Predicting Protein Folding Kinetics via Temporal Logic Model Checking

Christopher James Langmead[*†], **Sumit Kumar Jha**[*]

May 2007
CMU-CS-07-132

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

[*]Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA 15213.
[†] Department of Biological Sciences, Carnegie Mellon University, Pittsburgh, PA, USA 15213

E-mail: cjl@cs.cmu.edu

This research is supported by a Young Pioneer Award from the Pittsburgh Lifesciences Greenhouse and a CAREER award from the U.S. Department of Energy.

Keywords: protein folding, model checking, temporal logic

Abstract

We present a novel approach for predicting protein folding kinetics using techniques from the field of *model checking*. This represents the first time model checking has been applied to a problem in the field of structural biology. The protein's energy landscape is encoded symbolically using *Binary decision diagrams* and related data structures. Questions regarding the kinetics of folding are encoded as formulas in the temporal logic CTL. Model checking algorithms are then used to make quantitative predictions about the kinetics of folding. We show that our approach scales to state spaces as large as 10^{23} when using exact algorithms for model checking. This is at least 14 orders of magnitude larger than the number of configurations considered by comparable techniques. Furthermore, our approach scales to state spaces at least as large as 10^{32} unique configurations when using approximation algorithms for model checking. We tested our method on 19 test proteins. The quantitative predictions regarding folding rates for these test proteins are in good agreement with experimentally measured values, achieving a correlation coefficient of 0.87.

1 Introduction

In the world of proteins, form usually follows function. Consequently, proteins are often studied in terms of their atomic-resolution structures. A detailed analysis of an enzyme's active site, for example, may reveal the mechanism by which it catalyzes a given reaction. Protein structures are not static, however, and conformational changes often play important functional roles. Moreover, large-scale conformational changes are also associated with a number of diseases, most notably the prion-related diseases. For these reasons, and others, it is interesting to study *how* a given protein moves between conformations. Such examinations may provide valuable insights into basic biology and pathology, as well as to the design of therapeutic or preventative interventions for certain classes of disease.

In this paper, we focus on what is typically the largest conformational change a protein will exhibit — folding. By folding we refer to the act of moving from a completely *denatured* form to the so-called *native* configuration. Unfortunately, there is no experimental technology that can provide atomic-resolution detail into the entire process of folding (or any other large-scale conformational change, for that matter). For this reason, computational methods are used to study large-scale conformational changes, including folding. Our work builds on prior research on the protein *unfolding* problem. In contrast to the well-known protein folding problem, the unfolding problem assumes that the native structure is already known. The computational challenge is to find low energy pathways between the unfolded and folded states. More specifically, we consider the Gō theory of (un)folding [12] wherein the folding process is driven by the formation of *native contacts* between residues (i.e, those present in the native structure). Non-native interactions deemed negligible, and are therefore ignored. Obviously, this is a highly simplified theory of folding. Nevertheless, this theory has been shown capable of making accurate quantitative predictions regarding the kinetics of folding (e.g., [1, 8, 11, 16]).

Like previous algorithms for Gō-like theories, our algorithms operate on finite-state models of the protein's energy landscape. The primary contribution of our work lies in the observation that finite-state models of folding can be formally analyzed using techniques from the field of *model checking* [10]. Model checking refers to a family of algorithms for automatically verifying dynamic properties of concurrent reactive processes. Historically, model checking has been used to verify the correctness and safety of circuit designs, communications protocols, device drivers, and other classes of software. More recently, model checking algorithms have been introduced for analyzing the properties of stochastic systems. Such model checking algorithms for stochastic systems have been used in the field of systems biology to verify properties of biochemical and regulatory networks (e.g., [15]). To our knowledge, however, model checking has not been applied to any problem within the field of structural biology. This paper is the first to do so.

There are three primary advantages of a model-checking approach to studying protein folding pathways: *First*, model checking algorithms compute over symbolic representations of finite state models, not explicit representations. The computational complexity of model checking algorithms is polynomial in the size of the *encoding* of the finite-state model. Thus, if a given finite-state model can be compressed, extremely large state spaces can be considered. Unfortunately, finding a minimal encoding for an arbitrary finite-state model is NP-hard. However, good heuristics for finding compact encodings exist. For example, model checking algorithms have been able to verify

properties of systems having more than 10^{20} states since 1990 [7], and have been applied to systems with as many as 10^{120} states [5, 6]. In this paper, we show that using exact algorithms for model checking, energy landscapes with as many as 10^{23} states are tractable. This is at least 14 orders of magnitude larger than has been attempted by comparable algorithms for studying protein folding pathways. We also show that energy landscapes with at least 10^{32} states are tractable when using approximation algorithms for model checking. *Second*, model checking relies on formulas in a temporal logic to express precise queries about the behavior of the finite-state model. Temporal logics are very expressive and can be used to ask many questions of interest to protein folding. *Third*, model checking algorithms are exact; they are not simply a means for sampling or simulating the behavior of a system. There are, however, finite-state models that are too large for traditional model checking algorithms. For these, we use an algorithm for performing approximate model checking [19] which provides a guarantee on the quality of the computed result.

The organization of this paper is as follows: In Section 2, we define our model of protein folding. In Section 3, we briefly discuss model checking, and demonstrate how to encode the protein folding problem in a form suitable for model checking. In Section 4, we report the results of applying our method to 19 proteins and show that our quantitative predictions of folding rates are well-correlated with experimental values. We conclude with a discussion of ongoing work in applying model checking to the study of protein folding pathways.

2 A Simplified Model of Protein (Un)Folding

In this section we describe our model of protein folding; it is identical to that used in [16] and very similar to those reported elsewhere [1, 8, 11].

The thermodynamics of folding is governed by the Gibbs free-energy: $\Delta G = \Delta E - T\Delta S$. Here, E is the energy (in kcal mole^{-1}) of inter-residue interactions (e.g., hydrogen bonds, hydrophobic interactions, etc), S is the configurational entropy (in kcal mole^{-1} K^{-1}), and T is the absolute temperature (in Kelvin). Free energy is a balance between the stabilizing contributions of inter-residue interactions and the loss of configurational entropy as the protein folds.

Definitions: Let $P = \langle a_1, a_2, ..., a_n \rangle$ be a protein with n amino acids (aka residues) and m atoms. Let $\mathcal{C} \subset \mathbb{R}^{3m}$ be the set of possible configurations/embeddings of P such that each $C_i \in \mathcal{C}$ is consistent with the laws of physics. Let $C_F \in \mathcal{C}$ be the native configuration of P as determined by, say, X-ray crystallography. Following [16] we define a *contact* as two non-hydrogen atoms from two different residues that are within 4 Å of each other[1]. Contacts between residues (i, $i \pm 1$) and (i, $i \pm 2$) are ignored because they tend to be present in every configuration of P. A *contact map*, **M**, is an $n \times n$ matrix where element $\mathbf{M}(i, j)$ is the number of contacts between residues i and j. We define a separate *contact strength map*, $\mathbf{M}_S$, that is the same size as **M** but whose elements are obtained by mapping the elements of **M** as follows: 1-5 contacts $\mapsto$ 1; 6-10 contacts $\mapsto$ 2; 11-15 contacts $\mapsto$ 3; 16-20 contacts $\mapsto$ 4. Intuitively, $\mathbf{M}_S$ classifies contacts as being weak, medium, strong, or very strong.

[1] 1 Å = $10^{-10}m$.

The model assumes that folding is driven by the formation of the native contacts, and that non-native interactions are negligible. Therefore, the state space of the protein can be modeled using a binary string, $\mathbf{B} \in \{0,1\}^n$. Here, $\mathbf{B}(i)$ is 0 if the ith residue is completely unfolded and 1 if it is folded. There is an entropic penalty for each 1 in $\mathbf{B}$ which must be compensated for by the stabilizing energies of the native contacts. In particular, if $\mathbf{B}(i) = \mathbf{B}(j) = 1$, then we assume that the contacts between residues i and j (if any) are formed, and that the energy of that interaction can be used to offset the entropic penalty.

Under this model, there are 2^n possible states. Let $\mathbf{B}_U$ be the bit string containing all 0's, and let $\mathbf{B}_F$ be the bit string of all 1's. $\mathbf{B}_U$ and $\mathbf{B}_F$ correspond to the unfolded and folded states, respectively. Every other bit string corresponds to a partially folded state. Each state can be mapped to its free energy as follows:

$$G(B) = \sum_{i}^{n} \sum_{j>i}^{n} \mathbf{M}_S(i,j)\mathbf{B}(i)\mathbf{B}(j)\alpha - T\sum_{i}^{n} \mathbf{B}(i)\beta \tag{1}$$

where α is the strength of a single contact and β is the entropic penalty for folding a single residue [2]. The *Boltzmann factor* (i.e., *weight*) for any given configuration is a function of its energy, the gas constant (R) and the temperature, T; it is given by: $w(B) = \exp(-G(B)/RT)$. Since we are only interested in changes in free energy (i.e., ΔG), we arbitrarily set $G(\mathbf{B}_U) = 0$.

A protein's *energy landscape* is constructed by applying Eq. 1 to every possible configuration. In this paper, it can be thought of as an n-dimensional discrete function. Computationally, our task is to find a low-energy path (or a set of paths) between $\mathbf{B}_U$ and $\mathbf{B}_F$ in the energy landscape. Thus, we must define a set of allowable transitions. Under the model, state s can only transition to those states that are similar. In practice, this means that transition are only allowed between pairs of states whose bit vector representations have small Hamming distance. In this paper, we allow transitions between pairs of states with Hamming distance 1. A toy example of the model for a 3-residue protein is shown in Figure 1.

2.1 Kinetics

The reaction kinetics of folding are described in terms of an *energy profile* along a chosen *reaction coordinate*. A reaction coordinate is a projection of the energy landscape onto one of lower-dimension. The energy profile tracks the total energy for each position along the reaction coordinate. Given an appropriately chosen reaction coordinate, one can make quantitative predictions regarding the rate of folding from the energy profile. There are a number of potentially relevant reaction coordinates from which to chose when studying protein folding including radius of gyration, solvent accessible area, number of folded residues, and so forth. Following Muñoz and Eaton [16], we will use the number of folded residues (i.e., the number of 1's in $\mathbf{B}$) as our reaction coordinate.

For each position $0 \leq k \leq n$, there are $\binom{n}{k}$ binary strings, each with its own energy. Let $\mathbf{B}_k = \{\mathbf{B} \in \{0,1\}^n \mid \sum_{i=1}^{n} \mathbf{B}(i) = k\}$ be the set of bit strings with k 1's and $n-k$ 0's.

[2] See [16] for more details on contact energies and entropic penalties.

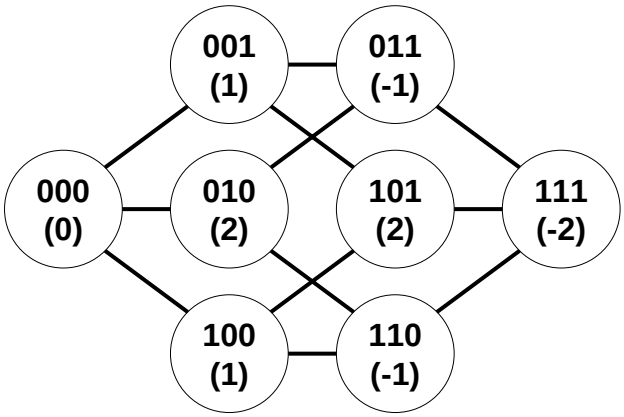

Figure 1: A toy example of the protein folding model. This finite-state model corresponds to a 3-residue protein. The state variables and the energy (in parens) are placed inside each node. The state labeled 000 is the unfolded state; the state labeled 111 is the folded state. In our experiments, we considered proteins with between 16 and 107 residues.

The Boltzman-weighted total energy for each position k along the reaction coordinate is $G_k = -RT \ln(\sum_{b \in \mathbf{B}_k} w(b))$. The energy profile for FKBP-12 is shown in Figure 2. In theory, it is possible to construct the energy profile by explicitly enumerating all 2^n binary strings. In practice, it is common to sample from the set of possible configurations. The algorithms reported in [1, 8, 11, 16], for example, operate on state space ranging in size from 10^4 to 10^9 configurations. In contrast, we seek to consider the *entire* space of binary strings by adopting symbolic techniques from the field of model checking.

We note that because the Boltzmann weight of a configuration is exponentially related to the negative energy of its configuration, we can compute an upper bound for each G_k by considering only the smallest-energy configurations for each k. It is these low-energy configurations we identify via model checking. Specifically, we seek to find the energy of the lowest-energy configuration for each k.[3] We will denote the lowest energy as $\tilde{G}_k$.

Given the value of $\tilde{G}_k$ for all $0 \leq k \leq n$, there are a number of ways to predict folding rates. Under a transition-state theory, for example, the folding rate, $k \propto k_0 \exp(-\Delta G^{\ddagger}/RT)$ where k_0 is a constant and $\Delta G^{\ddagger} = \mathrm{argmax}_k \, \tilde{G}_k - \tilde{G}_0$. In this paper, we use a more accurate way to predict the folding rate in terms of the rate of decay of the average number of folded residues starting from the folded state [16].

[3]It may be noted that our technique can be used to identify c lowest-energy configurations, for arbitrary integer c. For ease of presentation, we only consider the case of c=1 in this paper.

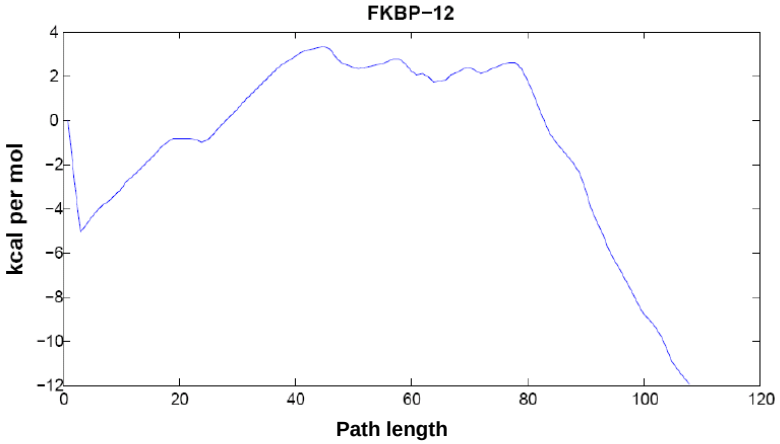

Figure 2: Energy profile for FKBP-12, as computed by our method.

3 Model Checking

The field of model checking was born from a need to formally verify the correctness of hardware designs. Since its inception in 1981, it has expanded to encompass a wide range of techniques for formally verifying finite-state transition systems, including those with stochastic behavior. Model checking algorithms are simultaneously theoretically very interesting and very useful in practice. Significantly, they have become the preferred method for formal verification in industrial settings over traditional verification methods like theorem proving, which often need guidance from an expert human user. A complete discussion of model checking theory and practice is beyond the scope of this paper. The interested reader is directed to [10] for a detailed treatment of the subject.

3.1 Modeling Concurrent Systems

Let AP be a set of atomic propositions. An atomic proposition, a, is a Boolean predicate referring to some property of the system. A *Kripke strucutre*, M, over AP is a tuple, $M = (S, S_0, R, L)$. Here, S is a finite set of states, $S_0 \subseteq S$ is the set of initial states, $R \subseteq S \times S$ is a total transition relation between states, and $L : S \mapsto 2^{AP}$ is a labeling function that labels each state with the set of atomic propositions that are true in that state. Variations on the basic Kripke structure exist. For example, if the system is stochastic, then we replace the transition relation, R, with a stochastic transition matrix, T where element $T(i, j)$ contains either a transition rates (for continuous-time Markov models) or a transition probability (for discrete-time Markov models).

3.1.1 Application to Energy Landscapes

The Kripke structure used in this paper closely follows the model of protein folding described in Section 2. The set of states, S, is isomorphic to the set of n-bit binary strings. The set of initial states, S_0, corresponds to ($\mathbf{B}_U$). The transition relation, R, allows transitions between pairs of states whose bit-vector representations have Hamming distance 1.

The labeling function, L, maps each state to an energy and works as follows: Recall that $\mathbf{B}_k$ is the set of bit strings where k bits are 1 and $n - k$ bits are 0. In this paper, our atomic propositions are generally of the form: "is the minimum energy of $\mathbf{B} \in \mathbf{B}_k = c$?". An interesting property of proteins is that that the energies of folding are bounded to a relatively small, constant-size range. In particular, the difference between $G(\mathbf{B}_U)$ and $G(\mathbf{B}_F)$ is generally 1 to 10 kcal mol^{-1}. The energy barrier which separates the unfolded and folded states is also typically 10 kcal mol^{-1} or smaller at room temperature. Indeed, the energy barrier *must* be small, or else folding won't occur. Thus, the domain of possible energies is, in effect, bounded by a constant of around 20 kcal mol^{-1}. This range is not related to the size of the protein. The set of possible states, on the other hand, is exponential in the size of the protein. Due to the discrete nature of our energy function and the fixed precision of the parameters α and β in Eq. 1, we can then apply the pigeonhole principle and conclude that the number of unique energy values is also constant. This will ultimately lead to a very efficient representation of the labeling function, as discussed in the next section.

In summary, assuming a Gō-like model of folding, we have shown that a protein's energy landscape can be encoded as a Kripke structure. In the model checking literature, Kripke structures are not represented explicitly, but rather symbolically. In the next section we discuss techniques for representing Kripke structures symbolically.

3.2 Symbolic Encodings of Kripke Structures

The basis for symbolic encodings of Kripke structures, which ultimately facilitated industrial applications of model checking, is the reduced ordered Binary Decision Diagrams (BDDs) introduced by Bryant [4]. BDDs are directed acyclic graphs that symbolically and compactly represent binary functions, $f : \{0,1\}^n \mapsto \{0,1\}$. While the idea of using decision trees to represent boolean formulae arose directly from Shannon's expansion for Boolean functions, two key extensions made to it were the use of a fixed variable ordering and the sharing of sub-graphs. The first extension made the data structure canonical while the second one allowed for compression in its storage. A third extension, also introduced in [4], is the development of an algorithm for applying Boolean operators to pairs of BDDs, as well as an algorithm for composing the BDD representations of pairs of functions. Briefly, if f and g are Boolean functions, the algorithms implementing operators APPLY(f,g,op) and COMPOSE(f,g) compute directly on the BDD representations of the functions in time proportional to $O(|f||g|)$, where $|f|$ is the size of the BDD encoding f. BDDs can be generalized to Multi-terminal BDDs (MTBDD) [9], which encode discrete, real-valued functions of the form $f : \{0,1\}^n \mapsto \mathbb{R}$. Significantly, MTBDDs can be used to encode real-valued vectors and matrices, and algorithms exist for performing matrix addition and multiplication over MTBDDs [9]. These algorithms play an important role in several model checking algorithms for stochastic systems [3].

3.2.1 Application to Energy Landscapes

As previously mentioned, we can encode energy landscapes using Kripke structures. It follows, therefore, that energy landscapes can be encoded symbolically using a combination of BDDs and MTBDDs. In particular, the transition relation, R, and the labeling function, L, can be encoded using BDDs and MTBDDs, respectively.

In practice, the construction of the BDDs and MTBDDs is done automatically from a high-level language describing the finite-state system and its behavior. Here, we use the specification formalism of reactive modules [2] as provided in the model checking tool PRISM [13]. Briefly, each residue is modeled as a separate two-state process (i.e., folded or unfolded). Residues change state asynchronously, and only one residue is allowed to change at any given time (thereby enforcing the Hamming-distance rule). The set of possible states of the system corresponds exactly to the set of n-bit strings. The set of allowable transitions is ultimately encoded as a BDD and the labeling function is encoded as a MTBDD.

3.3 Temporal Logics

Temporal logic is a formalism for describing behaviors in finite-state systems. They have been used since 1977 to reason about the properties of concurrent programs [18]. There are a number of different temporal logics from which to chose, and different logics have different expressive powers. In this paper, we use a small subset of the Computation Tree Logic (CTL). CTL formulae can express properties of *computation trees*. The root of a computation tree corresponds to the set of initial states (i.e., S_0) and the rest of the (infinite) tree corresponds to all possible paths from the root. A complete discussion of CTL and temporal logics is beyond the scope of this paper. The interested reader is directed to [10] for more information.

The syntax of CTL is given by the following minimal grammar:

$$\phi ::= a \mid true \mid (\neg\phi) \mid (\phi_1 \vee \phi_2) \mid \mathbf{EX}\phi \mid \mathbf{E}[\phi_1 \mathbf{U} \phi_2]$$

Here, $a \in AP$, is an atomic proposition (e.g., "does state s have energy c?"); "true" is a Boolean constant; $\neg$ and $\vee$ are the normal logical operators; **E** is the existential path quantifier (i.e., "there exists some path from the root in the computation tree"); and **X** and **U** are temporal operators corresponding to the notions of "in the next state" and "until", respectively. Given these, additional operators can be derived. For example, "false" can be derived from "$\neg true$" and the universal quantifier, $\mathbf{AX}\phi$, can be defined as $\neg\mathbf{EX}\neg\phi$.

Given some path $\pi = \langle \pi[0], \pi[1], \dots \rangle$ through the computation tree, the semantics of a CTL formula are defined recursively:

$\pi \models a$ iff a $\in$ L(π[0])

$\pi \models true, \forall\pi$

$\pi \models \neg\phi$ iff $\pi \not\models \phi$

$\pi \models \phi_1 \vee \phi_2$ iff $\pi \models \phi_1$ or $\pi \models \phi_2$

$\pi \models \mathbf{EX}\phi$ iff $\pi[1] \models \phi$

$\pi \models \mathbf{E}[\phi_1 \mathbf{U} \phi_2]$ iff $\exists i \geq 0, \pi[i] \models \phi_2 \wedge \forall j < i, \pi[j] \models \phi_1$

Here, the notation"$\pi \models \alpha$" means that π *satisfies* α.

3.3.1 Application to Protein Folding

Clearly, CTL formulas can express a rich set of properties concerning reachability (e.g., "will the protein end up in a particular configuration?") and/or the logical ordering of events (e.g., "will the second residue fold before the first one?"). Numerous extensions to CTL exist which facilitate questions regarding explicit timings (e.g., "will the protein fold within t milliseconds?") or likelihoods (e.g., "what is the probability that the protein fold within t milliseconds?"). In this paper, we only consider CTL formulas of the following form: let $a_{kc} \in AP$ be an atomic proposition that asks "does the state s have k folded residues and have energy c?", the CTL formula $\mathbf{E}[true\ \mathbf{U}\ a]$ asks "Is there a path from S_0 to some other state, $s \in S$, such that $s \models a$?" To find the minimum energy state for fixed k, we can perform a binary search over different values of c.[4] Recall, that we argued that the range of energies is bounded by a constant and that the number of unique energy values is also constant. Therefore, the cost of the binary search is $O(1)$.

3.4 Model Checking Algorithms

Having defined a Kripke structure and the CTL formula, we can then use existing model checking algorithms for verifying the formula, given a symbolic encoding of the Kripke structure. A model checking algorithm takes a Kripke structure, $M = (S, S_0, R, L)$, and a temporal logic formula, ϕ, and finds the set of states in S that satisfy ϕ: $\{s \in S \mid M, s \models \phi\}$. The complexity of model checking algorithms varies with the temporal logic and the operators used. For kinds of formulas used in this paper (i.e., $\mathbf{E}[\phi_1 \mathbf{U} \phi_2]$), an explicit state model checking algorithm requires time linear in the size of the finite-state model and in the length of the formula ([10] p 35-36).

Of course, for very large state spaces, even linear time is unacceptable. *Symbolic model checking* algorithms operate directly on BDD/MTBDD encodings of the Kripke structure and CTL formula. Briefly, the temporal operators of CTL can be characterized in terms of fixpoints. Let $\mathcal{P}(S)$ be the powerset of S. A set $S' \subseteq S$ is a fixpoint of a function $\tau : \mathcal{P}(S) \mapsto \mathcal{P}(S)$ if $\tau(S') = S'$. Symbolic model checking algorithms define an appropriate function, based on the formula, and then iteratively find the fixpoint of the function. This is done using set operations that operate directly on BDDs/MTBDDs. The fixpoint of the function corresponds exactly to $\{s \in S \mid M, s \models \phi\}$. The interested reader is encouraged to read [10], ch. 6 for more details.

Explicit-state and symbolic model checking algorithms are exact. There are also approximation algorithms for model checking algorithms (e.g., [19]), which employ sampling techniques and hypothesis testing. Such algorithms provide guarantees, in terms of the probability of the property being true, and can scale to much larger state spaces. These do not use BDDs/MTBDDs, but rather

[4] In our experiments, we make use of extensions to CTL provided in the tool PRISM that allows one to ask for the minimum energy value directly. Therefore, we do not perform an explicit binary search.

operate on the high-level language description of the finite-state model (see Sec. 3.2). We explored the use of both exact and approximate algorithms for model checking in our experiments.

4 Experiments and Results

We replicated the experiments of Muñoz and Eaton [16] who made predictions on 19 proteins[5] The largest protein in that set, FKBP-12 (PDB id 1FKB), has 107 residues. Muñoz and Eaton consider state spaces in the range of size $O(10^3)$ to $O(10^9)$ states. In contrast, we have successfully performed exact model checking on state spaces of size $2^{76} \approx 10^{23}$ using 2GB of memory on a single processor of a 4-node cluster. The time taken for these experiments is shown in Table 1. For proteins up to 74 residues, the longest runtime was under 30 minutes. Then, there is a jump to almost 7 hours for a 76-residue protein. The increase in time is due to thrashing of virtual memory. In general, The computation time is dominated by the time to construct the MTBDD. The actual cost of performing the model checking is under 90 seconds. Both load time and model checking time are correlated with the length of the protein for proteins up to 74 residues, with a correlations of 0.77 and 0.78, respectively, ($p = .02$). However, these are not monotonically related to length. No significant correlations between load times, model checking time and actual folding rates were observed. We also ran experiments with an approximation algorithm for model checking [19]. These all completed in under 11 minutes. The time to perform approximate model checking is strongly correlated with protein length ($R = 0.97, p \ll 0.001$). The largest state space we considered using the approximation algorithm has $2^{107} \approx 10^{32}$ states.

Figure 2 shows one sample energy profile computed using model checking for the protein FKBP-12. Using the technique described in [16] for transforming the free-energy profile into a quantitative prediction of folding time, we predicted the folding times for each of the 19 proteins. The correlations between the logarithms of the predicted folding rates and the experimentally measured values [14] are shown in Figure 3. The correlation coefficient between predicted and experimental values is 0.87. By comparison, Muñoz and Eaton achieve correlation coefficients between 0.83 and 0.87 on the same proteins, depending on which approximation was used. Plaxco and co-workers developed a simple method for predicting folding rates based on contact order (a length-normalized average sequential distance between contacting residues) [17]. Their correlation coefficient on 18 of the 19 proteins studied in this paper was 0.64. The mean absolute error of our predictions is 1.55. In comparison, the mean errors reported for two different techniques on a similar, but not identical, set of proteins in [8] was 2.77 and 3.42, respectively.

5 Conclusions and Future Work

We have presented an approach to predict the rate of folding using techniques from the field of model checking. We believe this paper represents the first application of model checking to a problem in structural biology. The key advantages of this approach are that it scales to extremely

[5]The PDB ids of the 19 proteins are: 1APS, 1COA, 1CSP, 1FKB, 1FNF, 1HDN, 1LMB, 1MJC, 1NYF, 1PBA, 1PGB, 1PKS, 1SHG, 1SRL, 1TEN, 1URN, 2ABD, 2AIT, 2PTL.

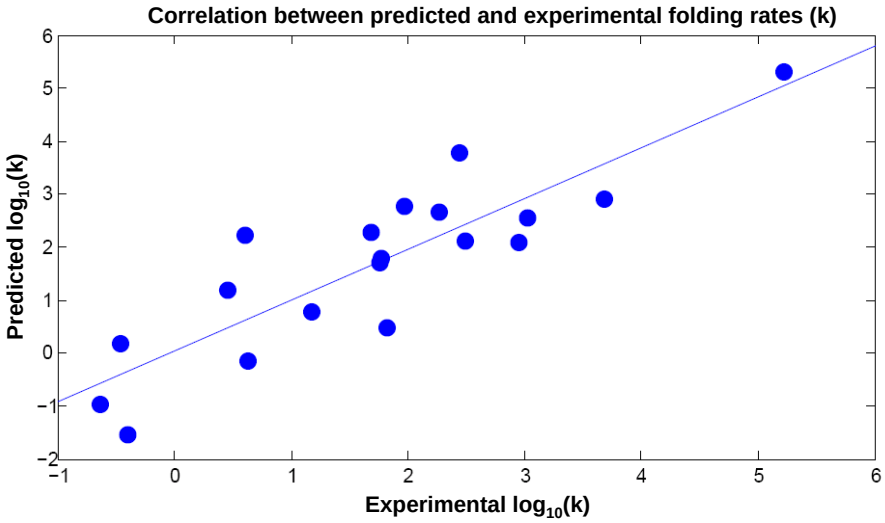

Figure 3: Scatter plot of log predicted (y-axis) and actual (x-axis) folding rates. The correlation coefficient is 0.87, $p \ll 0.001$

large state spaces and that it is exact. In terms of accuracy, our predictions of folding rate are well-correlated with experimentally determined values. However, it remains to be seen whether such levels of accuracy can be obtained when analyzing significantly larger proteins.

There are numerous extensions to this work that we intend to pursue. First, we have only begun to explore the kinds of queries that can be encoded in temporal logics. Second, a more thorough analysis of the relationship between the answers obtained via exact and approximate model checking is necessary. Finally, our model does not actually include any stochastic behavior. We have developed stochastic variants of our model of folding and we intend on applying model checking algorithms for stochastic systems to these. A comparison between the stochastic and non-stochastic techniques is planned.

Acknowledgments

We thank Dr. Edmund Clarke for helpful discussions on this topic. This research was supported by a U.S. Department of Energy Career Award (DE-FG02-05ER25696), and a Pittsburgh Life-Sciences Greenhouse Young Pioneer Award to C.J.L.

PDB Id	Residues	MTBDD Build Time (seconds)	MC Time (seconds)	Approximate MC Time (seconds)
1PGB	16	0.269	0.027	29.39
1SRL	56	313.546	18.083	188.69
1SHG	57	452.684	34.767	194.48
1NYF	58	712.788	64.882	195.41
1COA	64	1331.58	110.99	226.80
1CSP	67	973.664	6.57	248.75
1MJC	69	1963.879	86.139	267.32
2AIT	74	1753.331	85.205	318.15
1PKS	76	24647.21	10.55	319.61
2PTL	78	-	-	328.98
1PBA	81	-	-	335.82
1HDN	85	-	-	388.19
2ABD	86	-	-	378.94
1LMB	87	-	-	373.36
1TEN	90	-	-	415.54
1FNF	91	-	-	447.37
1URN	96	-	-	485.32
1APS	98	-	-	511.56
1FKB	107	-	-	611.59

Table 1: **Performance Statistics**. *MC = model checking*. Column 3 indicates whether exact or approximate model checking was used. MTBDD build times are only relevant to exact MC because approximate MC does not use MTBDDs. The approximation error bound was set to 1% of the energy for these experiments.

References

[1] E. Alm and D. Baker. Prediction of protein-folding mechanisms from free-energy landscapes derived from native structures. *Proc. Natl. Acad. Sci.*, 96(20):11305–11310, 1999.

[2] R. Alur and T. A. Henzinger. Reactive modules. *Formal Methods in System Design: An International Journal*, 15(1):7–48, 1999.

[3] C. Baier, E. Clarke, V. Hartonas-Garmhausen, M. Kwiatkowska, and M. Ryan. Symbolic model checking for probabilistic processes. In *Proc. 24th International Colloquium on Automata, Languages and Programming (ICALP'97)*, pages 430–440, 1997.

[4] Randal E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 35(8):677–691, 1986.

[5] J.R. Burch, E. M. Clarke, and D. E. Long. Symbolic model checking with partitioned transition relations. *Proc. 1991 Conf. on VLSI*, pages 49–58, 1991.

[6] J.R. Burch, E. M. Clarke, D. E. Long, K. L. McMillan, and D. L. Dill. Symbolic model checking for sequential circuit verification. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 3(4):401–424, 1993.

[7] J.R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Proc. Fifth Ann. IEEE Symposium on Logic in Computer Science*, pages 428–439, 1990.

[8] T. H. Chiang, M. S. Apaydin, D. L. Brutlag, D. Hsu, and J. C. Latombe. Predicting Experimental Quantities in Protein Folding Kinetics using Stochastic Roadmap Simulation. *Proceedings of the 2006 ACM International Conference on Research in Computational Molecular Biology (RECOMB)*, pages 410–424, 2006.

[9] E.M. Clarke, M. Fujita, P. C. McGeer, J.C.-Y. Yang, and X. Zhao. Multi-terminal binary decision diagrams: An efficient datastructure for matrix representation. *IWLS '93 International Workshop on Logic Synthesis*, 1993.

[10] E.M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, Cambridge, MA, 1999.

[11] S. O. Garbuzynskiy, A. V. Finkelstein, and O. V. Galzitskaya. Outlining folding nuclei in globular proteins. *J. Mol. Biol.*, 336:509–525, 2004.

[12] N. Gō and H. Taketomi. Studies on protein folding, unfolding and fluctuations by computer simulation. IV. Hydrophobic interactions. *Int J Pept Protein Res*, 13(5):447–461, 1979.

[13] A Hinton, M. Kwiatkowska, G. Norman, and D. Parker. PRISM: A tool for automatic verification of probabilistic systems. In H. Hermanns and J. Palsberg, editors, *Proc. 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'06)*, volume 3920, pages 441–444, 2006.

[14] SE. Jackson. How do small single-domain proteins fold? *Fold. Des.*, 3(4):R81–R91, 1998.

[15] M. Kwiatkowska, G. Norman, D. Parker, O. Tymchyshyn, J. Heath, and E. Gaffney. Simulation and verification for computational modelling of signalling pathways. pages 1666–1675, 2006.

[16] Munoz, V. and Eaton, W. A. A simple model for calculating the kinetics of protein folding from three-dimensional structures. *Proc. Natl. Acad. Sci.*, 96(20):11311–11316, 1999.

[17] K. W. Plaxco, K. T. Simon, and D. Baker. Contact order, transition state placement and the refolding rates of single domain proteins. *J. Mol. Biol.*, 277(4):985–994, 1998.

[18] Amir Pnueli. The temporal logic of programs. In *Proceedings of the 18th IEEE. Foundations of Computer Science (FOCS)*, pages 46–57, 1977.

[19] Håkan L. S. Younes and Reid G. Simmons. Probabilistic verification of discrete event systems using acceptance sampling. In *Proceedings of the 14th International Conference on Computer Aided Verification*, volume 2404, pages 223–235, Copenhagen, Denmark, July 2002. Springer.