Cells as computation

Aviv Regev and Ehud Shapiro

Although we are successfully consolidating our knowledge of the 'sequence' and 'structure' branches of molecular cell biology in an accessible manner, the mountains of knowledge about the function, activity and interaction of molecular systems in cells remain fragmented. Sequence and structure research use computers and computerized databases to share, compare, criticize and correct scientific knowledge, to reach a consensus quickly and effectively. Why can't the study of biomolecular systems make a similar computational leap? Both sequence and structure research have adopted good abstractions: DNA-as-string (a mathematical string is a finite sequence of symbols) and 'protein-as-three-dimensional-labelled-graph', respectively. Biomolecular systems research has yet to find a similarly successful one.

The hallmark of scientific understanding is the reduction of a natural phenomenon to simpler units. Equally important understanding comes from finding the appropriate abstraction with which to distill an aspect of knowledge. An abstraction — a mapping from a real-world domain to a mathematical domain — highlights some essential properties while ignoring other, complicating ones. For example, classical genetic analysis uses the 'gene as hereditary-unit' abstraction, ignoring the biochemical properties of genes as DNA sequences. A good scientific abstraction has four properties: it is relevant, capturing an essential property of the phenomenon; computable, bringing to bear computational knowledge about the mathematical representation; understandable, offering a conceptual framework for thinking about the scientific domain; and extensible, allowing the capture of additional real properties in the same mathematical framework.

For example, the DNA-as-string abstraction is relevant in capturing the primary sequence of nucleotides without including higher- and lower-order biochemical properties; it allows the application of a battery of string algorithms, including probabilistic analysis using hidden Markov models, as well as enabling the practical development of databases and common repositories; it is understandable, in that a string over the alphabet A, T, C, G is a universal format for discussing and conveying genetic information; and extensible, enabling, for example, the addition of a fifth symbol denoting methylated cytosine.

We believe that computer science can provide the much-needed abstraction for biomolecular systems. Advanced computer science concepts are being used to investigate the molecule-as-computation abstraction, in which a system of interacting molecular entities is described and modelled by a system of interacting computational entities. Abstract computer languages, such as Petri nets, Statecharts and the Pi-calculus, were developed for the specification and study of systems of interacting computational entities. Abstract computer languages, such as Petri nets, Statecharts and the Pi-calculus, were developed for the specification and study of systems of interacting computational entities. Abstract computer languages, such as Petri nets, Statecharts and the Pi-calculus, were developed for the specification and study of systems of interacting computational entities. Abstract computer languages, such as Petri nets, Statecharts and the Pi-calculus, were developed for the specification and study of systems of interacting computational entities. Abstract computer languages, such as Petri nets, Statecharts and the Pi-calculus, were developed for the specification and study of systems of interacting computational entities. Abstract computer languages, such as Petri nets, Statecharts and the Pi-calculus, were developed for the specification and study of systems of interacting computational entities. Abstract computer languages, such as Petri nets, Statecharts and the Pi-calculus, were developed for the specification and study of systems of interacting computational entities. Abstract computer languages, such as Petri nets, Statecharts and the Pi-calculus, were developed for the specification and study of systems of interacting computational entities. Abstract computer languages, such as Petri nets, Statecharts and the Pi-calculus, were developed for the specification and study of systems of interacting computational entities. Abstract computer languages, such as Petri nets, Statecharts and the Pi-calculus, were developed for the specification and study of systems of interacting computational entities. Abstract computer languages, such as Petri nets, Statecharts and the Pi-calculus, were developed for the specification and study of systems of interacting computational entities. Abstract computer languages, such as Petri nets, Statecharts and the Pi-calculus, were developed for the specification and study of systems of interacting computational entities. Abstract computer languages, such as Petri nets, Statecharts and the Pi-calculus, were developed for the specification and study of systems of interacting computational entities. Abstract computer languages, such as Petri nets, Statecharts and the Pi-calculus, were developed for the specification and study of systems of interacting computational entities. Abstract computer languages, such as Petri nets, Statecharts and the Pi-calculus, were developed for the specification and study of systems of interacting computational entities. Abstract computer languages, such as Petri nets, Statecharts and the Pi-calculus, were developed for the specification and study of systems of interacting computational entities. Abstract computer languages, such as Petri nets, Statecharts and the Pi-calculus, were developed for the specification and study of systems of interacting computational entities. Abstract computer languages, such as Petri nets, Statecharts and the Pi-calculus, were developed for the specification and study of systems of interacting computational entities. Abstract computer languages, such as Petri nets, Statecharts and the Pi-calculus, were developed for the specification and study of systems of interacting computational entities. Abstract computer languages, such as Petri nets, Statecharts and the Pi-calculus, were developed for the specification and study of systems of interacting computational entities. Abstract computer languages, such as Petri nets, Statecharts and the Pi-calculus, were developed for the specification and study of systems of interacting computational entities. Abstract computer languages, such as Petri nets, Statecharts and the Pi-calculus, were developed for the specification and study of systems of interacting computational entities. Abstract computer languages, such as Petri nets, Statecharts and the Pi-calculus, were developed for the specification and study of systems of interacting computational entities. Abstract computer languages, such as Petri nets, Statecharts and the Pi-calculus, were developed for the specification and study of systems of interacting computational entities. Abstract computer languages, such as Petri nets, Statecharts and the Pi-calculus, were developed for the specification and study of systems of interacting computational entities. Abstract computer languages, such as Petri nets, Statecharts and the Pi-calculus, were developed for the specification and study of systems of interacting computational entities. Abstract computer languages, such as Petri nets, Statecharts and the Pi-calculus, were developed for the specification and study of systems of interacting computational entities. Abstract computer languages, such as Petri nets, Statecharts and the Pi-calculus, were developed for the specification and study of systems of interacting computational entities. Abstract computer languages, such as Petri nets, Statecharts and the Pi-calculus, were developed for the specification and study of systems of interacting computational entities. Abstract computer languages, such as Petri nets, Statecharts and the Pi-calcula...