Distributions on Minimalist Grammar Derivations

Tim Hunter
Department of Linguistics
Cornell University
159 Central Ave., Ithaca, NY, 14853
tim.hunter@cornell.edu

Chris Dyer
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA, 15213
cdyer@cs.cmu.edu

Abstract

We present three ways of inducing probability distributions on derivation trees produced by Minimalist Grammars, and give their maximum likelihood estimators. We argue that a parameterization based on locally normalized log-linear models balances competing requirements for modeling expressiveness and computational tractability.

1 Introduction

Grammars that define not just sets of trees or strings but probability distributions over these objects have many uses both in natural language processing and in psycholinguistic models of such tasks as sentence processing and grammar acquisition. Minimalist Grammars (MGs) (Stabler, 1997) provide a computationally explicit formalism that incorporates the basic elements of one of the most common modern frameworks adopted by theoretical syntacticians, but these grammars have not often been put to use in probabilistic settings. In the few cases where they have (e.g. Hale (2006)), distributions over MG derivations have been over-parametrized in a manner that follows straightforwardly from a conceptualization of the derivation trees as those generated by a particular context-free grammar, but which does not respect the characteristic perspective of the underlying MG derivation. We propose an alternative approach with a smaller number of parameters that are straightforwardly interpretable in terms that relate to the theoretical primitives of the MG formalism. This improved parametrization opens up new possibilities for probabilistically-based empirical evaluation of MGs as a cognitive hypothesis about the discrete primitives of natural language grammars, and for the use of MGs in applied natural language processing.

In Section 2 we present MGs and their equivalence to MCFGs, which provides a context-free characterization of MG derivation trees. We demonstrate the problems with the straightforward method of supplementing a MG with probabilities that this equivalence permits in Section 3, and then introduce our proposed reparametrization that solves these problems in Section 4. Section 5 concludes and outlines some suggestions for future related work.

2 Minimalist Grammars and Multiple Context-Free Grammars

2.1 Minimalist Grammars

A Minimalist Grammar (MG) (Stabler and Keenan, 2003)[1] is a five-tuple $G = \langle \Sigma, Sel, Lic, Lex, \mathsf{c} \rangle$ where:

- Σ is a finite alphabet
- Sel ("selecting types") and Lic ("licensing types") are disjoint finite sets which together determine the set Syn ("syntactic features"), which is the union of the following four sets:

$$selectors = \{=f \mid f \in Sel\}$$
$$selectees = \{\ \ f \mid f \in Sel\}$$
$$licensors = \{+f \mid f \in Lic\}$$
$$licensees = \{-f \mid f \in Lic\}$$

- Lex ("the lexicon") is a finite subset of $\Sigma^* \times (selectors \cup licensors)^* \times selectees \times licensees^*$
- $\mathsf{c} \in Sel$ is a designated type of completed expressions

(A sample lexicon is shown in Fig. 3 below.)

[1] We restrict attention here to MGs without head movement as presented by Stabler and Keenan (2003). Weak generative capacity is unaffected by this choice (Stabler, 2001).

Given an MG G, an **expression** is an ordered binary tree with non-leaf nodes labeled by an element of $\{<, >\}$, and with leaf nodes labeled by an element of $\Sigma^* \times Syn^*$. We take elements of Lex to be one-node trees, hence expressions. We often write elements of $\Sigma^* \times Syn^*$ with the two components separated by a colon (e.g. *arrive* : `+d v`). Each application of one of the derivational operations MERGE and MOVE, defined below, "checks" or deletes syntactic features on the expression(s) to which it applies.

The **head** of a one-node expression is the expression's single node; the head of an expression $[_< e_1\ e_2]$ is the head of e_1; the head of an expression $[_> e_1\ e_2]$ is the head of e_2. An expression is **complete** iff the only syntactic feature on its head is a selectee feature `c` and there are no syntactic features on any of its other nodes. Given an expression e, $yield(e) \in \Sigma^*$ is result of concatenating the leaves of e in order, discarding all syntactic features.

$CL(G)$ is the set of expressions generated by taking the closure of Lex under the functions MERGE and MOVE, defined in Fig. 1; intuitive graphical illustrations are given in Fig 2. The language generated by G is $\{s \mid \exists e \in CL(G)$ such that e is complete and $yield(e) = s\}$.

An example derivation, using the grammar in Fig. 3, is shown in Fig. 4. This shows both the "history" of derivational operations — although operations are not shown explicitly, all binary-branching nodes correspond to applications of MERGE and all unary-branching nodes to MOVE — and the expression that results from each operation. Writing instead only MERGE or MOVE at each internal node would suffice to determine the eventual derived expression, since these operations are functions. A **derivation tree** is a tree that uses this less redundant labeling: more precisely, a derivation tree is either (i) a lexical item, or (ii) a tree $[_{\text{MERGE}}\ \tau_1\ \tau_2]$ such that $\text{MERGE}(eval(\tau_1), eval(\tau_2))$ is defined, or (iii) a tree $[_{\text{MOVE}}\ \tau]$ such that $\text{MOVE}(eval(\tau))$ is defined; where *eval* is the "interpretation" function that maps a derivation tree to an expression in the obvious way. We define $\Omega(G)$ to be the set of all derivation trees using the MG G.

An important property of the definition of MOVE is that it is only defined on $\tau[+f\alpha]$ if there is a *unique* subtree of this tree whose (head's) first feature is $-f$. From this it follows that in any

pierre : `d`	*who* : `d -wh`
marie : `d`	*will* : `=v =d t`
praise : `=d v`	ϵ : `=t c`
often : `=v v`	ϵ : `=t +wh c`

Figure 3: A Minimalist Grammar lexicon. The type of completed expressions is `c`.

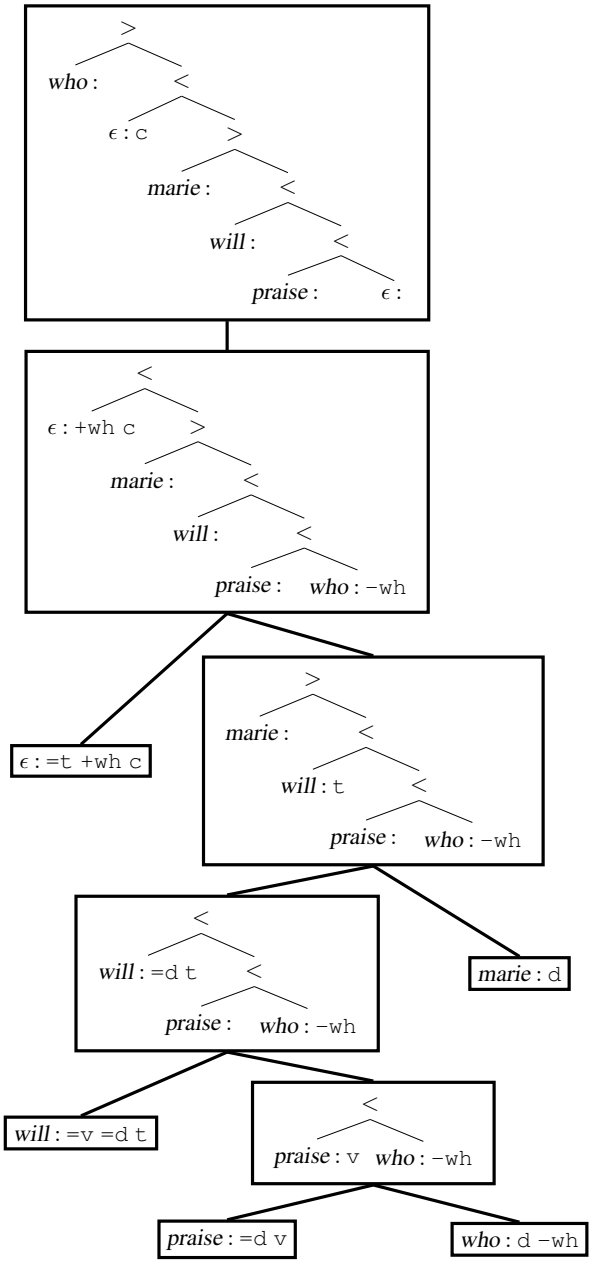


Figure 4: An MG derivation of an embedded question

$$\textsc{merge}\big(e_1[{=}f\ \alpha], e_2[f\ \beta]\big) = \begin{cases} [_< e_1[\alpha]\ e_2[\beta]] & \text{if } e_1[{=}f\ \alpha] \in \mathit{Lex} \\ [_> e_2[\beta]\ e_1[\alpha]] & \text{otherwise} \end{cases}$$

$$\textsc{move}\big(e_1[{+}f\ \alpha]\big) = [_> e_2[\beta]\ e_1'[\alpha]]$$

where $e_2[{-}f\ \beta]$ is a unique subtree of $e_1[{+}f\ \alpha]$

and e_1' is like e_1 but with $e_2[{-}f\ \beta]$ replaced by an empty leaf node $\epsilon : \epsilon$

Figure 1: Definitions of MG operations MERGE and MOVE. The first case of MERGE creates complements, the second specifiers. f ranges over $\mathit{Sel} \cup \mathit{Lic}$; α and β range over Syn^*; and $e[\alpha]$ is an MG expression whose head bears the feature-sequence α.

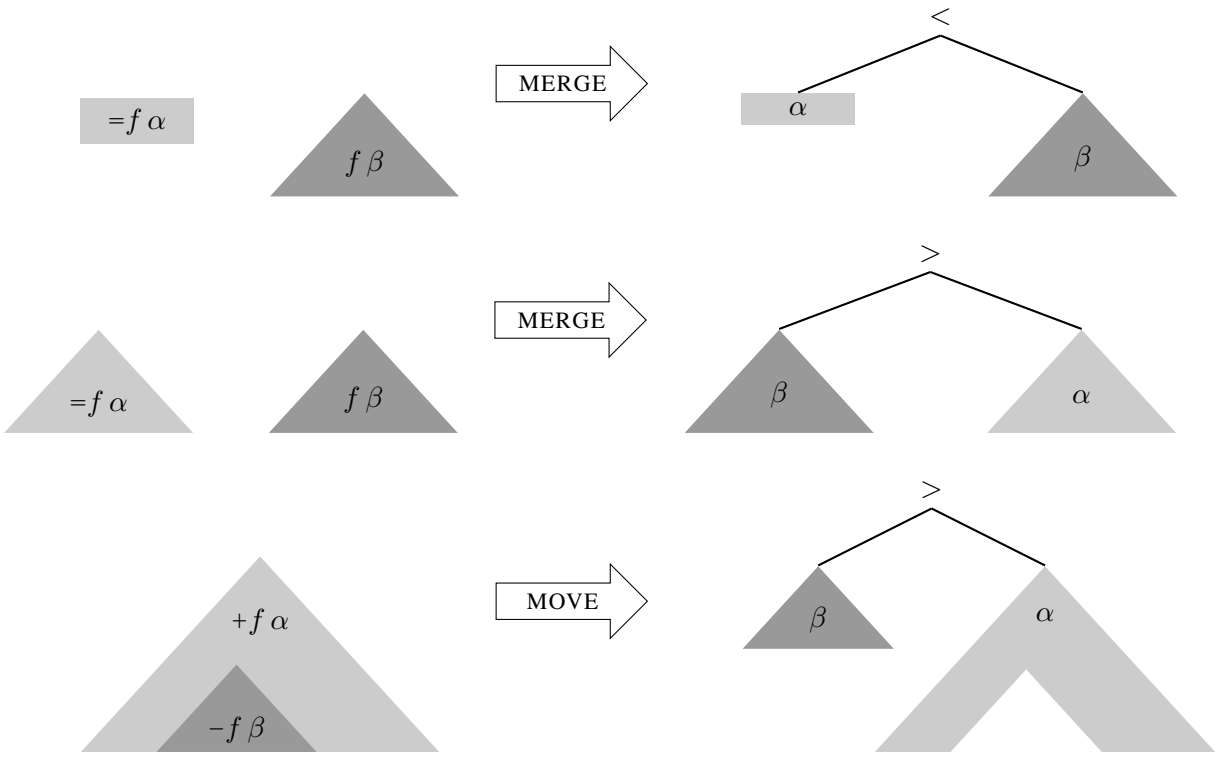


Figure 2: Graphical illustrations of definitions of MERGE and MOVE. Rectangles represent single-node trees. Triangles represent either single-node trees or complex trees, but the second case of MERGE applies only when the first case does not (i.e. when the $=f\ \alpha$ tree is complex).

derivation of a complete expression, every intermediate derived expression will have at most $|Lic|$ subtrees whose (head's) first feature is of the form $-g$ for any $g \in Lic$.

2.2 Multiple Context-Free Grammars

Multiple Context-Free Grammars (MCFGs) (Seki et al., 1991; Kallmeyer, 2010) are a mildly context-sensitive grammar formalism in the sense of Joshi (1985).[2] They bring additional expressive capacity over context-free grammars (CFGs) by generalizing to allow nonterminals to categorize not just single strings, but tuples of strings. For example, while a CFG might categorize *eats cake* as a VP and *the boy* as an NP, an MCFG could categorize the tuple $\langle$*says is tall*, *which girl*$\rangle$ as a VPWH (intuitively, a VP containing a WH which will move out of it). Correspondingly, MCFG production rules (construed as recipes for building expressions bottom-up) can specify not only, for example, how to combine a *string* which is an NP and a *string* which is a VP, but also how to combine a *string* which is an NP with a *tuple of strings* which is a VPWH. The CFG rule which would usually be written 'S $\rightarrow$ NP VP' is shown in (1) in a format that makes explicit the string-concatenation operation; (2) uses this notation to express an MCFG rule that combines an NP with a VPWH to form a string of category Q, an embedded question. (We often omit angle brackets around one-tuples.) An example application of this rule is shown in (3).

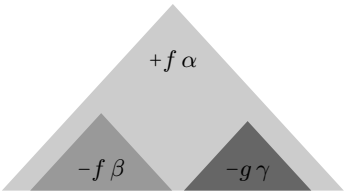
$$st :: \text{S} \quad \Rightarrow \quad s :: \text{NP} \qquad t :: \text{VP} \tag{1}$$

$$t_2 s t_1 :: \text{Q} \quad \Rightarrow \quad s :: \text{NP} \quad \langle t_1, t_2 \rangle :: \text{VPWH} \tag{2}$$

$$\textit{which girl the boy says is tall} :: \text{Q} \quad \Rightarrow$$
$$\textit{the boy} :: \text{NP} \quad \langle \textit{says is tall}, \textit{which girl} \rangle :: \text{VPWH} \tag{3}$$

Every nonterminal in an MCFG derives (only) n-tuples of strings, for some n known as the nonterminal's **rank**. In the examples above NP, VP, S and Q are of rank 1, and VPWH is of rank 2. A CFG is an MCFG where every nonterminal has rank 1.

Michaelis (2001) showed that it is possible to reformulate MGs in a way that uses categorized string-tuples, of the sort that MCFGs manipulate, as derived structures (or expressions) instead of trees. The "purpose" of the internal tree structure that we assign to derived objects is, in effect, to allow a future application of MOVE to break them apart and rearrange their pieces, as illustrated in Fig. 2. But since the placement of the syntactic features on a tree determines the parts that will be rearranged by a future application of MOVE (in any derivation of a complete expression), we lose no relevant information by splitting up a tree's yield into the components that will be rearranged and then ignoring all other internal structure. Thus the following tree:



(4)

becomes a tuple of categorized strings (we will explain the 0 subscript shortly):

$$\langle \quad s : +f\,\alpha \quad , \quad t : -f\,\beta \quad , \quad u : -g\,\gamma \quad \rangle_0$$

or, equivalently, a tuple of strings, categorized by a tuple-of-categories:

$$\langle s, t, u \rangle :: \langle +f\,\alpha, -f\,\beta, -g\,\gamma \rangle_0 \tag{5}$$

The order of the components is irrelevant *except for* the first component, which contains the entire structure's head node; intuitively, this is the component out of which the others move.

Based on this idea, Michaelis (2001) shows how to construct, for any MG, a corresponding MCFG whose nonterminals are tuples like $\langle +f\,\alpha, -f\,\beta, -g\,\gamma \rangle_0$ from above. The uniqueness requirement in the definition of MOVE ensures that we need only a finite number of such nonterminals. The feature sequences that comprise the MCFG nonterminals, in combination with the MG operations, determine the MCFG production rules in which each MCFG nonterminal appears. For example, the arrangement of features on the tree in (4) dictates that MOVE is the only MG operation that can apply to it; thus the internals of the complex category in (5) correspondingly dictate that the only MCFG production that takes (5) as "input" (again, thinking right-to-left or bottom-up as in (1) and (2)) is one that transforms it in accord with the effects of MOVE. If $\beta = \epsilon$, then this

[2] MCFGs are almost identical to Linear Context-Free Rewrite Systems (Vijay-Shanker et al., 1987). Seki et al. (1991) show that the two formalisms are weakly equivalent.

effect will be to transform the three-tuple into a two-tuple as shown in (6), since the t-component now has no remaining features and has therefore reached its final position:

$$\langle ts, u\rangle :: \langle \alpha, -g\,\gamma\rangle_0 \quad \Rightarrow \quad \langle s, t, u\rangle :: \langle +f\,\alpha, -f, -g\,\gamma\rangle_0 \quad (6)$$

This is analogous — modulo the presence of the additional $u : -g\,\gamma$ component — to the rule that is used in the final step of the derivation in Fig. 5, which is the MCFG equivalent of Fig. 4.

If, on the other hand, $\beta \neq \epsilon$, then the t-component will need to move again later in the derivation, and so we keep it as a separated component:

$$\langle s, t, u\rangle :: \langle \alpha, \beta, -g\,\gamma\rangle_0 \quad \Rightarrow \quad \langle s, t, u\rangle :: \langle +f\,\alpha, -f\,\beta, -g\,\gamma\rangle_0 \quad (7)$$

The subscript 0 on the tuples above indicates that the corresponding expressions are non-lexical; for lexical expressions, the subscript is 1. This information is not relevant to MOVE operations, but is crucial for distinguishing between the complement and specifier cases of MERGE. For example, in the simplest cases where no to-be-moved subconstituents are present, the constructed MCFG must contain two rules corresponding to MERGE as follows. (n matches either 1 or 0.)

$$st :: \langle \alpha\rangle_0 \quad \Rightarrow \quad s :: \langle =f\,\alpha\rangle_1 \quad t :: \langle f\rangle_n \quad (8)$$

$$ts :: \langle \alpha\rangle_0 \quad \Rightarrow \quad s :: \langle =f\,\alpha\rangle_0 \quad t :: \langle f\rangle_n \quad (9)$$

By similar logic, it is possible to construct all the necessary MCFG rules corresponding to MERGE and MOVE; see, for example, Stabler and Keenan (2003, p.347) for (a presentation of the MG operations that can also be straightforwardly be read as) the general schemas that generate these rules. One straightforward lexical/preterminal rule is added for each lexical item in the MG, and the MCFG's start symbol is $\langle \texttt{c}\rangle_0$.[3] The resulting MCFG is weakly equivalent to the original MG, and strongly equivalent in the sense that one can straightforwardly convert back and forth between the two grammars' derivation trees. The MCFG equivalent of the MG in Fig. 3 is shown in Fig. 6 (ignoring the weights for now, which we come to below).[4]

[3]We exclude $\langle \texttt{c}\rangle_1$ on the simplifying assumption that the MG has no lexical item whose only feature is the selectee c.

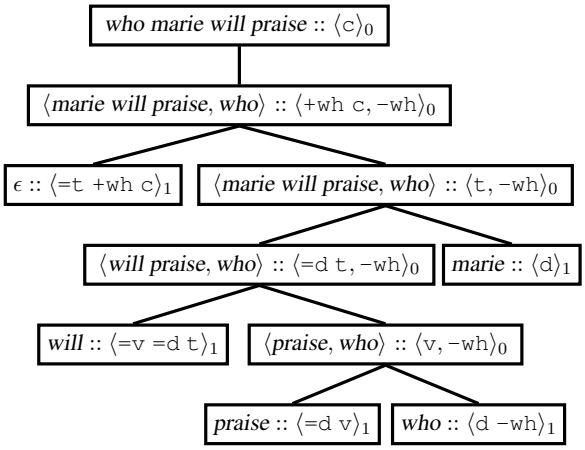


Figure 5: The MG derivation from Fig. 4 illustrated with tuples of strings instead of trees as the derived structures.

Notation. We define the above conversion process to be an (invertible) function π from MGs to MCFGs. That is, for an valid MG, G it holds that $\pi(G)$ is an equivalent MCFG and $\pi^{-1}(\pi(G)) = G$. By abuse of notation, we will use π as the function for converting from MG derivation trees to equivalent MCFG derivation trees. By an MCFG derivation tree we mean a tree like Fig. 5 but with non-leaf nodes labelled only by nonterminals (not tuples of strings). The derivation tree language of an MCFG is thus a local tree language, just as for a CFG; that of an MG is non-local but regular (Kobele et al., 2007).

3 Distributions on Derivations

Assume a Minimalist Grammar, G. In this section and the next, we will consider various ways of defining probability distributions on the derivation trees in $\Omega(G)$.[5] The first approach, introduced in Section 3.2, is conceptually straightforward but is problematic in certain respects that we discuss in Section 3.3. We present a different approach that resolves these problems in Section 4.

We also consider the problem of estimating the parameters of these distributions from a *finite sample* of training data, specified by a function $\tilde{f} : \Omega(G) \to \mathbb{N}$, where $\tilde{f}(\tau)$ is the number of times derivation τ occurs in the sample. To this end, it

[4]This MCFG includes only the rules that are "reachable" from the lexical items. For example, we leave aside rules involving the nonterminal $\langle =\texttt{c}\ \texttt{v}\ -\texttt{wh}\rangle_0$, even though the schemas in Stabler and Keenan (2003) generate them.

[5]We use the terms *derivation tree* and *derivation* interchangeably.

θ^{ERF}	
2/2	$\epsilon :: \langle \texttt{=t +wh c} \rangle_1$
95/95	$\epsilon :: \langle \texttt{=t c} \rangle_1$
97/97	*will* $:: \langle \texttt{=v =d t} \rangle_1$
6/6	*often* $:: \langle \texttt{=v v} \rangle_1$
97/97	*praise* $:: \langle \texttt{=d v} \rangle_1$
95/192	*marie* $:: \langle \texttt{d} \rangle_1$
97/192	*pierre* $:: \langle \texttt{d} \rangle_1$
2/2	*who* $:: \langle \texttt{d -wh} \rangle_1$

θ^{ERF}			
2/2	$\langle st, u \rangle :: \langle \texttt{+wh c, -wh} \rangle_0$	$\Rightarrow$	$s :: \langle \texttt{=t +wh c} \rangle_1 \quad \langle t, u \rangle :: \langle \texttt{t, -wh} \rangle_0$
95/95	$st :: \langle \texttt{=d t} \rangle_0$	$\Rightarrow$	$s :: \langle \texttt{=v =d t} \rangle_1 \quad t :: \langle \texttt{v} \rangle_0$
2/2	$\langle st, u \rangle :: \langle \texttt{=d t, -wh} \rangle_0$	$\Rightarrow$	$s :: \langle \texttt{=v =d t} \rangle_1 \quad \langle t, u \rangle :: \langle \texttt{v, -wh} \rangle_0$
2/97	$ts :: \langle \texttt{c} \rangle_0$	$\Rightarrow$	$\langle s, t \rangle :: \langle \texttt{+wh c, -wh} \rangle_0$
95/97	$st :: \langle \texttt{c} \rangle_0$	$\Rightarrow$	$s :: \langle \texttt{=t c} \rangle_1 \quad t :: \langle \texttt{t} \rangle_0$
95/95	$ts :: \langle \texttt{t} \rangle_0$	$\Rightarrow$	$s :: \langle \texttt{=d t} \rangle_0 \quad t :: \langle \texttt{d} \rangle_1$
2/2	$\langle ts, u \rangle :: \langle \texttt{t, -wh} \rangle_0$	$\Rightarrow$	$\langle s, u \rangle :: \langle \texttt{=d t, -wh} \rangle_0 \quad t :: \langle \texttt{d} \rangle_1$
95/100	$st :: \langle \texttt{v} \rangle_0$	$\Rightarrow$	$s :: \langle \texttt{=d v} \rangle_1 \quad t :: \langle \texttt{d} \rangle_1$
5/100	$st :: \langle \texttt{v} \rangle_0$	$\Rightarrow$	$s :: \langle \texttt{=v v} \rangle_1 \quad t :: \langle \texttt{v} \rangle_0$
2/3	$\langle s, t \rangle :: \langle \texttt{v, -wh} \rangle_0$	$\Rightarrow$	$s :: \langle \texttt{=d v} \rangle_1 \quad t :: \langle \texttt{d -wh} \rangle_1$
1/3	$\langle st, u \rangle :: \langle \texttt{v, -wh} \rangle_0$	$\Rightarrow$	$s :: \langle \texttt{=v v} \rangle_1 \quad \langle t, u \rangle :: \langle \texttt{v, -wh} \rangle_0$

Figure 6: The MCFG produced from the MG in Fig. 3, as described in Section 2.2; with weights computed by relative frequency estimation based on the naive parametrization, as described in Section 3.

will be useful to define the **empirical distribution** on derivations to be $\tilde{p}(\tau) = \tilde{f}(\tau)/\sum_{\tau'} \tilde{f}(\tau')$.

3.1 Stochastic MCFGs

As with CFGs, it is straightforward to imbue an MCFG, H, with production probabilities and thereby create a **stochastic MCFG**.[6] In stochastic MCFGs (as in CFGs) the probability of a non-terminal rewrite in a derivation is conditionally independent of all other rewrite decisions, given the non-terminal type. This formulation defines a distribution over MCFG derivations in terms of a random branching process that begins with probability 1 at the start symbol and recursively expands frontier nodes N, drawing branching decisions from the the conditional distribution $p(\cdot \mid N)$; the process terminates when lexical items have been produced on all frontiers.

If $p(\delta \mid N)$ is the probability that N rewrites as δ and $f_\tau(N \Rightarrow \delta)$ is the number of times $N \Rightarrow \delta$ occurs in derivation tree τ, then

$$p(\tau) = \prod_{(N \Rightarrow \delta) \in H} p(\delta \mid N)^{f_\tau(N \Rightarrow \delta)}. \qquad (10)$$

With mild assumptions to ensure consistency (Chi, 1999), the $p(\tau)$'s form a proper probability distribution over all derivations in H.[7]

Because the derivation trees of the MG G stand in a bijection with the derivation trees of the MCFG $\pi(G)$, stochastic MCFGs can be used to define a distribution on MG derivations.

[6] Although MCFGs have a greater generative capacity than CFGs, the statistical properties do not change at all, unless otherwise noted.

[7] The estimators that are based on empirical frequencies in a derivation bank which we use in this paper will always yield consistent estimates. Refer Chi (1999) for more detail.

3.2 The naive parametrization

The most straightforward way to parameterize a stochastic MCFG uses individual parameters $\theta_{\delta|N}$ to represent each production probability, i.e., $p(\delta \mid N) \doteq \theta_{\delta|N}$. When applied to an MCFG that is derived from an MG, we will refer to this as the **naive parametrization**.

This is the parametrization used by Hale (2006) to define a probability distribution over the derivations of MGs in order to explore the predictions of an information-theoretic hypothesis concerning sentence comprehension difficulty.

MLE. The arguably most standard technique for setting the parameters of a probability distribution is so that they maximize the likelihood of a sample of training data. In the naive parameterization, the maximum likelihood estimate (MLE) for each parameter $\hat{\theta}^{ERF}_{\delta|N}$ is the *empirical relative frequency* of the rewrite $N \Rightarrow \delta$ in the training data (Abney, 1997):

$$\hat{\theta}^{ERF}_{\delta|N} = \frac{\sum_\tau \tilde{f}(\tau) f_{\pi(\tau)}(N \Rightarrow \delta)}{\sum_\tau \tilde{f}(\tau) \sum_{(N \Rightarrow \delta') \in \pi(G)} f_{\pi(\tau)}(N \Rightarrow \delta')}.$$

3.3 Unfaithfulness to MGs

While the naive parameterization with MLE estimation is simple, it is arguably a poor choice for parameterizing distributions on MGs. The problem is that, relative to the independence assumptions encoded in the MG formalism, each step of the MCFG derivation both conditions on and predicts "too much" structure. As a result, commonalities across different applications of the same MG operation are modeled independently and do not share statistical strength. This arises because

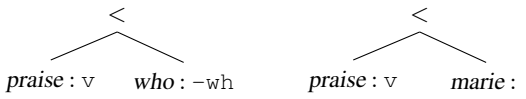
90	pierre will praise marie
5	pierre will often praise marie
1	who pierre will praise
1	who pierre will often praise

Figure 7: An artificial corpus of sentences derivable from the grammars in Figures 3 and 6.

of the way the MCFG's nonterminals multiply out all relevant arrangements of features.[8] We illustrate the problem with an example.

Consider the corpus in Fig. 7, where each sentence is preceded by its frequency. Since each sentence is assigned a unique derivation by our example MG, this is equivalent to a treebank.

One reasonable statistical interpretation of the first two lines is that a verb phrase comprises a verb and an object 95% of the time, and comprises the adverb *often* and another verb phrase 5% of the time (since *pierre will often praise marie* has two nested verb phrase constituents). The last two lines provide an analogous pair of sentences involving wh-movement of the object. A priori, one would expect that the 95:5 relative frequency that describes the presence of the adverb also applies here; however, the ERF estimator will use 2:1 instead. Why is this? The VP category in the MCFG is "split" into two to indicate whether it has a wh-feature inside it, and each has its own parameters. We criticize this on the grounds that it is not in line with our main goal of defining a distribution over the derivations *of the MG*: from the perspective of the MG, there is a sense in which it is "the same instance" of MERGE that combines *often* with a verb phrase, whether or not the verb phrase's object bears a $-\texttt{wh}$ feature. In other words, the differences between the following two trees seem unrelated to the way in which they are both candidates to be merged with *often* : $=\texttt{v}\ \texttt{v}$.



From the perspective of the MCFG, however, the introduction of the adverb is mediated by expansions of the nonterminal $\langle \texttt{v} \rangle_0$ in cases without object wh-movement, but by expansions of the distinct nonterminal $\langle \texttt{v}, -\texttt{wh} \rangle_0$ in cases with it. Therefore the information about adverb inclusion that is conveyed by the movement-free entries in the corpus is interpreted as only relevant to similarly movement-free derivations. This can be seen in the weights of the last four rules in Fig. 6, which were computed by relative frequency estimation on the basis of the corpus.

Relative to the underlying MG, the naive parametrization has too many degrees of freedom: the model is *overparameterized* and is capable of capturing statistical distinctions that we have theoretical reasons to dislike. Of course, it is possible that VPs have meaningfully different distributions depending on whether or not they contain a wh-feature; however, we would like a parameterization that provides the flexibility to treat these two different contexts as identical, as different, or to share statistical strength between them in some other way. In the next section we propose two alternative parametrizations that provide this control.

4 Log-linear MCFGs

4.1 Globally normalized log-linear models

An alternative mechanism for inducing a distribution on $\Omega(G)$ that provides more control over independence assumptions is the **globally normalized log-linear model** (also called a Markov random field, undirected model, or Gibbs distribution). Unlike the model in the previous section, log-linear models are *not* stochastic in nature—they assign probabilities to structured objects, but they do not rely on a random branching process to do so. Rather, they use a d-dimensional vector of feature functions $\boldsymbol{\Phi} = \langle \Phi_1, \Phi_2, \ldots, \Phi_d \rangle$, where $\Phi_i : \Omega(G) \to \mathbb{R}$, to extract *features* of the derivation, and a real-valued weight vector $\boldsymbol{\lambda} \in \mathbb{R}^d$.[9] Together, $\boldsymbol{\Phi}$ and $\boldsymbol{\lambda}$ define the *score* of a derivation τ as a monotonic function of the weighted sum of the feature values $\Phi_1(\tau), \ldots, \Phi_d(\tau)$:

$$s_{\boldsymbol{\lambda}}(\tau) = \exp(\boldsymbol{\lambda} \cdot \boldsymbol{\Phi}(\tau)).$$

Using this function, a **Gibbs distribution** on the derivations in $\Omega(G)$ is

$$p_{\boldsymbol{\lambda}}(\tau) = \frac{s_{\boldsymbol{\lambda}}(\tau)}{\sum_{\tau' \in \Omega(G)} s_{\boldsymbol{\lambda}}(\tau')}, \qquad (11)$$

[8] Stabler (forthcoming) also discusses the sense in which MCFG rules "miss generalizations" found in MGs.

[9] The term **feature** here refers to functions of a derivation; it should *not* be confused with the **syntactic features** discussed immediately above. However, in as much as syntactic features characterize the steps in a derivation, it is natural that they would play a central role in defining distributions over derivations, and indeed, our proposed feature functions examine syntactic features almost exclusively.

provided that the sum in the denominator is finite.[10]

Notice that (11) is similar to the formula for a relative frequency, the difference being that we use a derivation's score $s_{\boldsymbol{\lambda}}(\tau)$ rather than its empirical count. This use of scores provides a way to express the kind of "missed similarities" we discussed in Section 3.3 via the choice of feature functions. Returning to the example from above, in order to express the similarity between the two adverb-introducing rules — one involving the nonterminal $\langle \mathtt{v} \rangle_0$, the other involving $\langle \mathtt{v}, \mathtt{-wh} \rangle_0$ — we could define a particular feature function Φ_i that maps a derivation to 1 if it contains either one of these rules and 0 otherwise. Then, all else being equal, setting the corresponding parameter λ_i to a higher value will increase the score $s_{\boldsymbol{\lambda}}(\tau)$, and hence the probability $p_{\boldsymbol{\lambda}}(\tau)$, of *any* derivation τ that introduces an adverb, with or without wh-movement of the object.

MLE. As with the naive parameterization, the the parameters $\boldsymbol{\lambda}$ may be set to maximize the (log) likelihood of the training data, i.e.,

$$\begin{aligned}\hat{\boldsymbol{\lambda}} &= \arg\max_{\boldsymbol{\lambda}} \prod_{i=1}^{n} p_{\boldsymbol{\lambda}}(\tau_i)^{\tilde{f}(\tau_i)} \\ &= \arg\max_{\boldsymbol{\lambda}} \underbrace{\sum_{i=1}^{n} \tilde{f}(\tau_i) \log p_{\boldsymbol{\lambda}}(\tau_i)}_{=\mathcal{L}\text{ [log likelihood]}}. \end{aligned} \tag{12}$$

We remark that maximizing the log likelihood of data in this parameterization is equivalent to finding the distribution $p_{\boldsymbol{\lambda}}(\tau)$ in which the expected value of $\boldsymbol{\Phi}(\tau)$ is equal to the expected value of the same under the empirical distribution (i.e., under $\tilde{p}(\tau)$) and whose *entropy is maximized* (Della Pietra et al., 1997). This equivalence is particularly clear when the gradient of $\mathcal{L}$ (see (12)) with respect to $\boldsymbol{\lambda}$ is examined:

$$\nabla_{\boldsymbol{\lambda}} \mathcal{L} = \mathbb{E}_{\tilde{p}(\tau)}[\boldsymbol{\Phi}(\tau)] - \mathbb{E}_{p_{\boldsymbol{\lambda}}(\tau)}[\boldsymbol{\Phi}(\tau)]. \tag{13}$$

This form makes clear that $\mathcal{L}$ achieves an optimum when the expectations of $\boldsymbol{\Phi}$ match under the two distributions.[11]

[10] There are several conditions under which this is true. It is trivially true if $|\Omega(G)| < \infty$. When Ω is infinite, the denominator may still be finite if features functions grow (super) linearly with the derivation size in the limiting case as the size tends to infinity. Then, if feature weights are negative, the denominator will either be equal to or bounded from above by an infinite geometric series with a finite sum. Refer to Goodman (1999) and references therein.

[11] While the maximizing point cannot generally be solved for analytically, gradient based optimization techniques may be effectively used to find it (and it is both guaranteed to exist and guaranteed to be unique).

4.2 Feature locality

Notice that the approach just outlined is extremely general: the feature functions $\boldsymbol{\Phi}$ can examine the derivation trees as a whole. It is possible to define features that pay attention to arbitrary or global properties of a derivation. While such features might in fact generalize well to new data — for example, one could mimic a bigram language model by including features counting bigrams in the string that is generated by the derivation — these are intuitively "bad" since they ignore the derivation's structure. Furthermore, there is a substantial practical downside to allowing unrestricted feature definitions: features that do not "agree" with the derivation structure make inference computationally intractable. Specifically, finding the best most probable derivation of a sentence with "global" features is NP-hard (Koller and Friedman, 2009).

For these reasons, it is advantageous to require that $\boldsymbol{\Phi}$ *decompose additively* in terms of *local* feature functions, $\boldsymbol{\varphi}$ over the steps that make up a derivation. For defining distributions under an MG G, we will assume that feature functions decompose over the productions in a derivation under the MCFG projection $\pi(G)$, i.e.,

$$\boldsymbol{\Phi}(\tau) = \sum_{(N \Rightarrow \delta) \in \pi(\tau)} \boldsymbol{\varphi}\left(N \Rightarrow \delta\right).$$

Under the locality assumption, we may rewrite the score $s_{\boldsymbol{\lambda}}(\tau)$ as

$$\prod_{(N \Rightarrow \delta) \in \pi(G)} \left(\exp(\boldsymbol{\lambda} \cdot \boldsymbol{\varphi}(N \Rightarrow \delta))\right)^{f_{\pi(\tau)}(N \Rightarrow \delta)}.$$

This (partially) addresses the issue of computational tractability, enforces our intuition that the score of a derivation tree should be a function of scores of its component steps, and still gives us the ability to avoid the overconditioning that we identified in Section 3.3.[12]

4.3 Locally normalized log-linear models

Even with our assumption of feature locality, finding $\hat{\boldsymbol{\lambda}}$ remains challenging since the second term

[12] We say that the issue of computational tractability is only *partially* resolved because only certain operations — identifying the most probable derivation of a string — are truly efficient. Computing the model's normalization function, while no longer NP-hard, still not practical.

in (13) is difficult to compute.[13] In this section we suggest a parameterization that admits both efficient ML estimation and retains the ability to use feature functions to control the distribution.

To do so, we revisit the approach of defining distributions on derivations in terms of a stochastic process from Section 3.1, but rather than defining the branching distributions with independent parameters for each MCFG nonterminal rewrite type, we parameterize it in terms of locally normalized log-linear models, also called a **conditional logit model** (Murphy, 2012). Given an MG G, a weight vector $\boldsymbol{w} \in \mathbb{R}^d$, and rule-local feature functions $\boldsymbol{\varphi}$ as defined above,[14] let the branching probability

$$p_{\boldsymbol{w}}(\delta \mid N) \doteq \frac{\exp(\boldsymbol{w} \cdot \boldsymbol{\varphi}(N \Rightarrow \delta))}{\sum_{(N \Rightarrow \delta') \in \pi(G)} \exp(\boldsymbol{w} \cdot \boldsymbol{\varphi}(N \Rightarrow \delta'))}.$$

Like the parametrization in Section 4.1, this new parametrization is based on log-linear models and therefore allows us to express similarities among derivational operations via choices of feature functions. However, rather than defining feature functions Φ_i on entire derivations, these features can only "see" individual MCFG rules. Put differently, the same technique we used in Section 4.1 to define a probability distribution over the entire set of derivations, is used here to define each of the local conditional probability distributions over the expansions of a single MCFG nonterminal. Via the perspective familiar from stochastic MCFGs, these individual conditional probability distributions together define a distribution on the entire set of derivations.

MLE. As with the previous two models, we can set parameters $\boldsymbol{w}$ to maximize the likelihood of the training data. Here, the global likelihood is expressed in terms of the probabilities of conditionally independent rewrite events, each defined in a log-linear model:

$$\mathcal{L}^c = \sum_{\tau} \tilde{f}(\tau) \sum_{(N \Rightarrow \delta) \in \pi(\tau)} f_{\pi(\tau)}(N \Rightarrow \delta) \log p_{\boldsymbol{w}}(\delta \mid N).$$

Its gradient with respect to $\boldsymbol{w}$ is therefore

$$\nabla_{\boldsymbol{w}} \mathcal{L}^c = \sum_{\tau} \tilde{f}(\tau) \sum_{(N \Rightarrow \delta) \in \pi(\tau)} f_{\pi(\tau)}(N \Rightarrow \delta) \Big[\boldsymbol{\varphi}(N \Rightarrow \delta) - \mathbb{E}_{p_{\boldsymbol{w}}(\delta' \mid N)} \boldsymbol{\varphi}(N \Rightarrow \delta') \Big].$$

As with the globally normalized model, $\nabla_{\boldsymbol{w}} \mathcal{L}^c = 0$ has no closed form solution; however, gradient-based optimization is likewise effective. However, unlike (13), this gradient is straightforward to compute since it requires summing only over the different rewrites of each non-terminal category during each iteration of gradient ascent, rather than over all possible derivations in $\Omega(G)$!

4.4 Example parameter estimation

In this section we compare the probability estimates for productions in a stochastic MCFGs obtained using the naive parameterization discussed in Section 3.2 that conditions on "too much" information and those obtained using locally normalized log-linear models with grammar-appropriate feature functions. Our very simple feature set consists just of binary-valued feature functions that indicate:

- whether a MERGE step, MOVE step, or a terminating lexical-insertion step is being generated;
- what selector feature (in the case of MERGE steps) or licensor feature (in the case of MOVE steps) is being checked (e.g., `+wh` or `=d` or `=v`); and
- what lexical item is used (e.g., *marie* : `d` or ϵ : `=t c`), in the case of terminating lexical-insertion steps.

Table 1 shows the values of some of these features for a sample of the MCFG rules in Fig. 6.

Table 2 compares the production probabilities estimated for last four rules in Fig. 6 using the naive empirical frequency method and our recommended log-linear approach with the features defined as above.[15] The presence or absence of a `-wh` feature does not affect the log-linear model's probability of adding an adverb to a verb phrase, in keeping with the perspective suggested by the derivational operations of MGs.

[13] Specifically, it requires computing expectations under all possible derivations in $\Omega(\pi(G))$ during each step of gradient ascent, which requires polynomial space/time in the size of the lexicon to compute exactly.

[14] The notational shift from $\boldsymbol{\lambda}$ to $\boldsymbol{w}$ to emphasizes that these two parameter vectors have very different semantics. The former parameterizes potential functions in a globally normalized random field while the later is used to determine a family of conditional probability distributions used to define a stochastic process.

[15] The log-linear parameters were optimized using a standard quasi-Newtonian method (Liu and Nocedal, 1989).

Table 1: Selected feature values for a sample of MCFG rules. The first four rules are the ones that illustrated the problems with the naive parametrization in Section 3.3.

MCFG Rule	φ_{MERGE}	$\varphi_{\texttt{=d}}$	$\varphi_{\texttt{=v}}$	$\varphi_{\texttt{=t}}$	φ_{MOVE}	$\varphi_{\texttt{+wh}}$
$st :: \langle \texttt{v} \rangle_0 \Rightarrow s :: \langle \texttt{=d v} \rangle_1 \quad t :: \langle \texttt{d} \rangle_1$	1	1	0	0	0	0
$st :: \langle \texttt{v} \rangle_0 \Rightarrow s :: \langle \texttt{=v v} \rangle_1 \quad t :: \langle \texttt{v} \rangle_0$	1	0	1	0	0	0
$\langle s, t \rangle :: \langle \texttt{v, -wh} \rangle_0 \Rightarrow s :: \langle \texttt{=d v} \rangle_1 \quad t :: \langle \texttt{d -wh} \rangle_1$	1	1	0	0	0	0
$\langle st, u \rangle :: \langle \texttt{v, -wh} \rangle_0 \Rightarrow s :: \langle \texttt{=v v} \rangle_1 \quad \langle t, u \rangle :: \langle \texttt{v, -wh} \rangle_0$	1	0	1	0	0	0
$st :: \langle \texttt{c} \rangle_0 \Rightarrow s :: \langle \texttt{=t c} \rangle_1 \quad t :: \langle \texttt{t} \rangle_0$	1	0	0	1	0	0
$ts :: \langle \texttt{c} \rangle_0 \Rightarrow \langle s, t \rangle :: \langle \texttt{+wh c, -wh} \rangle_0$	0	0	0	0	1	1

Table 2: Comparison of probability estimators.

MCFG Rule	Naive $\hat{p}$	Log-linear $\hat{p}$
$st :: \langle \texttt{v} \rangle_0 \Rightarrow s :: \langle \texttt{=d v} \rangle_1 \quad t :: \langle \texttt{d} \rangle_1$	0.95	0.94
$st :: \langle \texttt{v} \rangle_0 \Rightarrow s :: \langle \texttt{=v v} \rangle_1 \quad t :: \langle \texttt{v} \rangle_0$	0.05	0.06
$\langle s, t \rangle :: \langle \texttt{v, -wh} \rangle_0 \Rightarrow s :: \langle \texttt{=d v} \rangle_1 \quad t :: \langle \texttt{d -wh} \rangle_1$	0.67	0.94
$\langle st, u \rangle :: \langle \texttt{v, -wh} \rangle_0 \Rightarrow s :: \langle \texttt{=v v} \rangle_1 \quad \langle t, u \rangle :: \langle \texttt{v, -wh} \rangle_0$	0.33	0.06

5 Conclusion and Future Work

We have presented a method for inducing a probability distribution on the derivations of a Minimalist Grammar in a way that remains faithful to the way the derivations are conceived of in this formalism, and for obtaining the maximum likelihood estimate of its parameters. Our proposal takes advantage of the MG-MCFG equivalence in the sense that it uses the underlying probabilistic branching process of a stochastic MCFG, but avoids the problems of overparametrization that come with the naive approach that reifies the MCFG itself.

Our parameterization has several applications worth noting. It provides a new way to compare variants of the MG formalism that propose slightly different sets of primitives (operations, types of features, etc.) but are equivalent once transformed into MCFGs. Examples of such variants include the addition of an ADJOIN operation (Frey and Gärtner, 2002), or replacing MERGE and MOVE with a single feature-checking operation (Stabler, 2006; Hunter, 2011). Derivations using these different versions of the formalism often boil down to the same string-concatenation operations and will therefore be expressible using equivalent sets of MCFG rules. The naive parametrization will therefore not distinguish them, but in the same way that our proposal above "respects" standard MGs' classification of MCFG rules according to one set of derivational primitives, one could define feature vectors that respect different classifications.

Outside of MGs, the strategy is applicable to any other formalisms whose derivations can be recast as those of MCFGs, such as TAGs and CCGs. More generally still, it could be applied to any formalism whose derivation tree languages can be characterized by a local tree grammar; in our case, the relevant local tree language is obtained via a projection from the regular tree language of MG derivation trees.

Acknowledgments

Thanks to John Hale for helpful discussion and to the anonymous reviewers for their insightful comments. This work was sponsored by NSF award number 0741666, and by the U. S. Army Research Laboratory and the U. S. Army Research Office under contract/grant number W911NF-10-1-0533.

References

Steven P. Abney. 1997. Stochastic attribute-value grammars. *Computational Linguistics*.

Zhiyi Chi. 1999. Statistical properties of probabilistic context-free grammars. *Computational Linguistics*.

Stephen Della Pietra, Vincent Della Pietra, and John Lafferty. 1997. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4).

Werner Frey and Hans-Martin Gärtner. 2002. On the treatment of scrambling and adjunction in minimalist grammars. In Gerhard Jäger, Paola Monachesi, Gerald Penn, and Shuly Wintner, editors, *Proceedings of Formal Grammar 2002*, pages 41–52.

Joshua Goodman. 1999. Semiring parsing. *Computational Linguistics*, 25(4).

John Hale. 2006. Uncertainty about the rest of the sentence. *Cognitive Science*, 30:643–672.

Tim Hunter. 2011. Insertion Minimalist Grammars: Eliminating redundancies between merge and move. In Makoto Kanazawa, András Kornai, Marcus Kracht, and Hiroyuki Seki, editors, *The Mathematics of Language (MOL 12 Proceedings)*, volume 6878 of *LNCS*, pages 90–107. Springer, Berlin Heidelberg.

Aravind Joshi. 1985. How much context-sensitivity is necessary for characterizing structural descriptions? In David Dowty, Lauri Karttunen, and Arnold Zwicky, editors, *Natural Language Processing: Theoretical, Computational and Psychological Perspectives*, pages 206–250. Cambridge University Press, New York.

Laura Kallmeyer. 2010. *Parsing Beyond Context-Free Grammars*. Springer-Verlag, Berlin Heidelberg.

Gregory M. Kobele, Christian Retoré, and Sylvain Salvati. 2007. An automata theoretic approach to minimalism. In James Rogers and Stephan Kepser, editors, *Proceedings of the Workshop on Model-Theoretic Syntax at 10; ESSLLI '07*.

Daphne Koller and Nir Friedman. 2009. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

Dong C. Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical Programming B*, 45(3):503–528.

Jens Michaelis. 2001. Derivational minimalism is mildly context-sensitive. In Michael Moortgat, editor, *Logical Aspects of Computational Linguistics, LACL 1998*, volume 2014 of *LNCS*, pages 179–198. Springer, Berlin Heidelberg.

Kevin P. Murphy. 2012. *Machine Learning: A Probabilistic Perspective*. MIT Press.

Hiroyuki Seki, Takashi Matsumara, Mamoru Fujii, and Tadao Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science*, 88:191–229.

Edward P. Stabler and Edward L. Keenan. 2003. Structural similarity within and among languages. *Theoretical Computer Science*, 293:345–363.

Edward P. Stabler. 1997. Derivational minimalism. In Christian Retoré, editor, *Logical Aspects of Computational Linguistics*, volume 1328 of *LNCS*, pages 68–95. Springer, Berlin Heidelberg.

Edward P. Stabler. 2001. Recognizing head movement. In Philippe de Groote, Glyn Morrill, and Christian Retoré, editors, *Logical Aspects of Computational Linguistics*, volume 2099 of *LNCS*, pages 254–260. Springer, Berlin Heidelberg.

Edward P. Stabler. 2006. Sidewards without copying. In Shuly Wintner, editor, *Proceedings of The 11th Conference on Formal Grammar*, pages 157–170. CSLI Publications, Stanford, CA.

Edward Stabler. forthcoming. Two models of minimalist, incremental syntactic analysis. *Topics in Cognitive Science*.

K. Vijay-Shanker, David J. Weir, and Aravind K. Joshi. 1987. Characterizing structural descriptions produced by various grammatical formalisms. In *Proc. 25th Meeting of Assoc. Computational Linguistics*, pages 104–111.