Uses and abuses of intersected languages

David Chiang
Department of Computer and Information Science
University of Pennsylvania
200 S 33rd St
Philadelphia PA 19104 USA

Abstract

In this paper we discuss the use of *intersection* as a tool for modeling syntactic phenomena and folding of biological molecules. We argue that intersection is useful but easily overestimated, because intersection coordinates grammars via their string languages, and if strong generative capacity is given priority over weak generative capacity, this kind of coordination turns out to be rather limited. We give two example uses of intersection which overstep this limit, one using CFGs and one using a range concatenation grammar (RCG). We conclude with an analysis and example of the different kinds of parallelism available in an RCG.

1 Introduction

Context-free languages (as well as the language classes of many other formalisms) are closed under union but not under intersection (Hopcroft and Ullman, 1979), the classic example being:

$$\{\mathbf{a}^*\mathbf{b}^n\mathbf{c}^n\} \cap \{\mathbf{a}^n\mathbf{b}^n\mathbf{c}^*\} = \{\mathbf{a}^n\mathbf{b}^n\mathbf{c}^n\} \tag{1}$$

which is easily shown by the pumping lemma to be beyond the power of CFG. Since recognizing the intersection of two CFLs takes only twice as long as recognizing a single CFL, this appears to be a way to obtain some of the power of grammar formalisms like TAG without their computational complexity. But this extra power appears less significant once we consider that strong generative capacity—the set of *structural descriptions* generated by a grammar (Chomsky, 1963)—is of primary importance for most applications of formal grammars.

Assume that a grammar G generates a set of structural descriptions $\Sigma(G)$, and for each such structural description D, a string $\overline{D}$ can be recovered, so that the string language $L(G)$ is defined as $\{\overline{D} \mid D \in \Sigma(G)\}$. Extending this definition to intersections, we define

$$\Sigma(G_1 \cap G_2) = \{D_1 \otimes D_2 \mid D_i \in \Sigma(G_i), \overline{D_1} = \overline{D_2}\} \tag{2}$$

where $\otimes$ is some operation for composing structural descriptions such that if $\overline{D_1} = \overline{D_2}$, then $\overline{D_1 \otimes D_2} = \overline{D_1} = \overline{D_2}$, otherwise $D_1 \otimes D_2$ is undefined. Note that in (2), D_1 and D_2 are correlated only by their yields; they do not directly constrain each other at all. Thus, from the point of view of strong generative capacity, language intersection is better thought of as adding a constraint to the tail end of otherwise independent parallel processes. We call this type of parallelism *weak parallelism* and argue that for real applications it is easy to overestimate how much control this kind of parallelism offers. We illustrate this in our first example, which uses CFGs for RNA pseudoknots.

We then consider the range-concatenation grammar (RCG) formalism (Boullier, 2000), which includes an intersection operation, allowing it to integrate weak parallelism more tightly into the operation of the grammar. However, weak parallelism is still susceptible to the caveat from above, which we illustrate with a second example, an analysis of German scrambling. Finally, we analyze more carefully the different kinds of parallelism available in an RCG and illustrate how they can be combined to model protein β-sheets.

2 Brown and Wilson's intersected-CFL analysis of RNA pseudoknots

Our first example comes from the RNA structure prediction literature. An RNA molecule can be thought of as a string over an alphabet of *nucleotides* or *bases* $\{\mathbf{a}, \mathbf{u}, \mathbf{c}, \mathbf{g}\}$. Certain pairs of bases, called *complementary* pairs, have an affinity for each other: **a** with **u**, **c** with **g**. This causes a molecule to fold up into a *secondary structure*, which depends on the sequence of bases. A central problem is predicting, given a sequence, what structure or structures the sequence will fold into.

TAG+7: Seventh International Workshop on Tree Adjoining Grammar and Related Formalisms.
May 20-22, 2004, Vancouver, BC, CA.
Pages 9-15.

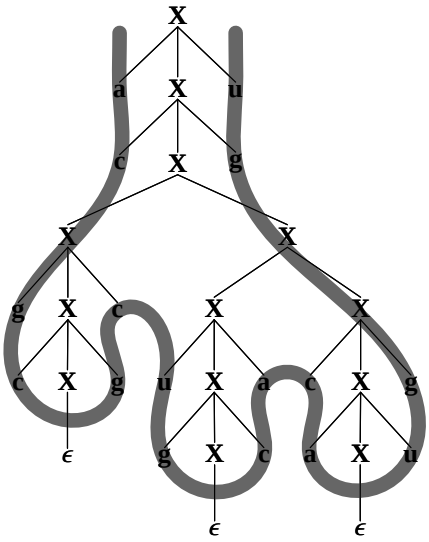

Figure 1: Example CFG derivation, with superimposed primary structure. Nonterminal symbols other than **X** are suppressed for clarity.

Searls (1992) was the first to observe similarities between this problem and syntactic analysis in natural language and to propose the use of formal grammars for biological sequence analysis. Consider the following CFG:

$$
\begin{aligned}
&\mathbf{S} \rightarrow \mathbf{Z} \\
&\mathbf{X} \rightarrow \mathbf{a}\overbrace{\mathbf{Z}\mathbf{u}} \mid \mathbf{u}\overbrace{\mathbf{Z}\mathbf{a}} \mid \mathbf{c}\overbrace{\mathbf{Z}\mathbf{g}} \mid \mathbf{g}\overbrace{\mathbf{Z}\mathbf{c}} \\
&\mathbf{Y} \rightarrow \mathbf{a}\mathbf{Y} \mid \mathbf{u}\mathbf{Y} \mid \mathbf{c}\mathbf{Y} \mid \mathbf{g}\mathbf{Y} \mid \epsilon \\
&\mathbf{Z} \rightarrow \mathbf{Y}\mathbf{X}\mathbf{Z} \mid \mathbf{Y}
\end{aligned}
\tag{3}
$$

This grammar generates the language Σ^*, but in such a way that only complementary bases are generated in the same derivation step (see Figure 1). RNA structures mostly contain only nested base pairings, like the *hairpin* (Figure 2a). For such structures, the above CFG is sufficient. However, some structures involve crossing dependencies, like the *pseudoknot* (Figure 2b), which crosses the nested base pairings of one hairpin with those of another.

There have been efforts to model pseudoknots using formalisms beyond CFG (Uemura et al., 1999; Rivas and Eddy, 2000). But Brown and Wilson (1996) attempt a different solution. They observe that $\{\mathbf{a}^m\mathbf{g}^*\mathbf{u}^m\mathbf{c}^*\}$ and $\{\mathbf{a}^*\mathbf{g}^n\mathbf{u}^*\mathbf{c}^n\}$ are context-free languages, but

$$\{\mathbf{a}^m\mathbf{g}^*\mathbf{u}^m\mathbf{c}^*\} \cap \{\mathbf{a}^*\mathbf{g}^n\mathbf{u}^*\mathbf{c}^n\} = \{\mathbf{a}^m\mathbf{g}^n\mathbf{u}^m\mathbf{c}^n\} \tag{4}$$

is beyond the power of CFG. Brown and Wilson propose to exploit this fact by interpreting the language (4) as a set of pseudoknots: the *m* **a**s and **u**s form one hairpin, and the *n* **g**s and **c**s are the other. And unlike with syntactic structures, there is an obvious way of combining the structural descriptions of the two component grammars: simply superimpose their base pairings.

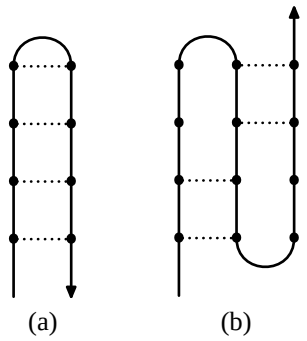

Figure 2: (a) RNA hairpin. (b) RNA pseudoknot.

However, in order for the pseudoknot to be well-formed, the two hairpins must interlock without colliding. That is, the base pairings must cross, but no two pairings should involve the same base. But the only reason the above example achieves this is because one hairpin has only **a**s and **u**s and the other has only **c**s and **g**s—that is, each symbol indicates overtly which hairpin it belongs to. For real molecules, both component grammars would have to generate at least all possible hairpins, or $\{vww^Rx\}$. In that case there would be no way of preventing the component grammars from missing each other or colliding.

Brown and Wilson recognize that there is a problem, but it is not clear whether they appreciate how serious it is. Their solution is to employ a special parsing strategy that uses the results of parsing with the first grammar to constrain the parse with the second; then the string is reparsed with the first, then again with the second. This procedure works only for their pair of grammars and only approximates the desired computation.

The root of the problem is that intersection only operates on strings, not structural descriptions. It allows parallel structural descriptions to be derived independently, then filters them on the basis of their string yields. We therefore call this kind of parallelism *weak parallelism*. The above example attempts to harness weak parallelism to generate only well-formed pseudoknots, but in order to do so it assumes that there is more information in the string languages than there really is.

3 Boullier's RCG analysis of German scrambling

Our second example comes from the range concatenation grammar (RCG) literature (Boullier, 2000). Here we briefly present a definition of a variant of RCG as a kind of deductive system. RCG clauses have the form

$$\psi :- \phi_1, \ldots, \phi_n.$$

(meaning "ψ is provable if $\phi_1, \ldots, \phi_n$ all are"). If $n = 0$, we simply write

$$\psi.$$

(which is trivially provable). The ψ and the ϕ_i in turn have the form

$$A(\alpha_1, \ldots, \alpha_m)$$

where A is a predicate (nonterminal) symbol and the α_j are strings of terminal symbols and variables (which range over strings of terminal symbols). Every α_j in ψ must be a substring of an $\alpha_{j'}$ in one of the ϕ_i. This condition ensures that in the derivation of a string w, all variables are instantiated only to substrings of w. (The standard definition of RCG does not have this requirement, because its variables range not over strings but pairs of string positions of w. The definition here is closer to that of simple literal movement grammars (Groenink, 1997).)

The language defined by an RCG is the set of all strings w such that $S(w)$ is provable, where S is a distinguished start predicate. The class of range-concatenation languages (RCLs) is exactly the set of languages recognizable in deterministic polynomial time (Bertsch and Nederhof, 2001).

Moreover, RCL, unlike CFL, is closed under intersection. The proof is very simple: given two grammars, we rename apart their predicate symbols; let S_1 and S_2 be the renamed start symbols and S be a new start symbol, and add the new clause

$$S(x) :- S_1(x), S_2(x).$$

Because the conjunction operator (comma) is part of the formalism, it can be used not only to intersect whole languages, but the yields of subderivations. This means that RCG gives finer control over weak parallelism, allowing us to localize it and use it in concert with other mechanisms in the grammar. The caveat from the previous section still applies, however, as we illustrate below.

Boullier (1999) explores possible uses of the extra power of RCG, and applies it to the phenomenon of German scrambling, in which the arguments of a verb cluster may appear in any order (see Figure 4). If we assume that an arbitrary number of verbs is allowed and arbitrary permutations of arguments is allowed, then scrambling can be shown to be beyond the power of linear context-free rewriting systems (Becker et al., 1992).

Boullier gives an RCG that he claims models German scrambling (Figure 3). The predicates **S**, **N**, and **V** use intersection to call the predicate **T** on every word, **N**′ on every noun, and **V**′ on every verb. This is an instance of *independent parallelism* (Rambow and Satta, 1999)—parallel processes in separate derivation branches—coupled with weak parallelism, which constrains the predicates to operate on the same strings.

$$\begin{aligned}
\mathbf{S}(XY) &:- \mathbf{N}(X, Y), \mathbf{V}(Y, X). \\
\mathbf{N}(nX, Y) &:- \mathbf{T}(n, X), \mathbf{N}'(n, Y), \mathbf{N}(X, Y). \\
\mathbf{N}(\epsilon, Y). & \\
\mathbf{V}(vX, Y) &:- \mathbf{T}(v, X), \mathbf{V}'(v, Y), \mathbf{V}(X, Y). \\
\mathbf{V}(\epsilon, Y). & \\
\mathbf{T}(a, bX) &:- \mathbf{T}(a, X). && a, b \in \Sigma, a \neq b \\
\mathbf{T}(a, \epsilon). & \\
\mathbf{N}'(n, vX). & && h(n) = v \\
\mathbf{N}'(n, vX) &:- \mathbf{N}'(n, X). && h(n) \neq v \\
\mathbf{V}'(v, nX). & && h(n) = v \\
\mathbf{V}'(v, nX) &:- \mathbf{V}'(v, X). && h(n) \neq v
\end{aligned}$$

Figure 3: Simplified version of Boullier's grammar. The function h maps from nouns to the verbs which take them as arguments.

These three predicates **T**, **N**′, and **V**′, in turn, check that each word is properly connected to the syntactic dependency structure. But as in Brown and Wilson's pseudoknot analysis, these three predicates rely on nonexistent information in the surface string.

First of all, **N**′ finds for each noun the verb on which it depends, and **V**′ likewise finds for each verb one of the nouns which depends on it. But whether a noun depends on a verb is assumed to be determinable (by the function h) from the noun and verb alone. In actuality, all that is known from a noun and verb is whether the noun can depend on the verb (because the verb might mark a certain case, for example), not whether it actually does. If h simply indicated the possibility of a noun depending on a verb, then this analysis' orchestration of its constraints would break down: several verbs might claim a single noun as an argument, or a verb might claim a noun which claims a different verb.

Second, the predicate **T** is used to check that all the nouns and all the verbs in each sentence are distinct, whereas in fact there is no reason why the same noun or verb would not be used twice in a single sentence. Passing over the fact that this constraint makes the generated language finite, all these constraints together indicate that the analysis assumes that dependency information is somehow overt. But this is not the case for real sentences. As in Brown and Wilson's system, this grammar tries to make weak parallelism do more than it can by assuming more information in the string language than is actually there.

(5) daß bisher noch niemand dem Kunden den Kühlschrank zu reparieren zu versuchen versprochen hat
that so far yet no one the client the refrigerator to repair to try promised has

that so far no one has promised to repair the refrigerator

(6) daß dem Kunden den Kühlschrank bisher noch niemand zu reparieren zu versuchen versprochen hat
that the client the refrigerator so far yet no one to repair to try promised has

that so far no one has promised to repair the refrigerator

Figure 4: Examples of German long-distance scrambling.

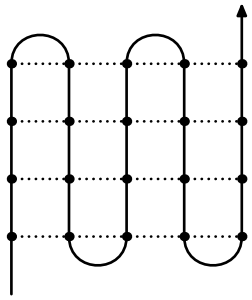

Figure 5: Protein β-sheet.

4 RCGs for protein β-sheets

What, then, can intersection be used for? Here we explore the possibility of using it for protein β-sheets. Like RNAs, proteins can be thought of as strings over an alphabet of bases, only the alphabet has 20 elements (amino acids) and the relationship between them is more complex than the complementary pairs of RNAs, and bases come together in *self-contacts* to form a folded structure. In one structural element, the β-sheet, multiple *strands* fold into a pattern like the one shown in Figure 5.

4.1 A multicomponent TAG analysis

A previous analysis (Abe and Mamitsuka, 1997) using a grammar formalism loosely related to set-local multicomponent TAG (Weir, 1988) uses *synchronous parallelism* (Rambow and Satta, 1999)—parallel processes in a single branch of a derivation—to model β-sheets. An equivalent multicomponent TAG is shown in Figure 6. This method has several strengths, which we will point out below, but two drawbacks. First, the number of strands generated is proportional to the number of components required; therefore the parsing complexity of a grammar that can generate k strands will be exponential in k. Furthermore, every grammar must impose some upper bound on the number of strands; no single grammar can generate all sheets.

A second problem is that this analysis is susceptible to a kind of spurious ambiguity in which a single structure can be derived in multiple ways. For example, consider Figure 7. In order to generate the β-sheet (a), we need trees like (b) and (c). But either of these trees can be used by itself to generate the β-sheet (d). The grammar must make room for the maximum number of strands, but when it does not use all of it, ambiguity can arise. It should be possible to carefully write the grammar to avoid much of this ambiguity, but we have not been able to eliminate all of it even for the single-component TAG case.

4.2 An RCG analysis

RCG, like many formalisms, has both synchronous parallelism (multiple arguments to a predicate) and independent parallelism (multiple predicate calls in a right-hand side). As mentioned above, it also has weak parallelism (multiple occurrences of a variable in a right-hand side), which can be coupled with either of the other two types of parallelism. We show below how these mechanisms can be used together to create an alternative model of β-sheets.

We start with some building blocks:

$$\mathbf{Anti}(a_1 X, Y a_2) \text{ :– } \mathbf{Anti}(X, Y). \qquad a_i \in \Sigma$$
$$\mathbf{Anti}(\epsilon, \epsilon).$$
$$\mathbf{Par}(a_1 X, a_2 Y) \text{ :– } \mathbf{Par}(X, Y). \qquad a_i \in \Sigma$$
$$\mathbf{Par}(\epsilon, \epsilon).$$
$$\mathbf{Adj}(X, Y) \text{ :– } \mathbf{Ant}(X, Y).$$
$$\mathbf{Adj}(X, Y) \text{ :– } \mathbf{Par}(X, Y).$$

The predicates **Anti** and **Par** generate pairs of antiparallel and parallel strands, respectively. This is an instance of synchronous parallelism, but only for pairs of strands, not all the strands together as in the multicomponent TAG analysis. Irregularities as in Figure 7a are also possible, but not shown here.

Then we can use intersection to combine them into a

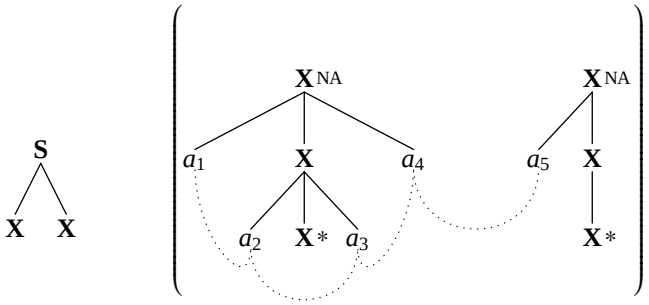

Figure 6: Set-local multicomponent TAG analysis of protein β-sheet with five alternating strands.

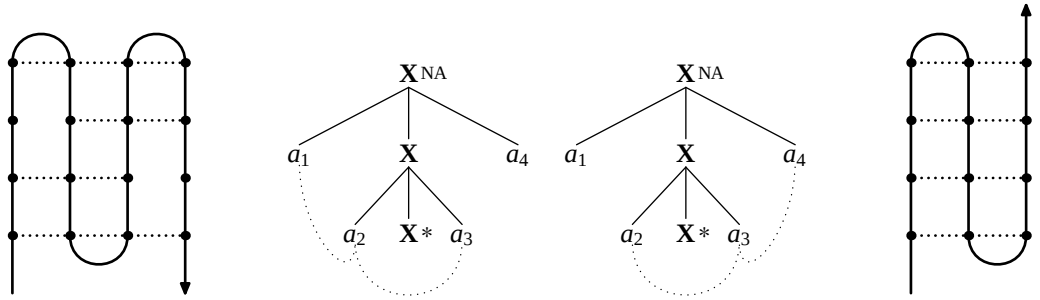

Figure 7: Illustration of spurious ambiguity.

sheet:

$$\textbf{Beta}(AB) \text{ :– } \textbf{B}(A, B).$$
$$\textbf{B}(ABY, B') \text{ :– } \textbf{B}(A, B), \textbf{Adj}(B, B').$$
$$\textbf{B}(BY, B') \text{ :– } \textbf{Adj}(B, B').$$

The first argument to **B** is a β-sheet minus the last strand, and the second argument is the last strand. The second production forms a larger β-sheet out of a smaller one by appending a new last strand and joining it to the previous last strand using **Adj**. This production has $O(n^5)$ possible instantiations (because it takes six indices to specify the variables on the left-hand side, but the arguments of **B** are always adjacent, eliminating one index), and therefore the parsing complexity of this grammar is also $O(n^5)$. Crucially, this complexity bound is not dependent on the number of strands, because each series of contacts is generated independently, not synchronously as in the multicomponent TAG analysis.

Weak parallelism is being used to ensure that each strand is consistent—that is, no two parts of the derivation that generate the same strand will disagree about the contents of the strand (including its length). Unlike with Brown and Wilson's analysis and Boullier's analysis, this is information that is really contained in the substring itself, so this is a legitimate use of weak parallelism.

Finally, even independent parallelism allows parallel subderivations to control each other via their root nonterminal (predicate) symbols, as illustrated in the following example. A β-sheet can be rolled into a cylinder to form a *β-barrel*. We can generate these as well, but we must keep track of the direction of each strand so as not to generate any Möbius strips:

$$\textbf{Barrel}(ABC) \text{ :– } \textbf{B}(A, B, C), \textbf{Par}(A, C).$$
$$\textbf{Barrel}(ABC) \text{ :– } \textbf{B}'(A, B, C), \textbf{Anti}(A, C).$$
$$\textbf{B}(A, BCY, C') \text{ :– } \textbf{B}'(A, B, C), \textbf{Anti}(C, C').$$
$$\textbf{B}(A, BCY, C') \text{ :– } \textbf{B}(A, B, C), \textbf{Par}(C, C').$$
$$\textbf{B}(A, Y, A') \text{ :– } \textbf{Par}(A, A').$$
$$\textbf{B}'(A, BCY, C') \text{ :– } \textbf{B}(A, B, C), \textbf{Anti}(C, C').$$
$$\textbf{B}'(A, BCY, C') \text{ :– } \textbf{B}'(A, B, C), \textbf{Par}(C, C').$$
$$\textbf{B}'(A, Y, A') \text{ :– } \textbf{Anti}(A, A').$$

Here **B** has three arguments: the first strand, the middle part, and the last strand; there is an additional predicate symbol **B**′ which is the same as **B**, except that **B**′ is for sheets with antiparallel first and last strands, whereas **B** is restricted here to sheets with parallel first and last strands. The first clause joins the first and last strands to form a barrel; it uses the information in the **B** vs. **B**′ distinction to join the strands so that no Möbius strips will be generated.

4.3 The importance of synchronous parallelism

The strands of β-sheets do not always appear in linear order; they can be permuted as in Figure 8. We can model such permutations by increasing the degree of synchronous parallelism (that is, the number of arguments to **B**), and therefore increasing parsing complexity. By contrast, since multicomponent TAG already uses synchronous parallelism to generate all the strands together, it allows permutations of strands at no extra cost.

Suppose we envision a sheet being built up one strand at a time, each successive strand being added to either side of the sheet:

$$\begin{aligned}
\textbf{Beta}(ABCD) &\text{ :– } \textbf{B}(A,B,C,D).\\
\textbf{B}(ABC,D,Y,B') &\text{ :– } \textbf{B}(A,B,C,D), \textbf{Adj}(B,B').\\
\textbf{B}(A,B,CDY,B') &\text{ :– } \textbf{B}(A,B,C,D), \textbf{Adj}(D,B').\\
\textbf{B}(\epsilon,B,Y,B') &\text{ :– } \textbf{Adj}(B,B').
\end{aligned}$$

Figure 8a shows an example sheet that can be generated by this grammar but not the previous ones. In this grammar, the second and fourth arguments to **B** are the leftmost and rightmost strands (*not* respectively) in the folded structure. The second clause adds a new strand on one side, and the third clause adds a new strand on the other. Both clauses have $O(n^7)$ possible instantiations if we take into account that the four arguments to **B** will always be adjacent.

Suppose we always build up a sheet out of two smaller sheets, as in Figure 9. Figure 8b shows an example sheet that can be generated by this grammar but not the previous ones. In this grammar, the second and fourth arguments are again the leftmost and rightmost strands (*not* respectively) in the folded structure. The second and third clauses join two β-sheets together in two different ways; there are conceivably four ways to join them together, but using only these two avoids spurious ambiguity. Both clauses have $O(n^{12})$ possible instantiations if we take into account that the five arguments to **B** will always be adjacent.

Figure 8c shows the only permutation of four strands that the above grammar cannot generate. This does not seem problematic, since, at least for sheets formed out of two hairpin motifs, this permutation was not known as of 1991 to occur in nature (Branden and Tooze, 1999, p. 31).

Another way in which synchronous parallelism might be important can be seen by comparing with some results for RNA structure prediction. Akutsu (2000) has shown that structure prediction for a similar class of RNA folds, "generalized pseudoknots," is NP-hard. The proof reduces another NP-hard problem (longest common subsequence) to RNA structure prediction using the assumption that in RNA structures, a single base may not participate in multiple pairings. The RCG approach illustrated above cannot enforce this assumption (just as Brown and Wilson's could not), because weak parallelism does not allow that level of control. By contrast, the multicomponent TAG analysis could enforce it easily. But for proteins this assumption does not hold, so this is not problematic for our grammars.

However, the weights we assign to productions must also respect the independence of intersected derivations. For example, the energy of a contact between two strands must not depend on other strands or contacts with them. Another NP-hardness result for RNA structure prediction (Lyngsø and Pedersen, 2000) relies crucially on such a dependency: it assumes that the energy of a base pairing (i, j) can be affected by another base pairing $(j-1, i')$ even if i and i' come from different "strands," or by $(j', i+1)$ even if j and j' come from different "strands." We leave for future work the question of how important dependencies of this sort are for β-sheets, and whether a limited amount of synchronous parallelism would suffice to approximate them.

5 Conclusion

The fundamental difficulty with the first two applications we have examined is a confusion between weak and strong generative capacity: it is misleading to speak of the "pseudoknot language" or the "scrambling language," even as abstractions, because what really matters in these two phenomena are the structures assigned to the strings rather than the strings themselves. This danger is heightened when dealing with intersected languages because intersection provides control over strings and only indirectly over structural descriptions. We have given two examples of applications which overestimate the power of this *weak parallelism* and illustrated how to use weak parallelism in concert with synchronous and independent parallelism in an RCG.

Acknowledgements

This research was supported in part by NSF ITR grant EIA-02-05456. I would like to thank Julia Hockenmaier, Laura Kallmeyer, Aravind Joshi, and the anonymous reviewers for their valuable help. *S. D. G.*

References

Naoki Abe and Hiroshi Mamitsuka. 1997. Predicting protein secondary structures based on stochastic tree grammars. *Machine Learning*, 29:275–301.

Tatuya Akutsu. 2000. Dynamic programming algorithms for RNA secondary structure prediction with pseudoknots. *Discrete Applied Mathematics*, 104:45–62.

Tilman Becker, Owen Rambow, and Michael Niv. 1992. The derivational generative power of formal systems,

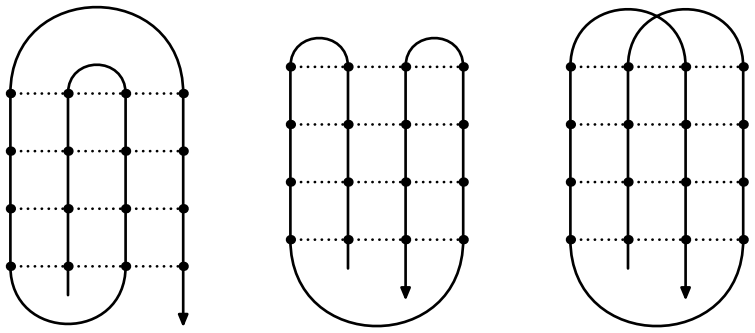

Figure 8: Permuted β-sheets.

$$\begin{aligned}
\mathbf{Beta}(ABCDE) &:\!- \mathbf{B}(A,B,C,D,E).\\
\mathbf{B}(ABC,D,EYA',B',C'D'E') &:\!- \mathbf{B}(A,B,C,D,E),\mathbf{B}(A',B',C',D',E'),\mathbf{Adj}(B,D').\\
\mathbf{B}(A,B,CDEYA',B',C',D',E') &:\!- \mathbf{B}(A,B,C,D,E),\mathbf{B}(A',B',C',D',E'),\mathbf{Adj}(D,D').\\
\mathbf{B}(\epsilon,B,C,D,\epsilon) &:\!- \mathbf{Adj}(B,D).
\end{aligned}$$

Figure 9: $O(n^{12})$-time RCG for β-sheets.

or, Scrambling is beyond LCFRS. Technical Report IRCS-92-38, Institute for Research in Cognitive Science, University of Pennsylvania. Presented at MOL3.

Eberhard Bertsch and Mark-Jan Nederhof. 2001. On the complexity of some extensions of RCG parsing. In *Proceedings of the Seventh International Workshop on Parsing Technologies (IWPT 2001)*, pages 66–77, Beijing.

Pierre Boullier. 1999. Chinese numbers, MIX, scrambling, and range concatenation grammars. In *Proceedings of the Ninth Conference of the European chapter of the Association for Computational Linguistics (EACL '99)*, pages 53–60.

Pierre Boullier. 2000. Range concatenation grammars. In *Proceedings of the Sixth International Workshop on Parsing Technologies (IWPT 2000)*, pages 53–64, Trento.

Carl-Ivar Branden and John Tooze. 1999. *Introduction to Protein Structure*. Garland Publishing, Inc., New York. 2nd ed.

Michael Brown and Charles Wilson. 1996. RNA pseudoknot modeling using intersections of stochastic context free grammars with applications to database search. In *Proceedings of the Pacific Symposium on Biocomputing 1996*.

Noam Chomsky. 1963. Formal properties of grammars. In R. Duncan Luce, Robert R. Bush, and Eugene Galanter, editors, *Handbook of Mathematical Psychology*, volume 2, pages 323–418. Wiley, New York.

Annius Victor Groenink. 1997. *Surface without Structure: Word order and tractability issues in natural language analysis*. Ph.D. thesis, University of Utrecht.

John E. Hopcroft and Jeffrey D. Ullman. 1979. *Introduction to automata theory, languages, and computation*. Addison-Wesley, Reading, MA.

Rune B. Lyngsø and Christian N. S. Pedersen. 2000. RNA pseudoknot prediction in energy-based models. *J. Computational Biology*, 7(3/4):409–427.

Owen Rambow and Giorgio Satta. 1999. Independent parallelism in finite copying parallel rewriting systems. *Theoretical Computer Science*, 223:887–120.

Elena Rivas and Sean R. Eddy. 2000. The language of RNA: a formal grammar that includes pseudoknots. *Bioinformatics*, 16(4):334–340.

David B. Searls. 1992. The linguistics of DNA. *American Scientist*, 80:579–591.

Yasuo Uemura, Aki Hasegawa, Satoshi Kobayashi, and Takashi Yokomori. 1999. Tree adjoining grammars for RNA structure prediction. *Theoretical Computer Science*, 210:277–303.

David J. Weir. 1988. *Characterizing Mildly Context-Sensitive Grammar Formalisms*. Ph.D. thesis, Univ. of Pennsylvania.