A grammar-based unification of several alignment and folding algorithms

Fabrice Lefebvre
LIX - École Polytechnique

June 24, 1996

[summary by Pierre Nicodème]

Abstract

We show that many popular models of folding and/or alignment may be described by a new formalism: multi-tape S-attribute grammars (MTSAGs). This formalism relieves the designer of biological models of implementation details. We present also a tool which, given a MTSAG, will output an efficient parser for this grammar and show that MTSAGs offer a new, efficient and useful way to handle stochastic context-free grammars. This summary is an extended abstract of [7].

1. Introduction

We shall see here that most popular models of alignment and/or folding of DNAs, RNAs or proteins, HMMs (Hidden Markov Models) [5], SCFGs (Stochastic Context-Free Grammars) [8] and CMs (Covariance Models) [3] share a common representation in terms of a new formalism: Multi-Tape S-Attribute Grammars (MTSAGs). This formalism is not only a help for the description of old or new methods. We designed and implemented a tool which, from the high-level description given by a MTSAG, will automatically generate the C source of an efficient C parser which is able to compute alignments and foldings. The speed and memory requirements of such generated parsers stand the comparison with programs manually written from dynamic programming relations. As a consequence, we show how to automatically build SCFGs from sets of unaligned, unfolded RNAs.

2. Definitions

We define a special "m-tape" alphabet which will handle sequence alignments, and then a "m-tape" extension of context-free grammars which will handle structures of alignments.

DEFINITION 1. A *m-tape alphabet* Σ is a product of m alphabets $\Sigma^{(i)}$ augmented with the empty string: $\Sigma = \bigotimes_{i=1\ldots m}(\Sigma^{(i)} \cup \{\epsilon\})$. An element $a_1 \cdots a_l$ of the free monoid Σ^*, generated by formal concatenation of m-tape elements of Σ, is called a *m-tape alignment* of length l. The empty alignment of Σ^* is denoted by ϵ.

EXAMPLE. $(abba, dcd)$ is a 2-tape input string on $\Sigma^{(1)} = \{a, b\}$ and $\Sigma^{(2)} = \{c, d\}$. We shall also write this 2-tape input string as $\begin{smallmatrix} abba \\ dcd \end{smallmatrix}$, which is a somewhat more natural notation in the context of alignments. This 2-tape input string has a 2-tape input substring $\begin{smallmatrix} bb \\ dc \end{smallmatrix}$.

DEFINITION 2. Given any m-tape alignment $a_1 \cdots a_l$, we get a m-tape input string by concatenation, or *ϵ-deletion*, of symbols of the projection of $a_1 \cdots a_l$ on every tape.

$$\Sigma^* \longrightarrow \langle \Sigma^* \rangle, a_1 \cdots a_l \longrightarrow \langle a_1 \cdots a_l \rangle.$$

$$
\begin{array}{rcll}
start & \longrightarrow & frame0 & (0) \\
frame0 & \longrightarrow & frame0\left[\begin{smallmatrix} X \\ X \end{smallmatrix}\right] \mid \left[\begin{smallmatrix} - \\ - \end{smallmatrix}\right] & (0) \\
 & \mid & frame0\left[\begin{smallmatrix} X \\ Y \end{smallmatrix}\right] & (2) \\
 & \mid & frame1\left[\begin{smallmatrix} - \\ X \end{smallmatrix}\right] \mid frame2\left[\begin{smallmatrix} X \\ - \end{smallmatrix}\right] & (1) \\
frame1 & \longrightarrow & frame1\left[\begin{smallmatrix} X \\ X \end{smallmatrix}\right] & (1) \\
 & \mid & frame1\left[\begin{smallmatrix} X \\ Y \end{smallmatrix}\right] & (3) \\
 & \mid & frame0\left[\begin{smallmatrix} X \\ - \end{smallmatrix}\right] \mid frame2\left[\begin{smallmatrix} - \\ X \end{smallmatrix}\right] & (3) \\
frame2 & \longrightarrow & frame2\left[\begin{smallmatrix} X \\ X \end{smallmatrix}\right] & (1) \\
 & \mid & frame2\left[\begin{smallmatrix} X \\ Y \end{smallmatrix}\right] & (3) \\
 & \mid & frame0\left[\begin{smallmatrix} - \\ X \end{smallmatrix}\right] \mid frame1\left[\begin{smallmatrix} X \\ - \end{smallmatrix}\right] & (3)
\end{array}
$$

FIGURE 1. In this weighted left-regular grammar, weights are written in parentheses after each group of productions having the same weight. Later on, weights will be turned into attribute evaluation functions.

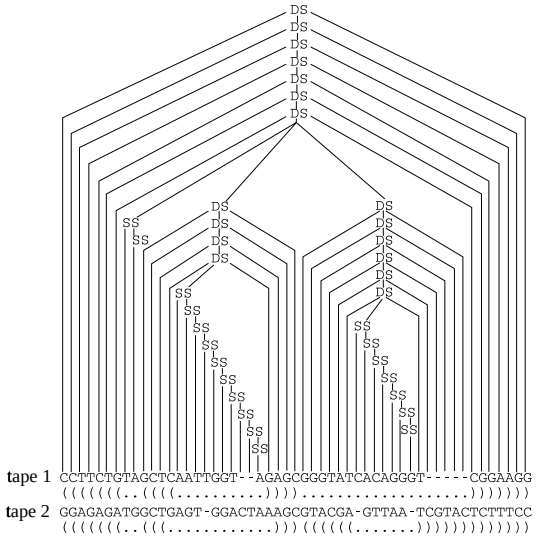

FIGURE 2. Derivation tree of an alignment of two RNAs.

EXAMPLE. Our 2-tape input string $\begin{smallmatrix} abba \\ dcd \end{smallmatrix}$ may be defined as an ϵ-deletion of the alignments $\left\langle \left[\begin{smallmatrix} \epsilon \\ d \end{smallmatrix}\right]\left[\begin{smallmatrix} a \\ \epsilon \end{smallmatrix}\right]\left[\begin{smallmatrix} b \\ \epsilon \end{smallmatrix}\right]\left[\begin{smallmatrix} b \\ c \end{smallmatrix}\right]\left[\begin{smallmatrix} a \\ d \end{smallmatrix}\right]\right\rangle$ or $\left\langle \left[\begin{smallmatrix} a \\ \epsilon \end{smallmatrix}\right]\left[\begin{smallmatrix} b \\ d \end{smallmatrix}\right]\left[\begin{smallmatrix} \epsilon \\ c \end{smallmatrix}\right]\left[\begin{smallmatrix} b \\ \epsilon \end{smallmatrix}\right]\left[\begin{smallmatrix} a \\ d \end{smallmatrix}\right]\right\rangle$.

Searls did show that the alignment of two strings according to some edit-distance may be carried out by some simple 2-tape nondeterministic finite automaton (NFA) with weighted transitions [9]. The sought alignment has a minimal total weight. The set of alignments recognized by a Searls' NFA is a regular language over our 2-tape terminal alphabet, and may be described by a regular grammar with weighted productions (see figure 1).

As regular grammars are a proper subset of context-free grammars, we found natural to generalize this idea of alignment to m-tape (i.e. the terminal alphabet is a subset of a m-tape alphabet) context-free grammars (MTCFGs) and their recognizing devices, namely m-tape nondeterministic pushdown automata (NPDA). Weighted transitions of NFA are easily translated into weighted pop-transitions of NPDA. The sought alignment is obtained from a sequence of pop-transitions of the NPDA which has an optimal (minimal for some problems, maximal for others, etc...) total weight.

Figure 2 shows how alignments and structures may be deduced from a single m-tape derivation. The underlying grammar may be easily recovered. Base pairings are inferred from derivations of DS (Double-Strand) and they are given below each tape. Notice that a double-strand has been defined as a substructure whose ends must be paired on at least one tape, whereas a single-strand (SS) may only have unpaired bases on both tapes.

DEFINITION 3. A ***m**-tape context-free grammar* $G = (V_T, V_N, P, S)$ consists of a finite set of terminals V_T such that V_T *is a subset of a* ***m**-tape alphabet*, a finite set of nonterminals V_N such that $V_N \cap V_T = \emptyset$, a finite set of productions (rewriting rules) P and a start symbol $S \in V_N$. Let $V = V_T \cup V_N$ denote the vocabulary of the grammar. Each production in P has the form $A \to \alpha$, where $A \in V_N$ and $\alpha \in V^*$. A is the left-hand side of the production and α its right-hand side.

A *derivation tree* is a planar representation of a sequence of derivations (replacements of a nonterminals A in a string of V^* by strings α such that $A \to \alpha$) and it is a result of *parsing*. The language $L(G)$ is the set of m-tape input strings generated by G: $L(G) = \{\langle u \rangle \in \langle V_T^* \rangle \mid S \to^* u\}$.

EXAMPLE. The following toy MTCFG will align two properly parenthesized strings interspersed with a:

$$S \rightarrow \left[\begin{smallmatrix}(\\(\end{smallmatrix}\right] S \left[\begin{smallmatrix})\\)\end{smallmatrix}\right] \mid \left[\begin{smallmatrix}a\\a\end{smallmatrix}\right] \mid \left[\begin{smallmatrix}a\\-\end{smallmatrix}\right] \mid \left[\begin{smallmatrix}-\\a\end{smallmatrix}\right] \mid \left[\begin{smallmatrix}-\\-\end{smallmatrix}\right] \mid SS$$

In this MTSAG, the structure defined by parentheses must be the same on both tapes, but substrings of a may be aligned with gaps (denoted by $-$ in terminals instead of ϵ, because a dash is appropriate, and even expected, in the context of alignments).

DEFINITION 4. Let $G = (V_T, V_N, P, S)$ be a proper m-tape context-free grammar. For every tape i $(1 \leq i \leq m)$, define the *projected grammar* $G^{(i)}$ as the conversion of the grammar $(V_T^{(i)}, V_N, P^{(i)}, S)$ into a proper grammar, where $V_T^{(i)}$ and $P^{(i)}$ are the sets of values on tape i of all the elements of V_T and P respectively.

EXAMPLE. The MTCFG of the preceding example has the same projected grammar on both tapes :

$$S \rightarrow (S) \mid () \mid a \mid SS$$

Projected grammars are useful for the study of the complexity of our parsing algorithm as a function of the ambiguity of MTCFGs.

We said earlier that we could assign a cost to each alignment or folding produced by a NPDA, thanks to weights on pop-transitions. This cost-evaluation step is essential for the determination of an optimal cost alignment or folding.

We use the general mechanism of synthesized attributes, or S-attributes which, together with MTCFGs, give us m-tape S-attribute context-free grammars, or MTSAGs. S-attributes are attributes which are assigned to every vertex of a derivation tree and which are computed from the bottom of a derivation tree (i.e. every terminal has a known S-attribute) to its root by means of attribute evaluation functions associated to grammar productions. Thanks to these functions, the computation of the final attribute of the derivation tree does not have to rely on a fixed, predetermined, operation (summation, multiplication, ...), as it would have been the case with weighted productions. In our implementation, attribute evaluation functions are C functions. We have already shown the effectiveness of S-attributes with our adaptation of the thermodynamic model of folding to context-free grammars [6]. This algorithm uses a parse table to store the shared forest of derivation trees of a m-tape input string.

DEFINITION 5. A *m-tape S-attribute grammar* is denoted by $G = (V_T, V_N, P, S, \mathcal{A}, S_{\mathcal{A}}, F_P)$. It is an extension of a m-tape context-free grammar $G = (V_T, V_N, P, S)$, where an attribute $x \in \mathcal{A}$ is attached to each symbol $X \in V$ and a string of attributes $\lambda \in \mathcal{A}^*$ to each string $\alpha \in V^*$. $S_{\mathcal{A}}$ is a function from V_T to $\mathcal{A}$ assigning attributes to terminals. F_P is a set of functions from $\mathcal{A}^*$ to $\mathcal{A}$. A function $f_{A \rightarrow \alpha}$ is in F_P iff $A \rightarrow \alpha$ is in P.

The attribute λ of a string α is the concatenation of the attributes of the symbols in α. When a function $f_{A \rightarrow \alpha}$ is applied to the attribute λ of a string α derived from A, it returns the attribute x of A (hence the bottom-up computation of attributes).

3. Syntax analysis for MTSAGs

A generalization of Cocke-Younger-Kasami's algorithm (CYK) would be an easy algorithm to parse m-tape input strings. This algorithm has a time complexity of $O(n^3)$ and a space complexity of $O(n^2)$ when only one tape of size n is considered [1]. A generalization to m tapes, each of size n, would lead to an algorithm having a complexity of $O(n^{3m})$ in time and $O(n^{2m})$ in space.

To overcome the limitations of CYK's algorithm, we generalized our parsing algorithm for 1-tape MTSAGs [6].

When constructing the parse table, a minimum condition of usefulness is applied. This condition means that an item is never add to an entry if it has no chance of being used in a derivation tree, up to the already parsed part of the m-tape input string. This condition is akin to a condition verified by Earley's parsing algorithm and it is the key to the lower parsing complexities of our algorithm when some projections of the underlying MTSAG are unambiguous.

In fact, out algorithm may be considered as an improvement of Earley's algorithm, where Earley's items $[\Delta \rightarrow \alpha \cdot \beta, i], (\alpha, \beta \in V^*)$ which share the same α and i are factorized into a single item $[\Delta \rightarrow \alpha, i]$. Also, non-kernel items of Earley's algorithm are replaced by much smaller sets of expected non-terminals.

PROPOSITION 1 (1-TAPE COMPLEXITY). *Let G be a proper 1-tape MTSAG and let $r \geq 1$ be the maximum number of nonterminals appearing at the right-hand side of a production of G. For a tape of length n, the time and space complexities of the previous parsing algorithm are, in order of decreasing constraints on G:*

- *Equal and at most $O(n)$ if G is LR(k) and not right-recursive (this encompasses left-regular grammars);*
- *equal and at most $O(n^2)$ if G is unambiguous;*
- *$O(n^{r+1})$ and $O(n^r)$ for a generic proper MTSAG.*

PROPOSITION 2 ($\boldsymbol{m}$-TAPE COMPLEXITY). *Let G be a proper m-tape MTSAG. The time complexity of our parsing algorithm on G is equal to the product of the parsing complexities of the same algorithm applied on each tape i with each projected grammar $G^{(i)}$. The same kind of result holds for space complexities. Hence the time complexity is at most $O(n^{m(r+1)})$, and the space complexity is at most $O(n^{mr})$, for m-tapes of size n*

In practice, MTSAGs that we used verified $r \leq 2$ and thus the time and space complexities of our parsers for those grammars were respectively $O(n^{3m})$ and $O(n^{2m})$ at most, but were sometimes much better.

4. Stochastic Context-Free Grammars

An essential aspect of MTSAGs is the ability to easily generate efficient parsers from grammars. On the basis of the tool we had already written to generate parsers from 1-tape S-attribute grammars, we designed a new tool, MTSAG2C, which automatically generates the C source of a parser from a given MTSAG. The generated parser is able to read tapes (thanks to a lexical analyzer provided by the user), parse tapes, and then output a single derivation tree which satisfies constraints given in the MTSAG.

When using 2-tape MTSAGs for SCFGs, we transfer on the first tape the high-level description of a family of RNAs usually used with SCFGs, and on the second tape the RNA to be folded and aligned. All rules used by the traditional SCFG generating tool to generate a SCFG from its high level description are then written down as a single, fixed, MTSAG. This has the additional benefit of shortening the development cycle (see figure 3b).

We compared the parser generated from a 1-tape version of a 97 nonterminals SCFG (this parser already proved to be quite fast [6]) to the parser generated from the 2-tape version of this SCFG (figure 4(a)).

Tests done on an Alpha 2100-500MP give the results:

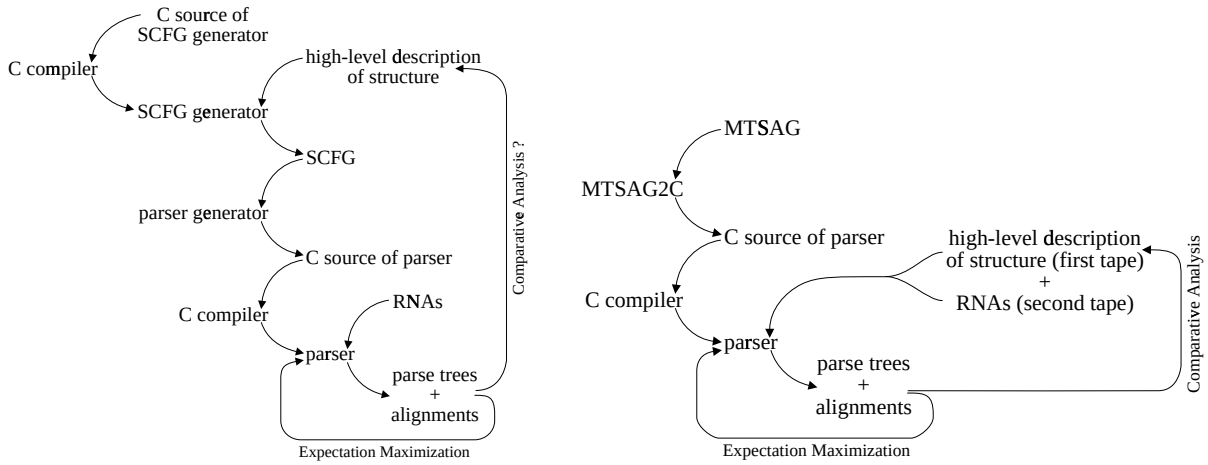

FIGURE 3. (left) 1-tape MTSAG; (right) 2-tape MTSAG. Development cycle of a MTSAG implementation of SCFGs. It has been suggested [4] that a comparative analysis of alignments resulting from parsing may be used to build a new SCFG or a new high-level description of it. With 2-tape MTSAGs, this kind of feedback is as easy to implement as the feedback designed for CMs by Eddy and Durbin [3].

	1-tape	2-tape
83 bases tRNA		
time in seconds:	0.45	0.33
space in Mbytes:	1.8	0.9

With MTSAGs you do not have to generate and compile another parser every time you modify the high-level description of your family (figure 3 (left)). Instead, we may use the following adaptation of the procedure of Eddy and Durbin to learn their CMs from initially unaligned and unfolded RNAs:

(1) Use a MTSAG adaptation of any folding algorithm (Sankoff, Zuker) to get a rough (and even wrong) initial folding. Convert this folding to a suitable first tape (by replacing all single strands by '*' for instance);
(2) Use Dirichlet priors to estimate probabilities;
(3) Align and fold all RNAs with a 2-tape MTSAG;
(4) Optimize probabilities from results of the previous step and repeat the previous step until probabilities converge;
(5) Use a comparative analysis algorithm on alignments of step 3 to get a new approximation of the common structural features of all RNAs. Then convert this approximation to a suitable first tape;
(6) Repeat steps 2 to 6 until the first tape converges.

5. Conclusion

We introduce a new way to describe SCGFs in the form of 2-tape MTSAGs and special first tapes. This new way alleviates the need for specialized SCFG building tools and for recompilations of parsers every time the model is changed (only the first tape has to be changed).

MTSAGs may also be applied to most useful sequence analysis methods which were usually expressed with dynamic programming relations (Smith-Waterman alignment model, global alignment, HMMs, simultaneous alignment and folding of RNAs). We believe that MTSAGs should be

```
(((((((..((((********)))).(((((.......)))))********((((.......)))))))))))).
CCUUCUGUAGCUCAAUUGGUAGAGCAUGUGACUGUAGAGUAUGCGGGUAUCACAGGGUCGCUGGUUCGAUUCCGGCCGGAAGG
```

(a) unaligned 2-tape input string.

```
(((((((..((((********)))).(((((.......)))))*******----------*(((((.......)))))))))))).
CCUUCUGUAGCUCAAUUGGUAGAGCAUGUGACUGUAGAGUAUGC--GG-GUAUCACAGGGUCGCUGGUUCGAUUCCGGCCGGAAGG
```

(b) 2-tape alignment of the previous 2-tape input string.

FIGURE 4. Unaligned and aligned version of a 2-tape input string. The first tape of this 2-tape input-string has a cloverleaf-like structure. This structure has two single strands which may have a variable length around 8 bases. The second tape is the RNA DY6050 extracted from a well known freely available compilation of tRNAs [10].

used instead of dynamic programming relations because these relations hinder the inventiveness of designers of new sequence analysis models.

We also gave a sketch of a method to build stochastic models from unaligned, unfolded RNAs. However, divide and conquer methods may be a prerequisite for long RNAs [2, 4]. We will try to apply MTSAGs to these methods.

Bibliography

[1] Aho (Alfred V.) and Ullman (Jeffrey D.). – *The Theory of Parsing, Translation, and Compiling.* – Prentice-Hall, 1972, vol. 1.

[2] Corpet (Florence) and Michot (Bernard). – RNAlign program: alignment of RNA sequences using both primary and secondary structures. *Computing Applications in the Biosciences*, vol. 10, n° 4, 1994, pp. 389–399.

[3] Eddy (Sean R.) and Durbin (Richard). – RNA sequence analysis using covariance models. *Nucleic Acids Research*, vol. 22, n° 11, 1994, pp. 2079–2088.

[4] Grate (Leslie). – Automatic RNA secondary structure determination with stochastic context-free grammars. In *Third International Conference on Intelligent Systems for Molecular Biology.* pp. 136–144. – AAAI Press, 1995.

[5] Krogh (A.), Brown (M.), Mian (I. S.), Sjölander (K.), and Haussler (D.). – Hidden Markov models in computational biology: Applications to protein modeling. *Journal of Molecular Biology*, vol. 235, 1994, pp. 1501–1531.

[6] Lefebvre (Fabrice). – An optimized parsing algorithm well suited to RNA folding. In *Third International Conference on Intelligent Systems for Molecular Biology.* pp. 222–230. – AAAI Press, 1995.

[7] Lefebvre (Fabrice). – A grammar-based unification of several alignment and folding algorithms. In *Fourth International Conference on Intelligent Systems for Molecular Biology.* pp. 143–154. – AAAI Press, 1996.

[8] Sakakibara (Yasubumi), Brown (Michael), Hughey (Richard), Mian (I. Saira), Sjölander (Kimmen), Underwood (Rebecca C.), and Haussler (David). – Stochastic context-free grammars for tRNA modeling. *Nucleic Acids Research*, vol. 22, 1994, pp. 5112–5120.

[9] Searls (David B.) and Murphy (Kevin P.). – Automata – theoretic models of mutation and alignment. In *Third International Conference on Intelligent Systems for Molecular Biology.* pp. 341–349. – AAAI Press, 1995.

[10] Steinberg (S.), Misch (A.), and Sprinzl (M.). – Compilation of tRNA sequences and sequences of tRNA genes. *Nucleic Acids Research*, vol. 21, 1993, pp. 3011–3015.