Towards Big Biology: High-Performance Verification of Large Concurrent Systems

Ph.D. Thesis, VU University Amsterdam, 2012

Elżbieta Krępska
e.krepska@vu.nl

Thesis summary

Executable models are the most important tool of systems biology: they are used to unambiguously encode our understanding of complex biological processes, and they allow to conduct experiments otherwise technically impossible or unethical. In order to make model-based predictions about biology, the models should faithfully represent nature; to this end, the models must be verified: checked against the known biological evidence.

In this dissertation we investigate *how to verify large discrete models of biological systems*. Such systems typically consist of a large number of concurrently-executing small components. The main challenge when verifying them is *state explosion*: exponential growth of the state space with the number of concurrent components in the system. Therefore, the main focus of this work is scalability: ability to handle systems with very large state spaces. In order to achieve scalability, we use existing techniques, as well as propose new ones, in the fields of high-performance computing and model checking.

In Chapter 2 we study *Monte Carlo simulations*. In this method, a system is analyzed by performing and examining a large number of simulations; intuitively, a population of animals is emulated undergoing the studied biological process. We apply this approach to our model of cell fate determination during formation of a vulva (an egg-laying organ) in the *C. elegans* worm; the size of the state space of this model is in the order of 2^{715}. For each of the 64 genetic perturbations of our model, we performed 5000 simulations. Besides aggressive optimization of a single simulation, we parallelized the Monte Carlo experiments for a cluster of computers, which resulted, on a distributed machine with 256 cores, in reducing the time needed to run the entire suite of verification experiments to less than an hour.

While simulations may not reach all corners of a state space, *formal methods*, *i.e.* analyzing a model as a computer program, are able to check all states or all paths of a system. One such method is *abstract interpretation*, which allows to prove a property about a system by interpreting only parts of it relevant to the property under consideration. In Chapter 3, we propose BioCheck, an efficient procedure to prove stabilization (reaching a unique fixpoint) of systems. The tool proves stabilization by building the global liveness property from a chain of small liveness properties, which are fast to prove. BioCheck achieves scalability by applying state space exploration only locally to small pieces of the system rather than the entire

system as a whole. We used it to prove stabilization of a 3-D mesh of $200x500x5$ mammalian skin cells; the state space of this model contains 2^{6mln} reachable states.

In Chapters 4 and 5, we treat a state space as a large sparse graph, which has to be split between multiple machines to mitigate the state explosion problem; an important consequence of this approach is that verification algorithms need to be *parallelized*. Chapter 4 introduces HIPG, a high-level framework for writing such distributed graph algorithms. The key idea in HIPG is that a user expresses a graph algorithm by defining data stored by a vertex, as well as the vertex' methods. The framework allows to seamlessly execute methods on any graph vertex, local or remote. HIPG parallelizes the graph application automatically and handles the details of execution on a distributed machine.

Using HIPG we implemented SPINJADI, a distributed *enumerative model checker*, which explores a state space in an *on-the-fly* fashion: it starts with an empty graph, explores the system's initial state, its successors, the successors of the successors, and so on, until a bug is found or the state space exhausted. Properties of infinite executions are checked using an on-the-fly cycle detection algorithm by Brim *et al*. Using SPINJADI, we checked two mutual exclusion protocols, as well as a biological model of T-cell activation during an immune response.

In Chapter 5, we introduce TSCCDC, an efficient distributed algorithm to find terminal strongly connected components (TSCCs) in large graphs. In biology, TSCCs correspond to states of terminal differentiation (when a cell stops specializing), or to steady states. TSCCDC is a parallel divide-and-conquer graph algorithm: using reachability computations, a graph is split into four independent subgraphs, which cannot be 'crossed' by SCCs, and so can be searched recursively in parallel. We found TSCCDC was the only algorithm able to process our case study: a model of human haematopoietic (blood) cells, of size in the order of 2^{34} of vertices and edges.

To summarize, the contributions of this dissertation are as follows. Chapter 2 introduces a new approach to modeling in biology, which we use to create a model of vulval development in *C. elegans*—this model includes two previously published but not modeled hypotheses about the process. Chapter 3 proposes a novel scalable technique to prove stabilization of concurrent systems. In Chapter 4, we design and implement a new framework to easily write distributed graph algorithms; in addition, the framework is used to implement a distributed on-the-fly enumerative model checker. Chapter 5 introduces a novel efficient distributed algorithm to find terminal strongly connected components in a large graph.