# Elsevier LaTeX template[☆]

Elsevier[1]

*Radarweg 29, Amsterdam*

*Elsevier Inc[a,b], Global Customer Service[b,*]*

*[a]1600 John F Kennedy Boulevard, Philadelphia*
*[b]360 Park Avenue South, New York*

**Abstract**

Our goal was to compare among different starting points in the process of language production. We accomplished this by building a model of language production that takes as input a set of words and attempts to construct a grammatical sentence from them. We pre-biased the model's initial utilities to start in the middle, beginning, and end of the sentence in order to add evidence to the debate of how incremental language production is. [Insert findings here] Language production is an interesting process that is difficult to examine with traditional means.

*Keywords:* language production, cognitive modelling
*2010 MSC:* 00-01, 99-00

## 1. Introduction

Studying language production in an effective way thus far has been hard. Psycholinguistics generally relies on carefully controlled experiments; however, studying production in this way is difficult because participants are presented with a large degree of choice. For example, if you ask a subject to describe a picture without controlling what he or she will say, there are a (theoretically)

[☆]Fully documented templates are available in the elsarticle package on CTAN.
[*]Corresponding author
*Email address:* `support@elsevier.com` (Global Customer Service)
*URL:* `www.elsevier.com` (Elsevier Inc)
[1]Since 1880.

infinite number of utterances he or she could produce. Due to the difficulty in collecting large amounts of data in such a setting, comparing a massive number of experimental conditions is not really feasible.

For instance, consider an experimenter who wishes to contrast the choice of a Double Object construction (The boy threw his dog the ball) with a Prepositional Object construction (The boy threw the ball to his dog). If the participant is simply describing a picture, there is no guarantee he or she will even reference the dog! The primary strategy for countering this problem has been to provide the participant with the syntactic structure of the sentence [1] or with the first few words of the utterance [2]. In the latter of these studies, the researchers still had to include an "Other" category for sentences that did not fit in any of the desired conditions.

While these controls have been useful, they could have the side effect of muddling certain parts of the process: such as planning. In order to produce a sentence, it must first be planned. Investigating the planning process is difficult: many paradigms frequently used to investigate sentence comprehension, such as eye tracking or the visual world, seem much more difficult to apply. Despite the difficulty in designing experiments to measure planning, it is clearly of interest to researchers, as can be evidenced by ongoing debates. One such debate is to what degree is the process *incremental*.

For instance, with sentences that contain center-embedded clauses, is the end planned before or after the embedded clause? Consider the sentence "The dog that was chasing the cat that seemed to have rabies fell over." An incremental derivation would rely on *surface order*: the actual order the words appear in the sentence. In other words, the speaker first plans "the dog", then "that was chasing the cat", then "that seemed to have rabies" and lastly "fell over." A less incremental derivation might be to plan "fell over" immediately after "the dog", and then to add the clauses later.

Some researchers seem to take it as a given that language processes are radically incremental (especially with regards to comprehension) [3][4]. However, the evidence (especially for production) is not so clear [1]. Due to the seeming difficulty in solving this debates such as these with more experimentation, we suggest turning to computational cognitive modeling.

## 2. Previous Work

Language processing, in terms of both comprehension and production, have been explored broadly by the cognitive modeling community.

In terms of comprehension, cognitive models have been created both to explain several phenomenon and more generally. **?** ] explained the lexical decision task as a by-product of chunk activation. **?** ] provides evidence that memory retrieval is likewise sufficient to explain whether nouns are treated as anaphoric: whether they refer to an antecedent or are a new reference. Both **?** ] and **?** ] make strides toward more general models of language comprehension.

The models to explain language production are thus far, more narrow. **?** ] created a model that produced references for the iMAP task. **(author?)** [5] was a cognitive model of syntactic priming; it demonstrated that priming can be explained by activation. It made linguistic choices, but only well-defined choices, such as whether to choose double object or prepositional object constructions.

While broader coverage models of language generation do exist [**?** ], they make no claims of cognitive plausibility. Thus, we see this paper as a step towards a broad, realistic model of language production.

## 3. Background

Language comprehension could be briefly stated as the process of transforming words into meaning. Language production is the opposite process. Language production thus consists of several tasks, including transforming some semantic representation into words, and then combining those words into ideas. For now, we primarily focus on the second task, which we accomplish with a grammar formalism, *Combinatory Categorial Grammar* (CCG) [6].

CCG is a grammar in the family of mildly-context sensitive grammars, along with Tree-Adjoining Grammar [**?** ]. Mildly-context sensitive grammars have

$$X/Y > Y = X$$

$$Y < X \setminus Y = X$$

$$X/Y >> Y/Z = X/Z$$

$$Y \setminus Z << X \setminus Y = X \setminus Z$$

Figure 1: Forward Application, Backward Application, Forward Composition, and Backward Composition respectively. Note that types $A, B$ could be any of combinatorially derived types of CCG

been shown to be capable of representing human speech, and computationally plausible in real-time [7]. Also importantly, CCG stores only a single type value at any given point for combined words, as opposed to TAG which stores the entire tree. The types in CCG can be arbitrarily complex, but its rules are quite simple. See Figure 1 for a demonstration of the rules. See Figure 2 for an example of a derivation.

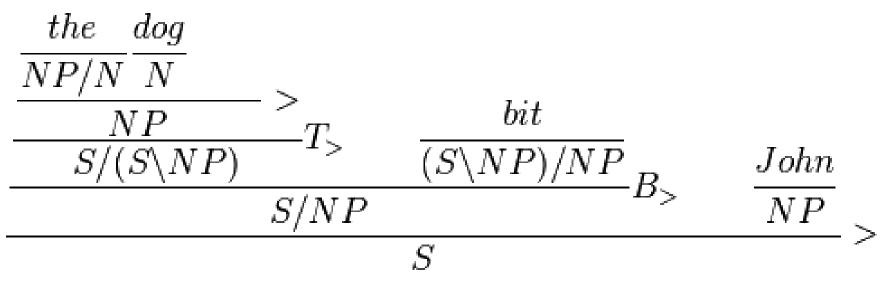

Figure 2: An example of a derivation of CCG. Note that the $T >$ is for type-raising, which is handled my declarative memory rather than production rules in my model. **Redo with my own application combinators**

## 4. Model

The presented model is implemented in jACT-R [**?** ], a full Java implementation of the ACT-R theory [8]. Unlike other potential variants [9], jACT-R has the same full set of constraints as ACT-R.

The presented model's chunks and productions presently largely have a one to one correspondence with CCG. The primary production rules besides memory retrievals are rules directly from CCG; the primary computational units are the types in CCG.

In theory, a model of language production should not make any more assumptions beyond ACT-R, in practice, the model relies on several simplifications. In the future, many of these could become parameters to a model to compare how well the model fits. The model's basic goal is to greedily combine lexical-syntactic chunks together to form a sentence. Right now, the process for that combination consists of the syntactic rules of CCG, and the model has no inherent preference for what to combine with what, besides as described later in experiments.

## *4.1. Rules and Chunks*

The model itself, due to its ability to generalize to any sentence, is programatically generated. The model has *classes* of rules and classes of chunks. A general layout of the production rules can be found in Figure 3. The organizational scheme of the chunks can be found in Figure 4.

### *4.1.1. Production Rules*

There are several classes of production rules. A class of production rules can be thought of as a type of operation, with each specific production rule being that operation on that chunk. For instance, it takes two separate production rules to combine two elements that are at different points in the sentence. Secondly, it takes a different production rule to determine which of the types the Lexsyn should use to combine. For instance, family can be either an adjective or a noun, and it could fill a slot earlier in the sentence ("My family and I went to eat dinner") or later ("I went with my family members to eat dinner"). The main classes of production rules are as follows:

- Move word $A$ into the model: The word moves from DM into the goal buffer. This class contains position information, so there are $N$ of them, where $N$ is the number of words in the sentence.

- Retrieve lexsyn $A_{lex}$ for the word $A$: The type info moves from DM into the goal buffer. This class contains position information, so there are $N$ of them, where $N$ is the number of words in the sentence.

- Apply Syntactic Rule to $A_{lex}$ and $B_{lex}$: CCG defines four basic and a few more complicated syntactic rules, which are described in Section 3. This class contains position and type information both two arguments, so there are $N * K * (N-1) * J$, where $N$ is the number of words in the sentence, $K$ is the number of types $A_{lex}$ has, and $J$ is the number of types $B_{lex}$ has

- Resolve Syntactic Rule on $A_{lex}$: In this step, the new type that results from the previous operation is retrieved and stored in $A_{lex}$, and $B_{lex}$ is removed from the goal buffer.

- Terminate and begin new sentence: If only a single lexsyn remains because all of them have been successfully combined, or no more rules are applicable, then the sentence is finished and the model will start its next goal.

#### *4.1.2. Declarative Memory*

Declarative Memory is composed of a few simple chunk types, described below.

- Word : A word simply has a name, which corresponds to its lexical information (e.g. family).

- Type : A type is an arbitrarily complicated CCG type. The types that exist in DM are the types that are used in the Switchboard CCG derivations.

- Lexsyn : A Lexsyn associates a Word with some number of Types. The chosen types are the types that the word has in the Switchboard corpus, including from type-raising.[2]

- Sentence : A sentence is the work area to create the final utterance. Thus, it starts with words, which eventually become several Lexsyns, which eventually combine into a single Lexsyn, representing the finished sentence.

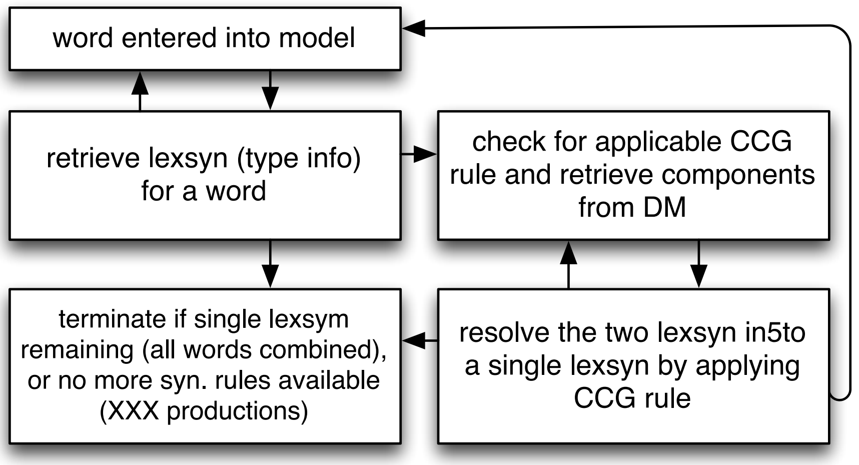
word entered into model
retrieve lexsyn (type info) for a word
check for applicable CCG rule and retrieve components from DM
terminate if single lexsym remaining (all words combined), or no more syn. rules available (XXX productions)
resolve the two lexsyn in5to a single lexsyn by applying CCG rule

Figure 3: General process flow of the production rules of the model. See Section 4.1.1 for a more detailed description.

## *4.2. Input*

Normally, to simulate language production, we would prefer the model start with a semantic representation. Unfortunately, starting with a true semantic representation requires both an evaluation of that representation and an implementation of lexical selection. Thus, we used a simple proxy for the semantic representation as an unordered set of words that could create at least one valid sentence. The words are taken from sentences in the Switchboard corpus.

[2]Type-raising is a CCG operation where a type changes to a symbolically different but effectively equivalent type. This allows derivations in different orders.

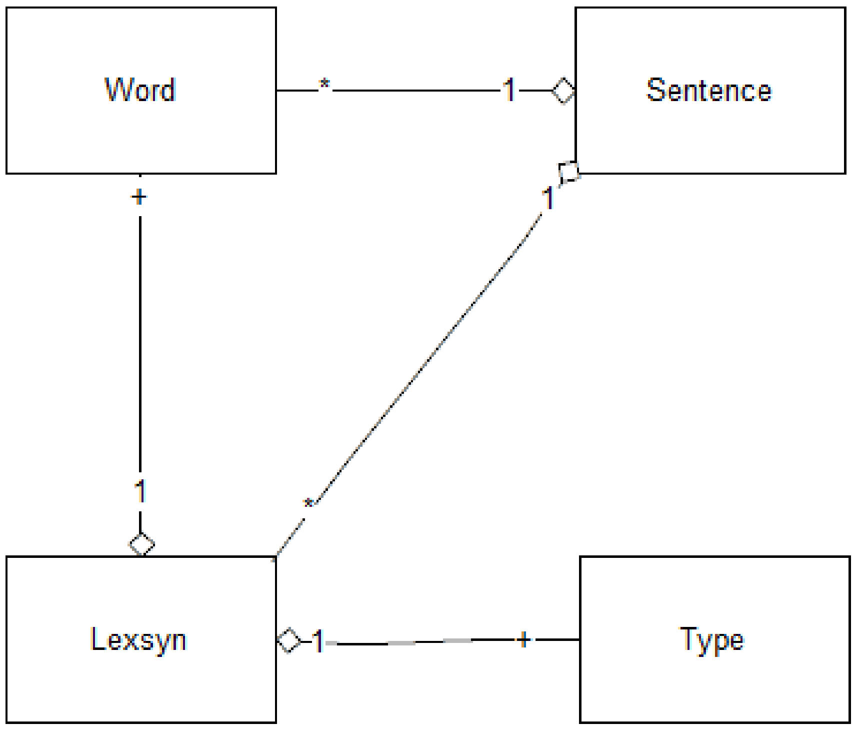

Figure 4: General data model of the declarative memory. See Section 4.1.2 for a more detailed description.

## *4.3. Output*

The model attempts to produce a sentence. However, it does not always successfully finish a sentence, sometimes producing a fragment or several disconnected fragments.

## *4.4. Evaluation*

Naturally, while the model produces utterances, those utterances don't always make sense. Quantifying exactly what it is to make sense is obviously difficult, so we went with a simpler metric: what is probable. We scored each sentence using a weighted bigram average using the SRILM toolkit trained on the spoken BNC corpus [10] inspired by **?** ]. While we could have compared the utterance to the original sentence, since we are mostly bypassing the semantic representation component, we are more concerned with the model producing any good utterances, rather than the ones with the intended meaning.

## 5. Methods

While we use the bigram method previously described to evaluate the utterances described, we also need something to compare to. To do this, we put randomly sampled Switchboard sentences through the same evaluation scheme. We additionally compare variants of the model to this condition, to see which one best fits the data. As described earlier, our primary research question is to determine the degree that language production is incremental. We believe the effectiveness of each model in fitting the human data can provide evidence toward answering that question.

- The *Incremental* model has an initial utility distribution that favors syntactic rules at the beginning of the sentence, decreasing for rules later in the sentence.
- The *Centered* model has an initial distribution favoring rules in the center of the sentence, decreasing as they get closer to the end and the beginning.
- The *Reverse* model has an initial distribution favoring rules at the end of the sentence, decreasing as they get closer to the beginning.
- The *Default* model has no initial utility distribution, with every rule starting at zero utility.

As a reminder, all of these utilities would change as the model progresses and makes sentences. However, the *Default* model has certain disadvantages. As there is an initial sentence the model receives, having any imposed structure would make the model more likely to produce an utterance closer to that original sentence, which is presumably slightly more plausible than any given grammatical utterance. Nonetheless, we hypothesize due to a decreased cognitive load and an increased flexibility in the usage of syntactic rules that the *Incremental* model will perform the best.

Each of the four models and the control condition were fed ten groups of one hundred sentences. Each of them is plotted and compared by their standard deviation, mean, and median.

## 6. Results

## References

[1] F. Ferreira, B. Swets, How Incremental Is Language Production? Evidence from the Production of Utterances Requiring the Computation of Arithmetic Sums, Journal of Memory and Language 46 (1) (2002) 57–84. `doi:10.1006/jmla.2001.2797`.

[2] It is there whether you hear it or not: Syntactic representation of missing arguments 136.

[3] V. Demberg, F. Keller, A psycholinguistically motivated version of TAG, in: Proceedings of the 9th International Workshop on Tree Adjoining Grammars and Related Formalisms. Tbingen, 2008, pp. 25–32.

[4] W. Menzel, Towards radically incremental parsing of natural language, Recent Advances in Natural Language Processing V: Selected Papers from Ranlp 2007 (2009) 41.

[5] D. Reitter, F. Keller, J. D. Moore, A Computational Cognitive Model of Syntactic Priming, Cognitive Science 35 (4) (2011) 587–637.

[6] M. Steedman, J. Baldridge, Combinatory categorial grammar, Non-Transformational Syntax: Formal and Explicit Models of Grammar. Wiley-Blackwell.

[7] A. K. Joshi, K. V. Shanker, D. Weir, The convergence of mildly context-sensitive grammar formalisms.

[8] J. Anderson, A Spreading Activation Theory of Memory, Journal of Verbal Learning and Verbal Behavior.

[9] D. Reitter, C. Lebiere, Accountable modeling in ACT-UP, a scalable, rapid-prototyping ACT-R implementation, in: Proceedings of the 10th international conference on cognitive modeling, Philadelphia, PA, Citeseer, 2010.

[10] A. Stolcke, SRILM-an extensible language modeling toolkit., in: Proceedings of the International Conference on Spoken Language Processing, Vol. 2, Denver, 2002, pp. 901–904.