Algebras and Languages for Molecular Programming

Luca Cardelli

Microsoft Research

Nucleic acids (DNA/RNA) encode information digitally, and are currently the only truly 'user-programmable' entities at the molecular scale. They can be used to manufacture nano-scale structures, to produce physical forces, to act as sensors and actuators, and to do computation in between. Eventually we will be able to interface them with biological machinery to detect and cure diseases at the cellular level under program control. The basic technology to create and manipulate these devices has existed for many years, but the imagination necessary to exploit them has been evolving slowly. Recently, some very simple computational schemes have been developed that are autonomous (run on their own once started) and involve only short (easily synthesizable) DNA strands with no other complex molecules.

We now need programming abstractions and tools that are suitable for molecular programming, and this requires a whole hierarchy of concepts to come together. Low-level molecular design is required to produce molecules that interact in the desired controllable ways. On that basis, we can then design various kinds of 'logic gates' and 'computational architectures', where much imagination is currently needed. We also need programming languages to organize complex designs both at the level of gate design, and at the level of circuit design. Since DNA computation is massively concurrent, some tricky and yet familiar programming issues arise: the need to formally verify circuit designs to avoid subtle deadlocks and race conditions, and the need to design high-level languages that exploit concurrency and stochasticity.