a new state is added, an edge from some existing state to this new state is created (steps 11, 19 in the algorithm). Also, there is no step that deletes edges. □

We now show that M is acyclic. Specifically, we show that the algorithm starts from an acyclic directed graph and adds edges (q, w, r) such that $q, r \in Q$ only if a path from q to r already exists in the graph.

Lemma 2. *When algorithm 1 adds an edge (q, w_i, r) to the automaton, either some edge (q, w_j, r) already exists or r is a new node.*

Proof. The algorithm only adds an edge between two existing nodes in step 17 and in the *Merge* function. Step 17 adds the edge (q, w_i, r) such that $q, r \in Q$. It is only performed if for some $w \in \Sigma$ and $r \in Q$, $\delta(q, w) = r$ (step 16).

In addition, the *Merge* algorithm (algorithm 3) can add edges between existing states. In step 14 of algorithm 1, *Merge* is called with q' and r'; this is conditioned on $q' = \delta(q, w_i)$ and $\delta(q, w_j) = r'$. Consequently, if an edge (q, w_j, q') is added by *Merge*, this is conditioned by an already existing edge (q, w_i, q'). □

Corollary 1. *M is acyclic.*

Proof. The algorithm starts from an acyclic automaton (representing the first utterance in the input) in which $q_0 \in Q$ is the initial state and $q_{end} \in Q$ is the final state. By lemma 2, when an edge (q, w_i, r) is added to the automaton either r is a new node with no outgoing edges, or some edge (q, w_j, r) already exists. Hence no edge is added that can close a cycle. □

Another important point is the deterministic character of the model.

Lemma 3. *If algorithm 1 adds an edge (q, w, r) to the automaton, no edge (q, w, r') exists in the automaton such that $r \neq r'$.*

Proof. The algorithm starts with a deterministic automaton and adds a new edge explicitly in steps 17, 20, and implicitly by calling the utility functions *AddNode* and *Merge*. In steps 17, 20 a new edge (q, w_i, r) is added only if $\delta(q, w_i)$ is not defined (this condition is verified in step 8 of the algorithm and the addition of an edge is only in the *else* section), so the lemma holds for these cases.

The utility function *AddNode* (algorithm 2) is called twice from algorithm 1. The first call (step 11) adds the edge (q, w_i, r) and is conditioned by an already existing edge (q, w_i, q_{end}), but these two edges are immediately merged. The second call to *AddNode* (step 19) adds the edge (q, w_i, r) and is performed only if $\delta(q, w_i)$ is not defined (this condition is verified in step 8 of the algorithm and the second call to *AddNode* is only in the *else* section).

When two states $q_i, q_j \in Q$ are found to be compatible and are merged, the resulting automaton could in principle become non-deterministic. For this to happen, there should be two edges (q_i, w, r) and (q_j, w, r') for some $w \in \Sigma$ such that $r \neq r'$. If this were the case, the *Merge* function would continue recursively and merge the states r, r' as a result of step 2–6 in algorithm 3. Note that the merge of q_i, q_j occurs only if r, r' are compatible and merged as well (step 6 in algorithm 3). Hence, if two edges (q_i, w, r) and (q_j, w, r') exist and merge occurs for q_i and q_j, it is assured that at the end of this merge there is only one transition to a next state labeled by $w \in \Sigma$. □

Corollary 2. *M is deterministic.*

Having established that M is a connected, acyclic, deterministic graph, we now prove that it is indeed a DSFA.

Lemma 4. *Let M, q, w be the input and r be the output of the function AddNode. Then, $in(r) = \sum_{(q',w',r)\in M} weight(q', w', r)$.*

Proof. *AddNode* adds the node r to M, with $in(r) = 1$ and $final(r) = 0$. Additionally, it sets q as the only node connected to r with $w(q, w, r) = 1$. Consequently, $\sum_{(q',w',r)\in M} weight(q', w', r) = w(q, w, r) = 1 = in(r)$. □

Lemma 5. *Let $M = \langle Q, \Sigma, P, q_0\rangle, q_1, q_2$ be the input of the function Merge. The following is invariant of the Merge algorithm: For every $q \neq q_0 \in Q$, $in(q) = \sum_{(q',w,q)\in M} weight(q', w, q)$.*

Proof. Observe that the function *Merge* only affects $in(q_1)$ and $in(q_2)$ and does not change weights of any edge in M. Steps 2–6 can only result in the function returning without affecting any node. If conditions 2–6 do not hold, all the incoming edges to q_2 are changed to go to q_1 and all the out edges from q_2 are changed to leave q_1 (steps 7–8). Then (step 9), $in(q_1)$ is changed to be the sum of $in(q_1)$ and $in(q_2)$. If the invariant holds on entering the function, $in(q_1) = \sum_{(q',w,q_1)\in M} weight(q', w, q_1)$ and $in(q_2) = \sum_{(q',w,q_2)\in M} weight(q', w, q_2)$. Consequently, after the algorithm terminates, $in(q_1) = \sum_{(q',w,q)\in M} weight(q', w, q)$, as required, and q_2 is deleted from M (step 9). □

Lemma 6. *Let $M_k = \langle q, \Sigma, P, q_0\rangle$ be the automaton created by algorithm 1 after $k > 0$ input utterances (step 20). For each $q \neq q_0 \in Q$, $in(q)$ is the sum of the weights of all incoming edges to q: $in(q) = \sum_{(q',w,q)\in M_k} weight(q', w, q)$.*

Proof. By induction on k.

Base: For $k = 1$, consider steps 1–2 of the algorithm. At the end of step 2, for every state $q \in Q$, $q \neq q_0$, the in-degree of q is 1, $in(q)$ is 1 and for every edge (q_i, σ, q_j), $weight(q_i, \sigma, q_j)$ is 1; hence the lemma holds for $k = 1$.

Step: Assume that the lemma holds for k. Let $w = w_1...w_n$ be the $k+1$-th input utterance. The following operations can take place (for each i, $1 \leq i \leq n$):

- Steps 8–15: if $i \neq n$ and $q' = q_{end}$, a new node r is added and the lemma holds as a result of lemma 4. Then, (q, w_i, q') is deleted, and in its stead (q, w_i, r) is added. At the same time, $in(r)$ is set to $weight(q, w_i, q')$ and $in(q_{end})$ is subtracted $weight(q, w_i, q')$. By the induction hypothesis, $in(q_{end})$ was the sum of the weights of all incoming edges to q_{end}. Hence, after (q, w_i, q') is deleted and $in(q_{end})$ is subtracted $weight(q, w_i, q')$, the lemma still holds. In addition, r is a new node and by adding (q, w_i, r) and at the same time setting $in(r)$ to $weight(q, w_i, q')$, the lemma holds for r as well. Otherwise, if $i = n$ or $q' \neq q_{end}$, since $\delta(q, w_i)$ is defined and $q' = \delta(q, w_i)$, $weight(q, w_i, q')$ is incremented by 1 and at the same time $in(q')$ is incremented by 1 (step 12). By the induction hypothesis, $in(q')$ was the sum of the weights of all incoming edges to q'. Hence, after $weight(q, w_i, q')$ is incremented by 1 and $in(q')$ is incremented by 1, the lemma still holds. If the condition of step 14 is satisfied, the function *Merge* is called and the lemma still holds as a result of lemma 5.

- Steps 16–20: if the condition of step 16 is satisfied, an edge (q, w_i, r) is added with $weight(q, w_i, r) = 1$ and at the same time $in(r)$ is incremented by 1. By the induction hypothesis, $in(r)$ was the sum of the weights of all incoming edges to r. Hence, after adding new income edge to r with $weight = 1$ and adding 1 to $in(r)$, the lemma still holds. Otherwise, if $i \neq n$, the function *AddNode* is called and the lemma holds as a result of lemma 4. Otherwise, a new incoming edge to q_{end} is added, (q, w_i, q_{end}), with $weight(q, w_i, q_{end}) = 1$ and at the same time $in(q_{end})$ is incremented by 1. By the induction hypothesis, $in(q_{end})$ was the sum of the weights of all incoming edges to q_{end}. Hence, after adding new income edge to q_{end} with $weight = 1$ and adding 1 to $in(q_{end})$, the lemma still holds.

□

Lemma 7. *Let $M_k = \langle q, \Sigma, P, q_0 \rangle$ be the automaton created by algorithm 1 after $k > 0$ input utterances (step 20). For each $q \in Q$, $in(q) = \sum_{(q,w,q') \in M_k} weight(q, w, q') + final(q)$.*

Proof. By induction on k.

Base: For $k = 1$, consider steps 1–2 of the algorithm. At the end of step 2, for every state $q \in Q$, $q \neq q_{end}$, the out-degree of q is 1, $in(q)$ is 1 and $final(q)$ is 0 and for every edge (q_j, σ, q_i), $weight(q_j, \sigma, q_i)$ is 1; hence the lemma holds for every state $q \in Q$, $q \neq q_{end}$. For q_{end}, its out-degree is 0, $in(q_{end})$ is 1 and $final(q_{end})$ is 1; hence the lemma holds for $k = 1$.

Step: Assume that the lemma holds for k. Let $w = w_1 ... w_n$ be the $k + 1$-th input utterance. Consider the loop on i in steps 6–20; we prove that for every $q \in Q$, $in(q)$ is incremented by 1 in the j-th iteration if and only if either $final(q)$ is incremented by 1 in the same iteration or $\sum_{(q,w,q') \in M_{k+1}} weight(q, w, q')$ is incremented by 1 in the $j + 1$-th iteration.

- Steps 8–15: $in(q')$ is incremented by 1 in step 12 and in the same iteration if $i = n$, $final(q')$ is incremented by 1. If $i \neq n$, one of the following can occur in the next iteration:
  - Steps 8–15: if $\delta(q, w_i)$ is defined, $weight(q, w_i, q')$, i.e., $\sum_{(q,w,q') \in M_{k+1}} weight(q, w, q')$ is incremented by 1.
  - Steps 16–20: if the condition of step 16 is satisfied, an edge (q, w_i, r) is added with $weight(q, w_i, r) = 1$, i.e., $\sum_{(q,w,q') \in M_{k+1}} weight(q, w, q')$ is incremented by 1.
- Steps 16–20: $in(r)$ is incremented by 1 in step 17. This is under the condition that $i \neq n$. Then, in the next iteration, one of the following can occur:
  - Steps 8–15: if $\delta(q, w_i)$ is defined, $weight(q, w_i, q')$, i.e. $\sum_{(q,w,q') \in M_{k+1}} weight(q, w, q')$ is incremented by 1.

– Steps 16–20: if the condition of step 16 is satisfied, an edge (q, w_i, r) is added with $weight(q, w_i, r) = 1$, i.e., $\sum_{(q,w,q')\in M_{k+1}} weight(q, w, q')$ is incremented by 1.

Also, in step 20, $in(q_{end})$ is incremented by 1 and in the same step $final(q_{end})$ is incremented by 1.

By the induction hypothesis, the lemma holds for k. As a result of the above, in the $k+1$-th input utterance, for every $q \in Q$, $in(q)$ is incremented by 1 in the j-th iteration if and only if either $final(q)$ is incremented by 1 in the same iteration or $\sum_{(q,w,q')\in M_{k+1}} weight(q, w, q')$ is incremented by 1 in the $j+1$-th iteration. Hence the lemma holds for $k+1$. □

Corollary 3. *M is a DSFA*

Proof. By lemma 7, for every node q_i, $\sum_{j=1}^{out(q_i)} weight(q_i, w_j, q_j) + final(q_i) = in(q_i)$. Hence, for every q_i, $\sum_{j=1}^{out(q_i)}(weight(q_i, w_j, q_j)/in(q_i)) + (final(q_i)/in(q_i)) = 1$ (dividing by $in(q_i)$). According to step 21 of the algorithm, $p_{ij}(w) = (weight(q_i, w, q_j)/in(q_i))$ and $p_i^f = (final(q_i)/in(q_i))$. If the edge (q_i, w, q_j) does not exist, it means that $p_{ij}(w) = 0$ for every $w \in \Sigma$. Therefore, for every node $q_i \in Q$, $\sum_{q_j\in Q}\sum_{w\in\Sigma} p_{ij}(w) + p_i^f = 1$. Consequently, the output of the algorithm is DSFA. □

4.4 Examples

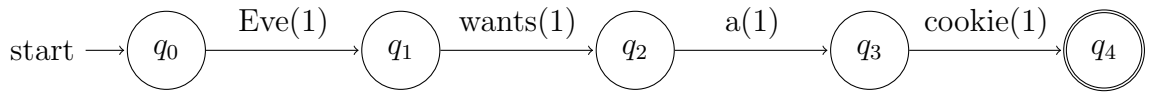
As an example of the operation of the algorithm, consider a corpus consisting of the following utterances:

Eve wants a cookie

Eve wants this milk

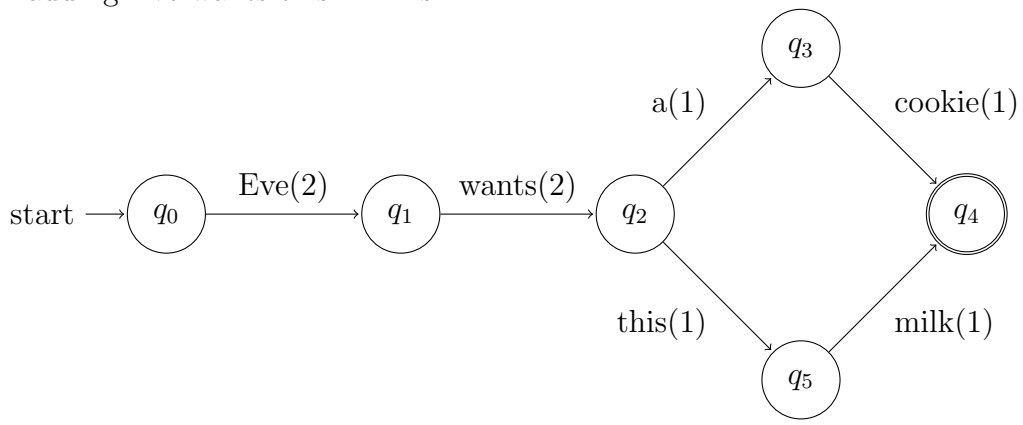
Mommy wants this cookie

First, an automaton is built to represent the first utterance (the numbers on the edges stand for the value of *weight*):



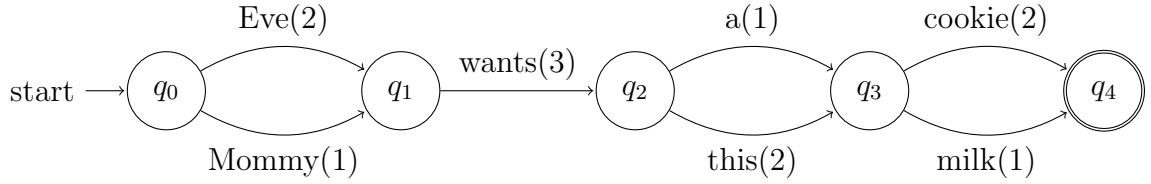
Properties of all $q_i \in Q$: $in(q_i) = 1$ for $0 \leq i \leq 4$, $final(q_i) = 0$ for $0 \leq i \leq 3$ and $final(q_4) = 1$.

Next, the utterance *Eve wants this milk* is added. The first two words; '*Eve wants*' induce only changes in the weights, following step 8. Then, the suffix *this milk* produces new states and a path that ends at $q_{end} = q_4$ as a result of steps 19 and 11; the automaton after adding *Eve wants this milk* is:



Properties of all $q_i \in Q$: $in(q_i) = 2$ for $0 \leq i \leq 2$, $in(q_i) = 1$ for $i = 3, 5$, $final(q_i) = 0$ for $i = 0, 1, 2, 3, 5$, $in(q_4) = 2$, $final(q_4) = 2$.

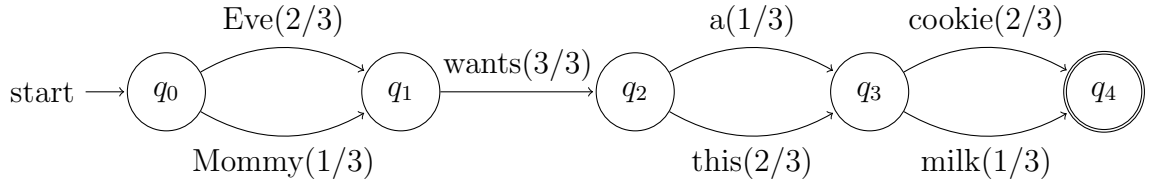
When the string *Mommy wants this cookie* is added to the automaton, *Mommy* is added as an edge between existing states, due to step 16 in the algorithm. Then, the word *wants* only causes weight increment as in step 8. When the word *this* is added, it causes weight increment in addition to merge as a result of condition 14. This is because the algorithm finds that q_3 already has an output edge labeled by the word *cookie* which is also the next word in the current input string, and therefore merges q_3 and q_5. The resulting automaton is:



Properties of all $q_i \in Q$: $in(q_i) = 3$ for $0 \leq i \leq 3$, $in(q_4) = 2$, $final(q_i) = 0$ for

$0 \leq i \leq 3$, $final(q_4) = 2$

Finally, as a result of step 21 of the algorithm, the probabilities are calculated and the resulting automaton is:



Note that the string s = *Mommy wants this milk*, which does not occur in the input, is accepted by the DFSA with $p(s) = p_{0,1}(Mommy) \times p_{1,2}(wants) \times p_{2,3}(this) \times p_{3,4}(milk) \times p_4^f = \frac{1}{3} \times \frac{3}{3} \times \frac{2}{3} \times \frac{1}{3} \times \frac{2}{2} = \frac{2}{27}$

5 Results and Evaluation

We apply our algorithm to several American English corpora available from CHILDES (MacWhinney, 2000), reflecting child–adult interactions involving children from age 1;6 to 3;0. Specifically, we use the Brown (1973) and the Suppes (1974) corpora. Each corpus is an independent experiment. We also use a one-month slice of the (British English) Thomas dense corpus (Lieven et al., 2009). This corpus is divided into two parts: Thomas-A and Thomas-B. Both record the same child, but Thomas-A is denser than Thomas-B. Thomas-A reflects a very intensive period in which Thomas is recorded for one hour, five times a week, every week for the entire period. In Thomas-B, Thomas is recorded for one hour, one week in every month. The use of this corpus will help us test the claim that data density has an effect on the learning outcome.

The files in each corpus are ordered chronologically; the test material in each case consists of the multi-word (length > 2) child utterances in the last file of the corpus; training material consists of all *adult* multi-word utterances in all earlier files, as well as the adult utterances in the last file. This reflects the assumption that children learn mainly from the input directed to them.

We execute Algorithm 1 on each of the training corpora. The result is a DSFA

accepting vastly more utterances than the input ones (since the DSFA is acyclic, this is still a finite number). Details on the corpora and the resulting DSFA are listed in Table 4.

Corpus	Age	Training (CDS only)		Test		$\lvert L(M)\rvert$
		Utterances	Tokens	Utterances	Tokens	Utterances
Eve	1;6—2;3	11,640	54,419	224	875	3e+24
Adam	2;3—3;0	8,606	40,310	792	3,166	2e+23
Sarah	2;3—2;7	4,125	17,182	106	252	5e+16
Nina	1;11—2;5	16,076	85,549	458	1,632	1e+28
Thomas-A	3;1–3;2	19,659	109,537	357	1,269	3e+32
Thomas-B	3;3–3;7	17,581	98,552	436	2,192	4e+32

Table 4: Size of the corpora and the resulting DSFA. Training data: CDS

To evaluate the recall of the model, we count the number of utterances in the test corpus that are accepted by the DSFA produced by the algorithm, and report the ratio of accepted utterances to the total number of test utterances. We also report the average utterance probability (see Definition 3) and its standard deviation (StD); the average probability and StD of only the *accepted* utterances; and the average *per-word* probability and StD of accepted utterances. The per-word probability of an utterance is calculated by summing the probabilities of the edges in the path accepting this utterance in the DSFA and dividing this sum by the length of the utterance. The results, which are summarized in table 5, show that between 59–79% of the child's utterances, depending on the test corpus, can be derived. Given the small size of the training corpora and their sparse nature, these are excellent results, comparable with the best reported results on this task (which, on much denser corpora, amount to 70-84% recall, see section 2.3).

Of course, excellent recall is always possible, at the expense of poor precision. As noted above, evaluating precision is more difficult. In order to assess the over-generation potential of our model, we create a corresponding corpus for each child corpus, containing all the utterances in the test corpus *in reverse word order*.[3] Since English is relatively rigid-order, many of these reversed utterances are clearly ungrammatical. We then test each DSFA on the reversed test-set. The results are shown in Table 5; only approxi-

[3]This is similar to our evaluation of the TBM, see section 3.2.

Corpus	Test	Accepted		Utterance Prob.		Accepted Prob.		Per-word Prob.	
		#	%	Avg.	StD	Avg.	StD	Avg.	StD
Eve	224	176	78.57	25	19	32	22	150,839	46,875
(reversed)		98	43.75	3	2	7	2	71,443	6,014
Adam	792	471	59.47	70	48	117	62	177,296	36,771
(reversed)		278	35.10	7	5	20	8	86,091	7,846
Sarah	106	62	58.49	19	3	33	4	96,398	7,182
(reversed)		31	29.24	8	3	28	5	107,464	9,815
Nina	458	353	77.07	13	7	16	8	106,849	74,816
(reversed)		166	36.24	0.2	0.1	0.5	0.1	63,712	5,402
Thomas-A	357	277	69.75	49	18	71	22	130,749	12,061
(reversed)		171	47.89	9	4	19	5	69,434	6,649
Thomas-B	436	280	64.22	45	19	70	23	133,973	19,172
(reversed)		212	48.63	15	10	31	14	91,113	16,130

Table 5: Evaluation results (training on CDS only). Probabilities and StD must be multiplied by 10^{-7}

mately 29–48% of the reversed utterances are accepted by the DSFA. More significantly, the average probability of the reversed utterances is dramatically lower than the average probability of the actual child utterances in each case: the latter is between 3 and 65 times higher than the former. These differences are retained also when only the accepted utterances are averaged, and when the per-word averages are compared. Note that these results hold for six different corpora. In all corpora the average probability of the reversed utterances is dramatically lower than the average probability of the actual child utterances.

Although it is assumed that children learn mainly from adult input, there is room to also test the model in a scenario where learning is from child *and* adult utterances. In this case, the training sets are larger, but presumably include less accurate data. The size of each corpus (the number of *multi-word* utterances in the corpora) is detailed in table 6 below.

The results, which are summarized in table 5, show that between 68–85% of the child's utterances, depending on the test corpus, can be derived while training data is CS and CDS. The results also show that a larger training set (e.g., the Nina corpus) yields higher recall. More significantly, only approximately 40–60% of the reversed utterances

Corpus	Training		Test		$\lvert L(M)\rvert$
	Utterances	Tokens	Utterances	Tokens	Utterances
Eve	19,536	85,350	224	875	6.8e+26
Adam	20,443	75,213	792	3,166	1.9e+24
Sarah	6,425	23,330	106	252	1.2e+17
Nina	38,736	175,748	458	1,632	8.5e+29
Thomas-A	25,776	132,836	357	1,269	1.2e+33
Thomas-B	25,110	131,652	436	2,192	1.56e+34

Table 6: Size of the corpora and the resulting DSFA. Training data: CDS and CS

are accepted by the DSFA and the average probability of the reversed utterances is dramatically lower than the average probability of the actual child utterances in each case: the latter is between 3 and 30 times (about one order of magnitude) higher than the former. These differences are retained also when only the accepted utterances are averaged, and when the per-word averages are compared. Comparing these results with the previous ones, where training data are only CDS, shows that when the training data is sparser but more accurate (CDS only), recall decreases slightly but over-generation much less so.

Corpus	Test	Accepted		Utterance Prob.		Accepted Prob.		Per-word Prob.	
		#	%	Avg.	StD	Avg.	StD	Avg.	StD
Eve	224	181	80.80	26	17	34	21	139,629	44,722
(reversed)		117	52.23	9	6	16	9	71,608	5,712
Adam	792	555	70.08	255	186	363	221	212,617	57,154
(reversed)		474	59.85	22	16	36	21	88,713	26,088
Sarah	106	72	67.92	83	16	123	18	171,088	110,451
(reversed)		41	38.68	15	5	38	8	81,211	65,940
Nina	458	389	84.93	34	17	40	19	120,337	74,169
(reversed)		276	60.26	1	5	2	6	88,455	7,107
Thomas-A	357	277	77.59	79	26	102	29	135,230	11,773
(reversed)		201	56.30	12	5	22	7	62,656	5,541
Thomas-B	436	326	74.77	66	28	88	32	141,397	13,617
(reversed)		258	59.17	16	9	28	12	80,726	5,721

Table 7: Evaluation results (training on CDS and CS). Probabilities and StD must be multiplied by 10^{-7}

We also examine the number of *novel utterances* in the input, since exact repetitions are trivially accepted by the model and if most utterances of the test corpus are exact

repetitions, excellent recall is trivial. In Table 8 we report the number of exact repetitions (non-novel utterances) in the test corpus. Note that all repeated utterance are non-novel, but in our model, sub-strings of input utterances are not necessarily themselves accepted utterances. The results show that in all corpora novel utterances are more than 75% of the test corpus.

Corpus	Test	Exact repetitions	
		#	%
Eve	224	7	3.12
Adam	792	79	9.97
Sarah	106	27	25.47
Nina	458	39	8.51
Thomas-A	357	51	14.28
Thomas-B	436	54	12.38

Table 8: The amount of exact repetitions in test corpus

5.1 Clustering of Words

A bi-product of learning is clustering of words that occur in similar contexts. In the output DSFA edges between two specified states are, in most cases, labeled by words in the same category. As a particular example, the DSFA constructed from the Eve training corpus includes a path consisting of three edges; the first is labeled by *I* (weight = 1293) and *you* (1171), among others; the second is labeled by several verbs, including *want* (229) and *have* (302). The third edge, which leads to an accepting state, has dozens of labels, including *nut* (12), *flower* (9), *crayons* (8), *plate* (7), *truck* (7), *train* (6), etc. As another example, the edges between two states are labeled by duck(2), dog(1) and squirrel(1). Another example is two states that are connected by edges labeled by verbs (in base form): go(4), see(3), get(3), find(3), play(3), drink(3), want(2), catch(1) and walk(1). One more evidence is some edges that lead from one state to the end state that are labeled by table(3), briefcase(1), spoon(1), wastebasket(1), box(1), napkin(1) and chair(1). Such examples abound, and indicate a certain natural categorization of words according to their contexts in the input.

5.2 Comparison to Other Learning Systems

We compare the evaluation results of our model to the results of two other learning algorithms: ADIOS (Solan et al., 2005) and EMILE (Adriaans, 1992, 2001; Adriaans and Vervoort, 2002) (for more details, see section 2.2).

In order to execute these algorithms we use the ADIOS demo available at `http://adios.tau.ac.il/algorithm.html` and an implementation of EMILE.[4] Since we use demos of the programs, both systems are limited by the amount of training data they can learn from, due to memory restrictions. Hence, the training data for the evaluation of the results include only 730 CDS utterances from the last file of the Eve corpus and the test corpus consists of 224 child utterances in the same file. Table 9 shows the results of the three algorithms.

system	corpus	Accepted		Utterance Prob.	
		#	%	Avg.	StD
EMILE	Test	33	15%		
	Reverse	0	0%		
ADIOS	Test	39	17%		
	Reverse	1	0%		
Our	Test	56	25%	794e-07	323e-07
	Reverse	12	5%	121e-07	8e-07

Table 9: Comparison of The Results

The results show that the recall of our algorithm is about 50% higher than the others (25% compare with 15-17%). In addition, although our algorithm succeeded to generate 12 out of 224 reversed-order utterances, the average probability of the reverse utterances is dramatically lower than the average probability of the test utterances: the latter is approximatly 7 times higher than the former.

[4]We are grateful to Bilal Saleh and Emeel Byadse for their implementation of the EMILE system.

6 Conclusion

We presented a formal, computational model of early syntactic acquisition. Inspired by various psycholinguistic observations, our model implements well-established findings of cognitive psychology as architectural features and constraining biases. At the same time, its mathematical properties are well understood, and it lends itself to rigorous computational evaluation. Indeed, we demonstrate that the language learning algorithm we propose can learn the early language of five different children with high accuracy. This work thus consolidates insights from cognitive science, psycholinguistics and computer science to provide a cognitively plausible computational model of language acquisition, limited to the earliest syntactic structures.

This work can be extended in various directions. First, the importance of morphology in syntactic acquisition is unquestionable. Our model currently ignores morphology altogether and views all words as atomic tokens. In the future, we intend to refine the model in a way that will facilitate the acquisition of morphology and syntax *in tandem*; obviously, finite-state automata are adequate computational devices for expressing the morphological structure of natural languages, so we are confident that only a rather orthodox, natural extension of our current model would be required.

Our algorithm currently creates acyclic automata. While we strongly believe that iteration and recursion are extremely constrained in actual natural language use, a model that allows cycles (corresponding to recurring structures) but constrains the number of times such cycles can be traversed, is clearly a more elegant way of expressing the structure of natural languages. This direction is left for future research.

Finally, we only evaluated our model on English. Several corpora of child–adult interactions are now available in over 25 languages, and we certainly intend to evaluate our algorithm on more languages in the future. We are quite confident that the results will remain as robust.

References

Pieter Adriaans. *Language Learning from a Categorial Perspective*. PhD thesis, Universiteit van Amsterdam, 1992.

Pieter Adriaans. Learning shallow context-free languages under simple distributions. In Ann Copestake and Kees Vermeulen, editors, *Algebras, Diagrams and Decisions in Language, Logic and Computation, CSLI/CUP*. CSLI, Stanford, 2001. URL `http://staff.science.uva.nl/~{}pietera/Publications/shallowlanguages.pdf`.

Pieter Adriaans and Marco Vervoort. The EMILE 4.1 grammar induction toolbox. In Pieter Adriaans, H. Fernau, and Menno van Zaanen, editors, *ICGI 2002*, volume 2481 of *LNAI*, pages 293–295. Springer, Berlin and Heidelberg, 2002. doi: 10.1007/3-540-45790-9_24. URL `http://dx.doi.org/10.1007/3-540-45790-9_24`.

Pieter W. Adriaans and Menno M. van Zaanen. Computational grammatical inference. In Dawn E. Holmes and Lakhmi C. Jain, editors, *Innovations in Machine Learning*, volume 194 of *Studies in Fuzziness and Soft Computing*, chapter 7. Springer-Verlag, Berlin Heidelberg, Germany, 2006. ISBN: 3-540-30609-9.

Afra Alishahi. *Computational modeling of human language acquisition*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool, 2011.

Lalit R. Bahl, Frederick Jelinek, and Robert L. Mercer. A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):179–190, 1983.

Colin Bannard and Elena Lieven. Repetition and reuse in child language learning. In R. Corrigan, E. Moravcsik, H. Ouali, and K. Wheatley, editors, *Formulaic Language*. John Benjamins, Amsterdam, 2009.

Colin Bannard, Elena Lieven, and Michael Tomasello. Early grammatical development is

piecemeal and lexically specific. *Proceedings of the National Academy of Science*, 106 (41):17284–17289, October 2009.

Heike Behrens. The input-output relationship in first language acquisition. *Language and Cognitive Processes*, 21(1-3):2–24, April 2006. ISSN 0169-0965. doi: 10.1080/01690960400001721. URL http://dx.doi.org/10.1080/01690960400001721.

Jonathan Berant, Yaron Gross, Matan Mussel, Ben Sandbank, and Shimon Edelman. Boosting unsupervised grammar induction by splitting complex sentences on function words. In *Proceedings of the 31st Boston University Conference on Language Development*, pages 93–104. Cascadilla Press, 2007.

Robert C. Berwick. Treebank parsing and knowledge of language: a cognitive perspective. In *Proceedings of the EACL 2009 Workshop on Cognitive Aspects of Computational Language Acquisition*, CACLA '09, pages 2:1–2:1, Morristown, NJ, USA, 2009. Association for Computational Linguistics. URL http://portal.acm.org/citation.cfm?id=1572461.1572463.

Rens Bod. Unsupervised parsing with U-DOP. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 85–92, New York City, June 2006. Association for Computational Linguistics. URL http://www.aclweb.org/anthology-new/W06/W06-2912.bib.

Gideon Borensztajn and Willem Zuidema. Bayesian model merging for unsupervised constituent labeling and grammar induction. ILLC Prepublication PP-2007-40, ILLC, University of Amsterdam, 2007.

Gideon Borensztajn, Jelle Zuidema, and Rens Bod. Children's grammars grow more abstract with age — evidence from an automatic procedure for identifying the productive units of language. In *Proceedings of CogSci 2008*, 2008.

Peter Brodsky, Heidi Waterfall, and Shimon Edelman. Characterizing motherese: On the

computational structure of child-directed language. In *Proceedings of the 29th Cognitive Science Society Conference*. Cognitive Science Society, 2007.

Roger Brown. *A first language: the Early stages*. Harvard University Press, Cambridge, Massachusetts, 1973.

Paula J. Buttery. Computational models for first language acquisition. Technical Report UCAM-CL-TR-675, University of Cambridge, Computer Laboratory, November 2006. URL `http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-675.pdf`.

R. Carrasco and J. Oncina. Learning stochastic regular grammars by means of a state merging method. In *Proc. 2nd International Colloquium on Grammatical Inference - ICGI '94*, volume 862, pages 139–150. Springer-Verlag, 1994. URL `citeseer.ist.psu.edu/carrasco94learning.html`.

Alexander Clark and Shalom Lappin. *Linguistic Nativism and the Poverty of the Stimulus*. Wiley Blackwell, Oxford and Malden, MA, 2011.

William Croft. *Radical Construction Grammar*. Oxford, 2001.

Ewa Dąbrowska and Elena Lieven. Towards a lexically specific grammar of children's question constructions. *Cognitive Linguistics*, 16(3):437–474, 2005. URL `http://www.reference-global.com/doi/full/10.1515/cogl.2005.16.3.437`.

Katherine Demuth. Exploiting corpora for language acquisition research. In Heike Behrens, editor, *Corpora in Language Acquisition Research: History, methods, perspectives*, pages 199–205. John Benjamins, Amsterdam, 2008.

Holger Diessel. Frequency effects in language acquisition, language use, and diachronic change. *New Ideas in Psychology*, 25:108–127, 2007.

Holger Diessel and Michael Tomasello. A new look at the acquisition of relative clauses. *Language*, 81:1–25, 2005.

Jeffrey L. Elman. Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99, 1993. doi: 10.1016/0010-0277(93)90058-4. URL http://www.isrl.uiuc.edu/~amag/langev/paper/elman93cognition.html.

Jerome A. Feldman. Computational cognitive linguistics. In *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*, page 1114, Morristown, NJ, USA, 2004. Association for Computational Linguistics.

Daniel Freudenthal, Julian M. Pine, and Fernand Gobet. Modelling the development of children's use of optional infinitives in Dutch and English using MOSAIC. *Cognitive Science*, 30:277–310, 2006.

Daniel Freudenthal, Julian M. Pine, and Fernand Gobet. Understanding the developmental dynamics of subject omission: the role of processing limitations in learning. *Journal of Child Language*, 34(01):83–110, 2007. doi: 10.1017/S0305000906007719. URL http://journals.cambridge.org/action/displayAbstract?fromPage=online&aid=658324&fulltextType=RA&fileId=S0305000906007719.

Daniel Freudenthal, Julian M. Pine, and Fernand Gobet. Simulating the referential properties of Dutch, German, and English root infinitives in MOSAIC. *Language Learning and Development*, 5:1–29, 2009. URL http://www.informaworld.com/10.1080/15475440802502437.

Daniel Freudenthal, Julian Pine, and Fernand Gobet. Explaining quantitative variation in the rate of optional infinitive errors across languages: a comparison of mosaic and the variational learning model. *J Child Lang*, 37(3): 643–69, 2010. ISSN 1469-7602. URL http://www.biomedsearch.com/nih/Explaining-quantitative-variation-in-rate/20334719.html.

E. Mark Gold. Language identification in the limit. *Information and Control*, 10(5): 447–474, May 1967.

Frederick Jelinek, Robert L. Mercer, Lalit R. Bahl, and J. K. Baker. Perplexity—a measure of the difficulty of speech recognition tasks. *Journal of the Acoustical Society of America*, 62:S63, November 1977. Supplement 1.

Lillian Lee. Learning of context-free languages: A survey of the literature. Technical Report TR-12-96, Harvard University, 1996. Available via ftp, ftp://deas-ftp.harvard.edu/techreports/tr-12-96.ps.gz.

Elena Lieven, Heike Behrens, Jennifer Speares, and Michael Tomasello. Early syntactic creativity: a usage-based approach. *Journal of Child Language*, 30(2):333–370, 2003. doi: http://dx.doi.org/10.1017/S0305000903005592. URL `http://dx.doi.org/10.1017/S0305000903005592`.

Elena Lieven, Dorothé Salomo, and Michael Tomasello. Two-year-old children's production of multiword utterances: a usage-based analysis. *Cognitive Linguistics*, 20(3): 481–507, 2009.

Elena V. Lieven, Julian M. Pine, and Gillian Baldwin. Lexically-based learning and early grammatical development. *Journal of Child Language*, 24(1):187–219, 1997. doi: http://dx.doi.org/10.1017/S0305000996002930. URL `http://dx.doi.org/10.1017/S0305000996002930`.

Erkki Luuk and Hendrik Luuk. The redundancy of recursion and infinity for natural language. *Cognitive Processing*, 2011. ISSN 1612-4782. URL `http://dx.doi.org/10.1007/s10339-010-0368-6`. 10.1007/s10339-010-0368-6.

Brian MacWhinney. Rules, rote, and analogy in morphological formations by Hungarian children. *Journal of Child Language*, 2:65–77, 1975.

Brian MacWhinney. The acquisition of morphophonology. *Monographs of the Society for Research in Child Development*, 43:1–123, 1978.

Brian MacWhinney. *The CHILDES Project: Tools for Analyzing Talk*. Lawrence Erlbaum Associates, Mahwah, NJ, third edition, 2000.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19(2): 313–330, June 1993.

Jose Oncina and Pedro García. Identifying regular languages in polynomial. In *Advances in Structural and Syntactic Pattern Recognition*, volume 5 of *Machine Perception and Artificial Intelligence*, pages 99–108. World Scientific, 1992.

Ann M. Peters. *The Units of Language Acquisition*. Monographs in Applied Psycholinguistics. Cambridge University Press, New York, 1983.

W. J. Rapaport. Syntactic semantics: Foundations of computational natural-language understanding. In J. H. Fetzer, editor, *Aspects of Artificial Intelligence*, pages 81–131. Kluwer, Dordrecht, Holland, 1988.

Caroline F. Rowland, Sarah L. Fletcher, and Daniel Freudenthal. Repetition and reuse in child language learning. In Heike Behrens, editor, *Corpora in Language Acquisition Research: History, methods, perspectives*, pages 1–24. John Benjamins, Amsterdam, 2008.

Kenji Sagae, Eric Davis, Alon Lavie, Brian MacWhinney, and Shuly Wintner. Morphosyntactic annotation of CHILDES transcripts. *Journal of Child Language*, 37(3):705–729, 2010. doi: 10.1017/S0305000909990407. URL `http://journals.cambridge.org/action/displayAbstract?fromPage=online&aid=7614676&fulltextType=RA&fileId=S0305000909990407`.

Dan I. Slobin. Cognitive prerequisites for the development of grammar. In D. I. Slobin and C. A. Ferguson, editors, *Studies of child language development*, pages 175–208. Holt, Rinehart and Winston, New York, 1973.

Zach Solan, David Horn, Eytan Ruppin, and Shimon Edelman. Unsupervised learning of natural languages. *Proceedings of the National Academy of Sciences of the United States of America*, 102(33):11629–11634, August 2005. doi: 10.1073/pnas.0409746102. URL `http://www.isrl.uiuc.edu/~amag/langev/paper/solan05languageLearningPNAS.html`.

Andreas Stolcke and Stephen M. Omohundro. Inducing probabilistic grammars by bayesian model merging. In *ICGI '94: Proceedings of the Second International Colloquium on Grammatical Inference and Applications*, pages 106–118, London, UK, 1994. Springer-Verlag. ISBN 3-540-58473-0.

Patrick Suppes. The semantics of children's language. *American Psychologist*, 29:103–114, 1974.

Michael Tomasello. *Constructing a language*. Harvard University Press, Cambridge and London, 2003.

Menno van Zaanen and Jeroen Geertzen. Problems with evaluation of unsupervised empirical grammatical inference systems. In *ICGI '08: Proceedings of the 9th international colloquium on Grammatical Inference*, pages 301–303, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 978-3-540-88008-0. doi: http://dx.doi.org/10.1007/978-3-540-88009-7_29.

Paul Vogt and Elena Lieven. Verifying theories of language acquisition using computer models of language evolution. *Adaptive Behavior Special issue on Language Evolution: Computer models for Empirical Data*, Forthcoming.