Finite Growth Models

Eric Sven Ristad   Peter N. Yianilos

Research Report* CS-TR-533-96
December, 1996 (revised April 1997)

Abstract

Finite growth models (FGM) are nonnegative functionals that arise from parametrically-weighted directed acyclic graphs and a tuple observation that affects these weights. The weight of a source-sink path is the product of the weights along it. The functional's value is the sum of the weights of all such paths. The mathematical foundations of hidden Markov modeling (HMM) and expectation maximization (EM) are generalized to address the problem of functional maximization given an observation. Probability models such as HMMs and stochastic context free grammars are examples that satisfy a particular constraint: that of summing or integrating to one. The FGM framework, algorithms, and data structures describe these and other similar stochastic models while providing a unified and natural way for computer scientists to learn and reason about them and their many variations.

Restricted to probabilistic form, FGMs correspond to stochastic automata that allow observations to be processed in many orderings and groupings – not just one-by-one in sequential order. As a result the parameters of a highly general form of stochastic transducer can be learned from examples, and the particular case of string edit distance is developed.

In the FGM framework one may direct learning by criteria beyond simple maximum-likelihood. The MAP (maximum *a posteriori estimate*) and MDL (minimum description length) are discussed along with the application of FGMs to causal-context probability models and unnormalized noncausal models.

Keywords: *Expectation Maximization (EM), Hidden Markov Model (HMM), Baum-Welch Reestimation, Stochastic: Model, Grammar, Transducer, Transduction, Metric Learning, Maximum Likelihood (ML), Maximum a Posteriori (MAP), Minimum Description Length (MDL), String Edit Distance, String Similarity.*

*Both authors are with the Department of Computer Science, Princeton University, 35 Olden Street, Princeton, NJ 08544. The second author is also with the NEC Research Institute, 4 Independence Way, Princeton, NJ 08540. The first author is partially supported by Young Investigator Award IRI-0258517 from the National Science Foundation. Email: {ristad,pny}@cs.princeton.edu.

1 Introduction

Hidden discrete-state stochastic approaches such as hidden Markov models (HMM) for time series analysis, stochastic context free grammars (SCFG) for natural language, and statistical mixture densities are used in many areas including speech and signal processing, pattern recognition, computational linguistics, and more recently computational biology. These are parametric models and are typically optimized using the Baum-Welch, inside-outside, and expectation maximization (EM) algorithms respectively. They share a common mathematical foundation and are shown to be instances of a single more general abstract recursive optimization paradigm which we refer to as the *finite growth model* framework (FGM) involving non-negative bounded functionals associated with finite directed acyclic graphs (DAG). These functionals need not be probabilities and an FGM need not correspond to a stochastic process.

With FGMs interesting new kinds of stochastic models may be designed, existing models may be improved, settings that are not strictly probabalistic in nature may be addressed, and it becomes easier to design and reason about a wide range of problems. This paper introduces and develops the FGM framework and then applies it to:

1. the definition of *k-way stochastic transducers* whose parameters may be conveniently optimized based on training data;

2. the special case of 2-way string transduction which is shown to correspond closely with the notion of *string edit distance* where parameter optimization corresponds to learning insert, delete, and substitute costs that best explain a training corpus of similar strings;

3. the inclusion of model complexity tensions such as *minimum description length* (MDL) and *maximum a posteriori parameter* (MAP) estimation within models such as HMMs while retaining the ability to easily optimize them;

4. show that context dependent observation functions may be used to extend the power of conventional stochastic models while preserving the convenience of parameter reestimation;

5. consider problems that are not strictly probabilistic in nature such as parameter estimation for unnormalized noncausal image models and investment portfolio optimization; and

6. provide a unified framework for understanding existing models such as SCFGs and HMMs along with their many variations by giving time and space efficient reductions to FGMs.

Graphical models [20, 27] represent a branch of related work in which the independence structure of a set of random variables is the main focus. One can express such models using FGMs – as well as considerably more general settings such as that in which no two variables are independent (their graphical model is fully connected), but they are jointly constrained in interesting ways.

An FGM is a directed acyclic graph with a single source and sink together with a collection of non-negative bounded weight functions $\{w_i\}$ that are associated with edges – one per edge. The value of a source-sink path is the product of the weights along it and the FGM's value is the sum over all such paths of their values. Section 2 presents a formal development but we will sketch the basic ideas here.

The weight function $w_i(z_e|\Psi_i)$ associated with edge e accepts an argument z_e and parameters Ψ_i. The objective of FGM optimization is to maximize the FGM's value over its parameters $\{\Psi_i\}$ which are assumed to be independent. The argument z_e is regarded as fixed. The same weight function may be associated with several edges but will in general assume different values since z_e is a function of the edge to which it is attached.

If each edge has a distinct associated weight function then optimization is immediately reduced to the task of independently maximizing the weight functions in isolation. The interesting situation arises when this is not the case. Here a change to a single parameter set Ψ may affect weights throughout the DAG. The values of many source-sink paths may be affected and an interaction between members of $\{\Psi\}$ arises, i.e. simple independent maximization is no longer possible.

Surprisingly it is still possible to decompose the optimization problem into certain independent subproblems. This is the key mathematical message of Baum-Welch and EM generalized to the FGM framework. However, even exact solution of the subproblems does not yield a maximum for the FGM. But a new combined parameter set does result that strictly improves it — unless one has already converged to a limiting value. Iteration is then used to *climb* to such a limiting value. Unlike gradient descent with line search this approach confidently moves to a sequence of increasingly better points in parameter space and in many cases reduces to a strikingly simple and intuitive algorithm. The cost functions may themselves be FGMs and the computation of an improved parameter set simply depends upon the ability to solve the primitive maximization problems at the bottom of the recursive hierarchy.[1]

The mathematical idea behind this decomposition begins with a simple equality having to do with the log of a sum of terms. Let $S = T_1 + \ldots + T_n$ and define proportions $P_i = T_i/S$. Notice $S = T_i/P_i$. For any nonnegative vector R with unity sum:

$$\begin{aligned} \log S &= \sum_i R_i \log S && (1)\\ &= \sum_i R_i \log T_i/P_i && (2)\\ &= \sum_i R_i \log T_i - \sum_i R_i \log P_i && (3) \end{aligned}$$

Observe that the second term is maximized when $P = R$ since both are stochastic vectors.

In our setting each term T_i corresponds to a source-sink path, and its value is not constant, but rather is a function of some parameter set Ψ'. Optimizing the FGM is then the problem of maximizing their sum over Ψ'. This is the same as optimizing their log sum; connecting the equality above to the problem of FGM optimization.

Accordingly we parameterize the equality by writing $S(\Psi') = T_1(\Psi') + \ldots + T_n(\Psi')$ and $P(\Psi') = T_i(\Psi')/S(\Psi')$ where our objective is to maximize S over Ψ'. The search for a maximum takes place iteratively. During each iteration we denote the *current* parameters by Ψ and vary Ψ' looking for an improved set.

To express this iterative approach the equality is specialized by defining R to be P evaluated at the current parameters Ψ, i.e. $R \triangleq P(\Psi)$. Then:

[1]We must remark that it is important to realize that convergence to a limiting value for the FGM is not the same as convergence of its parameters.

$$\log S(\Psi') = \sum_i P_i(\Psi) \log T_i(\Psi') - \sum_i P_i(\Psi) \log P_i(\Psi')$$

The right term may be ignored because any departure from Ψ will decrease it whereby $\log S$ and therefore S are increased. We then focus on maximizing the left term which corresponds to the Q function of the HMM/EM literature.

Elements of the remaining summation correspond to source-sink paths through the FGM's DAG and the proportions $P_i(\Psi)$ are the relative contribution of each path to the FGM's value. Each term $T_i(\Psi')$ is then a product of of weight functions and breaks up into a sum of individual log terms. The proportions are then distributed over them and the final step is to group terms by weight function. The problem is then decomposed as required into independent subproblems each involving a single weight function.

The argument above is made in much greater detail in section 2 and the formal definition of an FGM presented there is more involved in two respects. First, there are two weight functions, not one, associated with each edge. These are abstractions of the normally separate transition and observation generation functions of a stochastic automaton. Second, the z_e in our discussion above is defined to be some function of an observation tuple x. A simple such function is that of coordinate projection where, for example, the function's value might be that of a single dimension of x.

The contribution of section 2 is the development of a framework that has nothing directly to do with stochastic models or probability in which the essential ideas of Baum-Welch and EM nevertheless apply. Our view is that the essential content of these ideas is one of decomposability of certain optimization problems that are captured in a somewhat general way by our definition of an FGM.

The early literature, especially theorem 2.1 of [3], reveals an awareness that the mathematics of HMMs apply beyond strictly probabalistic settings. This direction of generalization does not however appear to have been followed until now. Thinking seems to have been dominated by the motivating problem: the optimization of Markovian stochastic models. Because of the limited computational capabilities of the time and their highly mathematical viewpoint a great deal of emphasis was also placed on the primitive weight functions for which exact maximization may be performed in *closed form*, and those which give rise to unique global maxima. By contrast we devote very little attention to the the specific characteristics of these primitive functions and view them instead as an abstract types that by assumption support a set of required operations.

In section 3 the FGM framework is specialized to the case of probabilistic models. The z_i above result from projections that select one or more coordinate dimensions such that along each source-sink path each dimension is selected exactly once. Also, the weight functions are restricted to be probability functions or densities. It is not important that the dimensions occur in any fixed order as one follows alternative source-sink routes; nor is it required that the dimensions be selected one at a time. The contribution of this outlook is that FGM-based stochastic models can generate observations in many orderings and groupings – not just one-by-one in sequential order as in HMMs – a capability exploited later in our discussion of stochastic transducers.

Section 4 begins by considering the case of infinite stochastic processes. Viewed generatively these models produce output of unbounded size but are used in practice to evaluate the probability of finite observations and are trained using a finite set of finite observations. Restricted in this way they correspond to FGMs that capture their *truncated* form. Moreover the conventional evaluation and learning algorithms associated with them arise naturally from this viewpoint.

Baum-Welch and EM are today regarded as *maximum-likelihood* parameter estimation techniques. A more modern *learning theoretic* perspective recognizes the important tension between model complexity and available training data. An important contribution of section 4 is the description of a construction that modifies the FGM associated with a stochastic model so that alternative criteria such as MDL or MAP can guide optimization. The optimization is still based on Baum-Welch/EM so we suggest that these should *not* be viewed as statistical ML techniques and that their broader utility is revealed by our more general functional optimization perspective.

The observation functions associated with each state in a conventional Hidden Markov model must satisfy an independence assumption: that their state dependency is limited to the current state (the Markov condition). Section 4 observes that conditioning on earlier portions of the observation sequence does not violate this assumption and demonstrates that using the FGM framework, it is easy to see that parameter reestimation for such context dependent HMMs decomposes into primitive context optimization problems. This generalizes Brown's use [7] of the previous speech window to condition generation of the current one. Thus the power of causal context models such as statistical language models may be added to the HMM framework and may lead to improved performance.

The section also presents another illustration that the reach of FGMs extends in a useful way beyond conventional and strictly stochastic models. Causal context models express the probability of an observation as a product of conditional probabilities such that each dimension is predicted once and does appear earlier as a conditioning variable. These restrictions are constrictive in settings such as image modeling where modeling each pixel based on its surrounding context seems natural. Simply relaxing the requirements that probabalistic modeling places on an FGM's source-sink projection functions proves that FGM optimization will improve even noncausal models like the image example above.

Stochastic models describe the manner in which an object is generated. This object is conventionally imagined to be something like an image, an audio recording, or a string of symbols. If instead one models pairs of such objects the focus shifts from describing the nature of an individual to uncovering the relationship between them. The pairs may be thought of as instances of the *are similar* concept and learning an effective model to describe them is an example of what the authors have called the *metric learning* paradigm.

Fixing the left element of the pair one might then ask what generative sequences could explain the right. This is the essential idea of 2-way stochastic transduction. Section 5 shows that a somewhat general automaton-based definition for k-way transduction may be reduced to problems within the FGM framework. One may view speech recognition as a transduction, perhaps in several stages, and the application of learned transducers to this field represents an interesting area for future work.

Perhaps the best known example of 2-way transduction is that of *string edit distance* which is defined as the least costly way to transform one string into another via a given set of weighted insertion, deletion, and substitution editing operations. The development in section 5.1 begins with a natural reformulation into stochastic terms first noted by Hall and Dowling in [12, P.390-1]. We refer to the result as *stochastic edit distance* [24].

Our focus is on the interesting problem of learning insert, delete, and substitute costs from a training corpus $(s_1, t_1), \ldots, (s_n, t_n)$ of string pairs. The objective is to minimize the total stochastic edit distance of the collection. In [24] the authors give such a learning algorithm independent of the FGM framework and report on experiments. Until now the costs which parameterize edit distance algorithms have been prescribed not learned. Section 5.1 of this paper describes the reduction of this learning problem to the FGM optimization framework and may be viewed as an adjunct to [24] providing a proof of correctness.

The optimized costs are often quite different from the initial ones. It is in this sense that we *learn* them. In previous work, the costs have been stipulated after study of the problem domain. Even worse, maximally uninformative unit costs (i.e., Levenshtein distance) are sometimes stipulated when the problem domain is nontrivial. It is now possible to improve any educated guess, or start virtually from scratch from unit costs, to arrive at a set of optimized ones. Algorithms 7 and 8 of section 5.1 run in time proportional to the product of the two string lengths, matching the complexity of the conventional edit distance computation. Since edit distance has found widespread application our discovery of a simple way to learn costs may lead to improved performance in many problem areas.

Section 5.1 also observes that the notion of string edit distance may be strengthened so that the cost of an edit operation depends not only on its target but also on context. For example, such context-sensitive string edit distances allow the cost of inserting a symbol to depend on the symbol left-adjacent to the insertion site. The result remains an FGM on one therefore the ability to easily improve costs given a training corpus is retained.

Several kinds of hidden discrete-state stochastic models are in widespread use today. Hidden Markov Models (HMM) were introduced in the 1960's [2, 3, 1] and are typically applied in time-series settings when it is reasonable to assume that the observed data are generated or approximated well by the output of an automaton which changes state and generates output values stochastically. A matrix-based approach is typically used to formalize this notion, and [21] and [14] provide nice overviews. Normal (Gaussian) mixtures find application in statistical pattern recognition where items to be classified are represented as real-valued vectors. Stochastic context free grammars (SCFG) lead to a natural probabilistic interpretation of natural language parsing. Other model types exist, and are easily formed by combining and modifying the standard ones.

The situation is complicated by *parameter tying*; the practice of reducing the number of parameters in a model by equating two or more different sets of formal parameters to a single set. Observations are sometimes discrete and sometimes continuous. Entire models may be mixed together or recursively combined. Some states or transitions may be *nonemitting* meaning that no observation is generated. Finally, *nonuniform* time-series models may predict several future data points. All of these factors make it difficult to write down a single high-level framework with which to reason about and verify the correctness of a model's associated algorithms.

The contribution of FGMs is to move instead to a lower-level representation which is common to all of the models and complicating variations above. As noted earlier isn't really the original model which is represented, but rather its truncation to the given collection of finite observations. By understanding FGMs it is possible to reason about the others as mappings from their individual description language to an FGM given finite observations.

Section 6 relates the models above to FGMs by describing a reduction for each. As further illustration it considers the portfolio optimization problem and shows that the simple iterative algorithm reported in [9] for optimization arises by reducing the problem to an FGM.

The work described in this paper began in 1993 with our efforts to model handwriting. The focus then was on transduction, in an attempt to evaluate and learn a notion for similarity for the off-line case. This effort exposed many limitations of the conventional modeling outlook – problems that extend far beyond handwriting. To address these limitations we had to break several *rules* of conventional modeling only to ultimately discover that these rules are in many cases unnecessary artifacts of the history that led to HMMs and EM. After three years of generalization, and exercises in specialization, we arrived at the FGM outlook reported here.

2 Finite Growth Models

In this section we develop the theory of Finite Growth Models (FGM) as a class of nonnegative functionals represented as annotated directed acyclic graphs. The next section focuses on the special case of probability models. Our formal definition of an FGM amounts to an intricate recursive *data structure*, in the computer science sense, that has been crafted to span a wide range of applications. As a result it may seem at first to be rather abstract and have little to do with stochastic modeling, but each feature is an abstraction of some characteristic required by a problem we considered. It is the mathematical engine at the center of these problems. In particular much of our development and notation parallels that of Hidden Markov Models (HMM).

Let x be a tuple observation with components $x_1, \ldots, x_d$, and $\mathcal{X}$ denote the space of all such observations. We will not specify the type of each component since for much of our development it matters only that appropriate functions exists on the space of values that a component may assume. Components may therefore represent discrete values such as letters of the alphabet, continuous values such as real numbers, or other more complex structures. Within the observation tuple, component types need not be the same. In simple *time series* probability-model settings, components do have the same type, and position within the tuple corresponds to the passage of time.

Definition 1 *A finite growth model F for a d-dimensional observation space $\mathcal{X}$, is an 8-tuple (V, E, M, C, S, m, c, s), such that:*

1. *(V, E) is a finite DAG with vertices $V = v_1, \ldots, v_n$ having a single source v_1 and sink v_n. For $e \in E$ we denote by $s(e)$ the index of its source vertex, and by $d(e)$ the index of its destination vertex.*
2. *M is a finite collection $\{M_i\}$ of parameterized nonnegative bounded observation functionals with corresponding parameters $\{\Psi_i\}$. We write $M_i(y|\Psi_i)$ to denote evaluation of the ith observation functional, where the domain is not yet specified.*
3. *C is a finite collection $\{C_i\}$ of discrete-domain parameterized nonnegative bounded choice functionals. The size of the domain of $C_i(j)$ is denoted $|C_i|$, and the domain consists of the integers $1, \ldots, |C_i|$. The parameters of C_i are denoted Υ_i and we write $C_i(j|\Upsilon_i)$ to denote evaluation of the ith choice functional.*
4. *S is a finite collection $\{S_i\}$ of selection functions with domain X where the range is not yet specified.*
5. *$m : E \to \{1, \ldots, |M|\}$ is a function associating an element of M with each edge.*
6. *$c : E \to \{1, \ldots, |C|\} \times \mathbb{N}$ is a function that associates each edge with a choice function and a member of its range; which is formalized by denoting the projections of c by c_1, c_2 and requiring:*
   - (a) *$s(e_1) = s(e_2) \Rightarrow c_1(e_1) = c_1(e_2), \forall e_1, e_2 \in E$, so that an element of C is in effect associated with each nonsink vertex, and;*
   - (b) *For each nonsink vertex v, c_2 restricted to $\{e|s(e) = v\}$ is 1-1 and onto the range of $C_{c_1(v)}$.*

The choice value corresponding to an edge e is then $C_{c_1(e)}(c_2(e)|\Upsilon_{c_1(e)})$ and is denoted $C(e|\Upsilon)$ for brevity where it is understood that in this context Υ refers to $\Upsilon_{c_1(e)}$ – but the subscript is sometimes written for clarity.

7. *$\varsigma : E \to \{1, \ldots, |S|\}$ is an function that associates each edge with a selection function such that for all edges e, the range of $S_{\varsigma(e)}$ matches the domain of $M_{m(e)}$. The observation value corresponding to an edge e is then $M_{m(e)}(S_{\varsigma(e)}(x)|\Psi_{m(e)})$ and is denoted $M(e, x|\Psi)$ for brevity where it is understood that in this context Ψ refers to $\Psi_{m(e)}$ – but the subscript is sometimes written for clarity.*

The combined observation and choice parameters are denoted Ψ and Υ respectively, and $\Phi \triangleq (\Psi, \Upsilon)$ denotes the parameters of F. The value of a source-sink path is the product of the observation and choice functional values along it. The value $F(x|\Phi)$ of the FGM is the sum over all source-sink paths, of the value of each. A choice or observation model whose value is the constant 1 does not affect an FGM's value and is therefore said to be void.

Notice that since an FGM is itself a nonnegative parameterized functional, the definition above may be applied recursively to define an FGM whose constituent functionals are themselves FGMs.

Direct evaluation of an FGM's value by enumerating all source-sink paths is of course not practical. Fortunately $F(x|\Phi)$ may be efficiently computed using dynamic programming. To express this precisely requires some notation. Let $v_1, \ldots, v_n$ be a topologically sorted vertex listing. Following the convention of the HMM literature, we define corresponding real variables $\alpha_1, \ldots, \alpha_n$. The sets of edges *into* and *out from* a vertex v are denoted by $\mathcal{I}(v)$ and $\mathcal{O}(v)$ respectively. Using the notation above, the contribution of an edge e to the value of a path containing it, is $C(e|\Upsilon) \cdot M(e, x|\Psi)$ The product of these contributions is the probability of the path. The complete dynamic program is then given by the following simple algorithm:

Algorithm 1

```
procedure forward
    α_1 = 1
    for i = 2, ..., n
        α_i = Σ_{e∈I(v_i)} α_{s(e)} · C(e|Υ) · M(e, x|Ψ)
```

This may be recognized as the forward step of the Baum-Welch HMM *forward-backward* algorithm [2, 1, 21] adapted to the more general FGM setting. Following execution α_n has value $F(x|\Phi)$.

Algorithm 1 is said to *evaluate* the FGM using its current set of parameters. The optimization problem we consider seeks a parameter set that maximizes the FGM's value

$$\overline{\Phi} = \mathrm{argmax}_{\Phi} F(x|\Phi) \tag{4}$$

but we will be content to find locally optimal solutions, and in some cases to simply make progress. Here x is fixed and it is the FGM's parameters that are varied.

The end-to-end concatenation of two FGMs clearly results in another whose value is the product of the two, so that given a set of observations $x_1, \ldots, x_m$, the optimization problem

$$\overline{\Phi} = \mathrm{argmax}_{\Phi} \prod_{i=1}^{m} F(x_i|\Phi) \tag{5}$$

is really just an instance of Eq. 4. Viewing $\{x_i\}$ as training examples the optimization then *learns* a parameter set, or in the language of statistics and assuming the FGM corresponds to a probability model, *estimates* its parameters.

Our approach to this problem extends the idea of Baum-Welch to FGMs. Several years following the development of HMMs, essentially the same mathematical ideas expressed in the their algorithm were rediscovered as the expectation maximization (EM) algorithm for mixture densities [11, 22]. This work focused on parameter estimation for mixture densities, and has had considerable influence on a somewhat independent community. In what follows we will use the phrase *Baum-Welch/EM* to refer to the adaptation of these results to the FGM setting.

The HMM and EM literature deals with probabalistic models, and Baum-Welch/EM is presented as a method for reestimating their parameters. Our mathematical contribution is the recognition that the method is not essentially a statement about probability but rather should be viewed as an iterative approach to the maximization of arbitrary parameterized nonnegative functionals of a certain form. The form consists of a sum of terms each one of which is a product of subterms. An FGM's DAG is a representation of these forms which, for many with appropriate structure, has the advantages of succinct representation and computational economy. In the interesting case, the subterms are partitioned or grouped into families sharing common parameters. The approach then consists of focusing on each family in isolation thus localizing the problem. Individual family maximization problems are constructed that include certain weighting factors. These factors are constant with respect to the family's optimization but are computed using the entire DAG and are therefore influenced by all families. Thus global information is used to decompose the original problem – localizing it to individual families. If parameters are found that improve any of the individual family problems, the FGM's value will strictly increase. The method is iterative since even exact maximization of every family's problem may not maximize the FGM's value. Instead the process is repeated.

To formalize this idea of simply making progress with respect to a given maximization problem, the notation:

$$\text{argclimb}_{a||b} f(a) \triangleq \{a | f(a) \geq f(b)\}$$

is introduced. Notice that the improvement need not be strict and that the result is a set of values.

Our definition of an FGM is clearly an abstraction of stochastic modeling. We remark that it is possible to start from a definition that is yet more general and in a sense simpler in which only choice models are used and a variable number may be attached to each edge. But then connecting our results to the mainly stochastic problems we are interested in becomes more complex, and it is more difficult to see the connection between our generalization and the original literature.

We therefore develop Baum-Welch/EM for FGMs starting from a lemma that is stated so as to illustrate its correspondence with the EM literature. It is a statement about improper mixture densities, i.e. where the whole and its constituent parts are not necessarily probabilities. This is relevant to FGMs because they may be regarded as immense mixtures of contributions from every source-sink path. We follow earlier lines in its proof without requiring proper densities. The lemma may also be derived starting from theorem 2.1 of [3] in its most general form.

Lemma 1 *(Baum-Welch/EM) Let* $f(x|\Phi) \triangleq \sum_{i=1}^{k} g(x|\omega_i, \Psi) h(\omega_i|\Upsilon)$ *be a finite parameterized mixture where* $\{g(\cdot|\omega_i)\}$ *and* h *are nonnegative parameterized functionals (but not necessarily probability* *functions or pdfs), ω_i selects a component of the mixture, and $\Phi \triangleq \{\Psi, \Upsilon\}$. Also define probability $P(\omega_i|x,\Phi) \triangleq g(x|\omega_i,\Psi)h(\omega_i|\Upsilon)/f(x|\Phi)$. Finally, assume $f(x|\Phi) > 0$. Then:*

$$\overline{\Phi} \in \mathrm{argclimb}_{\Psi',\Upsilon'||\Psi,\Upsilon} \left(\sum_{i=1}^{k} P(\omega_i|x,\Phi) \log g(x|\omega_i,\Psi') + \sum_{i=1}^{k} P(\omega_i|x,\Phi) \log h(\omega_i|\Upsilon') \right)$$

is an improved parameter set. That is, $f(x|\overline{\Phi}) \geq f(x|\Phi)$, with the inequality strict unless Φ is already optimal. It is understood that Ψ', Υ' vary over their defined domain.

proof: We begin by establishing the identity:

$$\begin{aligned} E_{\omega|x,\Phi}(\log f(x|\Phi')) &= \sum_{i=1}^{k} P(\omega_i|x,\Phi) \log f(x|\Phi') \\ &= \log f(x|\Phi') \sum_{i=1}^{k} P(\omega_i|x,\Phi) \\ &= \log f(x|\Phi') \end{aligned}$$

where ω is viewed as a discrete random variable distributed as $P(\omega_i|x,\Phi)$ and $E_{\omega|x,\Phi}$ denotes expectation with respect to it. The point is merely that $f(x|\Phi')$ is independent of ω since the former depends on Φ' not Φ.

Next with $f(x,\omega_i|\Phi') \triangleq g(x|\omega_i,\Psi')h(\omega_i|\Upsilon')$ we have $\log f(x|\Phi') = \log f(x,\omega|\Phi') - \log P(\omega|x,\Phi')$ since $f(x,\omega|\Phi') = P(\omega|x,\Phi')f(x|\Phi')$. So:

$$\begin{aligned} \log f(x|\Phi') &= E_{\omega|x,\Phi} \log f(x|\Phi') \\ &= E_{\omega|x,\Phi} \log f(x,\omega|\Phi') - E_{\omega|x,\Phi} \log P(\omega|x,\Phi') \\ &= \sum_{i=1}^{k} P(\omega_i|x,\Phi) \log f(x,\omega_i|\Phi') - \sum_{i=1}^{k} P(\omega_i|x,\Phi) \log P(\omega_i|x,\Phi') \end{aligned}$$

It follows from Jensen's inequality that the second summation is minimized by $\Phi = \Phi'$. This can also be seen by recognizing it as $-[D(\Phi\|\Phi') + H(\Phi)]$, where $D(\cdot\|\cdot)$ is the Kullbak-Leibler distance, and $H(\cdot)$ denotes entropy (see [10]). Thus maximizing the first term, written $Q(\Phi,\Phi')$ in the literature, will surely increase $\log f(x|\Phi')$ and hence $f(x|\Phi')$. But:

$$\sum_{i=1}^{k} P(\omega_i|x,\Phi) \log f(x,\omega_i|\Phi') = \sum_{i=1}^{k} P(\omega_i|x,\Phi) \log g(x|\omega_i,\Psi') + \sum_{i=1}^{k} P(\omega_i|x,\Phi) \log h(\omega_i|\Upsilon')$$

and the right side is recognized as the object of the argclimb operator in the lemma's statement. $\square$

The conditional expectation operator $E_{\omega|x,\Phi}$ can cause confusion but we use it to make clear the correspondence with the EM literature. Our discussion in section 1 starting with Eq. 1 represents an alternative approach that is free of explicit mention of expectation and probability.

To apply lemma 1 to FGM optimization we introduce two more simple algorithms. The first computes $F(x|\Phi)$ by sweeping through the DAG's vertices in reverse rather than forward order, establishing the values of a new set of real variables $\beta_1, \ldots, \beta_n$:

Algorithm 2

procedure *backward*

$\beta_n = 1$

for $i = n-1, \ldots, 1$

$\beta_i = \sum_{e \in \mathcal{O}(v_i)} \beta_{d(e)} \cdot C(e|\Upsilon) \cdot M(e, x|\Psi)$

This corresponds to the backward step of the forward-backward algorithm. After it has run, β_1 equals $F(x|\Phi)$, and therefore α_n.

We denote by $\omega_1, \ldots, \omega_N$ the source-sink paths through F, and by $E(\omega)$ the set of edges along a particular path ω, and by $\Omega(e)$ the set of paths that include edge e. The contribution of a single path ω to the overall value of the FGM is denoted $F(x, \omega|\Phi)$ and given by:

$$F(x, \omega|\Phi) = \prod_{e \in E(\omega)} C(e|\Upsilon) \cdot M(e, x|\Psi)$$

So that:

$$F(x|\Phi) = \sum_{i=1}^{N} F(x, \omega_i|\Phi)$$

Next define $\gamma_\omega \triangleq F(x, \omega|\Phi)/F(x|\Phi)$, and then for each edge e define:

$$\gamma_e \triangleq \sum_{\omega \in \Omega(e)} \gamma_\omega$$

Notice that by definition the γ_e arise from a partition of the set of all source-sink paths whence it is clear that:

Proposition 1 *The sum of the γ_e for any cut of DAG is unity.*

The α and β variables may be used to compute these γ_e via a third simple dynamic program corresponding to the *expectation step* of EM:

Algorithm 3

procedure *gamma*

for $i = n-1, \ldots, 1$

for $e \in \mathcal{O}(v_i)$

$\gamma_e = (\alpha_{s(e)} \cdot C(e|\Upsilon) \cdot M(e, x|\Psi) \cdot \beta_{d(e)})/F(x|\Phi)$

where either α_n or β_1 may be substituted for $F(x|\Phi)$. As a practical matter algorithms 3 and 2 may be combined.

We are now in position to state precisely how the objective of optimizing an FGM is decomposed into smaller problems.

Definition 2 *An inexact Baum-Welch/EM reestimate $\overline{\Phi} = (\overline{\Psi}, \overline{\Upsilon})$ of the current parameters $\Phi = (\Psi, \Upsilon)$ of an FGM F given an observation x is formed by solving two sets of independent optimization problems:*

1. *For each i the improved parameters $\overline{\Psi}_i$ of observation model M_i are given by:*

$$\overline{\Psi}_i \in \mathrm{argclimb}_{\Psi'_i \| \Psi_i} \sum_{e \in m^{-1}(i)} \gamma_e \cdot \log M(e, x|\Psi'_i)$$

2. *For each i the improved parameters $\overline{\Upsilon}_i$ of choice model C_i are given by:*

$$\overline{\Upsilon}_i \in \mathrm{argclimb}_{\Upsilon'_i \| \Upsilon_i} \sum_{e \in c_1^{-1}(i)} \gamma_e \cdot \log C(e|\Upsilon'_i)$$

where the γ_e are computed based on $\Phi = (\Psi, \Upsilon)$. The exact Baum-Welch/EM reestimate is formed by replacing the argclimb *operations with* argmax. *By $m^{-1}(\cdot)$ and $c^{-1}(\cdot)$ we mean the set valued inverses of functions m and c.*

The *exact* form corresponds to the historical definition of the Baum-Welch/EM reestimate. However, since this paper's focus is on decomposition, and not mathematical issues relating to particular choice and observation models, we will use "Baum-Welch reestimate" to refer to our *inexact* form defined above. This allows our framework to include models for which exact solution is not convenient. Also it is interesting to note that one may choose to not reestimate one or more models, and still improve the overall FGM.

Theorem 1 *The Baum-Welch/EM reestimate $\overline{\Phi}$ satisfies $F(x|\overline{\Phi}) \geq F(x|\Phi)$ with the inequality strict provided that progress is made in at least one subsidiary optimization problem.*

proof: We write:

$$F(x|\Phi) = \sum_{i=1}^{N} \underbrace{\left(\prod_{e \in E(\omega_i)} M(e, x|\Psi_i) \right)}_{g(x|\omega_i, \Psi)} \cdot \underbrace{\left(\prod_{e \in E(\omega_i)} C(e|\Upsilon_i) \right)}_{h(\omega_i|\Upsilon)}$$

which makes clear the correspondence between the FGM and lemma 1. Now by their definitions $\gamma_{\omega_i} = P(\omega_i|x, \Phi)$ so from the lemma we have:

$$\overline{\Phi} \in \mathrm{argmax}_{\Phi', \Upsilon'} \left[\sum_{i=1}^{N} \gamma_{\omega_i} \log \left(\prod_{e \in E(\omega_i)} M(e, x|\Psi'_i) \right) + \sum_{i=1}^{N} \gamma_{\omega_i} \log \left(\prod_{e \in E(\omega_i)} C(e|\Upsilon'_i) \right) \right]$$

$$\overline{\Phi} \in \mathrm{argmax}_{\Phi', \Upsilon'} \left[\sum_{i=1}^{N} \sum_{e \in E(\omega_i)} \gamma_{\omega_i} \log M(e, x|\Psi'_i) + \sum_{i=1}^{N} \sum_{e \in E(\omega_i)} \gamma_{\omega_i} \log C(e|\Upsilon'_i) \right]$$

The double summations above enumerate every edge along every path though the FGM. To complete the proof we have only to enumerate them in a different order:

$$\overline{\Phi} \in \mathrm{argmax}_{\Phi',\Upsilon'} \left[\sum_{i=1}^{|M|} \sum_{e \in m^{-1}(i)} \sum_{\omega \in \Omega(e)} \gamma_\omega \log M(e, x|\Psi'_i) + \sum_{i=1}^{|C|} \sum_{e \in c_1^{-1}(i)} \sum_{\omega \in \Omega(e)} \gamma_\omega \log C(e|\Upsilon'_i) \right]$$

Recall $\gamma_e = \sum_{\omega \in \Omega_e} \gamma_\omega$ so:

$$\overline{\Phi} \in \mathrm{argmax}_{\Phi',\Upsilon'} \left[\sum_{i=1}^{|M|} \left(\sum_{e \in m^{-1}(M_i)} \gamma_e \log M(e, x|\Psi'_i) \right) + \sum_{i=1}^{|C|} \left(\sum_{e \in c_1^{-1}(i)} \gamma_e \log C(e|\Upsilon'_i) \right) \right]$$

and the parenthesized terms are the independent subproblems of definition 2. $\square$

We now change our focus to computational and data structural issues. The first part of our discussion deals with the intricacies of recursion. That is, when an FGM invokes another FGM as one of its observation models. Here, proper algorithm and data structure design can lead to polynomial rather than exponential complexity as illustrated by the case of stochastic context free grammars described in section 6.2. We will confine ourselves to finite recursion. The second issue has to do with the arithmetic demands of FGM computation.

The idea behind reducing the complexity of recursive FGM operations is a simple one: avoid duplicate work. If a particular combination of an observation model and selection function occurs multiple times, the combination should only be evaluated once. Since choice models do not depend on the observation, each should be evaluated only once. The computation is then ordered so that when an edge is considered, its choice and observation functions have already been evaluated and their values may be looked up in constant time.

To formalize this idea recall that $M(e, x|\Psi)$ is defined as $M_{m(e)}(S_{\varsigma(e)}(x)|\Psi_{m(e)})$. Notice that edge e appears as a function argument. We say $M(e_1, x|\Psi) \equiv M(e_2, x|\Psi)$ if their definitions are the same. Each equivalence class is referred to as a *model application* and expressing algorithms in terms of these classes avoids duplicate work. If for two edges $\varsigma(e_1) \neq \varsigma(e_2)$ then by this definition $M(e_1, x|\Psi) \not\equiv M(e_2, x|\Psi)$ – even if $S_{\varsigma(e_1)}$ is the same function as $S_{\varsigma(e_2)}$. For this reason we adjust the definition to account for equivalences that may exist among the $\{S_i\}$.

This adjustment is subtle and we therefore illustrate it by example. Suppose that the observation model associated with an edge e_i of the top level FGM is itself an FGM F_a and that selection function S_q is used by e_i. Now within F_a assume that some edge e_j uses an observation model m with selection function S_r. Then to evaluate m, selection functions S_q and S_r are composed. Now focus on another top level edge e_k to which an FGM F_b is attached using selection function S_t. Within F_b consider an edge e_ℓ using the same observation model m as does e_j, but in combination with selection function S_v. To evaluate m here S_t and S_v are composed. So m should only be evaluated once if $S_r(S_q(\cdot))$ and $S_v(S_t(\cdot))$ are the same function. This may be the case despite the fact that S_r and S_v are not. Thus, whenever possible, equivalence classes should be defined so as to account for possibly equivalent compositions of selection functions. This is easily implemented for the class of *projection* selection functions introduced in the next section.

Mathematically definition 1 states that each edge has an attached observation model via mapping m. Computationally the edge instead selects (in practice points to) a model application structure which identifies the observation model and the corresponding selection function. We denote the set of observation model applications by $\{\mathcal{M}_i\}$. Inclusion within the recursive hierarchy induces a partial ordering on this set where it is important to realize that a given model application may be referenced at more than one recursive level. We extend this ordering by specifying that all primitive observation models are dominated by any FGM. In what follows we will then assume that $\{\mathcal{M}_i\}$ is indexed in topological order so that the top-level FGM is first and the primitive models form the list's end.

We also introduce *choice model application structures* $\{\mathcal{C}_i\}$. Since choice models do not depend on x or involve a selection function, these structures are in 1:1 correspondence with the choice models themselves and are identically indexed. Many references may exist to a choice model, and its corresponding application structure avoids redundant model evaluation by recording the model's current value. It also accumulates γ contributions during expectation step processing so that a single call may be made to maximize the model. This is explained later in greater detail.

If an FGM F_c is a recursive child of a parent F_p, one alternative is to eliminate the recursion and expand the child *inline* within the parent by adding vertices and edges to its DAG. If F_c is invoked along edge e between vertices i and j then the first step of this inline expansion disconnects e from j and replaces the invocation of F_c with a void model. The child F_c is then inserted between the disconnected end and vertex j.

When evaluating F_p this inline expansion clearly isn't necessary since one can evaluate F_c in *isolation*, record its value in the model application structure, and in constant time refer to that value where needed in F_p. The same is true for parameter reestimation although a brief argument is needed.

Proposition 2 *Suppose edge e of FGM F_p invokes model F_c, also an FGM. Then if F_c is expanded inline within F_p, the γ values of its edges are exactly those computed in isolation multiplied by γ_e.*

proof: Let e connect vertices i and j and let ω denote some source-sink path of F_c and γ_ω its weight. If F_c is expanded inline the value of ω will be $\alpha_i C(e|\Upsilon)\gamma_\omega\beta_i/F_p(x|\Phi)$. In isolation it is $\gamma_\omega/F_c(x|\Phi)$. But $\gamma_e = \alpha_i C(e|\Upsilon)F_c(x|\Phi)\beta_i/F_p(x|\Phi)$ from which the proposition follows. $\Box$

So if the same model application $\mathcal{M}_i$ is pointed to by many FGM edges, the right computational approach is to first accumulate their γ values and then to process $\mathcal{M}_i$. Here we emphasize that this approach is more efficient *and* mathematically identical to inline expansion. So if at the bottom of the recursion the required argmax problems are solved, then at all higher levels they are too.

An observation model M_i that is not an FGM is said to be *primitive*, and is represented by an abstract type for which the following operations are defined:

1. $M_i : evaluate(z)$

2. $M_i : Estep(z, \Gamma)$

3. $M_i : Mstep()$

Here z is an element of the range of a selection function and Γ is a non-negative weight propagated down as described in proposition 2. The *evaluate* function returns the value of M_i at z. Notice that the model's parameters Ψ_i are hidden by the abstract type. In practice a mutator may be provided to set them and a selector provided for readout. The names *Estep* and *Mstep* follow the EM literature even though FGMs are not restricted to maximum-likelihood estimation of probability functions and densities.

Given a sequence of values $z_1, \ldots, z_m$, and nonnegative *multiplicities* $\Gamma_1, \ldots, \Gamma_m$ the *Estep* and *Mstep* procedures address the following optimization problem:

$$\overline{\Psi}_i = \text{argmax}_{\Psi'_i} \prod_{j=1}^{m} M_i^{\Gamma_j}(z_j|\Psi'_i) \tag{6}$$

which generalizes Eq. 5. The *Estep* procedure is called for each (z_i, Γ_i) and then *Mstep* is called to reestimate the model's parameters. For some models the result is the unique optimum parameter set, but in general the process is repeated. For example, if the z_i are letters of the alphabet, the Γ_i are positive *frequencies*, and the model's value is a probability for each letter, then *Estep* need only record the Γ values provided and *Mstep* can easily determine the unique optimal model. That is:

$$P(i) = \frac{\Gamma_i}{\sum_j \Gamma_j}$$

We point out that if the probability of symbol i is zero prior to reestimation, then γ_i and therefore its reestimate $P(i)$ are zero. Such parameters in this simple discrete model are then effectively disabled since their corresponding choice of alphabet symbol will always have probability zero.

A simple unique solution also exists for the normal density and others. In the discrete example above the model is constrained to produce values which sum to one over the alphabet. Normal densities are constrained to integrate to one. These constraints are obviously important so that the overall optimization problem has a bounded solution. But it is important to observe that their precise nature is irrelevant and invisible to the FGM framework. In particular there is no need for a model to represent a probability function There is also no need to restrict one's attention to models for which unique optimum solutions are readily available. As noted earlier, any progress made for even one model, will translate to progress in the overall FGM optimization.

A choice C_i is represented by an abstract type for which the following operations are defined:

1. C_i : *evaluate*(j)
2. C_i : *Mstep*($\Gamma[\ldots]$)

Here j is the index of an edge and the *evaluate* function returns its value under C_i given the model's current parameters Υ_i. The *Mstep* procedure is passed the vector $(\Gamma_1, \ldots, \Gamma_{|C_i|})$ and addresses the following optimization problem:

$$\overline{\Upsilon}_i = \text{argmax}_{\Upsilon'_i} \prod_{j=1}^{|C_i|} C_i^{\Gamma_j}(j|\Upsilon'_i) \tag{7}$$

If C_i is a probability function then the problem is exactly that of Eq. 6 and an exact solution is available. In both cases the solution is mathematically optimal but other problem constraints might be incorporated into C_i or M_i that sometimes forbid it. For example, one might prohibit probabilities from falling below some fixed threshold. This amounts to a change in parameterization and we will see in a later section that this capability may be used to alter the meaning of *optimal* in useful ways. Note that we might instead have treated a choice model as an observation model over the natural numbers but decided that their roles are sufficiently distinct to warrant separate designs.

We now return to our discussion of model applications and begin by describing the information that must be associated with these structures. An observation application structure $\mathcal{M}_i$ contains:

1. $\mathcal{M}_i : model$ — the integer index of the observation model associated with this application.

2. $\mathcal{M}_i : selector$ — the integer index of associated selector function.

3. $\mathcal{M}_i : value$ — a real number representing the most recent model application evaluation.

4. $\mathcal{M}_i : \Gamma$ — an accumulator to which values are added by the possibly many places within the overall model that refer to $\mathcal{M}_i$. In the case of primitive models we may then call *Estep* a single time. For FGMs the value corresponds to γ_e of proposition 2 and is used to propagate γ values down the recursive hierarchy.

A choice application structure $\mathcal{C}_i$ contains:

1. $\mathcal{C}_i : value[\ldots]$ — a vector of real numbers representing the choice model's value at each integer of its domain based on its current parameters.

2. $\mathcal{C}_i : \Gamma[\ldots]$ — a vector accumulator to which values are added by the possibly many places within the overall model that refer to $\mathcal{C}_i$.

No *model* variable is needed since indexing corresponds to that of the choice models themselves.

Recall that the mathematical definition 1 of an FGM associates edges directly with elements of M, C, S via functions m, c, s – there are no model application structures. While the FGM along with its associated model application structure lists might be generated incrementally, it is simpler to imagine that the FGM is first specified *mathematically* and then *compiled* into a form that includes these lists. This compiled form is denoted $\hat{F}$ and merges the M, C, S sets of each FGM into single densely enumerated sets. In compiled form FGM edges do not point directly to observation models but rather to observation model application structures. The m function is replaced by m_a to effect this change.

We present evaluation, expectation step, and maximization step algorithms for compiled FGMs. These are not recursive functions but rather operate by traversing the global model application lists built during compilation.

Before any of these are called the FGM must be initialized. This procedure $initialize(\hat{F})$ consists of setting the Γ variables to zero in all choice model application structures and then evaluating each choice model and recording its value there.

Evaluating an FGM and performing an expectation step are carried out by bottom-up (reverse order) and top-down (in order) traversals respectively of the model application lists. Evaluation returns a single value for $\hat{F}$ given observation x and also records the value of all subsidiary models within the observation model application structures.

Algorithm 4

function $evaluate(\hat{F}, x)$
    for each $\mathcal{M}_i$ in descending index order
        if "primitive"
            $j = \mathcal{M}_i : selector$
            $k = \mathcal{M}_i : model$
            $\mathcal{M}_i : value = M_k : evaluate(S_j(x))$
        else "an FGM"
            $\mathcal{M}_i : value =$ result of *algorithm 1*
    return $\mathcal{M}_1 : value$

where algorithm 1 refers to the appropriate observation and choice application structure value variables rather than computing $C(e|\Upsilon)$ and $M(e, x|\Psi)$.

As for primitive observation models the FGM expectation step procedure accepts a Γ argument. Supplying a value of 1 for each x observed corresponds to the optimization problem of Eq. 5. In general these Γ values generalize the problem by appearing as exponents to the FGM's value as in *Eq.* 6. The FGM is first evaluated so that the observation model application structures contain current values. Their Γ values are then set to zero except for the first which corresponds to the top-level FGM and is set to the function's Γ argument. The main loop proceeds top-down to propagate Γ contributions to lower observation model application structures. Eventually primitive structures are reached and a primitive expectation step is performed. The FGM's value is returned as a convenience since it was necessary to compute it.

Algorithm 5

function $Estep(\hat{F}, x, \Gamma)$
    $evaluate(\hat{F}, x)$
    for each $\mathcal{M}_i$
        if $i = 1$, $\mathcal{M}_i : \Gamma = \Gamma$, else $\mathcal{M}_i : \Gamma = 0$
    for each $\mathcal{M}_i$ in increasing index order
        if "primitive"
            $j = \mathcal{M}_i : selector$
            $k = \mathcal{M}_i : model$
            $M_k : estep(S_j(x), \mathcal{M}_i : \Gamma)$
        else "an FGM"
            Set $\{\gamma_e\}$ using algorithms 1,2,3
            for each edge e
                $\mathcal{M}_{m_a(e)} : \Gamma += \gamma_e \cdot \mathcal{M}_i : \Gamma$
                $\mathcal{C}_{c_1(e)} : \Gamma[c_2(e)] += \gamma_e \cdot \mathcal{M}_i : \Gamma$
    return $\mathcal{M}_1 : value$

The FGM maximization procedure maximizes the primitive observation models and all choice models. The FGM is then reinitialized.

Algorithm 6

function $Mstep(\hat{F})$
  for each M_i
    $M_i : maximize()$
  for each C_i
    $C_i : maximize(\mathcal{C}_i : \Gamma[\ldots])$
  $initialize(\hat{F})$

Observe that we do not specify that the models are maximized in any particular order since if their parameter sets are disjoint so is the act of maximizing them. Our notation thus-far has suggested that the parameter sets are in fact independent but in general they need not be. For example, two models might share the same parameters but compute two different functionals depending on them. In this case the two corresponding optimization subproblems of definition 2 are simply combined into one. From an implementation viewpoint this kind of situation can be handled entirely at the model type level through communication between them that is invisible at higher levels. So the fact that algorithm 6 specifies no order for maximization does not imply that we are limited to the case of disjoint parameter sets.

Analysis of the algorithms above is straightforward from the structure of the compiled FGM $\hat{F}$. Let N_E denote the sum over nonprimitive observation model applications, of the number of edges in the FGM corresponding to each. Then both evaluation and expectation step processing require $O(N_E) + T_p$ time where T_p denotes the time needed to perform the required primitive observation model type operations. These are described by the final entries in the list of observation model applications. When these primitive operations are $O(1)$ time the overall complexity is just $O(N_E)$ since their number cannot exceed N_E. Each choice model is evaluated once during *initialize*. The *maximize* operation time-complexity is just that required to perform a maximization step for each observation and choice model in the system. Ignoring the internal requirements of the observation and choice models, it is obvious that $\hat{F}$ requires $O(N_E)$ space.

The algorithms above are conceptually straightforward, but an important numerical issue must be faced in their implementation. As one sweeps through the DAG, the α and β variables can easily become too large or small to represent as conventional floating point values. This is true even of the final FGM value. If the FGM is a probability model, exponent-range underflow is the issue and corresponds to an extremely small probability of generating any particular observation. For complex problems this behavior is the rule not the exception since element probabilities in general decline exponentially with dimension. Logarithmic representation is one solution but the authors have also explored a floating point representation with extended exponent range as reported in [23]. Because γ variables are normalized by $F(x)$ they may be adequately represented using standard floating point.

We now briefly discuss the computational complexity of FGM maximization. A restricted class of FGMs is identified for which maximization is shown to be NP-complete. Members of this class have constant choice functions that assume value 1. There are two selection functions s_0 and s_1 that assume constant values 0 and 1 respectively. Notice that there is no dependency on any observation. Each observation model has a single parameter bit b and given boolean argument x assumes value $(b \wedge \bar{x}) \vee (\bar{b} \wedge x)$. The FGM then assumes a bounded integral value and the maximization problem is well-posed. The corresponding decision problem is: does there exist a set of parameter bits such that the FGM's value is greater than a given integer i? This problem is clearly in NP since given a set of parameter bits, FGM evaluation, which runs in linear time, may be used to answer the question.

We will refer to this setting as the LFGM (logical FGM) problem.

Theorem 2 *LFGM is NP-complete.*

proof: The NP-complete SAT problem is easily reduced to LFGM by first associating an observation model with each variable. If the variable $v, \bar{v}$ occurs in a clause then selector s_1 or s_0 is used respectively. Clauses gives rise to trivial FGMs in which each term corresponds to a distinct edge from source to sink with the appropriate observation model and selection function attached. The FGMs for each clause are then concatenated to form the final FGM. The set of clauses are then satisfied if and only if the FGM assumes a value greater than zero. The reduction's complexity is linear whence LFGM is NP-complete. $\Box$

It is possible to construct reductions like that above where the observation models are continuous not discrete functions; but further discussion of the complexity of continuous optimization is beyond the scope of this paper. These continuous formulations correspond to stochastic *continuous embedding* approaches to SAT and represent an interesting area for future work.

This concludes our development of FGMs as general mixture-forms and in closing we remark that our framework is certainly open to additional generalization and refinement. For example one might introduce choice models that also depend on the observation, or replace the m, c, s edge association functions with distributions. Many such variations are possible but our goal was a development somewhere between empty generalization and over specialization.

3 FGM-based Probability Models

In this section we consider specializations of Finite Growth Models such that their value represents a probability. The associated DAG and attached models may then be viewed as an acyclic automaton directing the stochastic generation of objects. Stochastic modeling is the original motivation for FGMs and much of our nomenclature is drawn from this viewpoint.

The first step is to specialize the selection functions of definition 1 to projections. That is, functions that select some subset of the dimensions of tuple observation x. A projection is characterized by a subset g of the dimension indices of $\mathcal{X}$. We associate some subset g_e with every edge e of the FGM and then denote by $\pi_{g_e}(x)$, the *projection* of observation tuple x corresponding to selection of the components in g_e. This then is the selection function associated with e. For some edge e suppose $g_e = \{2, 4\}$. Then $\pi_{g_e}(x)$ is a tuple with two components which are equal in value to the second and fourth components of x. We assume for now that $g_e \neq \emptyset$ but will later see that this is unnecessary. A composition of projections, applied to x, is again a projection characterized by some dimensional subset; whence it is straightforward to define equivalence for the purpose of identifying truly distinct model applications for recursively combined FGMs. The use of projections corresponds to the generation of observations in arbitrary groupings and orderings – a fact we will rely on later in our discussion of stochastic transduction.

Next we assume that all choice and observation models are probability functions or densities. Each C_i then corresponds to a stochastic *choice* among edges to follow. Each M_i corresponds to stochastic *generation* of some portion of x (since our selectors are projections).

Finally we specialize the g_e subsets so that along any source-sink path their intersection is void and their union is the complete set of dimensions. Each path then represents one way in which

x might be stochastically generated or *grown* and corresponds to a permutation of the dimensional indices $1, \ldots, d$. The "finite growth model" framework is so named because we imagine that complete observations are grown, starting from the empty set, in many possible orderings, as specified by the model's underlying graph. The value of each source-sink path through the FGM is the probability of following that path *and* generating x. The FGM's value is then a probability function $P(x|\Phi)$ on $\mathcal{X}$, i.e. its sum/integral is one. When it is necessary to make clear that we are referring to an FGM with the restrictions above, we will use the term *stochastic finite growth model* (SFGM).

After algorithm 1 has run, α_n gives $P(x|\Phi)$. However the other α values have meaning as well. In general α_i is the probability of arriving at v_i *and* observing that part of x corresponding to the paths between v_i and the source. Note that because each source-sink path corresponds to a permutation of the observation tuple, every path from the source to v_i must generate the same observation components — possibly however in different orders. Dually, α_i is the probability that random operation of the machine encounters v_i, while at the same time generating those observation components accounted for by the source to v_i paths. Algorithm 1 sums over all paths through the FGM but is easily modified to find instead a single optimal path. This is referred to as the *Viterbii decode* of the FGM and answers the question: what is the single most likely explanation for the observation?

Algorithm 2 also computes $P(x|\Phi)$ as β_1. In general β_i is the probability of observing that part of x corresponding to the paths between v_i and the sink, given passage through v_i. Dually, β_i is the probability that the machine passes through v_i and proceeds on to generate those observation components accounted for by the v_i to sink paths. Denoting by L the part of x corresponding to paths from the source to v_i, and by R the part of x corresponding to paths from v_i to the sink, $\beta_i = P(R|v_i)$ and $\alpha_i = P(L, v_i)$. But $P(R|v_i) = P(R|L, v_i)$ because an SFGM is a Markov process. So $\alpha_i \cdot \beta_i = P(L, v_i, R)$, i.e. the probability of generating x and passing through v_i.

Observe that the meaning of β_i is not simply a time-reversed interpretation of α_i. This is because an SFGM has an intrinsic directionality. In-bound DAG edge probabilities need not sum to one so simply reversing edge direction does not leave a well-formed SFGM.

For SFGMs the γ_e values computed by algorithm 3 are the probabilities $P(e|x) = P(e, x)/P(x)$; where 'e' refers to the event of a transition over edge e. That is, γ_e gives the *a posteriori* probability of following edge e conditioned on observation (generation) of x. Given an SFGM and any x, the γ_e values must satisfy the following constraint which can in practice be used as an implementation self-check. It is a specialization of proposition 1 to SFGMs:

Proposition 3 *Given a stochastic finite growth model, an observation with nonzero probability, and γ values computed by algorithms 1, 2, and 3, then:*

1. *If $|g_e| = 1, \forall e$, then $\sum_e \gamma_e = d$.*
2. *In general, $\sum_e \gamma_e \cdot |g_e| = d$.*

proof: The key idea is that the edges that observe a given observation dimension induce a cut in the DAG but we present the proof from a probabilistic viewpoint. Assume $|g_e| = 1, \forall e$. Because every source sink path path generates each observation component exactly once, the set of edges E_j which generate a particular component j, corresponds to a set of mutually exclusive and exhaustive events. Hence $\sum_{e \in E_j} P(e|x) = \sum_{e \in E_j} \gamma_e = 1$. There are d observation components, and these partition E into disjoint sets $\{E_i\}$. So $\sum_{e \in E} \gamma_e = d$, establishing the first part of the proposition. In general, the

$\{E_i\}$ are not disjoint, and the number of sets to which an edge e belongs is $|g_e|$. Multiplying each γ_e by $|g_e|$ in effect gives each set its own copy, so that the sum remains d. $\square$

We required above that $g_e \neq \emptyset$. If necessary this restriction may be effectively removed and one may include so called *nonemitting* edges in a model. To do so the observation tuple is augmented with *dummy* components each capable of assuming only a single value. Nonemitting edges then observe these components assigning them probability one. The result is a probability model on the augmented observation space, and since its extension was trivial, on the original space.

The m and c functions of our FGM framework may be used to implement the *parameter tying* concept from the stochastic modeling literature. When $m(e_1) = m(e_2)$ the observation models attached to these edges are said to be tied. If the edges leaving vertex v_1 map under c to the same choice model as those leaving another vertex v_2, the choice models at v_1 and v_2 are said to be tied. In crafting a model there are two reasons for parameter tying. The first is that it reduces the number of free parameters in the model and as such combats overtraining. The second is that problem domain knowledge may suggest that two models represent the same underlying phenomenon.

Equation 5 formalizes the problem of parameter optimization given a training set $x_1, \ldots, x_m$ by forming a cascade of identical FGMs. More generally we may write:

$$\overline{\Phi} = \text{argmax}_{\Phi} \prod_{i=1}^{m} F_i(x_i|\Phi) \tag{8}$$

where now a possibly different FGM is associated with each training observation but all are tied to a single underlying collection of choice and observation models. This is an important conceptual point since it represents the technique used to deal with training examples of varying dimension. For example, each x_i might represent a finite time series. Typically the FGMs are instances of a single design – varied in order to accommodate the structure of the observation tuple. Equivalently the entire cascade may be regarded as a single FGM operating on a single observation formed by concatenation of the entire training set. This cascading and concatenation is of conceptual value only. In practice one merely performs expectation steps for each pair (x_i, F_i) followed by a maximization step. There is no need to actually form the cascade and concatenation. In an SFGM setting Eq. 8 represents the training set's *likelihood* and our goal of maximizing it corresponds to *maximum-likelihood* parameter estimation of statistics.

Ultimately one is faced with the task of maximizing primitive models. For simple discrete models and many continuous ones (notably normal densities) each independent maximization problem arising from theorem 1 has a unique globally optimal solution and satisfies certain other conditions. Lemma 1 and theorem 1 may then be strengthened to say that strict progress is made unless one is already at a critical point in parameter space. Even this does not imply parametric convergence however. See [3, 11, 22] for discussion of the convergence and other properties of the EM algorithm. These mathematical issues including rate of convergence, while important, are not our focus and we will not consider them further. Clearly relating various conditions on primitive models to the resulting behavior of the overall FGM represents an important area for future work.

We saw in the last section that simple discrete models are easily maximized. If M_i is a multivariate normal density then maximization is also easily accomplished since the weighted sample mean and weighted sample covariance define the maximum-likelihood parameter set. The weights are γ_e/Γ where Γ denotes the sum of all γ_e associated with M_i. In general, any density for which a maximum-likelihood estimate is readily available may be used as an observation model, e.g. beta densities for random variables confined to $[0, 1]$. See [14] for a treatment of HMMs including a compact discussion of the discrete and continuous optimization problems discussed above.

4 Beyond Finite Probability Models

SFGMs as defined in the previous section are probability functions on an associated finite dimensional observation space. They arise by restricting the FGM framework in several ways. We now relax certain of these restrictions and show that FGMs may be applied to a much broader class of stochastic modeling problems and settings.

4.1 Truncated Stochastic Processes

In contrast to an SFGM, *stochastic models* are frequently defined as either infinite generative processes, or processes that generate finite objects of unbounded size. Time series methods such as hidden Markov models are an example of the first kind and stochastic context free grammars are an example of the second. In the first case the probability of a finite object refers to the aggregated probability of all infinite objects that include it – in the case of strings the combined probability of all infinite strings with a particular finite prefix.

Processes such as these have an associated infinitely deep finite branching structure which is conceptually *truncated* to a finite DAG that explains observations of some fixed length. This DAG is regarded as part of an FGM whose primitive parameterized models correspond to those of the original process. Given a set of finite observations, a set of corresponding FGMs is constructed as discussed in earlier sections. In this way the parameters of an infinite process that best explain a set of finite observations may be learned using FGMs.

This truncation is conveniently effected within the SFGM framework through the introduction of *null observation models*. These are simply functionals that assume the value zero everywhere and as such lie trivially within the FGM framework. When generation via the original infinite branching process would result in an observation component beyond the finite set of interest, a void model is attached to that edge and it is redirected to the FGM's sink. In this way the processes infinite graph becomes a finite DAG with a single source and sink. This is illustrated by our discussion of stochastic transducers and hidden Markov models in later sections.

A more subtle issue is addressed by the introduction of *void choice models* which assume value 1 for each outgoing edge. They are useful when the events associated with future generation may be partitioned into mutually exclusive sets. Our discussion of stochastic context free grammars in a later section provides illustration.

4.2 Alternative Maximization Criteria

Maximum likelihood (ML) parameter estimation is a *best-fit* strategy and is therefore susceptible to overtraining. For example, if a discrete model never observes a given symbol, its ML parameters assign the symbol probability zero. The result can be poor generalization. Also, our discussion thus far has tacitly assumed that a single stochastic model represented as an FGM is used to explain a set of observations. Here too if this model is very complex in comparison to the available training data, generalization will suffer. The *model selection problem* consists of somehow choosing or forming a model of appropriate complexity.

These two issues are highly related and we will now briefly describe how both may addressed within the FGM framework. The result is a conceptually clear and computationally tractable approach to implementing more sophisticated estimation schemes for both existing and new stochastic models. That EM could accomplish this in general was observed in [11] but does not appear to have impacted the use of EM in practice. We consider two approaches: maximum *a posteriori* parameter (MAP) estimation, and minimum description length (MDL). Our discussion begins with MAP.

Earlier sections have shown that Baum-Welch/EM may be used to address the problem of finding Φ that maximizes $F(x|\Phi)$. The form of this objective corresponds to maximum-likelihood estimation but we will see that it actually not so limited. Given a prior $P(\Phi)$ on model parameters the MAP objective is to instead maximize $P(\Phi|x)$. From the Bayes' rule it follows that this is the same as maximizing $P(x|\Phi) \cdot P(\Phi)$ since this quantity is proportional to $P(\Phi|x)$, i.e. the denominator is fixed. We resolve Φ into its constituent parts $\{\Psi_i\}$ and $\{\Upsilon_i\}$ corresponding to the FGM's observation and choice models respectively, and define the overall prior based on independent parts:

$$P(\Phi) \triangleq \left(\prod_{i=1}^{|M|} P(\Psi_i) \right) \left(\prod_{i=1}^{|M|} P(\Upsilon_i) \right)$$

This product may itself be regarded as a linear FGM denoted F_P with null observation models, and 1-way choice models having values corresponding to each product term. Alternatively each term may be represented as an observation model that assumes a constant value for each setting of its parameters. The two FGMs F and F_P are then combined end-to-end, or in the case of multiple observations F_P is attached to the end of $F_1, \ldots, F_M$. The FGM F computes $P(x|\Phi)$ and F_P computes $P(\Phi)$ so that the cascade thus formed has value $P(x|\Phi) \cdot P(\Phi)$. Maximizing it will then effect MAP-guided learning.

The minimum description length principle states that one should choose a model and its parameters such that the combined cost of *describing* these followed by the encoded observations is minimized. The related notion of Kolmogorov complexity deals with the smallest such description and is generally not computationally tractable. By contrast MDL practitioners are content to fix some parameterized scheme of representation and optimize the parameters. This does not fully implement the MDL principle but captures the important tension that must exist between model and data complexity. The cost of describing the observations is $-\log_2 F(x|\Phi)$ bits where it is assumed that F is a probability function. The cost of describing the model's structure is upper bounded by choosing some reasonable representation and counting its bits. The cost of describing a real valued parameter may be upper bounded by choosing some number of quantization levels Q, assuming uniform quantization (typically), and describing $-\log 2Q$ bits per parameter. Many variations are possible. The final codelength is denoted $L(\Phi)$ and the total description length is $L(\Phi) + -\log_2 F(x|\Phi)$. As in the case of MAP above we assume that $L(\Phi)$ is formed by summing separate contributions from primitive models. As before these may be regarded as a linear FGM where the value of the attached models is just the corresponding codelength exponentiated. Notice that the the MDL and MAP constructions are essentially identical and for this reason we view MDL as a Bayesian approach in which the prior is not necessarily a probability model.

The important message is that the original maximization problem may be decomposed into separate primitive problems even when the guiding criterion is not maximum-likelihood. Any primitive model for which maximization may be conveniently performed subject to the selected criterion, may be used.

Since the basic FGM is fixed in the discussion above, the cost of describing its representation in the MDL framework is also fixed with respect to FGM maximization. However an important part of the MDL outlook is selecting a model of appropriate complexity. This tension is captured in the FGM framework by forming a finite mixture of models over a range of complexities. Each model is made up of a basic FGM augmented with an MDL extension as described above. The cost of describing the model's design now enters the optimization problem since it differs across mixture components. Rather than selecting a single model complexity, reestimation will learn a distribution on complexity. Each mixture component is the *a posteriori* probability of its corresponding complexity given the observations. Similarly the MAP setting above may be extended to include the learning of a distribution on model complexity. The model with maximum *a posteriori* probability may be selected if it is important to identify a single *best* model. Finally, in the case of MDL the pedantic implementor will account for the complexity of the initial mixture in the overall optimization.

The result is not MDL at all in its original sense since a single model and parameter set is not selected. It is more properly regarded as MAP with MDL-inspired priors.

Our discussion above started by placing all penalty terms together at the front of an FGM. In the model selection example, however, some were placed instead following an initial choice. This may be seen as a special case of a general strategy in which penalty terms are placed as deep as possible within the DAG. That is, just before their first use.
Our discussion has then established:

Theorem 3 *Let F be an SFGM or arise from the truncation of an infinite stochastic process. Also assume that the primitive models employed support MDL (or MAP) maximization. Then a new FGM F' may easily be constructed with at most twice the number of edges such that its maximization corresponds to MDL (or MAP) guided maximization of F.*

The practical significance of this theorem is Baum-Welch/EM may be used to efficiently implement estimation criteria more sophisticated than ML with which it is historically associated – and via FGMs it is clear how this can be done for a wide variety of models.

4.3 Alternative Discrete Models

The trivial discrete probability model on an alphabet of k members has $k-1$ nonnegative free parameters that sum to at most unity. Many other possibilities exist. For example, if alphabet symbols correspond to states in a truncated Poisson process then the Poisson probability function $P(x|\lambda)$ may be used. This is true of a choice model as well and has application in the modeling of event durations. In general we remark that the use of nontrivially parameterized choice functions represents an interesting direction that has received only limited attention.

4.4 Observation Context Conditioned Probability Models

In our SFGM framework each edge generates some subset of the observation tuple's components such that as one moves from source to sink each is generated once. The values generated depend only on the edge over which generation is transiting. This corresponds to the first-order Markov assumption of stochastic modeling. In particular the value cannot depend upon route followed to reach the edge. But dependence on observation components generated earlier along the path does not violate the Markov assumption. Formally the observation models are then conditional probability functions where the conditioning is on earlier observation components – *not* earlier edge sequences. Each source-sink path then corresponds to a causal chain of conditional probabilities so that the FGM's value is a probability.

Brown observes in his thesis [7] that the HMM output independence assumption may be relaxed to allow generation of the current speech sample window to depend on the previous one. He then uses Baum-Welch reestimation without reproof. His focus is on minimizing the number of model parameters – and we suspect that it is only for this reason that he did not propose longer context dependencies.

Our contribution is then the formalization of this message in general form within the FGM framework. That is: the problem of optimizing FGMs that employ conditional observation models is reduced to corresponding primitive conditional maximization problems.

4.5 Noncausal Unnormalized Models

Given a $k \times k$ pixel real-valued image one may model each pixel position as a function of some surrounding context. If each model's value is a probability and the context-induced dependency graph has no cycles, then the individual pixel probabilities may be multiplied to result in a *causal* probability model. Causal models are commonly built by fixing a scanning sequence and choosing contexts with respect to the implied temporal ordering. Since there is frequently no single natural scanning sequence the causality restriction may limit model effectiveness by preventing all of a pixel's neighbors from contributing to its prediction.

If the causality restriction is discarded each pixel's model is strengthened but their product is no longer a probability. This corresponds to relaxing the requirement that the projection functions in an SFGM correspond to a permutation of the observation dimensions. Noncausal neighborhood systems have proven useful in image processing [8].

Since reestimation must nevertheless climb we have the result that even a noncausal unnormalized models like that sketched above may be improved within the FGM framework.

4.6 Dynamic Choice Function

As remarked earlier we might have allowed choice functions to depend on the observation x. Given choices $c_1, \ldots, c_k$ the FGM may then follow each with probability $p(c_i|x)$. In contrast with the conventional static notion of state transition probability, such choice models are dynamic – responding to the observation. This provides interesting new modeling flexibility. If the portion of x used to influence the choice is generated earlier in the DAG (closer to the sink), then one can build probability models using dynamic choices. We remark, however, that useful models may exist that violate this assumption and therefore generate values that without normalization do not represent probabilities.

5 Stochastic Transducers

The notion of *transduction* has its roots in classical formal language theory (e.g. [6]) and represents the formalization of the idea of a machine that converts an input sequence into an output sequence. Later work within the syntactic pattern recognition outlook addressed the induction problem for a particular subclass of finite transducer [19]. More recently a stochastic approach was taken in [4] where hidden Markov models were used to capture an input-output relationship between synchronous time series.

Finite growth models can be used to derive Baum-Welch/EM based learning algorithms for *stochastic transducers*. This interesting class of stochastic models capture in a somewhat general way the relationship between dependent symbol streams. In particular there is no assumption that the streams are synchronous, and the symbols singly or jointly generated during transduction can depend upon all context information, i.e. that in every series involved.

Many problems of pattern recognition and artificial intelligence such as speech or handwriting recognition may be viewed as transductions from an input signal to an output stream of discrete symbols – perhaps via one or more intermediate languages. In this conceptual framework it is interesting to note that speech synthesis is merely transduction in the reverse direction. In this section we first discuss transduction in general terms, and then focus in detail on the simple *memoryless* transducer that corresponds to the widely-used notion of string edit distance.

Definition 3 *A k-way stochastic transducer over alphabets $\Sigma_1, \ldots, \Sigma_k$ is a stochastic automaton $A = (S, M)$ where S denotes its states and M its matrix of transition probabilities – such that associated with each transition e is a probability function p_e on $\Sigma_1^* \times \ldots \times \Sigma_k^*$ such that $p_e(\epsilon, \ldots, \epsilon) = 0$ where ϵ denotes the empty string. In the generative view of transduction the machine outputs nonempty k-tuples which are appended to k initially empty output strings. If any of the p_e values depend not just on e but also on the output generated thus-far, then the transducer is said to have memory. Otherwise it is memoryless. Automaton A is assumed to start from state zero, and the possibility that the alphabets are continuous is left open.*

Given observed strings $S = (s_1, \ldots, s_k)$ the transducer T assigns a probability $P_T(S)$ which represents the sum over generation sequences that generate S, of the probability of each. The FGM formalism provides a natural way to understand and reason about this complicated process and we now describe the reduction of a stochastic transducer and associated finite observation to an FGM.

Denoting the states of A by $\{S_i\}$ the transducer's infinitely deep temporal branching structure is truncated to form an FGM with states $\{S_i^{\ell_1,\ldots,\ell_k}\}$ where each ℓ_j is a nonnegative integer and $S_0^{0,\ldots,0}$ is the source. The superscript indices correspond to the number of symbols that have been written to the transducer's k output tapes.

The FGM contains another set of states $\{T_{i,j}^{\ell_1,\ldots,\ell_k}\}$ defined similarly except that the subscript refers to a pair of states in the original automaton. Edges exist between these two state sets but not between members of each. There are edges leaving each state $S_i^{\ell_1,\ldots,\ell_k}$ leading to $T_{i,j}^{\ell_1,\ldots,\ell_k}, j = 1, \ldots, |S|$. Row i of stochastic transition matrix M provides the corresponding choice model where the matrix entries are the parameters. The observation model attached to each such edge assumes the constant value 1.

Directed edges exist between T states and S states reflecting the writing of k-tuples to the output tape. For each transition e from state S_i to S_j in the original automaton, several FGM edges may be required because the function p_e may generate output of differing lengths. More formally for all $i_1, \ldots, i_k \in \mathbb{N}^k$ an edge exists leaving $T_{i,j}^{\ell_1,\ldots,\ell_k}$ leading to $S_j^{\ell_1+i_1,\ldots,\ell_k+i_k}$. Each such edge generates observations according to p_e, but only string tuples with lengths $i_1, \ldots, i_k \in \mathbb{N}^k$ are allowed. Void choice models are used for all edges from T to S.

Finally positions in the string-tuple observation are enumerated and appropriate projectors serve as the FGMs selection functions and associate observation models with the generation particular positions in the output tape. Notice that along a source-sink path the corresponding projections are disjoint and do cover all positions but do not in general do so in a simple left-right fashion. This completes our reduction of T to an FGM.

We now evaluate the complexity of FGM operations by counting edges. In general each function p_e may generate output tuples of unbounded total length. But in some cases such as that of string edit distance discussed shortly, the length of the string generated in each tuple position is bounded by some value D.

If L denotes the length of the longest string in the observation tuple, the FGM has $L^k|S|$ states $\{S_i^{\ell_1,\ldots,\ell_k}\}$ and there are $|S|$ edges leaving each of them leading to T states. The T states number $L^k|S|^2$. The out-degree of a T state is clearly $O(D^k)$ making the total number of edges $O(L^k D^k |S|^2)$ and this gives the time and space complexity of FGM operations assuming constant time for primitive model operations. The out-degree of a T state is at most L^k in the FGM so an alternative expression $O(L^{2k}|S|^2)$ applies even when f has infinite support.

Having reduced T to an FGM we identify several important operations and briefly describe their corresponding FGM computation:

1. A transducer learning step consists of improving the parameters of T (those of M and of each p_e) based on a training set $(s_1^1, \ldots, s_k^1), \ldots, (s_1^m, \ldots, s_k^m)$. This computation is performed using standard FGM Baum-Welch/EM steps.

2. There are several transducer evaluation operations:

   (a) Joint — yields the probability of the observation tuple. This is accomplished by standard FGM evaluation and amounts to computing the joint probability of the strings that comprise it.

   (b) Marginal — gives the probability of some subset of the tuple's dimensions, i.e. marginalizes over the remaining dimension. This is easily computed by computing the appropriate marginal of each primitive observation model. That is, if such a model involves one or more marginalized tuple dimensions, the corresponding primitive marginal becomes the value of the observation model.

   (c) Conditional — gives the conditional probability of some subset of the tuple's dimensions given the others. This is best computed by dividing the joint probability by the corresponding marginal.

3. The transduction decode operation consists of maximizing over free-tuple positions their probability conditioned on the other position which are fixed. In the case of 2-way transduction this means fixing one string and finding the most likely second. Since the denominator is constant in the quotient defining the conditional we have only to maximize the numerator, i.e. the joint probability of the selected tuple positions given the other fixed positions. This maximization is performed via Viterbii decode where the fixed positions are regarded as constant. This identifies the most likely transduction path. The result is built by following this path, and for each attached observation model outputting values corresponding to its mode.

A *cascade* operation may also be defined where the output of one transducer becomes the input to another. The development above has then established

Theorem 4 *By reduction to an FGM the evaluation, learning step, and decode operations for a stochastic transducer may be performed in $O(L^k D^k |S|^2)$ or $O(L^{2k}|S|^2)$ time and space assuming constant time primitive model operations. The constant time assumption holds in particular given the use of canonical discrete probability functions for all models.*

Note that the space complexity of evaluation and decode is actually smaller since only a forward pass is performed and there is no need to maintain a fully formed DAG in memory. Also, our earlier comments regarding alternative maximization criteria apply to transducers as well and other generalizations are possible including multiple start states and parameter tying.

5.1 String Edit Distance

The edit distance between two finite strings s, t over finite alphabet Σ is the least costly way to transform s into t via single-symbol insertions, deletions, and substitutions. The non-negative costs of these primitive edit operations are parameters to the algorithm. These costs are represented as a table $c_{i,j}$ of size $|\Sigma| + 1 \times |\Sigma| + 1$, where row and column 0 correspond to an additional alphabet member ϵ representing the null character. See [26] for review or [12] for a compact discussion. The entry in table position i, j gives the cost of substituting symbol j of s for symbol i of t. The first row gives insertion costs, the first column gives deletion costs, and the remaining entries specify substitution costs. The table need not be symmetric. Recall that a simple dynamic program finds the edit distance in $O(|s| \cdot |t|)$ time.

The task of learning an optimal cost table is clearly under-constrained, because the string edit distance for all strings may be trivially minimized by setting all edit costs to zero. Our outlook is stochastic and we therefore interpret each cost as $-\log_2()$ of the corresponding event probability. For example, an entry in the top row expresses the probability of choosing to insert a particular symbol into the left string. The natural constraint then requires that the probabilities that underly the cost table must sum to one. Expressed in terms of costs $c_{i,j}$, this translates to:

$$\sum_{i,j} 2^{-c_{i,j}} = 1$$

Rather than imagining an editing process that transforms s into t, we equivalently imagine that the pair (s, t) is *grown* in a series of steps of three kinds: joint steps, left steps, and right steps. This outlook was first introduced in section 3.2.2 of [12]. Our contribution is its further development to include the learning of costs and the construction of more sophisticated distance functions as described later in this section. In [24] the authors implement the learning of costs and give experimental results.

The process is formally a 2-way stochastic transducer with a single state, no memory, and $D = 2$. The initial state of the process is $(\emptyset, \emptyset)$. A joint step concatenates symbols to both elements of the pair. A right step adds a symbol to only the right element of the pair, while a left step adds a symbol to the left element. In the language of string edit distance, a joint step corresponds to a substitution operation. When a joint step adds the same symbol to both strings, one is substituting a symbol for itself. A right step corresponds to an insertion operation, and a left step corresponds to a deletion operation.

Figure 1 depicts the table now denoted $\chi_{i,j}$ after conversion to probabilities. The zero entries correspond to infinite costs. The stochastic choice between left, right, and joint generation is implicit in this table. That is, χ combines choice and observation probabilities. For example, the sum of the top row may be interpreted as the probability of choosing to perform a left insertion.

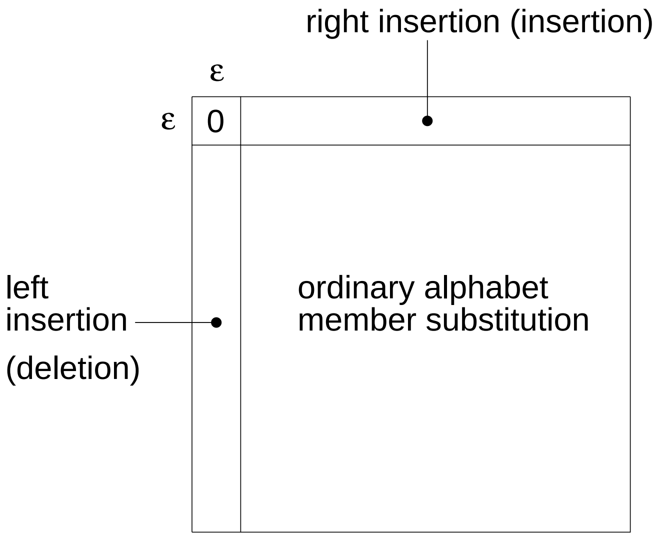

Figure 1: Augmented table of edit distance probabilities. This table sums to one, and the corresponding edit costs are $-\log_2()$ of its values.

Definition 4 *The "stochastic edit distance" between strings s and t, is $-\log_2(P((s,t)|\chi)$. Here the probability refers to the infinite generative stochastic process in which symbols are inserted at the end of the right string, or at the end of the left string, or jointly at the end of both. Specifically, it is the probability that pair (s,t) will occur as an intermediate stage of generation, during a random trial.*

Figure 2 depicts the FGM corresponding to stochastic edit distance between strings of length 4 and 5. Its 3-way branching pattern follows from the structure of the corresponding growth process and matches exactly the dependency structure of the dynamic program for conventional edit distance.

Notice that all but the sink and the bottom and rightmost vertices have ternary out-degree, and that the edges leaving them are drawn with solid lines. These are all in tied. Both their choice and observation models are specified by χ corresponding to normal operation of the generative stochastic process. Edges drawn with dotted lines implement truncation of the process and employ null models. These might have been depicted instead as individual long edges directly to the sink.

The $-\log_2()$ operation converts the probability to a non-negative "distance", which we will later see corresponds closely to the standard notion of edit distance. Also notice that in the definition, we refer to the probability of the ordered pair (s,t). This is an important distinction since χ need not be symmetric, and it might be that $P((s,t)|\chi) \neq P((t,s)|\chi)$.

It is also possible to interpret $P((s,t)|\chi)$ with respect to finite event spaces. If we denote by $S_{|s|,|t|}$ the FGM state corresponding to the argument string lengths, then the probability defined above may be written: $P((s,t), S_{|s|,|t|}|\chi)$. That is, the probability of generating s and t while passing through FGM vertex with coordinates $(|s|,|t|)$. It is straightforward to compute the probability of event $S_{|s|,|t|}|\chi)$. Division then yields the conditional probability $P((s,t)|\chi, S_{|s|,|t|})$ which amounts to conditioning on string lengths. It is also possible to define the FGM differently in order to directly implement conditioning on string lengths [24].

The stochastic edit distance $-\log_2 P((s,t)|\chi)$ is highly related but not identical to conventional edit distance. If stochastic table χ is converted by $-\log_2()$ to a table of non-negative costs then conventional edit distance corresponds to the log-probability of the *single best path* through the FGM – i.e. the Viterbi decode. Stochastic edit distance is the sum over *all paths*. Basing decisions on the Viterbi decode is analogous to focusing on the mode of a distribution while stochastic edit distance corresponds to the mean. In classification systems we expect that both forms will lead to nearly identical decisions. The improved costs we learn in the next section will provably reduce stochastic edit distances, but we also expect that in practice they will also improve results for the conventional form.

We formalize the application of FGMs to stochastic edit distance with the following:

Corollary 1 *Let χ be a table of stochastic edit distance probabilities, and $(s_1, t_1), (s_2, t_2), \ldots, (s_T, t_T)$ denote a set of string pairs over finite alphabet Σ, where ℓ bounds the length of any single string. Next let $E_\chi(s,t)$ denote the stochastic edit distance $-\log_2 P((s,t)|\chi)$. Then:*

1. *There exists an $O(|s| \cdot |t|)$ time, $O(\min(|s|, |t|))$ space algorithm to compute $E_\chi(s,t)$ – matching the complexity of the conventional dynamic program.*
2. *There exists an $O(\ell^2 T)$ time and space algorithm for producing a revised cost table χ' using Baum-Welch/EM such that:*

$$\sum_{i=1}^{T} E_{\chi'}(s_i, t_i) \leq \sum_{i=1}^{T} E_\chi(s_i, t_i)$$

where the inequality is strict unless χ is already a critical point.

proof: Apply theorem 4. □

The FGM formalism is useful to arrive at a correct solution but because of the extensive parameter tying employed in our construction, the associated computations may be greatly simplified. There is no need to build actual DAGs during the computation. Also, as remarked earlier, the choice model is implicit in χ so that creation of separate choice and observation models is unnecessary when computing $E_\chi(s,t)$. The probability of passing over an edge is just the corresponding entry in χ.

The result is algorithm 7. It requires an $(|s|+1) \times 2$ array of α values. We write $\alpha_{i,j}$ to denote an element, where $1 \leq i \leq |s|+1$ and j is 1 or 2. This array corresponds to the vertices within two adjacent columns of the FGM as depicted in figure 2. The logarithm E_χ of the joint probability of s and t is returned rather than the probability itself, so that the caller need not deal with extended range floating point values. As observed by [12], the algorithm's structure resembles that of the standard dynamic program for edit distance.

Algorithm 7

procedure *Evaluate(s, t)*
  for $j = 1, \ldots, |t| + 1$
    $k = 2 - j \bmod 2$
    $\ell = 1 + j \bmod 2$
    for $i = 1, \ldots, |s| + 1$
      if $i > 1 \vee j > 1$
        $\alpha_{i,k} = 0$
      else
        $\alpha_{1,1} = 1$
      if $i > 1$
        $\alpha_{i,k} \mathrel{+}= \alpha_{(i-1),k} \cdot \chi_{s_{(i-1)},0}$
      if $j > 1$
        $\alpha_{i,k} \mathrel{+}= \alpha_{i,\ell} \cdot \chi_{0,t_{(j-1)}}$
      if $i > 1 \wedge j > 1$
        $\alpha_{i,k} \mathrel{+}= \alpha_{(i-1),\ell} \cdot \chi_{s_{(i-1)},t_{(j-1)}}$
  return $\log_2 \alpha_{(|s|+1),k}$

As in the case of evaluation, there is no need to create separate choice and observation models when implementing reestimation. A brief argument establishes this fact. Assume we do have separate models. Then each edge e is of type *left*, *right*, or *joint*, and during Baum-Welch/EM the quantity γ_e will be added to a choice accumulator and also to an accumulator corresponding to the observed alphabet symbol or symbols. So the total γ contribution to each of the three choice accumulators will exactly match the contribution to its corresponding observation model. Hence, if we instead add the γ values directly into an initially zero array $\overline{\chi}$, the total contributions to the left column, top row, and interior will exactly correspond to the choice accumulator contributions above. So after normalization, $\overline{\chi}$ will represent the same model as that resulting from reestimation based on separate models.

Algorithm 8 is the central routine in the stochastic edit distance learning process. It is called repeatedly with pairs of strings. The current cost table $\chi_{i,j}$ must be initialized before the first call and the improved cost table $\overline{\chi}$ must be set to zeros. As each pair is processed, non-negative values are added to locations within $\overline{\chi}$. After all the pairs have been presented the $\overline{\chi}$ array is simply normalized so that the sum of its entries is one, and then represents the set of improved costs. This entire is repeated to further improve the model. That is, $\overline{\chi}$ is copied to χ and $\overline{\chi}$ is again set to zeros. The sequence of training pairs is then presented again, and so on.

The algorithm uses two work arrays, $\alpha_{i,j}$ and $\beta_{i,j}$, corresponding to the algorithms 1 and 2 respectively. It is assumed that these arrays are large enough to accommodate the training pairs. They need not be initialized by the caller. As in algorithm 7 it is possible to reduce the size either α or β but not both. Doing so results in a constant 2:1 space savings but for clarity we present the algorithm using complete arrays.

Algorithm 8

procedure $LearnCosts(s,\ t)$
  for $i = 1, \ldots, |s|+1$
    for $j = 1, \ldots, |t|+1$
      if $i > 1 \vee j > 1$
        $\alpha_{i,j} = 0$
      else
        $\alpha_{1,1} = 1$
      if $i > 1$
        $\alpha_{i,j} \mathrel{+}= \alpha_{(i-1),j} \cdot \chi_{s_{(i-1)},0}$
      if $j > 1$
        $\alpha_{i,j} \mathrel{+}= \alpha_{i,(j-1)} \cdot \chi_{0,t_{(j-1)}}$
      if $i > 1 \wedge j > 1$
        $\alpha_{i,j} \mathrel{+}= \alpha_{(i-1),(j-1)} \cdot \chi_{s_{(i-1)},t_{(j-1)}}$
  for $i = |s|+1, \ldots, 1$
    for $j = |t|+1, \ldots, 1$
      if $i \leq |s| \vee j \leq |t|$
        $\beta_{i,j} = 0$
      else
        $\beta_{(|s|+1),(|t|+1|)} = 1$
      if $i \leq |s|$
        $\beta_{i,j} \mathrel{+}= \beta_{(i+1),j} \cdot \chi_{s_i,0}$
        $\overline{\chi}_{s_i,0} \mathrel{+}= (\alpha_{i,j} \cdot \chi_{s_i,0} \cdot \beta_{(i+1),j}) / \alpha_{(|s|+1),(|t|+1)}$
      if $j \leq |t|$
        $\beta_{i,j} \mathrel{+}= \beta_{i,(j+1)} \cdot \chi_{0,t_j}$
        $\overline{\chi}_{0,t_j} \mathrel{+}= (\alpha_{i,j} \cdot \chi_{0,t_j} \cdot \beta_{i,(j+1)}) / \alpha_{(|s|+1),(|t|+1)}$
      if $i \leq |s| \wedge j \leq |t|$
        $\beta_{i,j} \mathrel{+}= \beta_{(i+1),(j+1)} \cdot \chi_{s_i,t_j}$
        $\overline{\chi}_{s_i,t_j} \mathrel{+}= (\alpha_{i,j} \cdot \chi_{s_i,t_j} \cdot \beta_{(i+1),(j+1)}) / \alpha_{(|s|+1),(|t|+1)}$

In algorithm 8 we remark that one should escape after the α computation in the event of a zero probability observation. Also, the self-check of proposition 3 has a particularly simple interpretation here: if one accumulates all contributions to $\overline{\chi}$, doubling those corresponding to substitutions, the result is exactly $|s| + |t|$.

We now illustrate by example that one has no guarantee of convergence to a global optimum. Let $s = ABB$ and $t = CC$ provide the sole training pair. If the probability of substitutions A, C and B, C are 0.32, and that of left-inserting B is also 0.32, and left-inserting A has probability 0.04, then learning converges to a local optimum in which these operations have probability 1/3, 1/3, 1/3, and 0 respectively. The algorithm has in effect chosen to edit s into t by substituting A, C, then B, C and finally left-inserting A. At convergence the pair's distance is 3.75 bits. But it is clearly better to left-insert A and perform two B, C substitutions. Initializing the edit operations near to this objective leads learning to the desired solution and the pair's distance is reduced to about 2.75 bits. Initializing the four parameters to 0.25 yields a third local maximum giving distance 3.92 bits.

String edit distance is a very simple example of stochastic transduction. Using it to ground our discussion we now return to more general observations.

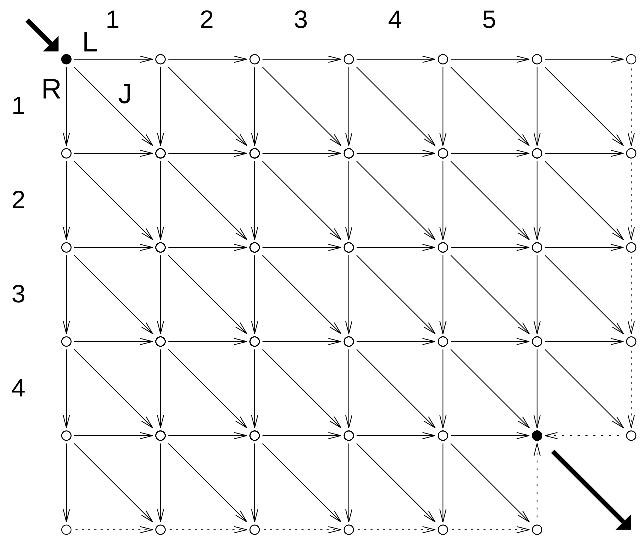

Figure 2: Finite growth model DAG corresponding to computation of the edit distance between strings of length 4 and 5.

The costs employed in the edit distance computation depend only on the targets of the edit operations. They are independent of the context in which the edit operation occurs. Yet in most naturally occurring edit processes the likelihood of an edit operation depends strongly on the context in which the operation occurs. Within the FGM framework one may strengthen the model for edit distance so that the cost of an edit operation depends on the context in which that operation is performed. Insertion and deletion costs can depend on one or more positions of trailing context in a single string while substitution costs can depend on a joint context constructed from both strings. This modeling technique should lead to more powerful similarity functions and is applicable to transducers in general.

Context models may also be built by encoding context into the state space. To do this one limits observation models to trivial probability functions that assign probability 1 to a single alphabet member. After passing over an edge one then has certain knowledge of the most recent symbol. Beyond this the graph can be designed so that arrival in a state implies exact knowledge of a trailing finite context of any desired length. In effect the modeling *work* is moved entirely to the choice models. Having arrived at a vertex the choice model predicts the next symbol conditioned on the past. Note that this technique is an example of something that *does not* generalize to continuous observations.

To specify simple stochastic edit distance, we must learn only a single χ table. With contexts, many more free parameters are added to the model. In principle very large context models can be built, but in practice their size is limited by the amount of training data available increasing the need for MDL or MAP guided optimization.

The learning of a 2-way transducer's parameters represents one formalization of what the authors have called the *metric learning* paradigm. The training set may be regarded as positive examples of the *are similar* concept and after learning, the negated logarithm of the FGM's joint evaluation or conditional evaluation is viewed as a measure of *dissimilarity*. The mathematical metric axioms are not in general satisfied so by "metric learning" we really mean *any learned quantification of* *similarity or dissimilarity.*

This idea applies beyond string settings and for processes more complex than our definition of transduction which specifies that a tuple observation is grown from left to right by concatenation. The string edit distance DAG need not be viewed as arising from 2-way transduction. The salient characteristic that implements metric learning is that some edges observe both strings while others see only one. It is not essential that as one moves from source to sink the string positions considered within each tuple position also move from left to right.

It is then clear that distances analogous to string edit distance may be defined for objects whose natural growth description is more complex than the linear DAG that corresponds to a string. We do not fully develop this direction but now make precise the corresponding DAG structure.

In the case of edit distance each FGM vertex faces three choices. Two choices correspond to single symbol generation, and a third choice selects joint generation. One may view the resulting graph as a product of two linear FGMs, each of which grows a single string from left to right. We now generalize this notion. Given two FGMs F_A, F_B we define the generation DAG of the product FGM denoted $F_A \otimes F_B$ as follows:

1. The vertices of $F_A \otimes F_B$ are pairs (v, w) where v, w are vertices of F_A and F_B respectively.
2. Edge $(v, w) \to (r, s)$ exists in $F_A \otimes F_B$ iff:
    (a) $v \to r \in F_A$ and $w = s$. Edges of this type *run* the left FGM only while keeping the right *idle*. For edit distance this corresponds to insertion into the left string, or;
    (b) $w \to s \in F_B$ and $v = r$ corresponding in edit distance to right string insertion, or;
    (c) $v \to r \in F_A$ and $w \to s \in F_B$. Here both machines run and a pair of symbols are generated corresponding to edit distance substitution.

It is readily verified that $F_A \otimes F_B$ has a single source and sink, and that each path corresponds to a permutation of the joint observation tuple so that after specifying the necessary choice and observation models, a well-formed SFGM results.

6 New Perspectives on Existing Models

In this section we relate FGMs to three models in widespread use: hidden Markov models, stochastic context free grammars, and normal mixtures. The problem of optimizing the log return of a fixed portfolio is also shown to correspond to a trivial FGM. We focus on reduction of these problems in standard form to FGMs but remark that any of them may be improved by using MDL or MAP guided optimization as described earlier.

6.1 Hidden Markov Models

A hidden Markov model (HMM) is an automaton that changes state stochastically after generating an observation. See [21, 14] for reviews. Its state transitions are given by a stochastic $n \times n$ matrix M, a vector π gives the probability of starting in each of its n states, an output function b_i is associated with each state i, and there are one or more designated *final* states. As time passes the machine changes states and generates observations which form an output *time series.*

An HMM is essentially a 1-way stochastic transducer restricted so that all transitions *from* a state specify the same observation model and generate a single observation component ($D = 1$). If there are more than one possible starting states then the entire corresponding FGM is preceded by a choice that plays the role of vector π. The edges leading to the sink are weighted one or zero according to membership in the set of final states.

Because output distributions are associated with states not edges in an HMM, every FGM edge leaving a state is associated with the same distribution. For this reason the transducer reduction can be simplified by eliminating the T states whose only purpose is to remember the previous state. So if the HMM states are denoted $S_1, \ldots, S_n$ then the FGM states are just $\{S_i^t\}$ where the superscript corresponds to time.

This reduction of an HMM to an FGM amounts to the simple *unfolding* in time of the HMM's operation and is therefore done with no loss of efficiency. In some sense it is a simpler representation of the HMM because the temporal order of events has been made explicit.

Reduction in the other direction is problematic. Given a finite alphabet it is not hard to construct an equivalent HMM, i.e. one that imposes the same extrinsic probability distribution. But the canonical such machine may have exponentially more states because an FGM can generate observations in arbitrary order. Moreover, since an FGM can generate observations jointly, the standard HMM formulae for reestimation do not apply. Finally, an FGM that generates joint continuous observations will correspond to an HMM only if these joint densities are trivial, i.e. a product of independent terms. This suggests that FGMs may be more expressive than HMMs.

We now discuss an important variation: *time duration HMMs* [17, 25] and sketch their rederivation in FGM terms illustrating the utility of parameterized choice models. See [14] pages 218–221 for a brief review. If in a conventional HMM a state transitions to itself with probability p, then duration in that state is geometrically distributed, i.e. as $p^{t-1}(1-p)$. This functional form restricts the model's ability to match the characteristics of some applications such as phoneme recognition.

In an ordinary HMM, diagonal transition matrix entries give the self-transition probabilities. These will be modeled separately so M is assumed to have a zero diagonal. The reduction of a time duration HMM to an FGM starts with the same state set $\{S_i^t\}$ but requires an additional set $\{V_i^t\}$. Associated with a state S_i^t is a parameterized choice model that assigns a probability to each possible state duration Δ from $1 - \infty$. Edges exist from S_i^t to states $V_i^{t+\Delta}$ and along each, Δ observations are generated according to the same distribution. To express this strictly within the FGM framework a linear sequence of Δ edges is necessary with a single observation model attached to each. Associated with $V_i^{t+\Delta}$ is a simple discrete choice model induced by transition matrix M. Edges are then directed to the corresponding $S_j^{t+\Delta}$ state with no attached observation model. Notice that since the diagonal of M is zero there is no edge leading to $S_i^{t+\Delta}$. Even though the choice model associated with each S state has infinite domain, the observation is finite so only a bounded number of edges are included in the FGM.

The simplest such infinite choice model has a parameter for every duration value – a nonparametric approach. Since the training set is finite this reduces to a standard discrete choice model which is easily maximized. Because of limited training data one might prefer a model with fewer parameters. A simple way to reduce parameters is to group them into bins and assume that within a bin probability is uniform. For example bin sizes might be $1, 2, 4, 8, \ldots$. Here duration 1 lies in the first bin, durations $2, 3$ in the second, $4, 5, 6, 7$ in the third, and so on. Parametric models may also be used.

We close our discussion of HMMs with the topic of *nonemitting* states, i.e. states that do not generate any output. These complicate our reduction to FGMs and conventional processing as well since cycles among them may have unbounded duration. Common practice is to preprocess models with such states in order to eliminate self-cycles. Denote by N the set of nonemitting states and by $S - N$ the rest. Now suppose a transition occurs from an ordinary state to a member of N. The next transition may leave N immediately or cycle through its states for some time before doing so. But since no observations are generated we are merely interested in the ultimate departure event. For each state in N it is straightforward to compute the probability of ultimately exiting to one of the states in $S - N$. These values become entries in the transition matrix associated with that state. All entries associated with self-transitions among N are set to zero. This modified machine will impose the same extrinsic distribution as the original but is free of nonemitting cycles and may be reduced to an FGM.

6.2 Stochastic Context Free Grammars

A *stochastic context free grammar* (SCFG) is a conventional context free grammar [13] with a probability attached to each production such that for every nonterminal A the probabilities attached to all productions "$A \rightarrow \ldots$" sum to unity. Applied to a sentence of finite length these grammars have an interesting recursive FGM structure. Baum-Welch/EM may then be used to learn the attached probabilities from a corpus of training sentences. The result is essentially the same as the *inside-outside* algorithm described in [21].

We assume the grammar G is in Chomsky normal form and use t to denote the input sentence. Viewing G as a stochastic generator of strings each nonterminal A may expand into sentences of many lengths. We write $P(A \rightarrow t[i \ldots j])$ to denote the probability that A will generate the substring of t consisting of its ith through jth positions. In particular the probability of t is $P(S \rightarrow t[1 \ldots L])$ where L denotes the length of t and S is the start symbol. For a terminal b, $P(b \rightarrow t[i \ldots j]) = 1$ if $i = j$ and $t[i] = b$, and zero otherwise. The SCFG is a probability function over all sentences, so that it is a subprobability model for those of any fixed length. For this reason null models (described earlier) are employed in the construction to truncate the normally infinite generation process.

The FGM corresponding to G restricted to length L is a recursive construction starting with the start symbol. We name the top level FGM $S^{1 \ldots L}$ because it computes the probability of S applied to $t[1 \ldots L]$. The construction is the same for each nonterminal A with corresponding productions $A_1, \ldots, A_{N(A)}$ where $N(A)$ denotes the number of productions of the form "$A \rightarrow \ldots$". The FGM $A^{i \ldots j}$ has $N(A)$ edges from source to sink and computes $P(A \rightarrow t[i \ldots j])$. Its single choice model corresponds to the probabilities associated with A in the grammar. Each edge generates observations $i, \ldots, j$ and the attached models are denoted $A_1^{i \ldots j}, \ldots, A_{N(A)}^{i \ldots j}$ where each $A_\ell^{i \ldots j}$ is defined as follows:

1. If production A_ℓ is of the form $A \rightarrow b$ where b is a terminal, then the model's value is $P(b \rightarrow t[i \ldots j])$ as defined above. Within the FGM formalism this is achieved by using a null model when $i \neq j$, and otherwise a discrete model with $P(b) = 1$.

2. If production A_ℓ is of the form $A \rightarrow BC$ and $i = j$ then a null model is used since each of B and C must ultimately generate at least one terminal. If $j > i$ then the attached model consists of an FGM $[BC]^{i \ldots j}$ which encodes all ways in which BC might generate $t[i \ldots j]$. Each of its $j - i$ source-sink paths is distinct and consists of two edges. Along the kth path numbered from zero, the first edge invokes FGM $B^{i \ldots i+k}$ and the second $C^{j-k \ldots j}$. The generation events

corresponding to each path are mutually exclusive so a void choice model is employed. That is:

$$P(BC \to t[i\ldots j]) = \sum_{k=0}^{j-i-1} P(B \to t[i\ldots i+k]) \cdot P(C \to t[j-k\ldots j])$$

Each FGM's $B^{i\ldots i+k}$ and $C^{j-k\ldots j}$ generate strictly shorter substrings of t than does $A^{i\cdots j}$ whence the recursion is bounded and ultimately descends to productions of the form $A \to b$ described above. The choice models are tied across occurrences of each FGM $A^{i\cdots j}$ in this recursive hierarchy.

The time and space complexity of the resulting model follows easily from an analysis of its structure. There are $O(L^2)$ FGMs of the form $A^{i\cdots j}$ and $O(L^2)$ of the form $[BC]^{i\cdots j}$. The first kind contains $O(1)$ edges while the second has $O(L)$ edges. The result is $O(L^3)$ time and space complexity. The same counting of FGMs and edges demonstrates that complexity is linear in the grammar's size.

Beyond emulating conventional SCFGs, our FGM construction suggests more complex models. For example, it is possible to learn a probability for each production conditioned on the length of the sentence it generates. Here the single value associated with each production is replaced with a vector of probabilities indexed by sentence length. The vector's length may be set to the maximum length of a sentence in the training corpus. The result is a model which generates only sentences of finite length. More generally each vector position may correspond to a range of lengths, e.g. less than 5, between 6 and 20, 21 and over. If the final range is open-ended then the model generates sentences of unbounded length, as does a SCFG. This new model has more parameters, but given enough training data may outperform an SCFG. To implement it one merely ties the choice models associated with each instance of each FGM $A^{i\cdots j}$ differently. In the construction above they are all tied. Instead we tie the choice model for $A^{i\cdots j}$ to $A^{k\ldots \ell}$ provided $j - i = \ell - k$, i.e. both instances generate output of the same length.

6.3 Normal Mixtures

A k-component normal mixture is a probability density function of the form:

$$f(x|\mu_1,\ldots,\mu_k,\Sigma_1,\ldots,\Sigma_k) = \sum_{i=1}^{k} c_i \cdot N_{\mu_i,\Sigma_i}(x)$$

where $\sum_{i=1}^{k} c_i = 1$, $c_i \geq 0$ for $i = 1,\ldots,k$, and $N(\cdot)$ denotes a multivariate normal density. Such a mixture corresponds to a trivial FGM that contains exactly k DAG edges, each from source to sink, and a normal density along each. The FGM's single choice model corresponds to the mixing parameters $c_1,\ldots,c_k$. Unlike transduction, HMMs, and SCFGs, normal mixtures are an example of an FGM which generates an observation tuple of fixed finite dimension. They may be used as the observation model along any edge in a parent FGM. Our earlier reduction of HMMs to FGMs then results in the mixture-density HMMs widely used in speech recognition. To illustrate the variations possible we observe that tying every component with the same index to a single density yields the *semi-continuous* variation [15].

6.4 Portfolio Optimization

Our final example consists of a problem that is not strictly speaking a stochastic probability model but nevertheless fits easily into the FGM formalism. Given *stocks* $x_1, \ldots, x_d$ and observations of their returns over many fixed time periods, the objective is to decide how much of each should be held in an optimal investment portfolio. A trivial reduction to FGMs results in a simple iterative algorithm. This problem is considered in [9] where the same algorithm, essentially a rediscovery of Baum-Welch/EM, is described. The *multiplicative update* perspective of [9] is essentially the growth transform of [3].

We describe this problem using the notation of [9] and reduce it to an FGM. A stock which appreciates by 25% in say one month, is said to have a return of 1.25. The returns of each stock over time period i forms a vector denoted X^i. We use stochastic vector b to denote our portfolio allocation and the portfolio's return over period i is then $b^t X^i$. This is captured by a simple FGM consisting of a d-way choice corresponding to b with each edge leading directly to the sink. An unnormalized observation model is attached to each edge and assumes the value of the selected stock's return. Given T observations $X^1, \ldots, X^T$, and letting diag(b) denote the diagonal matrix formed from b, one Baum-Welch/EM iteration is given by:

$$\overline{b} = \sum_{j=1}^{T} \frac{\mathrm{diag}(b) X^j}{b^t X^j}$$

after $\overline{b}$ is normalized to have unity sum. This matches the equation given in [9]. Assuming the X^j occur with equal probability then maximizing the training set's likelihood also maximizes our expected log return from the portfolio.

6.5 Other Applications

Pearl's Bayes nets [20], also known as graphical models [16], can be represented in their simplest forms as FGMs in which vertices correspond to variables, edge weights to conditional probabilities, and vertex *a posteriori* probabilities (γ) to belief. More complex networks can still be represented as FGMs although the reduction is more involved. The relationship between these networks and hidden Markov models is elucidated in [27].

Exploring the relationship of FGMs to error correcting codes such as those based on trellises, and the recently developed class of *turbo codes* [5], is a subject for future work. We wonder broadly whether our framework's generality might lead to better codes, and in particular whether DAGs that encode nonlinear time orderings might have value. Such a possibility is suggested by [29, 28]. The relationship between Turbo codes and Bayes nets is examined in [18] and understanding the relationship of FGMs to this specific outlook is another interesting area for future work.

Acknowledgments

We thank Andrew Appel, Vince Poor, Bob Sedgewick, and Bob Tarjan for their comments on this work, and Joe Kupin for helpful discussions regarding stochastic transduction.

References

[1] L. E. Baum, *An inequality and associated maximization technique in statistical estimatation of probabilistic functions of Markov processes*, Inequalities, 3 (1972), pp. 1–8.

[2] L. E. Baum and J. E. Eagon, *An inequality with application to statistical estimation for probabalistic functions of a Markov process and to models for ecology*, Bull. AMS, 73 (1967), pp. 360–363.

[3] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, *A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains*, Ann. Math Stat., 41 (1970), pp. 164–171.

[4] Y. Bengio and P. Frasconi, *Input/output hmms for sequence processing*, IEEE Transactions on Neural Networks, 7 (1996).

[5] C. Berrou, A. Glavieux, and P. Thitimajshima, *Near shannon limit error-correcting coding and decoding: Turbo codes (1)*, in Proc. ICC'93, Geneva, Switzerland, 1993, pp. 1064–1070.

[6] J. Berstel, *Transductions and Context-Free Languages*, Teubner, Stuttgart, 1979.

[7] P. F. Brown, *Acoustic modeling problem in automatic speech recognition*, PhD thesis, Carnegie-Mellon University, Department of Computer Science, 1987.

[8] R. Chellappa and A. Jain, eds., *Markov Random Fields: Theory and Application*, Academic Press, 1993.

[9] T. M. Cover, *An algorithm for maximizing expected log investment return*, IEEE Transactions on Information Theory, (1984).

[10] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, Wiley, 1991.

[11] A. P. Dempster, N. M. Laird, and D. B. Rubin, *Maximum-likelihood from incomplete data via the EM algorithm*, J. Royal Statistical Society Ser. B (methodological), 39 (1977), pp. 1–38.

[12] P. A. V. Hall and G. R. Dowling, *Approximate string matching*, Computing Surveys, 12 (1980), pp. 381–402.

[13] J. E. Hopcroft and J. D. Ullman, *Introduction fo Automata Theory, Languages, and Computation*, Addison-Wesley, 1979.

[14] X. D. Huang, Y. Ariki, and M. A. Jack, *Hidden Markov Models for Speech Recognition*, Edinburgh University Press, 1990.

[15] X. D. Huang and M. A. Jack, *Unified modeling of vector quantization and hidden markov models using semi-continuous hidden markov models*, in Proc. ICASSP, 1989, pp. 639–642.

[16] S. L. Lauritzen, *Graphical Models*, Clarendon Press, Oxford, 1996.

[17] S. E. Levinson, *Continuously variable duration hidden markov models for automatic speech recognition*, Computer Speech and Language, 1 (1986), pp. 29–45.

[18] R. J. McEliece, D. J. C. MacKay, and J. Cheng, *Turbo decoding as an instance of pearl's 'belief propagation' algorithm*, Submitted to IEEE Journal on Selected Areas in Communication, (1996).

[19] J. Oncina, P. García, and E. Vidal, *Learning subsequential transducers for pattern recognition interpretation tasks*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 15 (1993).

[20] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, 1988.

[21] A. B. Poritz, *Hidden Markov models: a guided tour*, in Proc. ICASSP-88, 1988, pp. 7–13.

[22] R. A. Redner and H. F. Walker, *Mixture densities, maximum likelihood, and the EM algorithm*, SIAM Review, 26 (1984), pp. 195–239.

[23] E. S. Ristad and P. N. Yianilos, *Probability value library*, tech. rep., Princeton University, Department of Computer Science, 1994.

[24] ——, *Learning string edit distance*, tech. rep., Princeton University, Department of Computer Science, 1996.

[25] M. J. Russell and A. E. Cook, *Experimental evaluation of duration modelling techniques for automatic speech recognition*, in Proc. ICASSP-85, 1985, pp. 5–8.

[26] D. Sankoff and J. B. Kruskal, *Macromolecules: The Theory and Practice of Sequence Comparison*, Addison-Wesley, 1983, 1983.

[27] P. Smith, D. Heckerman, and M. I. Jordan, *Probabilistic independence networks for hidden probability models*, Tech. Rep. MSR-TR-93-03, Microsoft Research, 1996.

[28] N. Wiberg, *Codes and Decoding on General Graphs*, PhD thesis, Linköping Studies in Science and Technology, Sweden, 1996. No. 440.

[29] N. Wiberg, H. Loeliger, and R. Kotter, *Codes and iterative decoding on general graphs*, European Transactions on Telecommunications, 6 (1995), pp. 513–526.