SONY COMPUTER SCIENCE LABORATORY

Sampling Markov Models under Constraints: Complexity Results for Binary Equalities and Grammar Membership

Stéphane Rivaud
email: stephane.rivaud@sony.com
François Pachet
email: pachetcsl@gmail.com

November 29, 2017

We aim at enforcing hard constraints to impose a global structure on sequences generated from Markov models. In this report, we study the complexity of sampling Markov sequences under two classes of constraints: Binary Equalities and Grammar Membership Constraints. First, we give a sketch of proof of #P-completeness for binary equalities and identify three sub-cases where sampling is polynomial. We then give a proof of #P-completeness for grammar membership, and identify two cases where sampling is tractable. The first polynomial sub-case where sampling is tractable is when the grammar is proven to be unambiguous. Our main contribution is to identify a new, broader class of grammars for which sampling is tractable. We provide algorithm along with time and space complexity for all the polynomial cases we have identified.

1 INTRODUCTION

Markov models are widely used in probabilistic sequential data modeling. Their associated graphical model is a linear chain which is a simple topology, allowing inference tasks like

computing marginals to be achieved in polynomial time. They are easy to implement and can be trained to fit perfectly a dataset in one pass. This simplicity makes them suitable for many tasks such as automatic content generation.

However, the Markov assumption underlying these models is quite restrictive. For a d-th order Markov chain $s = s_1, \cdots, s_n$, the assumption states that the probability of a value at position i in a sequence only depends on the d preceding values:

$$\mathbb{P}(s_i|s_1, s_2, \cdots, s_{i-1}) = \mathbb{P}(s_i|s_{i-d}, \cdots, s_{i-1}) \tag{1.1}$$

A classical way to increase the expressivity of Markov models is to increase the order, but this comes at an exponential cost in time and space complexity along with over-fitting issues when the training set is small. As a result, sequences generated from Markov models usually lack global structure and are not able to capture long range dependencies.

In order to address this issue, we investigate the use of hard constraints to enforce such global properties at the sampling phase on the generated sequence. Our goal is to express these properties as a Constraint Satisfaction Problem (CSP), and to sample solutions of the CSP with the probability given by the Markov model. Several results have been obtained toward this aim [10, 13, 12, 11].

In this paper, we review complexity results previously obtained on the sampling of Markov sequences constrained with binary equalities [16]. We also extend previous work on the Regular constraint [13, 12, 17] by investigating a more expressive constraint, the grammar membership constraint [18], and give complexity results for sampling under such constraints.

2 Sampling Markov Sequences under Binary Equality Constraints

Homogeneous Markov Model: Let $X_1, X_2, \ldots$ be a sequence of random variables taking values in a set $\mathscr{A}$. A homogeneous Markov model $\mathscr{M}$ consists of a transition matrix $T_{\mathscr{M}}$ and a vector P_0 such that:

- $T_{\mathscr{M}}(i, j) = \mathbb{P}(X_{k+1} = i | X_k = j)$
- $P_0(i) = \mathbb{P}(X_0 = i)$

The Markov assumption underlying a Markov model states that for all integers i, j and k such that $0 \leqslant i < k < j$, we have $X_i \perp\!\!\!\perp X_j | X_k$, i.e. given X_k, X_i and X_j are conditionally independent.

Binary Equality Constraint: A binary equality constraint between two variables X_i and X_j taking values in $\mathscr{A}$ is satisfied whenever $X_i = X_j$. These constraints are easily filtered and solutions can be uniformly sampled efficiently. We can extend the binary equalities to any binary relation where the value of X_i determines the value of X_j and vice-versa, e.g. $X_i = \sigma(X_j)$ where σ is a permutation of the set $\mathscr{A}$. These are particularly useful when one wants to obtain specific patterns in the sequences sampled from Markov models. However, even if

the problem is easily solved when the random variables are independent, it becomes #P-complete when imposing a Markov distribution on the sequence. The complete proof as well as some examples of chord sequence generation can be found in [16]. Here we give a sketch of the proof for the problem of sampling Markov sequences subject to binary equality constraints.

In this paper, we will assume that a sampling problem associated with a CSP is equivalent to its (approximate) counting version.

Counting the number of solutions in a binary CSP (and more generally pairwise Markov network) is #P-hard. Our proof in [16] consists in building a reduction from the problem of counting the number of solutions of a binary CSP to the problem of sampling Markov sequences constrained by binary equality constraints. The idea is to "unwrap" the factor graph of the binary CSP into a linear chain constrained with binary equalities, and to normalize the factors into a Markov transition matrix. Thus, any binary CSP can be seen as the contraction of a Markov model with binary equalities.

Let G be a factor graph associated to a binary CSP. The algorithm works in 3 steps:

1. Add dummy edges to make the graph G Eulerian.
2. Find an Eulerian path for G and build a Markov model $\mathscr{M}$ out of this path.
3. Impose equality constraints between variables corresponding to the same vertex.

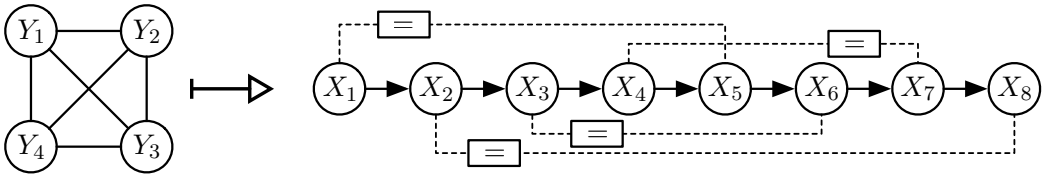

Figure 2.1: On the left is a factor graph representing a binary CSP.
On the right is the equivalent Markov model with a set of equalities.

We can show that if one is able to perform perfect sampling with respect to a Markov model subject to binary equality constraints, he can also count the number of solutions of a binary CSP, which establishes the #P-completeness of the problem.

However, we can identify specific cases where inference is tractable by looking at the topology of the quotient graph. Three cases can be useful in practice:

- **Non-crossing equalities** when the variables inside the equalities do not overlap. The time complexity is in $\mathscr{O}(|\mathscr{A}|^2 n)$ with n being the length of the sequence.
- **Repeated sections** when, given a list of binary equalities among variables, the topological order of the first variable for each equality is the same as for the second variable for each equality. The time complexity is in $\mathscr{O}(|\mathscr{A}|^3 n)$.

- **Palindromic sections** when the set of equalities can be written $\{X_{i_1} = X_{j_1}, \ldots, X_{i_K} = X_{j_K}\}$ with $i_1 < \ldots < i_K < j_K < \ldots < j_1$. The time complexity is in $\mathcal{O}(|\mathcal{A}|^2 n)$.

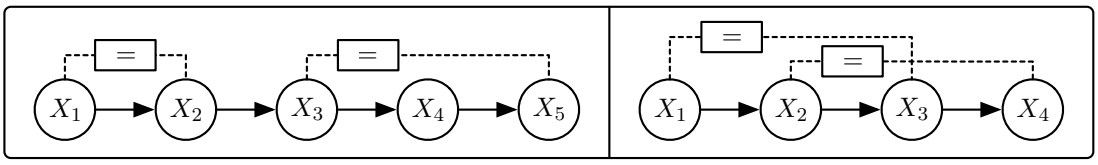

Figure 2.2: On the left, a non-crossing equalities configuration. On the right, a repeated section configuration.

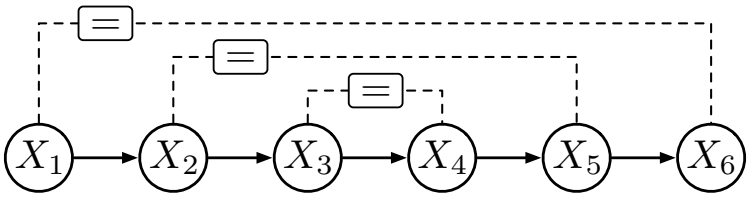

Figure 2.3: Graph representing the palindromic case of binary equalities with Markov models.

3 Sampling Syntactically Correct Markov Sequences

Algebraic languages, or equivalently Context-Free Grammars (CFGs) form the highest level in the Chomsky hierarchy for which the word problem is polynomial, i.e. the problem of deciding if a word w is generated by a given grammar G. The use of algebraic languages in Constraint Programming was studied in [18, 15], with discussions on the efficiency of implementation in [5, 6].

To deal with probabilities, a classical extension of CFGs are Probabilistic CFGs (PCFGs), which define a probability distribution on the set of words of length n. Another extension, Weighted Context-Free Grammars, has been proposed in [7]. In this approach, weights are assigned to each parsing tree without the restriction that it must define a probability distribution, but it has been proven to be as expressive as PCFGs [20]. There have been numerous applications of PCFGs in many domains [2] including Natural Language Processing [21], secondary structure discovery in proteins [4] and automatic music generation [3, 8, 9, 1]. However, training of these models is difficult due to ambiguity issues [19] and approximate iterative methods are often used in practice. This situation makes them unsuitable for on-line learning and incremental sampling where we have to sample one variable at each time-step, which is not the case when dealing with Markov models. On the other hand, *Regular* constraints can easily be expressed in the framework of grammar membership constraints. Having one global

framework to perform constrained sampling would allow a user to specify a wide variety of constraints without changing the implementation. This is typically useful in a work-flow when one would like to implement an additional control tool on a user interface without modifying the underlying engine [14]. We therefore investigate the complexity of imposing a grammar membership constraint on a Markov model.

More formally, a Context-Free Grammar (CFG) is a quadruplet $G = (\mathcal{V}, \mathcal{A}, R, S_0)$ where

- $\mathcal{V}$ is a set of variables called non-terminals.
- $\mathcal{A}$ is a set of variables called terminals, disjoint from V.
- $R \subset \mathcal{V} \times (\mathcal{V} \cup \mathcal{A})^*$ is a set of production rules.
- $S_0 \in \mathcal{V}$ is the initial symbol.

It is a rewriting system used to generate words by starting with the initial symbol S_0, and recursively applying rules from R to the letter of the word until it belongs to $\mathcal{A}^*$. Each word generated with this process inherently possesses a parse-tree representing the sequence of rules used to generate it. In the sequel, we give a proof that the general case is #P-complete, and we investigate two sub-cases where inference is tractable and give polynomial algorithms for these cases.

3.1 General case is #P-complete

Performing inference tasks like computing marginals or the normalization constant with respect to a model $\mathcal{M}$ subject to a constraint $\mathcal{C}$ is harder than counting the number of assignments satisfying the constraint $\mathcal{C}$. We show that counting the number of strings of an ambiguous grammar is #P-complete, even when the given grammar is Regular. An ambiguous Regular grammar can be represented with a non-deterministic finite automaton (NFA). We show that by reducing the #2SAT enumeration problem to the one of counting the strings of a given length n accepted by a NFA.

Let φ be a 2SAT Boolean formula over variables $x_1, \ldots, x_n$ in conjunctive normal form, i.e. $\varphi = \varphi_1 \wedge \ldots \wedge \varphi_K$ with $\varphi_k = l^k_{i_1} \vee l^k_{i_2}$ for all $k \in [1, K]$ and l^k_i is a literal equal to a variable or its negation. We denote by SAT(φ) the set of assignment that satisfy φ and UNSAT(φ) the ones who falsify φ. As there is 2^n possible assignment, we have that #SAT(φ) = 2^n - #UNSAT(φ). We will build an NFA exactly recognizing sequences in UNSAT(φ), thus counting these sequences is equivalent to counting #UNSAT(φ) and thus #SAT(φ).

A falsifying NFA: Let s_0 denote the initial state. For each clause $\varphi_k = l^k_{i_1} \vee l^k_{i_2}$, we add n state $s^k_1, \ldots, s^k_n$. We then add a transition (s_0, s^k_1), and the transitions (s^k_{i-1}, s^k_i) for all $i \in [2, N]$ with the following label:

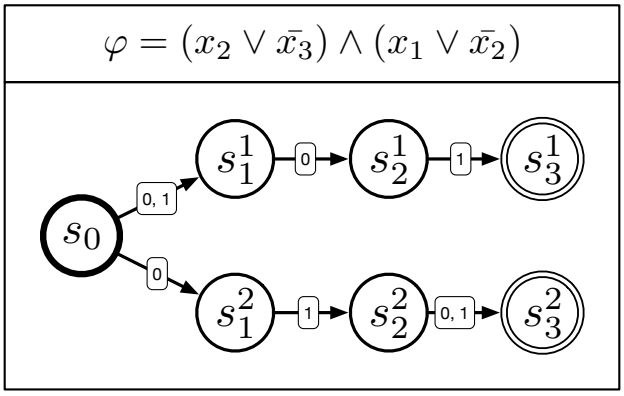

Figure 3.1: On the top is a 2-SAT formula.
On the bottom, its associated falsifying NFA.

- if $i_1 = 1$ then if $l_{i_1}^k$ is a positive literal, then (s_0, s_1^k) is labeled with 0, otherwise it is labeled with 1. if $i_1 \neq 1$ then (s_0, s_1^k) is labeled with 0 and 1.
- if a literal l_i^k is positive then the transition (s_{i-1}^k, s_i^k) is labeled with 0, otherwise it is labeled with 1.
- the remaining transitions are labeled with 0 and 1.

We then set s_n^k for all $k \in [1, K]$ as acceptance states. If a word $x_1 \dots x_n$ reaches a state s_n^k, then it falsifies the clause φ_k and thus φ. Therefore, the number of strings of length n accepted by the constructed automaton is the number of assignments in UNSAT(φ).

COMPLEXITY : The automaton built with our procedure comprises 1 initial state and n additional states per clause. There are K clauses, thus the complexity of the reduction is $\mathcal{O}(Kn)$, which is linear in the size of the formula. Since counting the number of satisfying assignments for a 2SAT formula is #P-complete [22], then counting the number of strings of length n accepted by an NFA is #P-complete, and so is the problem of counting the number of strings of length n generated by an ambiguous grammar.

It is worth noting that counting the number of strings accepted by a deterministic finite automaton can easily be done in polynomial time, with message passing algorithms for example. Therefore, the non-deterministic aspect of the automaton, which translates into ambiguity in the corresponding Regular grammar makes the counting problem #P-complete. In the sequel, we investigate restricted classes of context-free grammars where the counting problem can be answered in polynomial time. We then give an algorithm to perform common inference tasks with respect to a Markov model subject to a grammar membership constraint.

3.2 THE UNAMBIGUOUS CASE

One of the most common inference task is to compute the normalization constant of a distribution. For example, given $n \in \mathbb{N}$, a grammar G and its associated language $\mathscr{L}_G$, it is useful to compute $\mathbb{P}_{\mathscr{M}}(\mathscr{L}_G^n)$ where $\mathscr{L}_G^n$ is the set of words of length n that belong to $\mathscr{L}_G$, i.e. $\mathscr{L}_G^n = \{w \in \mathscr{A}^n \mid w \in \mathscr{L}_G, |w| = n\}$. In order to do so, we need to decompose the probability in a computationally efficient way. We therefore introduce some notations and additional variables.

HIDDEN VARIABLES: Given a non-terminal symbol $A \in \mathscr{V}$ and a word $w \in \mathscr{A}^*$, we note $A \rightarrow^* w$ if we can derive w from the non-terminal A. We note $\mathscr{L}_G(A)$ the set of words that can be derived starting from A, i.e. $\mathscr{L}_G(A) = \{w \in \mathscr{A}^* \mid A \rightarrow^* w\}$. We also denote by $\mathscr{L}_G^n(A)$ the set of word of length n in $\mathscr{L}_G(A)$. We introduce the random variables S_i^j for j from 1 to n, i from 1 to $n-j+1$ and $S \in \mathscr{V}$ taken from the filtering algorithm presented in [18]. These binary variables follow a conditional distribution $\mathbb{P}_{\mathscr{M}}(V_i^j = 1 \mid X_i, \dots, X_{i+j-1})$ which is equal to 1 if $V \rightarrow^* X_i \dots X_{i+j-1}$ and 0 otherwise. We therefore have:

$$\mathbb{P}_{\mathscr{M}}(V_i^j = 1) = \sum_{X_i, \dots, X_{i+j-1}} \mathbb{P}_{\mathscr{M}}(X_i, \dots, X_{i+j-1}) \mathbb{1}_{\mathscr{L}_G^j(V)}(X_i, \dots, X_{i+j-1})$$

It is important to note that for $A, B \in \mathscr{V}$ such that $A \neq B$, then A_i^j and B_i^j are different random variables, i.e. $\sum_{V \in \mathscr{V}} \mathbb{P}_{\mathscr{M}}(V_i^j) \neq 1$. Each of these random variables partitions the space $\mathscr{A}^n$ into two sets, namely $\{V_i^j = 1\}$ and $\{V_i^j = 0\}$. In the sequel, we will write $\mathbb{P}_{\mathscr{M}}(V_i^j)$ in lieu of $\mathbb{P}_{\mathscr{M}}(V_i^j = 1)$ to denote the probability of the set.

DYNAMIC PROGRAMMING: With these notations, we can decompose the probability with the following recurrence formula:

$$\mathbb{P}_{\mathscr{M}}(V_i^j) = \sum_{k=1}^{j-1} \sum_{(V \rightarrow AB) \in \mathscr{V}} \mathbb{P}_{\mathscr{M}}(V_i^j \mid A_i^k, B_{i+k}^{j-k}) \mathbb{P}_{\mathscr{M}}(A_i^k, B_{i+k}^{j-k}) \tag{3.1}$$

where $\mathbb{P}_{\mathscr{M}}(V_i^j \mid A_i^k, B_{i+k}^{j-k}) = 1$ if and only if $V \rightarrow AB$ is a valid production rule. This decomposition basically computes the weighted sum of the trees rooted in V_i^j where each tree is associated to the Markov probability of its corresponding word. As the grammar is unambiguous, there is a one-to-one mapping between the set of parse trees and the set of words in $\mathscr{L}_G$.

MARKOV DEPENDENCIES: The last quantity we need to compute is $\mathbb{P}_{\mathscr{M}}(A_i^k, B_{i+k}^{j-k})$. It represents the probability that $A \rightarrow^* X_i \dots X_{i+k-1}$ and $B \rightarrow^* X_{i+k} \dots X_{i+j-1}$. If the random variables $(X_1, \dots, X_n)$ were independent, we would simply have $\mathbb{P}_{\mathscr{M}}(A_i^k, B_{i+k}^{j-k}) = \mathbb{P}_{\mathscr{M}}(A_i^k) \mathbb{P}_{\mathscr{M}}(B_{i+k}^{j-k})$. In the Markov case, we use the Markov assumption to keep the decomposition tractable. Namely,

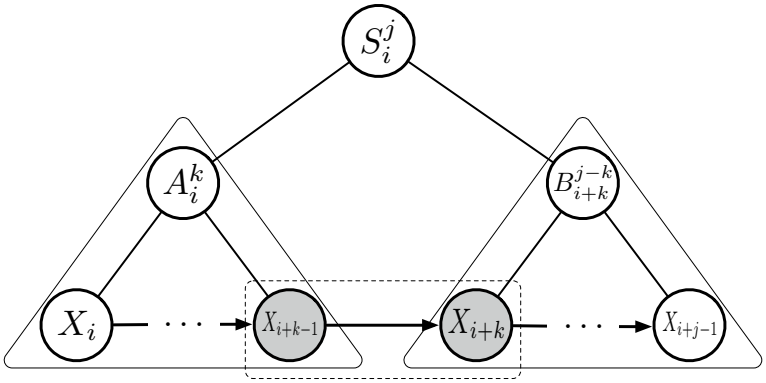

Figure 3.2: Graph representing the dependencies among variables. The arrows represent the Markov dependencies, and the lines represent the dependencies induced by the grammar constraint.

we have that given X_{i+k-1} or X_{i+k}, the variables A_i^k and B_{i+k}^{j-k} are conditionally independent:

$$\mathbb{P}_{\mathscr{M}}(A_i^k, B_{i+k}^{j-k}) = \sum_{X_{i+k}} \mathbb{P}_{\mathscr{M}}(B_{i+k}^{j-k}|X_{i+k}) \sum_{X_{i+k-1}} \mathbb{P}_{\mathscr{M}}(X_{i+k}|X_{i+k-1})\mathbb{P}_{\mathscr{M}}(X_{i+k-1}|A_i^k)\mathbb{P}_{\mathscr{M}}(A_i^k) \quad (3.2)$$

COMPLEXITY: Different trade-offs can be obtained between time and space complexity. According to our decomposition, we need to store $\mathbb{P}_{\mathscr{M}}(V_i^j|X_i)$ and $\mathbb{P}_{\mathscr{M}}(V_i^j|X_{i+j-1})$ which costs $\mathscr{O}(n^2|G|^2)$. We can also store the marginals $\mathbb{P}_{\mathscr{M}}(V_i^j)$ at the cost of $\mathscr{O}(n^2|G|)$ without increasing the asymptotic space complexity. The matrix $\mathbb{P}_{\mathscr{M}}(X_{i+k}|X_{i+k-1})$ is provided by the input as it is part of the Markov model, and the question to store the marginals $\mathbb{P}(X_k)$ is left to the trade-off between time or space efficiency.

The time complexity can be read through the different summations we need to perform in equation (3.1) and (3.2). Equation (3.2) performs summation over X_{i+k-1} and X_{i+k}. We usually perform summation over X_{i+k-1}, store the result for each values of X_{i+k}, and then perform summation over X_{i+k}. This requires $\mathscr{O}(|G|)$ in space but yields a double summation in time $\mathscr{O}(|G|)$. We apply (3.2) for all i, j, k such that $1 \leq i < i+k < j \leq n$ which is $\mathscr{O}(n^2)$. Providing all entries to perform (3.1) thus costs $\mathscr{O}(n^2|G|)$.

Given the necessary data, equation (3.1) performs summation over all integers $k \in [i, j]$ and production rules $(V \to AB) \in G$ thus requiring $\mathscr{O}(n|G|)$ for each variable V_i^j. We apply equation (3.1) for all $1 \leq i < j \leq n$, which represent $\frac{n(n+1)}{2} = \mathscr{O}(n^2)$ variables, resulting in $\mathscr{O}(n^3|G|^3)$ time complexity. This dominates the complexity associated with (3.2) and gives a total time complexity of $\mathscr{O}(n^3|G|^3)$.

3.3 A WEAKLY-AMBIGUOUS CASE

As shown earlier, there is little hope to find a polynomial algorithm in the general ambiguous case, but we can relax the unambiguity hypothesis into a weakly-ambiguous hypothesis.

Definition 1. *A context-free grammar G is weakly ambiguous if the languages generated by its symbols in $\mathcal{V}$ are disjoint, i.e. $\mathcal{L}_G(A) \cap \mathcal{L}_G(B) = \emptyset$ for all $A, B \in \mathcal{V}$ such that $A \neq B$.*

This condition imposes that if a substring $x_i \dots x_{i+j}$ appears in a word of length n, then there is only one symbol that could have generated it. Therefore, if there are two parsing trees associated with a word, then the two trees must "intersect" in some way. We formalize this idea of intersection in this section.

ORDERING THE TREES: We denote by $\mathscr{C}_n$ the set of syntactic trees having n leaves, $\mathscr{C}_n(S)$ the subset of trees in $\mathscr{C}_n$ that has a root labeled with $S \in \mathcal{V}$, $\mathscr{C}_n^k(S)$ the subset of trees in $\mathscr{C}_n(S)$ having their left sub-trees in $\mathscr{C}_k$ and $\mathscr{C}_n^k(S, A, B)$ the subset of trees in $\mathscr{C}_n^k(S)$ having their left sub-tree in $\mathscr{C}_k(A)$ and their right sub-tree in $\mathscr{C}_{n-k}(B)$.

The sets $\hat{\mathscr{C}}_n(S)$ for all $S \in \mathcal{V}$ (resp. $\mathscr{C}_n^k(S)$ for all $S \in \mathcal{V}$ and all $k \in \{1, \dots, n-1\}$, $\mathscr{C}_n^k(S, A, B)$ for all $S, A, B \in \mathcal{V}$ and all $k \in \{1, \dots, n-1\}$) partition the set of syntactic parse trees. As there is a natural mapping from the set of parse trees into $\mathcal{L}_G$, we can associate each set of trees to its corresponding sets of words. However if the grammar is ambiguous they do not induce a partition on the set words generated by the grammar, because one word might be associated with multiple parse trees.

We build an order on syntactic trees with the following recursive procedure. First, we impose that for any $T_1 \in \mathscr{C}_p$ and $T_2 \in \mathscr{C}_q$, we have $T_1 < T_2$ if $p < q$. Then, we choose an order on $\mathcal{V} \cup \mathscr{A}$, which can always be done because the set is assumed to be finite. Finally for $T_1, T_2 \in \mathscr{C}_n$, we have:

1. if $n = 1$, let's note s_1 and s_2 the label of the unique node in T_1 and T_2. We say that $T_1 < T_2$ if $s_1 < s_2$ for the order imposed on $\mathcal{V} \cup \mathscr{A}$.
2. if $n > 1$, we note $T_1 = (s_1, L_1, R_1)$ and $T_2 = (s_2, L_2, R_2)$ where s_i, L_i, R_i are the label of the root in T_i, the left sub-tree, and the right sub-tree. We say that $T_1 < T_2$ if $(s_1, L_1, R_1) < (s_2, L_2, R_2)$ for the lexicographical order.

PARTITIONING THE WORDS: Let us now consider a word $w \in \mathscr{A}^n$ and let us note $T(w)$ the set of parse trees associated with w. As $T(w)$ is finite and ordered the application

$$\hat{T} : w \mapsto \min_{t \in T(w)} t$$

is well defined and induces a partition on $\mathcal{L}_G^n$ for any $n \in \mathbb{N}^*$. We denote by $\widehat{\mathscr{C}_n(S)}$ (resp. $\hat{\mathscr{C}}_n^k(S), \widehat{\mathscr{C}_n^k(S, A, B)}$) the set of words w such that $\hat{T}(w)$ belongs to $\mathscr{C}_n(S)$ (resp. $\mathscr{C}_n^k(S), \mathscr{C}_n^k(S, A, B)$). With these notations, we can write the following partition:

$$\mathcal{L}_G^n = \widehat{\mathscr{C}_n(S)} = \cup_{k=1}^n \cup_{A,B} \widehat{\mathscr{C}_n^k}(S, A, B)$$

Considering $\mathscr{C}_n^k(S,A,B)$ as a set of words, it is composed of all the words that have a syntactic parse tree $T=(S,L,R)$ with L in $\mathscr{C}_k(A)$ and R in $\mathscr{C}_k(A)$. The set $\widehat{\mathscr{C}_n^k}(S,A,B)$ is the set of words such that $\widehat{T}(w)$ belongs to $\mathscr{C}_n^k(S,A,B)$. We can thus write:

$$\widehat{\mathscr{C}_n^k}(S,A,B) = \mathscr{C}_n^k(S,A,B) \setminus \mathscr{C}_n^{<k}(S,A,B)$$

where $\mathscr{C}_n^{<k}(S,A,B))$ is the set of words w in $\mathscr{C}_n^k(S,A,B)$ such that $\hat{T}(w)$ is not in $\mathscr{C}_n^k(S,A,B)$. We can decompose this set in the following way:

$$\mathscr{C}_n^{<k}(S,A,B) = \mathscr{C}_n^k(S,A,B) \cap \left(\cup_{(k',C,D)<(k,A,B)} \widehat{\mathscr{C}_n^k}(S,C,D) \right)$$

where $(k',C,D) < (k,A,B)$ is the lexicographical order.

DYNAMIC PROGRAMMING: We want to compute $\mathbb{P}_{\mathscr{M}}(S_1^n)$. In order to decompose the probability in a tractable way, we use the previous notations:

$$\mathbb{P}_{\mathscr{M}}(S_i^j) = \sum_{k=1}^{j-1} \sum_{(S\to AB)\in G} \mathbb{P}_{\mathscr{M}}(\widehat{\mathscr{C}_{i,j}^k}(S,A,B))$$

with

$$\mathbb{P}_{\mathscr{M}}(\widehat{\mathscr{C}_{i,j}^k}(S,A,B)) = \mathbb{P}_{\mathscr{M}}(S_i^j, A_i^k, B_{i+k}^{j-k}) - \mathbb{P}_{\mathscr{M}}(\mathscr{C}_{i,j}^{<k}(S,A,B))$$

We have

$$\mathbb{P}_{\mathscr{M}}(S_i^j, A_i^k, B_{i+k}^{j-k}) = \begin{cases} \mathbb{P}_{\mathscr{M}}(A_i^k, B_{i+k}^{j-k}) & \text{if} \quad S\to AB \quad \text{is a valid production rule} \\ 0 & \text{otherwise} \end{cases}$$

and

$$\mathbb{P}_{\mathscr{M}}(\mathscr{C}_{i,j}^{<k}(S,A,B)) = \sum_{(k',C,D)<(k,A,B)} \mathbb{P}_{\mathscr{M}}(\widehat{\mathscr{C}_{i,j}^{k'}}(S,C,D), \mathscr{C}_{i,j}^k(S,A,B))$$

This last term can be computed with the following decomposition:

$$\mathbb{P}_{\mathscr{M}}(\widehat{\mathscr{C}_{i,j}^{k'}}(S,C,D), \mathscr{C}_{i,j}^k(S,A,B)) = \sum_{E_{i+k'}^{k-k'}} \mathbb{P}_{\mathscr{M}}(S_i^j, C_i^{k'}, D_{i+k'}^{j-k'}, A_i^k, B_{i+k}^{j-k}, E_{i+k'}^{k-k'})$$

This decomposition is correct since we assume that our grammar is weakly ambiguous. Let us suppose there are two parse trees T_1 and T_2 associated to a word $x_1 \dots x_n$ with $T_1 \in \mathscr{C}_n^{k'}(S,C,D)$ and $T_2 \in \mathscr{C}_n^k(S,A,B)$. If $k=k'$ then $A=C$ and $B=D$ due to the weak ambiguity. Let us suppose then that $k'<k$:

- The symbol generating $x_1 \dots x_{k'}$ in T_1 and T_2 is C.
- The symbol generating $x_k \dots x_n$ in T_1 and T_2 is E.
- Their is $E \in \mathscr{V}$ such that the E is the symbol generating $x_{k'} \dots x_k$ in both T_1 and T_2.

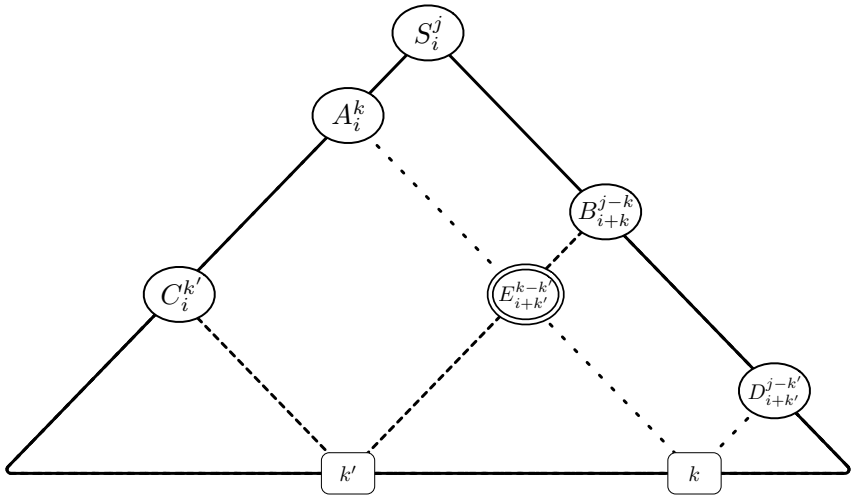

Figure 3.3: Graph representing the point how two parse trees representing the same word should intersect.

Therefore, to compute $\mathbb{P}_{\mathscr{M}}(\mathscr{C}_{i,j}^{<k}(S,A,B))$, we need to account for all trees in $\mathscr{C}_{i,j}^{k}(S,A,B)$ such that there is a tree in $\mathscr{C}_{i,j}^{k'}(S,C,D)$ fulfilling the above conditions for all $C,D \in \mathscr{V}$ and $k' < k$. When these conditions are satisfied, we have:

$$\mathbb{P}_{\mathscr{M}}(S_i^j, C_i^{k'}, D_{i+k'}^{j-k'}, A_i^k, B_{i+k}^{j-k}, E_{i+k'}^{k-k'}) = \mathbb{P}_{\mathscr{M}}(C_i^{k'}, E_{i+k'}^{k-k'}, D_{i+k'}^{j-k'})$$

MARKOV DEPENDENCIES: The last quantities we need to compute are $\mathbb{P}_{\mathscr{M}}(A_i^k, B_{i+k}^{j-k})$ and $\mathbb{P}_{\mathscr{M}}(C_i^{k'}, E_{i+k'}^{k-k'}, B_{i+k}^{j-k})$, which takes into account the Markov dependencies among variables. The first term $\mathbb{P}_{\mathscr{M}}(A_i^k, B_{i+k}^{j-k})$ can be computed the same way it as in the unambiguous case.

With the same approach, we have: given $X_{i+k'-1}$ or $X_{i+k'}$, and X_{i+k-1} or X_{i+k}, the variables $C_i^{k'}, E_{i+k'}^{k-k'}, B_{i+k}^{j-k}$ are conditionally independent. We can therefore write

$$\begin{aligned}\mathbb{P}_{\mathscr{M}}(C_i^{k'}, E_{i+k'}^{k-k'}, B_{i+k}^{j-k}) = \sum_{X_{i+k'}, C_i^{k'}, E_{i+k'}^{k-k'}, X_{i+k-1}, B_{i+k}^{j-k}} & \mathbb{P}_{\mathscr{M}}(X_{i+k'})\mathbb{P}_{\mathscr{M}}(C_i^{k'}, E_{i+k'}^{k-k'} | X_{i+k'}) \\ & \mathbb{P}_{\mathscr{M}}(X_{i+k-1} | E_{i+k'}^{k-k'}, X_{i+k'})\mathbb{P}_{\mathscr{M}}(B_{i+k}^{j-k} | X_{i+k-1})\end{aligned}$$

COMPLEXITY: The algorithm we propose needs to store the following quantities:

- $\mathbb{P}_{\mathscr{M}}(S_i^j | X_i, X_{i+j-1})$ for all i, j, S, which takes $\mathscr{O}(n^2|G|^3)$ space.
- $\mathbb{P}_{\mathscr{M}}(S_i^j, A_i^k, B_{i+k}^{j-k}, \widehat{C_j^k}(S,A,B))$ for all i, j, k, S, A, B which takes at most $\mathscr{O}(n^3|G|^3)$ space.

Therefore, we have a space complexity cubic in both the size of the sequence and the size of the grammar. The run time, due to the summation involved is in $\mathcal{O}(n^2|G|^4)$ per variable, which gives a total complexity of $\mathcal{O}(n^4|G|^5)$.

4 Conclusion

We have investigated the complexity of imposing several hard constraints on Markov sequences at the sampling phase: binary equalities and grammar membership. We give a sketch of proof for the #P-completeness of the binary equality constraint and identify three polynomial sub-cases. We also give a proof for the #P-completeness of the grammar membership constraint, and we identify two polynomial sub-cases, the unambiguous case and the new weakly-ambiguous case. Note that the subtle differences between perfect sampling, almost perfect sampling, exact counting and approximate counting could be investigated in more details to refine our statements.

References

[1] Samer A Abdallah and Nicolas E Gold. Exploring probabilistic grammars of symbolic music using prism. 2014.

[2] Stuart Geman and Mark Johnson. Probabilistic grammars and their applications. *International Encyclopedia of the Social & Behavioral Sciences*, 2002:12075–12082, 2002.

[3] Ryan Groves. *Automatic Melodic Analysis*. PhD thesis, McGill University Libraries, 2016.

[4] Why RNA Is Interesting. Stochastic context free grammars for rna modeling. 2001.

[5] Serdar Kadioglu and Meinolf Sellmann. Efficient context-free grammar constraints. In *AAAI*, pages 310–316, 2008.

[6] George Katsirelos, Sebastian Maneth, Nina Narodytska, and Toby Walsh. Restricted global grammar constraints. In *CP*, pages 501–508. Springer, 2009.

[7] George Katsirelos, Nina Narodytska, and Toby Walsh. The weighted grammar constraint. *Annals of Operations Research*, 184(1):179–207, 2011.

[8] Andrew McLeod and Mark Steedman. Meter detection in symbolic music using a lexicalized pcfg. 2017.

[9] Eita Nakamura, Masatoshi Hamanaka, Keiji Hirata, and Kazuyoshi Yoshii. Tree-structured probabilistic model of monophonic written music based on the generative theory of tonal music. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 276–280. IEEE, 2016.

[10] François Pachet, Pierre Roy, Alexandre Papadopoulos, and Jason Sakellariou. Generating 1/f noise sequences as constraint satisfaction: The voss constraint. In *IJCAI*, pages 2482–2488, 2015.

[11] A. Papadopoulos, P. Roy, and F. Pachet. Avoiding plagiarism in markov sequence generation. In *28th Conference on Artificial Intelligence (AAAI 2014)*, pages 2731–2737, Quebec (Canada), July 2014.

[12] Alexandre Papadopoulos, François Pachet, and Pierre Roy. Generating non-plagiaristic markov sequences with max order sampling. In *Creativity and Universality in Language*, pages 85–103. Springer, 2016.

[13] Alexandre Papadopoulos, François Pachet, Pierre Roy, and Jason Sakellariou. Exact sampling for regular and markov constraints with belief propagation. In *International Conference on Principles and Practice of Constraint Programming*, pages 341–350. Springer, 2015.

[14] Alexandre Papadopoulos, Pierre Roy, and François Pachet. Assisted lead sheet composition using flowcomposer. In *International Conference on Principles and Practice of Constraint Programming*, pages 769–785. Springer, 2016.

[15] Claude-Guy Quimper and Toby Walsh. Global grammar constraints. *Principles and Practice of Constraint Programming-CP 2006*, pages 751–755, 2006.

[16] S. Rivaud, F. Pachet, and P. Roy. Sampling markov models under binary equality constraints is hard. In *Journées Francophones sur les Réseaux Bayésiens et les Modèles Graphiques Probabilistes*, Clermont-Ferrand, France, June 2016.

[17] Pierre Roy and François Pachet. Enforcing meter in finite-length markov sequences. In *AAAI*, 2013.

[18] Meinolf Sellmann. The theory of grammar constraints. In *International Conference on Principles and Practice of Constraint Programming*, pages 530–544. Springer, 2006.

[19] Khalil Sima'an. Computational complexity of probabilistic disambiguation. *Grammars*, 5(2):125–151, 2002.

[20] Noah A Smith and Mark Johnson. Weighted and probabilistic context-free grammars are equally expressive. *Computational Linguistics*, 33(4):477–491, 2007.

[21] S Sundararajan. Probabilistic context-free grammars in natural language processing.

[22] Leslie G Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.