

Feature Article

# Computational linguistics: A new tool for exploring biopolymer structures and statistical mechanics

Ken A. Dill<sup>a,\*</sup>, Adam Lucas<sup>b</sup>, Julia Hockenmaier<sup>c</sup>, Liang Huang<sup>c</sup>,  
David Chiang<sup>d</sup>, Aravind K. Joshi<sup>c</sup>

<sup>a</sup> Department of Pharmaceutical Chemistry, University of California, San Francisco, N 472-F, 600 16th Street, San Francisco, CA 94143-2240, United States

<sup>b</sup> Department of Mathematics and Computer Science, Saint Mary's College of California, Moraga, CA 94575, United States

<sup>c</sup> Institute for Research in Cognitive Science and Department of Computer and Information Science, University of Pennsylvania, 3401 Walnut Street, Suite 400A, Philadelphia, PA 19104, United States

<sup>d</sup> USC Information Sciences Institute, 4676 Admiralty Way, Suite 1001, Marina del Rey, CA 90292, United States

Received 1 March 2007; received in revised form 4 May 2007; accepted 8 May 2007

Available online 23 May 2007

## Abstract

Unlike homopolymers, biopolymers are composed of specific sequences of different types of monomers. In proteins and RNA molecules, one-dimensional sequence information encodes a three-dimensional fold, leading to a corresponding molecular function. Such folded structures are not treated adequately through traditional methods of polymer statistical mechanics. A promising new way to solve problems of the statistical mechanics of biomolecules comes from *computational linguistics*, the field that uses computers to parse and understand the sentences in natural languages. Here, we give two examples. First, we show that a dynamic programming method of computational linguistics gives a fast way to search protein models for native structures. Interestingly, the *computational search* process closely resembles the *physical folding* process. Second, linguistics-based dynamic programming methods are also useful for computing partition functions and densities of states for some foldable biopolymers – helix-bundle proteins are reviewed here. In these ways, computational linguistics is helping to solve problems of the searching and counting of biopolymer conformations.

© 2007 Elsevier Ltd. All rights reserved.

**Keywords:** Biopolymers; Proteins; Lattice models

## 1. How computational linguistics applies to the structures of RNA and proteins

We review here new computational ways to enumerate the conformations of biopolymers, drawn from the seemingly distant field of *computational linguistics*. Why is computational linguistics relevant to biopolymer statistical mechanics? A biopolymer chain encodes a one-dimensional information, resembling the way a sentence encodes information in a linear string of words. Just as a sentence is a linear chain of words taken from a vocabulary, a linear heteropolymer molecule

is a covalent linear chain of monomers taken from an “alphabet” of different chemical moieties. Just as every sentence in a natural language has a particular grammatical structure that encodes its meaning in a one-dimensional string of words, proteins encode their three-dimensional structures (and functions) in a one-dimensional string of amino acids. In this paper, we illustrate the computational linguistics approach with two examples: (1) an efficient algorithm for predicting the native states and folding routes of proteins and (2) an algorithm for computing the partition functions and stabilities of helix-bundle proteins.

Consider the process of extracting information from a spoken or written sentence. This is called *diagramming* or *parsing* a sentence. Parsing is a process by which a listener: (a) begins with the one-dimensional string of words, (b) searches through

\* Corresponding author.

E-mail address: [dill@maxwell.ucsf.edu](mailto:dill@maxwell.ucsf.edu) (K.A. Dill).

a potentially large number of different *topologies* that represent the many possible relationships among different words and phrases, and (c) chooses the one that conveys the correct single meaning of the sentence. This is how a listener comes to understand a sentence. A similar process is needed for predicting the three-dimensional structure of a protein molecule from its sequence of monomer units (see Fig. 1). Computer protein structure prediction, too, is a process: (a) starting with a one-dimensional string information, (b) considering all the possible topologies (conformations) that could represent the possible native structure of the protein, and then (c) choosing the one (native structure) having the global minimum free energy.

In this review, we describe how the automated diagramming or parsing algorithms used in computational linguistics are beginning to contribute insights into biopolymer structures and statistical mechanics. These algorithms are all instances of *dynamic programming*. Dynamic programming algorithms can solve large search problems (such as the search for the lowest energy structure of a protein) by recursively decomposing them into smaller problems that can be solved independently (e.g. the search for the lowest energy structure of fragments of this protein) and using the solutions of these smaller problems to solve the larger problem. When applied to language, such algorithms require a *formal grammar*, a mathematical description of the possible sentences and associated grammatical structures for a particular language.

The key insight is in recognizing that for certain topological optimization problems – including the parsing of a sentence or the folding of a protein, the globally optimal state can usually be found efficiently by making small local independent decisions at first (deciding whether small phrases of a sentence

make sense, or determining whether short peptide fragments of a protein chain are metastable) and then combining those decisions hierarchically until a solution is found to the full problem. While not guaranteed to find the global optimum in general, it is found that this divide-and-conquer approach almost always works for these two specific types of problems (see below).

First, we describe the CKY (Cocke–Kasami–Younger) algorithm [1,2], a commonly used dynamic programming technique for parsing sentences. Then we describe a variant of CKY, which we call ZAMDP (Zipping and Assembly Mechanism by Dynamic Programming), that searches protein models to find native states in a recursive, hierarchical fashion [3]. Finally, we describe how related dynamic programming methods of computational linguistics can help computing statistical mechanical partition functions of folded polymers [4,5], such as proteins and RNA, a problem that has traditionally been challenging.

## 2. Parsing sentences using dynamic programming

In order to understand and distinguish the meaning of sentences such as ‘*We eat sushi with tuna*’ and ‘*We eat sushi with chopsticks*’, it is necessary to parse them, i.e., to identify their correct syntactic structures. Fig. 1 shows the possible structures, or parses, for both sentences. We use *phrase structure trees* to represent the parses of our example sentences, although Fig. 1 also shows the corresponding dependency graphs, which resemble polymer graphs. Such trees can be generated by *context-free grammars* [6], a particular kind of formal grammar that is able to generate recursive and hierarchically nested structures. Each node of the parse tree represents a ‘constituent’ of the sentence, and is labeled by its corresponding syntactic category, e.g. S (sentence), NP (noun phrase), VP (verb phrase) or PP (prepositional phrase). These parse trees can be generated by the grammar shown in Fig. 2. Rules such as ‘ $S \rightarrow NP VP$ ’ are statements that a constituent of category S (sentence) can be formed from a constituent of category NP (noun phrase) that is immediately followed by a constituent of category VP (verb phrase). Lexical rules of the form ‘ $NP \rightarrow we$ ’ specify that the word *we* forms a constituent of category NP.

### 2.1. How the CKY method works: the details

Fig. 3 shows how the CKY algorithm identifies all possible structures for an input sentence. Chart parsing algorithms such as CKY are dynamic programming techniques that exploit the independence assumptions implicit in the tree representation to search all possible trees efficiently and systematically.

$S \rightarrow NP VP$	$V \rightarrow eat$
$VP \rightarrow V NP$	$NP \rightarrow we$
$VP \rightarrow VP PP$	$NP \rightarrow sushi$
$NP \rightarrow NP PP$	$NP \rightarrow tuna$
$PP \rightarrow P NP$	$P \rightarrow with$

Fig. 2. A formal (context-free) grammar that describes the sentences in Fig. 1.

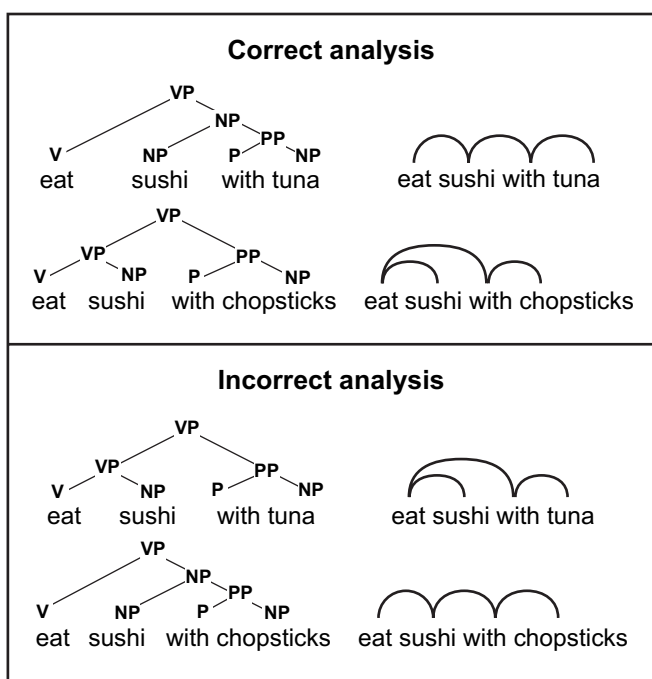


Fig. 1. Phrase structure trees (left) or dependency graphs (right) show the grammatical structures of sentences. Here, S stands for sentence, NP for noun phrase, N for noun, VP for verb phrase, V for verb, PP for prepositional phrase, and P for preposition.

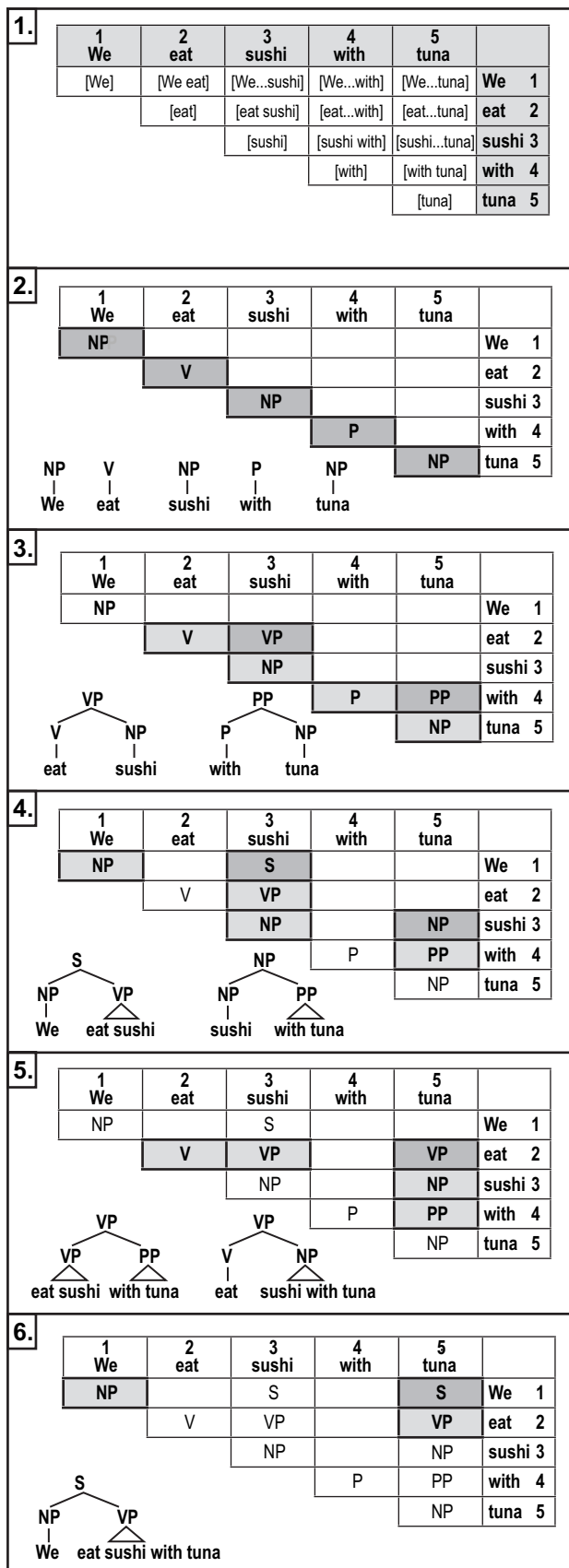


Fig. 3. The CKY algorithm: (1.) the chart is initialized; (2.) the lexicon entries are entered into the chart; (3.–6.) the chart is filled.

In order to parse the sentence *We eat sushi with tuna*, CKY first creates a table, called the *parse chart*. Each cell in this matrix is expressed as  $chart[i][j]$ , which represents the part of the sentence from word  $i$  to word  $j$  (Fig. 3(1.)). CKY then fills this chart *bottom-up*, starting with the cells along the main diagonal, i.e., the individual words themselves (Fig. 3(2.)). Next, CKY moves to the next diagonal away from the main diagonal and fills all the cells  $chart[i][i + 1]$ , followed by the next diagonal, which are all the cells  $chart[i][i + 2]$ , etc., until the top cell  $chart[1][n]$  is filled (Fig. 3(2.–6.)).

The cell  $chart[i][j]$  is filled in the following manner: if  $chart[i][k]$  contains a constituent with category  $Y$ ,  $chart[k + 1][j]$  contains a constituent with category  $Z$ , and if there is a grammar rule  $X \rightarrow Y Z$ , a new constituent with category  $X$  is entered into  $chart[i][j]$ . In order to recover the grammatical structure of  $X$ , a pair of backpointers from  $X$  to  $Y$  and  $Z$  is added. If  $chart[i][j]$  already contains a constituent  $X$ , it is sufficient to add a new pair of backpointers to that  $X$ . Therefore, the cell for *eat sushi with tuna* only contains one VP. The cell  $chart[i][j]$  remains empty if there are no previous chart entries  $Y$  in  $chart[i][k]$  and  $Z$  in  $chart[k + 1][j]$  and corresponding grammar rules  $X \Rightarrow Y Z$  that allow it to be filled.

If, at the end of the process, the top cell  $chart[1][n]$  is empty, it means that the input sentence cannot be generated by the grammar, and therefore does not belong to the language. Sometimes, parsing a sentence involves making probabilistic decisions: one possible meaning of a sentence may be more likely than another, for example. In those cases, *statistical parsers* (e.g. see Refs. [8,9]) are used. Statistical parsers require a probability model to rank competing analyses. This model is also typically used to prune away unlikely structures early in the process in order to speed up the search.

We now describe how a similar algorithm can be applied to protein folding.

### 3. A dynamic programming algorithm for hierarchical protein folding

Research in protein folding has long faced the “Levinthal paradox”, the question of how a protein can find its unique native structure — which is a small speck in its large conformational space — very quickly (in microseconds, for some proteins). One hypothesis is that proteins fold by the *Zippering and Assembly Mechanism (ZAM)* [10–16]. Accordingly, in the earliest stages of folding (submicroseconds, for small proteins), peptide pieces of the chain search independently their local conformations and converge upon metastable partial substructures. At later stages, either one such metastable piece grows additional structure by accreting nearby unstructured chain or multiple peptide pieces assemble together into larger structures. This hierarchical process of “local first, global later” is identical to the way the CKY method parses sentences. On this basis, we tested a variant of CKY as a computational strategy for predicting the native states of proteins from amino acid sequences. We call the dynamic programming method *ZAMDP (Zippering and Assembly Mechanism by Dynamic Programming)*. We have explained ZAMDP in more detail elsewhere [3].

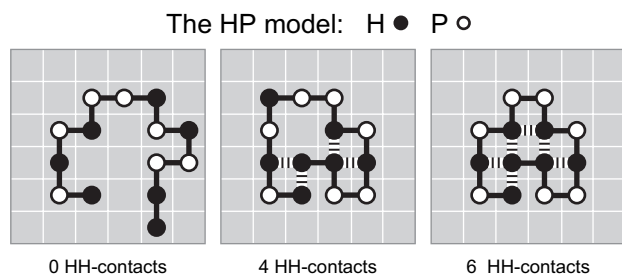


Fig. 4. The HP model of proteins.

Dynamic programming techniques have also been proposed for the identification of optimal combinations of independent secondary structure elements [17].

To test this hierarchical search principle, we used a simple exact lattice model, called the HP model [18]. In the HP model (Fig. 4), each monomer unit is represented as a single bead. To capture the different amino acid sequences in the simplest possible way, we lump the 20 types of amino acids into just two types of monomers: hydrophobic (H) and polar (P), chained together in different specific sequences. Folding is driven when two H monomers noncovalently associate with each other in different conformations. The energy function of the HP model is based on the number of (native or non-native) HH contacts in a conformation. In this, the HP model differs from, and is more physically plausible than, the superficially similar lattice variants of  $G\bar{o}$  models, which use potentials that simply enforce the native structure. We consider short chains (up to 20 monomers) on two-dimensional square lattices. Exhaustive enumeration on the computer allows us to explore the full conformational space without approximation. Despite its simplicity, the HP model has all the essential ingredients of proteins needed to serve as a workbench for testing computational search methods: it contains the basic physics, where chain entropy and excluded volume oppose a collapse driven by solvation interactions encoded throughout the sequence, different sequences fold to different structures, and the native structures are often unique in a conformational space that grows exponentially with the chain length. With this model, we have been able to test whether the ZAMDP search method is able to find the single globally optimal native conformation, and do so without searching the full conformational space. We found that this CKY-like searching is an efficient way to find the native states of these model proteins.

ZAMDP for protein folding follows the chart parsing method described above. It predicts the structure of a protein chain through a hierarchical search process that: (a) identifies the possible structures of small chain fragments, (b) stores them in a lookup table (like a parse chart), and then (c) iteratively combines adjacent pairs of such fragments, to grow structure in the molecule toward the native state. Unlike standard CKY, ZAMDP does not use a grammar, but simply concatenates adjacent chain fragments like pieces of a jigsaw puzzle, and explores all their local configurations. When two pieces are brought together, we search all the viable ways they can be configured and keep only those having lowest energies.

### 3.1. The hierarchical search method is also a useful model for the physical folding process

A computational search method, such as the dynamic programming algorithm proposed here, need not necessarily represent the kinetic processes that are involved when a protein physically folds up in a test tube. However, because the processes in our hierarchical search method so closely resemble the physical processes that we believe are involved in protein folding — parallel, local, independent decisions of peptide pieces at first, and more serial, nonlocal, global, and interdependent decisions later — we tested whether its computational steps might also correspond to the physical microscopic routes by which proteins fold. We find that the algorithmic steps closely resemble the physical folding routes, as noted below.

#### 3.1.1. This search method can account for the Levinthal paradox

Because proteins can fold up so quickly, it has been clear since Levinthal first noted it in 1968 [19] that proteins must typically avoid searching vast stretches of conformational space. What conformations do the protein avoid? And, how does it avoid them? The approach described above gives a simple answer: the large global optimization problem of protein folding can be solved, at least for most foldable monomer sequences, by breaking the problem into independent local optimizations on small local pieces of chain at first, and then making increasingly hierarchical decisions on those pieces until the protein is folded.

The search method is efficient because it never searches more than a few degrees of freedom at a time and eliminates high-energy conformations early in the search. Even though our method explores only a small fraction of the search space, it correctly identifies the single globally optimal conformation (known from prior exhaustive enumeration) for 96.6% of all 24,900 HP sequences of length 20. This local-first-global later Zipping and Assembly Mechanism explains how almost any protein can physically fold so quickly to its unique native state despite its large number of degrees of freedom and its complex energy landscape.

#### 3.1.2. Physical kinetics is more parallel than in Monte Carlo

Because ZAMDP represents folding routes as trees, it appears to be a more accurate way of representing the microscopic parallelness of folding than Monte Carlo algorithms are; the latter are inherently sequential. Unlike Monte Carlo algorithms, our hierarchical dynamic programming method explores and returns all possible routes simultaneously.

#### 3.1.3. This algorithm captures the Plaxco–Simons–Baker relationship between the native structure and folding rate

Protein folding rates are approximately predictable from their native structures: folding is fastest for proteins having the most *local* contacts (i.e., near neighbors in the chain sequence, as in helical proteins) and slowest for proteins having the most *nonlocal* contacts (mainly  $\beta$ -sheet proteins). Over

about eight orders of magnitude, the logarithm of folding rates diminishes with a quantity called the average *contact order* of the native structure [20], a measure of the average separation along the chain of the contacting monomers. Correspondingly, we find that, when averaged over all 20mers in the HP model with the same contact order, a “computational folding rate”, which takes the accessibility of the native state as well as the computational search time required for CKY to reach it into account, follows a similar dependence on the average contact order (see Fig. 5).

### 3.1.4. ZAMDP identifies slow- and fast-folding proteins

Since our algorithmic folding times correlate with physical folding times, the routes, mechanisms and bottlenecks that the dynamic programming method predict may also have some similarity to the physical processes. What are the search bottlenecks? Fig. 7 shows the difference between fast- and slow-folding proteins. It shows that fast folders are fast because: (a) each local decision (say, to form a turn of a helix) is rapid (because it does not involve much conformational searching), (b) there are many parallel options (a helix can start at any turn), and (c) each local contact is helpful (because it reduces the space of remaining conformations), even if it is non-native. A local contact is established after less searching than a nonlocal contact, and once a contact has been established, less subsequent searching is required.

### 3.1.5. ZAMDP identifies slow- and fast-folding routes

The ZAMDP identifies all direct folding routes that lead to the native state. We display ensembles of folding routes for the same HP sequence by projecting them onto the parse chart such that a cell is colored black if all routes pass through it and white if none do. Under the assumption that each

combination of two adjacent conformations requires one time step and different parts of the same chain (corresponding to different branches of a folding route tree) fold simultaneously, we can also compute search times for each folding route. Fig. 6 shows that, despite a multitude of microscopic routes, the partially folded substructures in the ensemble of fast routes are clearly distinct from those in the ensemble of slow-folding routes.

In summary, ZAMDP, the CKY-like dynamic programming method that is useful for parsing sentences, proves also to be a useful conformational search method for finding the native structures of proteins, closely resembling the Zipping and Assembly Mechanism by which proteins physically fold up.

## 4. Dynamic programming is also useful for computing statistical mechanical partition functions of heteropolymers

Now, we switch to a different problem of biopolymer theory. We are interested in the stabilities and folding cooperativities of proteins and RNA molecules. Traditional polymer statistical mechanics theories readily compute the partition functions for two different types of single-chain transitions: helix–coil transitions or coil–globule collapse transitions. It has been considerably more challenging, however, to compute partition functions for foldable sequence-specific biomolecules such as proteins. Folded proteins have a type of ordered structure that is much more complex than a simple helix or than a simple compact globule. The folded structure of a protein typically has some  $\alpha$ -helical structure in some places,  $\beta$ -sheet structure in other places, and different native chain folds are encoded by different monomer sequences. Traditional statistical mechanical averaging over whole structures

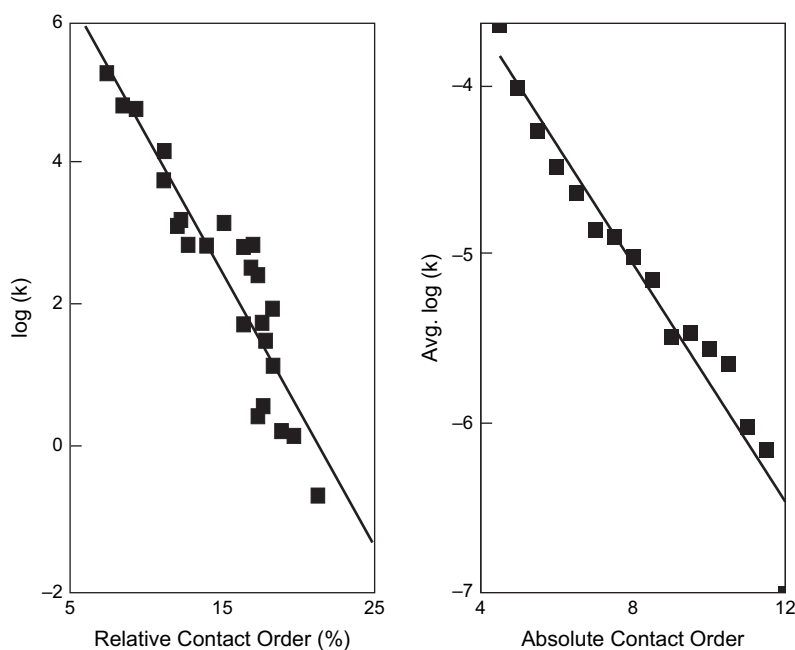


Fig. 5. Left (from Ref. [7]): experimental folding is fastest for proteins having the most local structures (helices and turns). Right: ZAMDP searching speeds are also the fastest for proteins having the most local native structures.



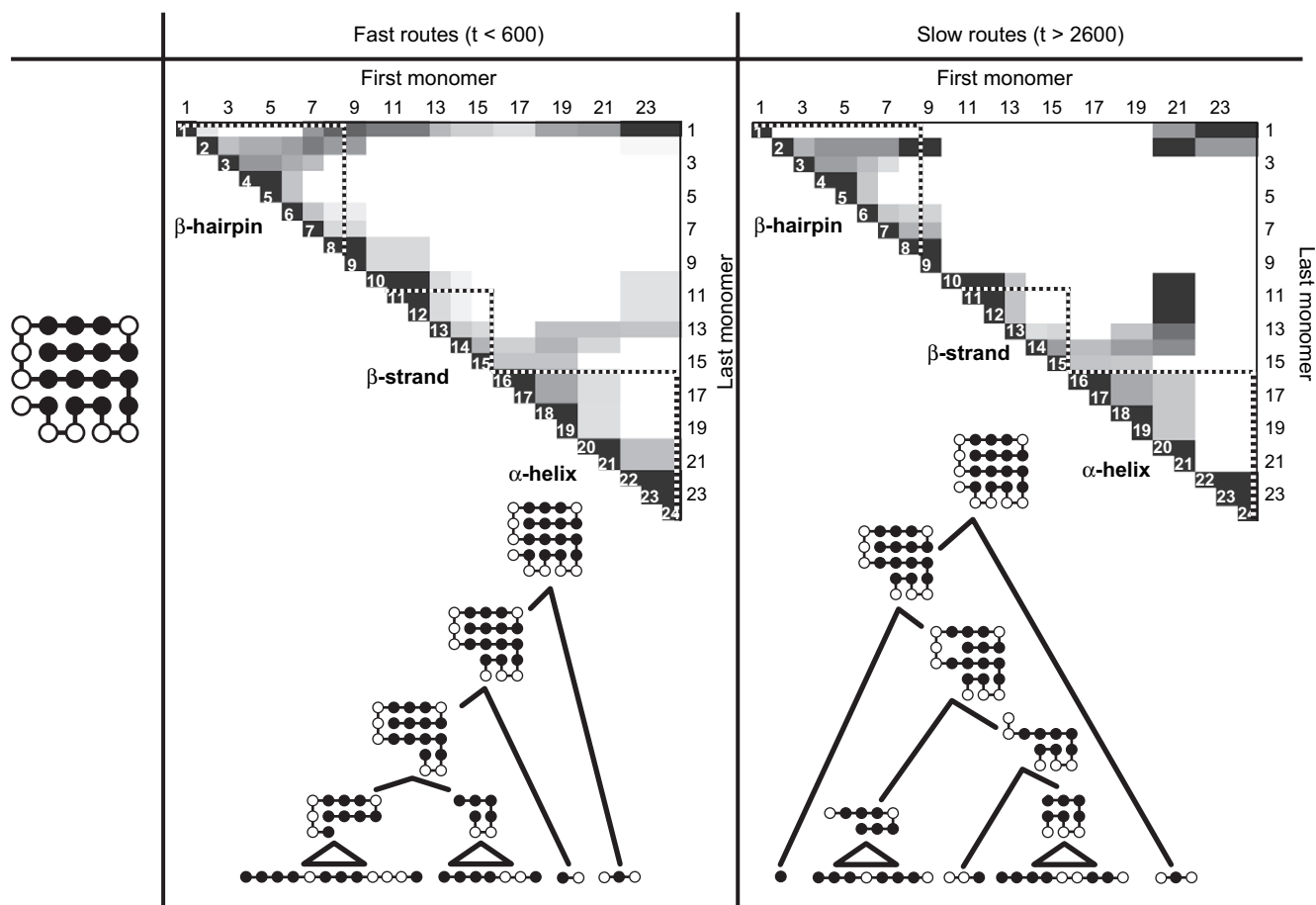


Fig. 6. The ZAMDP algorithm identifies all folding routes. Here, fast (left) and slow (right) routes. The diagrams at the top show where the trees show a representative fast or slow-folding route. We have colored each cell of the parse chart according to the fraction of native routes that go through it.

or whole sequences is not applicable to foldamers. Correspondingly, it is also challenging to define proper order parameters and reaction coordinates for foldamers. We describe here how computational linguistics-inspired dynamic programming techniques can compute partition functions for some simple protein and RNA molecule structures.

We focus on helix-bundle proteins, which are proteins whose only secondary structure elements are helices – usually three or more, see Fig. 8. Often, the tight packing constraints within a

native protein imply that several of the helices (which are approximately cylindrical) are usually approximately aligned, as with rods in a box. Our model can be adapted to study  $\beta$ -sheet proteins and is currently being used to study the equilibrium unfolding of the Formin binding protein WW domain.

A key question: why do helix-bundle proteins fold so cooperatively? Small helix-bundle proteins fold via a *two-state* transition – equivalent to a first-order transition in a macroscopic system. That is, at the midpoint of the folding transition,

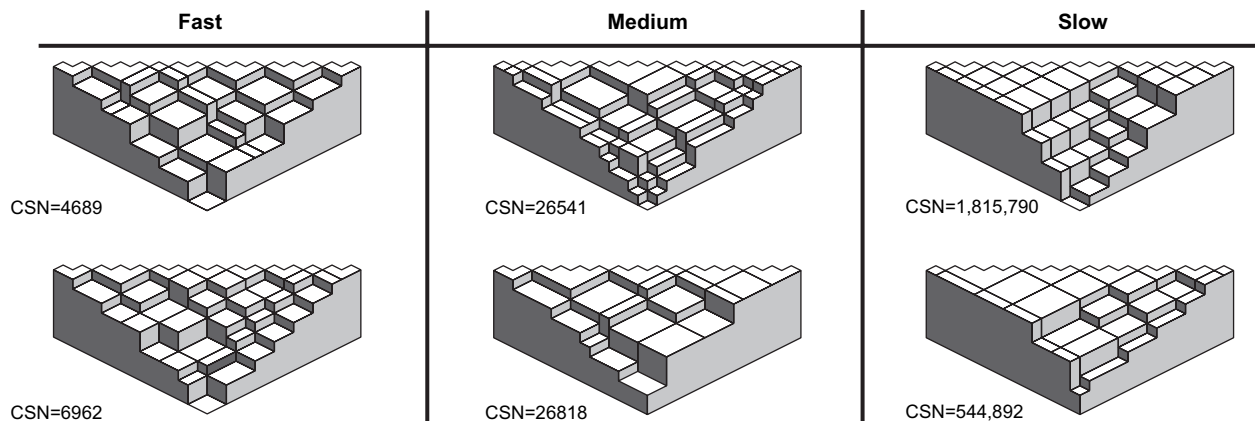


Fig. 7. The *chart landscape* shows the lowest accessible energy level for each fragment. Its shape predicts the amount of search required to fold a protein.

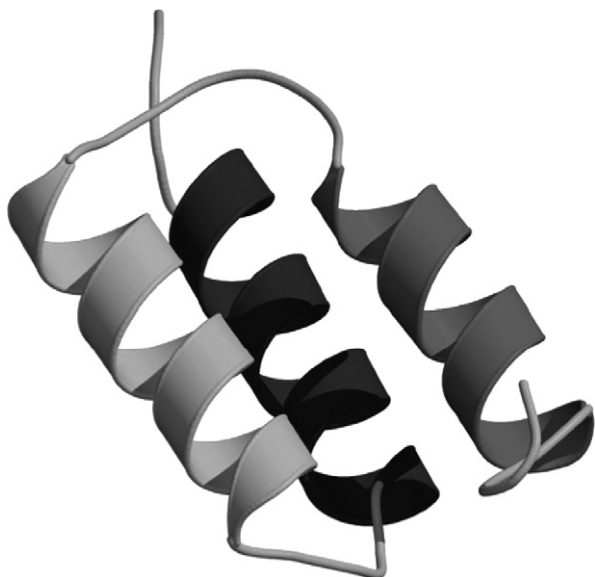


Fig. 8. A 3-helix-bundle protein: the  $\beta$  domain of protein A.

half the protein molecules are native and half are denatured and there is essentially no population of intermediate structures. One possibility is that the protein's cooperativity arises from the hydrogen bonding within each helix: a helical stretch of the protein chain prefers to be fully helical or coil-like, but not partly helical. Another possibility is that the protein's cooperativity arises because of the hydrophobically driven collapse. In that case, the chain prefers to be either expanded or completely compact, fully excluding water, but not partially collapsed. A third possibility is that both local and nonlocal factors cooperate: as each helix forms, it promotes the formation of the other helices and promotes the collective assembly of the helices together as a bundle. We describe below a model for helix-bundle folding, based on computational linguistics dynamic programming.

#### 4.1. The Ascending Level Model (ALM) of helix bundles

Our aim in this modeling is more modest than to treat the full conformational space. Rather, our aim is to begin with the knowledge that the native structure of our amino acid sequence is a helix bundle, and to enumerate only the conformations most relevant to the folding routes and thermodynamics. For example, small helix-bundle proteins fold very quickly, indicating that they do not get stuck in non-native kinetic traps. So, we consider only increasing native-like structures from the denatured conformations. This approximation allows us to use an efficient dynamic programming method for finding the approximate partition function.

To compute the populations of the different conformations of a helix-bundle protein, we need to enumerate the densities of states and the Boltzmann factors for each energy level. We use a lattice model. The lattice has layers (slices) that are perpendicular to the axes of the aligned helices in the native structure, see Fig. 9. A helix bundle is a linear stack of levels, with each level being either a helical (h) or coil (c) monomer (see Fig. 9).

In principle, as with many problems in polymer statistical mechanics, the count of the number of conformations increases exponentially with chain length. However, our problem is substantially reduced because of two features of the model. First, using dynamic programming (DP), described below, the time required to compute the numbers of states scales as no worse than  $O(l^5)$  for an  $l$ -level 3-helix bundle. An  $l$ -level 3-helix bundle is represented by 3 strands, each strand having  $l$  possible helical turns. If we have  $s$  strands and  $l$  levels, the number of states will be  $2^{sl}$ . For a 4-level 3-helix bundle, for example, this gives  $2^{12} = 4096$  different states (one of which is shown in Fig. 9). Since there are 3 monomers per turn, these 4096 states represent a 36-mer chain. Second, our degrees of freedom are associated with the layers, not with the monomers, reducing the conformational explosion.

In principle, as  $l$ , the number of levels, increases, the number of states (i.e., the size of the *search space*) increases exponentially: a 150-level helix bundle would have  $2^{150}$  states. However, dynamic programming (DP) makes an independence assumption that allows us to factorize the partition function. In particular, we assume that consecutive coil conformations of monomers, separated by a helix conformation, are independent, reducing the search space enormously, to  $150^2$ . DP also gains efficiency from building up the partition function from subcomponents, without the need for recomputation of the parts.

In our modeling to date, we have considered three types of interactions: hydrogen bonds within the helices, hydrophobic interactions (among any pair of monomers, in the simplest case, we have considered a hydrophobic homopolymer), and possible additional cooperative interactions that result when two helices pack together. The energy,  $E$ , of a chain configuration is:

$$E = N_{\text{hb}}\epsilon_{\text{hb}} + N_{\phi}\epsilon_{\phi} + N_c\epsilon_c \quad (1)$$

where  $\epsilon_{\text{hb}}$ ,  $\epsilon_{\phi}$ , and  $\epsilon_c$  are the energies (in units of kcal/mol), respectively, for a hydrogen bond, a hydrophobic interaction and a cooperative interaction. The count of each such type of interaction is  $N_{\text{hb}}$ ,  $N_{\phi}$  and  $N_c$ , respectively. The details to compute these quantities can be found in Ref. [14].

The dynamic programming method gives the density of states,  $g(E)$ , the count of all the different conformations of the

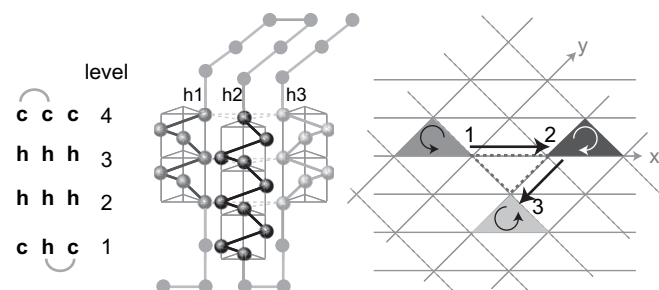


Fig. 9. The Ascending Level Model: the helices in the bundle are represented in terms of levels or layers on the lattice. Here, a 4-level 3-helix bundle is shown on a cubic lattice together with its associated h c sequence. Also shown is a slice through the  $XY$  plane, indicating how the 3 strands are situated in a given layer on the lattice. The dotted line indicates inter-helical interactions.

chain, from which we can then get the Boltzmann-weighted conformational populations,

$$p(E) = Q^{-1}g(E)\exp(-\beta E),$$

where  $\beta = 1/kT$  is the reciprocal of the Boltzmann constant times temperature and  $Q$  is the partition function.

#### 4.2. Predictions of the thermal properties of helices and helix bundles

Any helix-bundle model must first be able to make acceptable predictions for the transition between a single helix and its coil ensemble state. Fig. 10 shows that the ALM model predictions [4] are in good agreement with calorimetric and spectroscopic data on the thermal denaturation of a 50-mer helical peptide, called the Baldwin peptide, that has been studied extensively experimentally [21]. Fig. 11 shows that the model predicts well chemical denaturation for various chain lengths.

A more challenging test is on helix bundles, which are chain folds where both local and nonlocal interactions may contribute to the protein stability. Fig. 12 shows a prediction [4] for folding stability of the 58-residue 3-helix-bundle B domain of protein A [22] and Fig. 13 shows the corresponding

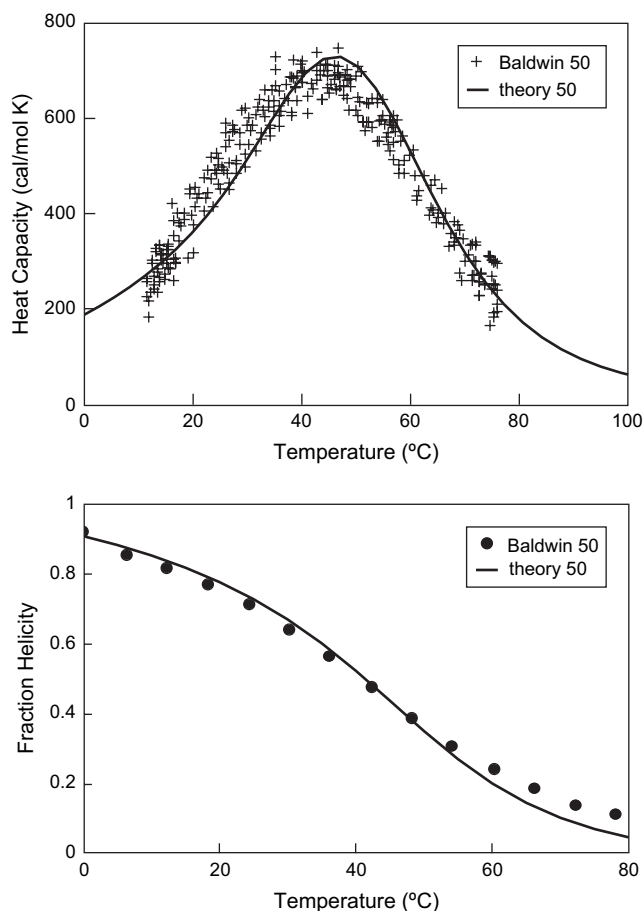


Fig. 10. The ALM model captures the helix–coil transition heat capacity (a) and denaturation profile (b) for the Baldwin peptide, Ac–Y(AEAAKA)<sub>8</sub>F–NH<sub>2</sub> [21]. The interaction energy for hydrophobic interactions is taken to be 0.2 kcal/mol and for hydrogen bonds is 0.8 kcal/mol.

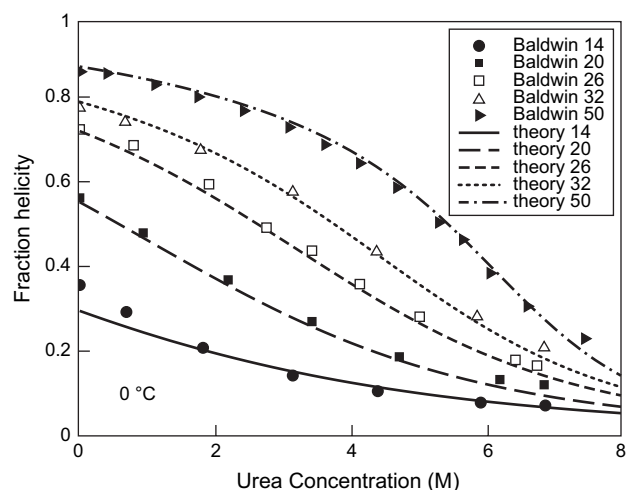


Fig. 11. The ALM model captures well urea denaturation for Baldwin peptides with chain lengths from 14 to 50 monomers.

predictions for combined thermal and guanidine denaturation of a different 3-helix bundle, called  $\alpha_3C$  [23].

Recently, Chan and his colleagues have shown that even the most cooperative models, based on nonphysical potentials, such as G $\bar{o}$  models, do not predict cooperativities as high as those that are observed in protein folding experiments [24–26]. The most cooperative folding model is currently that of Kaya and Chan which is based on a favorable coupling between helix formation and the packing of the native core, in addition to an extra stabilizing energy for the ground state [27]. Their model predicts a  $\Delta H_{vH}/\Delta H_{cal}$  ratio of 0.91 for a 55-mer chain sequence compared with a  $\Delta H_{vH}/\Delta H_{cal}$  ratio of 0.83 for a similar length sequence [4]. Our model allows us to interpret the stabilities and cooperativities in helix-bundle protein folding. In particular, it shows that folding cooperativity resides neither in the helix–coil transitions of the individual helices nor in a simple hydrophobic collapse, but rather in a coupling between the local and nonlocal interactions [4]. Our model is able to provide the first well-sampled energy landscape prediction for any real protein (see Fig. 14). It shows that the thermodynamic barrier to fold is very deep on the landscape and corresponds to the step of taking two partial helices and pairing them together to seed an assembly of two helices in the bundle.

A related model [5] of two-helix bundles uses formal grammar theory to combine the Zimm–Bragg model for local interactions in alpha helices [28], which can be written as a weighted finite-state automaton, with a model of Chen and Dill for nonlocal interactions in RNA-like hairpins [29,30], which can be written as a weighted context-free grammar. This approach allows us to use dynamic programming to analyze sequences in  $O(n^3)$  time and agrees quite well with exact enumeration on a lattice, which takes exponential time.

The power of the computational linguistics-based dynamic programming method is in its ability to provide a systematic enumeration of fairly complex chain folds that can include both local and nonlocal interactions through factorizable partition functions.



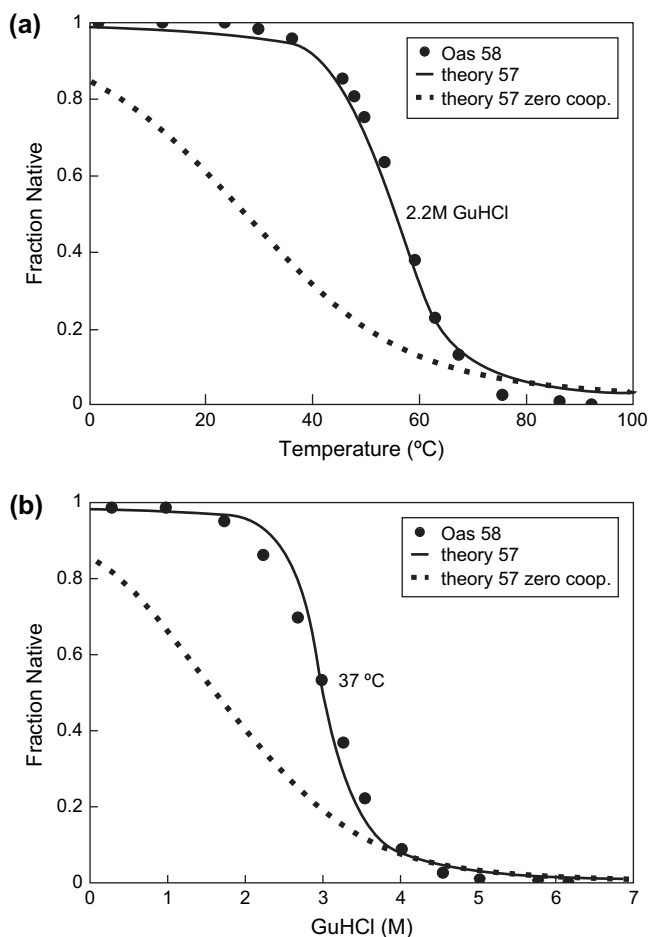


Fig. 12. Predictions of the 3-helix-bundle model vs. experiments of Oas et al. for denaturation by: (a) temperature and (b) denaturant guanidinium hydrochloride, for F13W\*. Thermal denaturation is carried out at 2.2 M GuHCl and chemical denaturation is carried out at 37 °C. The dotted curve in each plot shows the model prediction, with the cooperative interaction energy set to zero.

## 5. Other applications of computational linguistics to biological polymers

In this paper, we have reviewed two novel applications of computational linguistics to polymer conformational simulations. The common thread between the structure of natural language, on the one hand, and polymer conformations, on the other hand, is graphs that represent relationships among the different words in a sentence or among the different monomers in a polymer chain [31]. These graphs can be rank-ordered along the so-called *Chomsky hierarchy* (see Fig. 15). Graphs higher up on this hierarchy represent linguistic constructions or polymer conformations of increasing complexity. At the bottom of the hierarchy are simple *regular languages*, corresponding to conformations whose contacts are not recursively nested or crossed. These languages are generated by finite-state automata or Markov chains. The time required to parse strings of such languages is proportional to their chain length. Although these models are not generally expressive enough to capture the structures of biological polymer chains, the Zimm–Bragg model for  $\alpha$  helices can be cast as a regular language [5], because it treats a helix as a linear succession of turns. Hidden Markov Models [32], which were

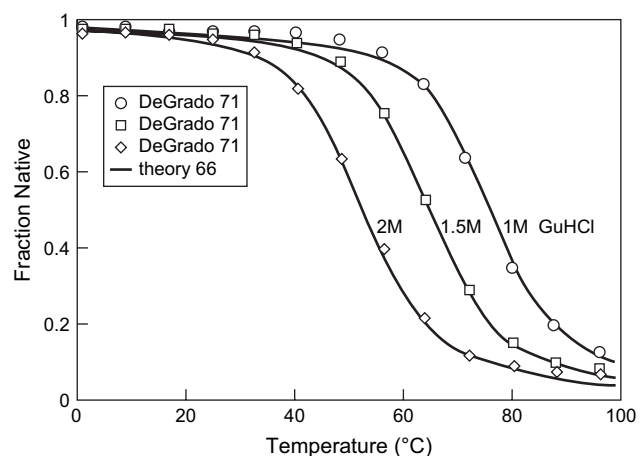


Fig. 13. Thermal denaturation for protein  $\alpha_3C$  at different GuHCl concentrations.

originally developed for computer speech recognition in 1970s and have been applied extensively to various bioinformatics problems such as gene finding and modeling of sequence families [33–35], also fall into this class of models. At the next level up are *context-free languages* (CFL), which include recursively *nested links*. In polymers, these are conformations that correspond to the stem loops of RNA secondary structure or  $\beta$  hairpins in proteins. Like HMMs, context-free grammars have been used in bioinformatics applications [36]. The time required to parse strings of CFLs, e.g. by algorithms such as CKY, is cubic in the length of the string ( $O(n^3)$ ). Therefore, dynamic programming algorithms for RNA sequences [37,38], which ignore pseudoknots and other crossing links, are equivalent to standard CKY [39] and hence also have cubic runtime.

However, natural languages and a number of more challenging biopolymer structures require graphs that contain crossing links. For example, Dutch is a language that cannot be parsed with context-free grammars. Protein conformations with crossing links include helix bundles and  $\beta$  sheets. RNA secondary structures such as pseudoknots also contain crossing links. In

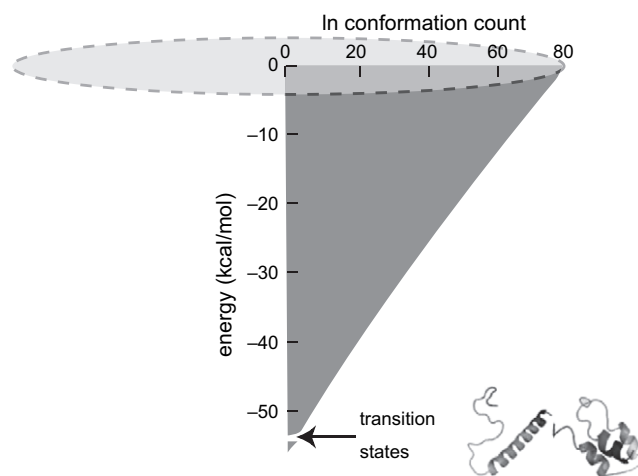


Fig. 14. The energy landscape for the 3-helix bundle F13W\*. The transition states are single helices and partially folded 3-helix bundles. Notice that the bottleneck state is far down the landscape.


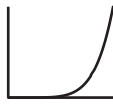
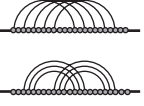
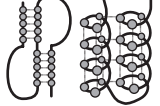
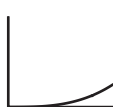
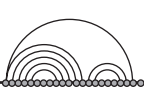
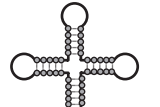

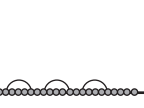

	Parsing	Graphs	Confs.
<b>CSL</b> Context-sensitive		?	?
<b>MCSL</b> Mildly Context-sensitive			
<b>CFL</b> Context-free			
<b>RL</b> Regular			

Fig. 15. The Chomsky hierarchy gives the different levels of complexity for parsing sentences, which are also useful for characterizing polymer conformations. At the bottom of the hierarchy are regular languages, corresponding to isolated polymer contacts. At the next level up are context-free languages, corresponding to RNA stem loops or simple  $\beta$  sheets in proteins (no crossing links; conformational searches can be done in polynomial time). At the next higher level are the mildly context-sensitive languages, corresponding to more complex chain folds such as RNA pseudoknots and helix bundles. These languages can still be parsed in polynomial time. Only the fully context-sensitive languages require exponential-time parsing algorithms.

Chomsky's original hierarchy, the third lowest level is that of the context-sensitive languages (CSL), which cannot be parsed in polynomial time. However, computational linguists have developed grammar formalisms that can capture the limited range of crossing structures that seem to exist in natural language in a way that makes it still possible to use efficient parsing algorithms with polynomial runtime. These formalisms systematically capture intermediate levels of the Chomsky hierarchy that lie above CFL, but well below CSL. One of these formalisms is the Tree-Adjoining Grammar (TAG) [40]. TAGs and their generalizations characterize an intermediate class of languages, called *mildly context-sensitive languages* [41,42]. It appears that crossings links in biopolymer conformations are limited in a similar way as the crossing dependencies in natural languages [43]. When applied to polymer conformational problems, these formalisms lead therefore to the prospect of efficient algorithms that are able to handle a variety of complex folded states of chains, such as those of proteins and RNA molecules. Some researchers have begun to explore these more powerful types of grammars for biological sequence analysis and structure prediction, including, for example, proposals for TAG-based models of RNA pseudoknots [44].

## 6. Conclusions

For many years, methods that originated in computational linguistics such as Hidden Markov Models and context-free

grammars have been important in bioinformatics [39], e.g. for the alignment of DNA sequences or for the prediction of simple RNA secondary structures. We have reviewed here two examples showing that new computational linguistics techniques are now allowing us to go beyond sequence comparisons and the prediction of simple secondary structure elements, giving insights into structures, folding mechanisms and the statistical physics of RNA and protein molecules. Computational linguistics-inspired dynamic programming methods can enumerate possible polymer conformations, leading to densities of states, partition functions, and predictions for the thermal properties of proteins and RNA molecules. Dynamic programming methods can also search conformational space efficiently, leading to new ways to predict native protein structures and explore folding mechanisms.

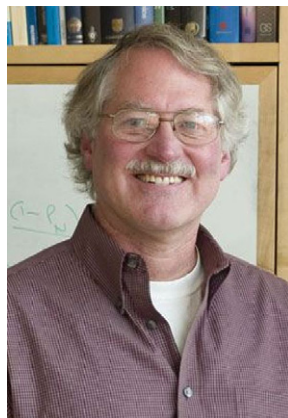
## Acknowledgements

This work is supported by NSF ITR grant 0205456. We would like to thank Sarina Bromberg for her invaluable help with the figures for this paper.

## References

- [1] Kasami T. An efficient recognition and syntax algorithm for context-free languages, Scientific report AFCRL-65-758. Bedford, MA: Air Force Cambridge Research Laboratory; 1965.
- [2] Younger D. Recognition and parsing of context-free languages in time  $n^3$ . *Information and Control* 1967;10:189–208.
- [3] Hockenmaier J, Joshi AK, Dill KA. Routes are trees: the parsing perspective on protein folding. *Proteins: Structure, Function, and Bioinformatics* 2007;66:1–15.
- [4] Lucas A, Huang L, Joshi A, Dill KA. Statistical mechanics of helix bundles using a dynamic programming approach. *Journal of American Chemical Society* 2007;129(14):4272–81.
- [5] Chiang D, Joshi AK, Dill KA. A grammatical theory for the conformational changes of simple helix bundles. *Journal of Computational Biology* 2006;13:21–42.
- [6] Chomsky N. *Syntactic structures*. The Hague: Mouton; 1957.
- [7] Plaxco KW, Simons KT, Ruczinski I, Baker D. Topology, stability, sequence, and length: defining the determinants of two-state protein folding kinetics. *Biochemistry* 2000;39:11177–83.
- [8] Collins M. Head-driven statistical models for natural language parsing. Ph.D. thesis, University of Pennsylvania; 1999.
- [9] Charniak E. A maximum-entropy-inspired parser. In: *Proceedings of the first meeting of the North American chapter of the Association for Computational Linguistics*. Seattle, WA; 2000. p. 132–9.
- [10] Fiebig KM, Dill KA. Protein core assembly processes. *The Journal of Chemical Physics* 1993;98.
- [11] Dill KA, Fiebig KM, Chan HS. Cooperativity in protein folding kinetics. *Proceedings of the National Academy of Sciences of the United States of America* 1993;90:1942–6.
- [12] Weikl TR, Dill KA. Folding rates and low-entropy-loss routes of two-state proteins. *Journal of Molecular Biology* 2003;329:585–98.
- [13] Weikl TR, Dill KA. Folding kinetics of two-state proteins: effect of circularization, permutation and crosslinks. *Journal of Molecular Biology* 2003;332:953–63.
- [14] Weikl TR, Palassini M, Dill KA. Cooperativity in 2-state protein folding kinetics. *Protein Science* 2004;13:822–9.
- [15] Merlo C, Dill KA, Weikl TR.  $\phi$  values in protein-folding kinetics have energetic and structural components. *Proceedings of the National Academy of Sciences of the United States of America* 2005;102:10171–5.

- [16] Voelz VA, Dill KA. Exploring zipping and assembly as a protein folding principle. *Proteins-Structure Function and Bioinformatics* 2007;66(4): 877–88.
- [17] Jernigan RL, Szu SC. Conformational energy minimization in the approximation of limited range interactions. *Macromolecules* 1979; 12:1156–9.
- [18] Lau K, Dill K. A lattice statistical mechanics model of the conformational and sequence spaces of proteins. *Macromolecules* 1989;22:638–42.
- [19] Levinthal C. Are there pathways for protein folding? *Journal de Chimie Physique* 1968;65:44–5.
- [20] Plaxco KW, Simons KT, Baker D. Contact order, transition state placement and the refolding rates of single domain proteins. *Journal of Molecular Biology* 1998;277:985–94.
- [21] Scholtz J, Marqusee S, Baldwin R, York E, Stewart J, Santoro M, et al. Calorimetric determination of the enthalpy change for the  $\alpha$ -helix to coil transition of an alanine peptide in water. *Proceedings of the National Academy of Sciences of the United States of America* 1991;88:2854–8.
- [22] Dimitriadis G, Drysdale A, Myers J, Arora P, Radford S, Oas T, et al. Microsecond folding dynamics of the F13W G29A mutant of the B domain of staphylococcal protein A by laser-induced temperature jump. *Proceedings of the National Academy of Sciences of the United States of America* 2004;101:3809–14.
- [23] Bryson J, Desjarlais J, Handel T, DeGrado W. From coiled coils to small globular proteins: design of a native-like three helix bundle. *Protein Science* 1998;7:1404–14.
- [24] Chan H. Modeling protein density of states: additive hydrophobic effects are insufficient for calorimetric two-state cooperativity. *Proteins: Structure, Function, and Genetics* 2000;40:543–71.
- [25] Moghaddam MS, Shimizu S, Chan H. Temperature dependence of three-body hydrophobic interactions: potential of mean force, enthalpy, entropy, heat capacity, and nonadditivity. *Journal of the American Chemical Society* 2005;127:303–16.
- [26] Chan H, Shimizu S, Kaya H. Cooperativity principles in protein folding. *Methods in Enzymology* 2004;380:350–79.
- [27] Kaya H, Chan H. Simple two-state protein folding kinetics requires near-Levinthal thermodynamic cooperativity. *Proteins: Structure, Function, and Genetics* 2003;52:510–23.
- [28] Zimm B, Bragg J. Theory of the one-dimensional phase transition in polypeptide chains. *The Journal of Chemical Physics* 1958;28:1246–7.
- [29] Chen S, Dill K. A statistical mechanical model for hydrogen exchange in globular proteins. *The Journal of Chemical Physics* 1995;103:5802–13.
- [30] Chen S, Dill K. Theory for the conformational changes of double-stranded chain molecules. *The Journal of Chemical Physics* 1998;109:4602–16.
- [31] Searls DB. The linguistics of DNA. *American Scientist* 1992;80:579–91.
- [32] Rabiner LR. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE* 1989;77:257–86.
- [33] Churchill GA. Stochastic models for heterogeneous DNA sequences. *Bulletin of Mathematical Biology* 1989;51:79–94.
- [34] Krogh A, Brown M, Mian IS, Sjolander K, Haussler D. Hidden Markov models in computational biology: applications to protein modeling. *Journal of Molecular Biology* 1994;51:79–94.
- [35] Eddy SR. Profile hidden Markov models. *Bioinformatics* 1998;14:755–63.
- [36] Sakakibara Y. Grammatical inference in bioinformatics. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2005;27:1051–62.
- [37] Nussinov R, Pieczenik G, Griggs JR, Kleitman DJ. Algorithms for loop matchings. *SIAM Journal on Applied Mathematics* 1978;35:68–82.
- [38] Zuker M. On finding all suboptimal foldings of an RNA molecule. *Science* 1989;244:48–52.
- [39] Durbin R, Eddy SR, Krogh A, Mitchison G. *Biological sequence analysis*. Cambridge University Press; 1998.
- [40] Joshi AK, Levy L, Takahashi M. Tree adjunct grammars. *Journal of Computer and System Sciences* 1975;10:136–63.
- [41] Vijay-Shanker K, Weir D, Joshi A. Characterizing structural descriptions produced by various grammatical formalisms. In: *Proceedings of the 25th annual meeting of the Association for Computational Linguistics*; 1987. p. 104–11.
- [42] Weir DJ. *Characterizing mildly context-sensitive grammar formalisms*. Ph.D. thesis, University of Pennsylvania; 1988.
- [43] Chiang D, Joshi AK, Searls DB. Grammatical representations of macromolecular structure. *Journal of Computational Biology* 2006;13:1077–100.
- [44] Uemura Y, Hasegawa A, Kobayashi S, Yokomori T. Tree adjoining grammars for RNA structure prediction. *Theoretical Computer Science* 1999;210:277–303.



**Ken A. Dill** is a Professor of Biophysics at the University of California, San Francisco, and Associate Dean of Research in the UCSF School of Pharmacy. He received his Ph.D. with Bruno H Zimm at UC, San Diego, and did postdoctoral work with Paul J. Flory at Stanford. He was a recipient of the Hans Neurath Award of the Protein Society in 1998; is a co-author (with Sarina Bromberg) of a physical chemistry textbook called *Molecular Driving Forces*; is a Fellow of the American Physical Society, the Biophysical Society, and the Institute of Physics; and was the President of the US Biophysical Society in 1998. He is interested in polymer statistical mechanics and its applications to understand the structures and properties of proteins and other biological systems.



**Adam Lucas** is currently an Assistant Professor of Mathematics and Computer Science at Saint Mary's College of California. He is also a visiting Research Associate with the Dill Group in the Department of Pharmaceutical Chemistry at UCSF. After a degree in Biochemistry from McGill University he did his Ph.D. in the area of Representation Theory at MIT. In 1999, he received an NIH postdoctoral fellowship to study protein folding at UCSF. Adam's current interests include polymer statistical mechanics, quantum information theory, and the use of peer instruction in mathematics education.



**Julia Hockenmaier** is currently a Postdoctoral Research Associate at the University of Pennsylvania. After a degree in Computational Linguistics from the University of Stuttgart and an MSc in Cognitive Science from the University of Edinburgh, she received her Ph.D. from the School of Informatics at the University of Edinburgh in 2003. She is interested in statistical parsing with linguistically expressive grammars and protein folding.





**Liang Huang** received his B.S. from Shanghai Jiao Tong University in 2003 and M.S.E. from the University of Pennsylvania in 2005, both in Computer Science. He is currently a Ph.D. candidate in the Department of Computer and Information Science, University of Pennsylvania. His research interests include computational linguistics, theoretical computer science, and their applications to computational biology.



**Aravind K. Joshi** is the Henry Salvatori Professor of Computer and Cognitive Science and a former Director of the Institute for Research in Cognitive Science (IRCS) both at the University of Pennsylvania. He is a Fellow of IEEE, ACM, AAAI, and a member of the National Academy of Engineering. Some of his recent honors include the Lifetime Achievement Award by the Association for Computational Linguistics, and the David Rumelhart Award from the Cognitive Science Society. He works in the areas of natural language processing (NLP), mathematical and processing models of language, artificial intelligence, and cognitive science. Recently he is also working on the applications of NLP techniques to some aspects of the modeling of folded structures of biological sequences.



**David Chiang** is a Research Assistant Professor at the University of Southern California and a Computer Scientist at the USC Information Sciences Institute. He received an AB/SM in Computer Science from Harvard University in 1997, and a Ph.D. in Computer and Information Science from the University of Pennsylvania in 2004. After a research fellowship at the University of Maryland Institute for Advanced Computer Studies, he joined the USC Information Sciences Institute in 2006, where he currently works on formal grammars for statistical machine translation.